

Article

Entropy-Guided BIST Instruction Set Compression for Auto-CP Embedded Flash Testing

Qi Liu * and David Wei Zhang

State Key Lab of ASIC & Systems, Fudan University, Shanghai 201203, China; dwzhang@fudan.edu.cn

* Correspondence: 17112020028@fudan.edu.cn; Tel.: +86-185-1605-6508

Abstract

Embedded Flash chip probing (CP) testing in an automotive Microcontroller Unit (MCU) involves long test flows and numerous test items. Conventional BIST-based testing relies on step-by-step Automated Test Equipment (ATE) instruction configuration, status polling, and result readout, causing considerable ATE interaction overhead that grows with the number of test items. This paper proposes an Auto-CP Built-In Self-Test (BIST) method with entropy-guided two-stage instruction set compression and on-chip autonomous scheduling for embedded Flash CP testing. Unlike conventional lossless compression that operates directly on raw instructions, the proposed method uses information entropy to quantify instruction redundancy and decomposes the original instruction stream into reusable instruction templates and a sequence scheduling stream. Structural compression removes repeated long instruction fields, while Huffman lossless coding shortens high-frequency symbols. On-chip decoding and scheduling logic then enable the ATE to load the compressed test program once and read the final results, eliminating step-by-step control of each test item. For a 40 nm 1 MB embedded Flash CP test at 15 MHz using an Advantest 93000 ATE, Auto-CP reduces the measured total CP test time from 42.65 s to 36.25 s, i.e., by 15%. The average ATE overhead ratio is lowered from 26% to 13%. The results show that the proposed method effectively reduces the ATE interaction overhead and improves the embedded Flash mass production CP test efficiency.

Keywords: automotive MCU; embedded flash; BIST; auto-CP; information entropy; Huffman compression; ATE overhead

1. Introduction

The Microcontroller Unit (MCU) serves as a core component in the field of automotive electronics. Driven by the growth of the global electric vehicle industry, automotive MCUs face rising requirements for advanced functionality, performance, design-for-test (DFT) [1–9], and efficient chip probing (CP) test strategies. As automotive electronic systems evolve toward higher levels of functional safety, the ISO 26262 standard imposes stringent requirements on the testability and reliability of safety-critical components [10,11]. In this context, automotive MCUs must undergo comprehensive CP testing that covers not only basic read/write verification but also read stress, program/erase cycling, margin read, and reliability-related screening [12,13]. Among these, embedded Flash within automotive-grade MCUs stands out as a critical module. Notably, the mass production CP testing cost of these MCUs is dominated by the Flash component, accounting for approximately 70% to 80% of the overall MCU test cost. For instance, in the case of a 40nm MCU Flash, the CP testing cost per ten thousand wafers can reach 1 to 2 million US dollars. Reducing



Academic Editor: John A. Kalomiros

Received: 7 September 2026

Revised: 8 October 2026

Accepted: 13 October 2026

Published: 15 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

the testing time by even 10% could yield substantial cost savings, potentially lowering expenses by 100,000 to 200,000 US dollars.

During embedded Flash testing, upgrading existing ATE hardware remains challenging, making the acceleration of high-voltage analog measurement a critical path to reducing the CP test time. In conventional BIST/ATE cooperative testing, the ATE repeatedly configures BIST registers, triggers tests, polls status, and reads results for each test item. This step-by-step interaction introduces significant overhead, especially for long test sequences. The total CP test time consists of the actual Flash test time, BIST internal overhead, and step-by-step ATE interaction overhead. Among these, the ATE interaction overhead is dominated by repeated configuration, invocation, polling, and readout, and is closely tied to the number of ATE interactions. The Flash test time itself is determined by physical characteristics and is difficult to reduce through test program optimization alone. Therefore, reducing the number of ATE interactions is a key optimization target. Furthermore, the trimming of high-voltage analog signals (VCG, VSL, and VEG) in automotive Flash macros requires traversing multiple trimming levels with frequent ATE measurement, accounting for nearly 7% of the total CP test time.

Existing studies have addressed embedded Flash CP testing from several directions. Recent works [14–17] have advanced embedded memory test and BIST in process, diagnosis, scheduling, and waveform generation, but they do not target this interaction overhead. Reference [5] align ATE resources with MCU Flash test requirements, and the MCU Flash BIST engine supports parallel testing of heterogeneous macros to improve multi-Flash efficiency. Previous work [18] proposed an auto-trimming DFT method that removes ATE from analog measurements by integrating a high-precision voltage comparator and BIST logic, reducing the trimming time from 2.94 s to 0.23 s and total CP time by 6.35%. However, these methods still depend on ATE for high-voltage analog testing and do not eliminate the repeated ATE interactions in conventional BIST/ATE cooperative testing. For automotive MCUs, long CP sequences, massive address traversals, and repeated configurations produce a highly redundant BIST instruction stream, causing significant ATE overhead through frequent instruction loading and status polling.

To address these limitations, this paper extends our previous auto-trimming method [18] with an entropy-guided two-stage compression and on-chip autonomous scheduling scheme. Because BIST instruction usage in a fixed CP flow is highly non-uniform and contains repeated templates, entropy quantifies instruction stream redundancy, while template/sequence decomposition removes repeated fields. Structural compression then prepares a compact symbol set for Huffman coding. Unlike conventional test data compression on raw vectors, the proposed method targets the BIST instruction stream, enabling one-time loading and autonomous on-chip execution.

The main contributions of this work are as follows: (1) an entropy-guided redundancy analysis is presented to quantify the compression potential of the CP BIST instruction stream; (2) an instruction/sequence decomposition mechanism is developed to separate reusable operation templates from flow control scheduling information; (3) Huffman lossless coding is adopted to further shorten the code length of high-frequency symbols; and (4) an on-chip Auto-CP controller is implemented to decode and autonomously execute the compressed test program.

The remainder of this paper is organized as follows. Section 2 analyzes the conventional BIST/ATE interaction flow and the source of the test overhead. Section 3 presents the overall Auto-CP BIST architecture with on-chip autonomous scheduling. Section 4 details the entropy-guided instruction set compression algorithm, including redundancy analysis, structural compression, and Huffman coding. Section 5 reports the experimental results and discusses the test efficiency and area overhead. Section 6 concludes the paper.

2. Conventional Flash BIST/ATE Flow and Overhead Model

The purpose of embedded Flash CP testing is to screen out dies with functional failures, marginal behavior, or potential reliability risks at the wafer level. Unlike SRAM or RAM, embedded Flash testing involves erase, program, high-voltage operation, margin read, stress conditions, and reliability-related screening in addition to basic read/write verification. Since conventional RAM March tests are not directly applicable to Flash memories, dedicated Flash test procedures are required.

From the perspective of test flow organization, embedded Flash CP testing is characterized by diverse operations, long test sequences, and repeated configurations. Multiple test items require repeated configuration of test modes, address ranges, data backgrounds, and result-checking conditions. Similar configurations and batch operations are repeatedly applied across different address regions, data patterns, and test modes. As a result, the generated BIST instruction stream is large and contains significant structural redundancy.

In a conventional embedded Flash BIST architecture, as shown in Figure 1, the external test program configures on-chip BIST-related registers through a test access interface or bus interface. Existing disclosed eFlash BIST solutions include command, pattern, data, and status registers; the BIST controller initiates test operations according to the register configuration, and the results are stored in status registers for external readout. Functionally, Flash BIST instructions can be grouped into configuration instructions, functional operation instructions, test mode instructions, and batch BIST instructions. Memory BIST results are typically reported through DONE/FAIL status signals, and some implementations also use address or data registers to locate failing positions.

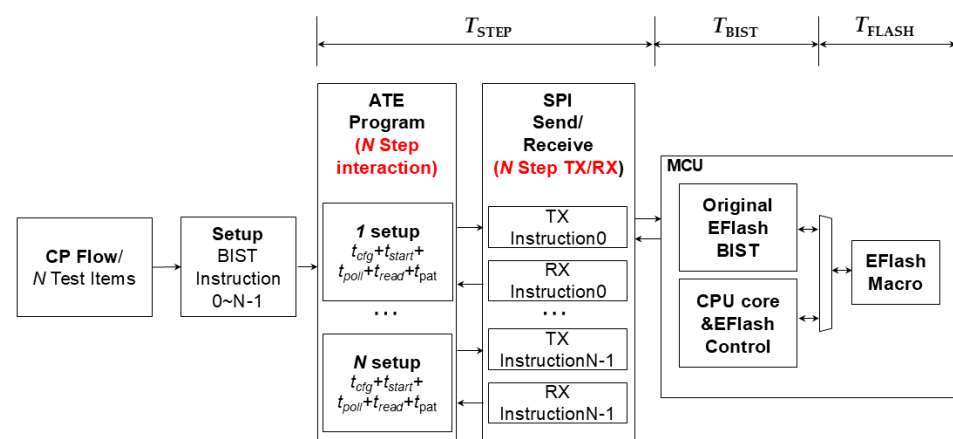


Figure 1. Conventional Flash BIST/ATE interaction flow.

Consequently, in conventional BIST/ATE cooperative testing, the ATE repeatedly configures BIST instructions or registers, waits for on-chip execution, reads status or fail information, and determines the next test step. The total CP test time of the conventional flow can be expressed as

$$T_{CP} = T_{FLASH} + T_{BIST} + T_{STEP} \quad (1)$$

where T_{FLASH} denotes the actual test time applied to the Flash macro. It is mainly determined by the physical read, program, and erase characteristics of the embedded Flash and by the required CP test items, and it is difficult to significantly reduce through test program optimization alone. T_{BIST} denotes the on-chip BIST internal execution and control overhead, including the translation of BIST instructions into actual Flash test patterns inside the BIST circuit. It does not include the SPI transfer of the compressed program, which is accounted for separately in the Auto-CP timing model. T_{STEP} denotes the step-by-step ATE interaction

overhead caused by repeated command configuration, test invocation, status polling, and result readout.

In a conventional flow, T_{STEP} is mainly caused by repeated command configuration, test invocation, status polling, and result readout. It can be abstracted as

$$T_{\text{STEP}} \approx N \times (t_{\text{cfg}} + t_{\text{start}} + t_{\text{poll}} + t_{\text{read}} + t_{\text{pat}}) \quad (2)$$

where N is the number of ATE/BIST interactions; t_{cfg} , t_{start} , t_{poll} , and t_{read} represent the time consumed by configuration, test triggering, status polling, and result readout, respectively; and t_{pat} represents additional pattern-related ATE overhead.

The actual Flash test time T_{FLASH} is mainly determined by the physical read, program, and erase characteristics of the Flash macro and by the required CP test items. It is difficult to significantly reduce this component through test program optimization alone. In contrast, T_{STEP} is strongly related to the number of instruction interactions and the organization of the test flow.

As indicated by Equation (2), since N grows with the number of CP test items, test mode configurations, status checks, and result readouts, reducing N becomes a key optimization target for shortening the total CP test time. This observation directly motivates the proposed one-time loading and on-chip autonomous scheduling scheme, in which the compressed test program is loaded once and the on-chip Auto-CP controller executes the complete CP flow without repeated ATE intervention.

3. Proposed Auto-CP BIST Method with Instruction Set Compression and On-Chip Autonomous

The proposed Auto-CP method changes the test flow control mechanism, as shown in Figure 2. Instead of configuring and triggering BIST operations step by step, the ATE loads compressed instruction and sequence data into the BIST circuit once. The on-chip Auto-CP scheduler then decodes the sequence stream, invokes the corresponding instruction templates, executes Flash operations, records the intermediate results, and reports the final results to the ATE after the complete CP test flow is finished.

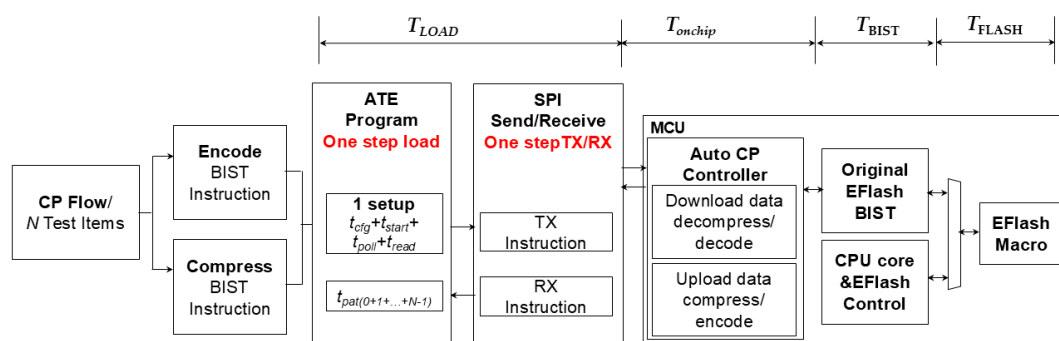


Figure 2. The detailed flow of the proposed Auto-CP BIST method.

With Auto-CP, the total CP test time can be expressed as

$$T_{\text{CP}} = T_{\text{FLASH}} + T_{\text{BIST}} + T_{\text{LOAD}} + T_{\text{onchip}} \quad (3)$$

where T_{LOAD} denotes the one-time compressed program loading and final result readout overhead, and T_{onchip} denotes the additional on-chip decoding and scheduling overhead.

Since the proposed method does not change the Flash test items themselves, T_{FLASH} remains almost unchanged and is not the main optimization target. The expected test

time reduction mainly comes from replacing repeated ATE interactions with one-time compressed loading and on-chip scheduling, which can be expressed as

$$\Delta T = T_{\text{STEP}} - (T_{\text{LOAD}} + T_{\text{onchip}}) \quad (4)$$

According to Equation (4), the reduction in the CP test time is obtained when the removed step-by-step ATE interaction overhead exceeds the additional overhead introduced by compressed program loading and on-chip execution. In conventional embedded Flash CP flows, the number of ATE/BIST interactions N increases with the number of test items, test mode configurations, status checks, and result readouts. Therefore, for long CP sequences with repeated operations, T_{STEP} can become a significant part of the total test time. For example, using the measured timings $T_{\text{STEP}} = 11.1$ s and $T_{\text{LOAD}} + T_{\text{onchip}} = 4.7$ s, Equation (4) gives $\Delta T = 11.1 - 4.7 = 6.4$ s.

The proposed Auto-CP method reduces this overhead through two related mechanisms. First, the original BIST command stream is reorganized into reusable instruction templates and a sequence scheduling stream. This representation removes repeated command fields and converts the step-by-step test flow into a compact program that can be loaded into the BIST circuit. Second, Huffman coding further reduces the size of the compressed instruction data, thereby decreasing the loading overhead T_{LOAD} . After the compressed program is loaded, the on-chip scheduler performs instruction decoding, sequence control, and result collection inside the BIST circuit. As a result, the effective number of ATE-controlled interactions is reduced from repeated per-test item operations to program loading and final result readout.

The added term T_{onchip} represents the cost of on-chip decoding and scheduling. This overhead is introduced by the Auto-CP controller, but it is incurred inside the chip and is amortized over the complete CP sequence.

4. Proposed Entropy-Guided Instruction Set Compression

4.1. Entropy-Guided Analysis of Instruction Redundancy

Given a fixed embedded Flash macro, BIST instruction set, and CP test flow, the mapping from test actions to BIST instructions is deterministic, and the generated instruction sequence consists of a finite set of instruction symbols. Since the occurrence counts of these symbols can be statistically obtained from the CP flow, the CP BIST instruction stream can be modeled as a discrete information source with a well-defined probability distribution.

Information entropy is a basic measure of uncertainty in information theory. Shannon formulated communication as the problem of reproducing a message selected from a set of possible messages and adopted a logarithmic measure for information. For a discrete random variable X with probability mass function $p(x)$, the entropy is defined as

$$H(X) = -\sum p(x) \log_2 p(x). \quad (5)$$

For the instruction set $C = \{C_1, C_2, \dots, C_n\}$ with occurrence probability p_i , the instruction stream entropy is

$$H(C) = -\sum p_i \log_2 p_i, \quad (6)$$

which gives the theoretical lower bound of the average code length for lossless coding. If $H(C)$ is much smaller than the original instruction width W_{orig} , the instruction encoding contains redundancy and has compression potential. To quantify this potential for each instruction class, the entropy-to-width ratio is defined as

$$R_{\text{EW}} = \frac{H(C)}{W_{\text{orig}}} \quad (7)$$

where W_{orig} denotes the original instruction width. A smaller R_{EW} indicates a larger gap between the original encoding width and the entropy lower bound, and therefore a larger compression potential.

As shown in Table 1, the entropy values are much smaller than the original instruction widths, e.g., 3.58 for the 8-bit set versus 5.05 for the 72-bit set. This gap arises because the CISC-style BIST instruction set contains many parameter fields and its usage frequency in a fixed CP flow is highly non-uniform: a few instructions are repeated thousands of times, while many others appear only a few times. Consequently, $H(C)$ is much smaller than W_{orig} , confirming the inherent compression potential of the CP instruction stream. A smaller R_{EW} indicates a larger gap and thus greater compression potential.

Table 1. Information Entropy Statistic of BIST Instruction.

Instruction Width or W_{orig}	Usage Count	Data Volume	$H(C)$	R_{EW}
8 bits	6591	52,728 bits	3.58	44.75%
24 bits	117	2808 bits	0.77	3.21%
40 bits	330	13,200 bits	3.31	8.28%
48 bits	2058	97,824 bits	2.58	5.38%
56 bits	66	3696 bits	3.74	6.68%
72 bits	3189	229,608 bits	5.05	7.01%

4.2. Instruction-Based Structural Compression

Although the entropy analysis shows that the number of distinct instruction types is limited (e.g., 8-bit, 24-bit, etc.), each type can potentially derive thousands of encoding variants after parameter adjustments. This is primarily because the CP test includes instructions that traverse specific address ranges in the embedded Flash, as shown in Table 1. Such an excessive number of encoding variants makes the instruction distribution highly scattered, posing a significant challenge for further compression at the bit level. Moreover, from the perspective of decompression circuit design, the large number of distinct encoding variants necessitates a larger storage space, which increases the area of the decompression circuit and consequently raises the chip cost.

To address these challenges, structural compression is required for the BIST instruction set. The instruction sequence required for the CP flow is divided into two parts, sequence and instruction, according to specific encoding rules. By employing secondary encoding and establishing a new queue structure, this method eliminates consecutive repeated instructions without altering the original BIST architecture. It significantly reduces the storage space occupied by the instruction sequence and minimizes the actual amount of instruction data transmitted from the ATE to the MCU, thereby lowering the ATE overhead and improving the test efficiency. This reduction makes it feasible to load all instructions into the BIST circuit in segments for automatic execution.

Accordingly, the original instruction stream is decomposed into two parts: (1) instruction templates, which store operation types, address modes, data patterns, target Flash identifiers, test modes, and read/write modes; and (2) sequence scheduling streams, which describe the invocation order of instruction templates, loop counts, conditional branches, result checks, and end flags. This decomposition effectively transforms repeated long instructions into shorter indices and flow control information. The instruction table can be regarded as a set of reusable functions, while the sequence stream acts as a calling script. Consequently, frequently used operations such as read, program, erase, margin read, and test mode switching are stored only once in the instruction table and repeatedly invoked by the sequence stream.

To implement this structural compression, the original instruction sequence undergoes a systematic re-encoding process, as shown in Figure 3. First, instructions are categorized according to their bit widths (8/24/32/40/48/56/72 bits) using a 3-bit Command Code. Second, these categorized instructions are reordered so that adjacent instructions of the same type are grouped together. Finally, the reordered sequence is encoded to compress identical adjacent instructions, ensuring that each unique instruction is stored only once. Crucially, given that the CP flow extensively relies on loop operations (e.g., traversing addresses where the address increments while other bits remain constant), a dedicated frame structure is incorporated into the sequence stream to encode the loop status, loop times, and step sizes. This allows a single sequence entry to represent hundreds or even thousands of repeated instructions, drastically reducing the required storage space and the data volume transferred from the ATE. This secondary encoding ultimately enables segmented instruction loading and on-chip autonomous execution.

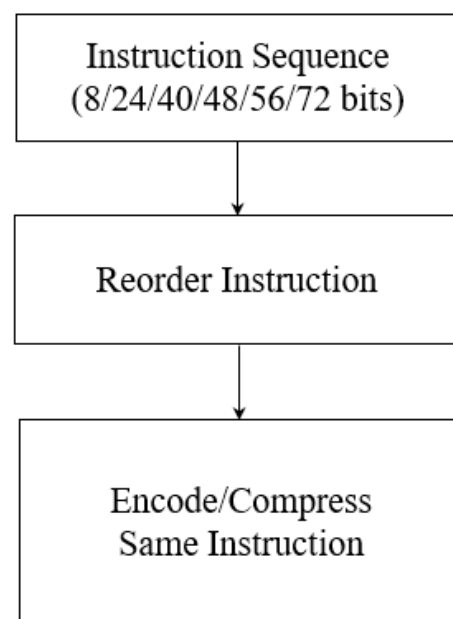


Figure 3. The process of structural compression.

To validate the effectiveness of the two-stage compression, Table 2 reports the data volume reduction for each instruction type. The overall data volume is reduced from 12,685 bits to 403 bits, achieving an average compression ratio of 3.2%. Among all instruction types, the 48-bit instructions show the most significant compression (to 0.5% of the original volume), as they are dominated by repetitive address traversal patterns that are efficiently captured by the sequence scheduling stream. This result directly confirms the entropy-based analysis in Section 4.1: the original instruction encoding contains substantial redundancy, and the proposed structural + Huffman compression effectively removes it.

Table 2. Structure Compression Ratio Result of BIST Instructions.

Instruction Width	8 Bits	24 Bits	40 Bits	48 Bits	56 Bits	72 Bits	Summary
Before compression (bits)	3368	144	640	528	224	7776	12,685
After compression (bits)	216	7	31	3	14	264	535
Compression Ratio	6.4%	5.1%	4.8%	0.5%	6.1%	3.4%	3.2%

4.3. Huffman Compression

The proposed compression is a two-stage process. In the first stage, structural compression decomposes the original CISC-style BIST instruction stream into instruction templates and a sequence scheduling stream, which reduces the instruction data volume and makes the symbol distribution more concentrated and suitable for subsequent coding. In the second stage, Huffman lossless coding is applied to the structurally compressed instruction set to further shorten the code length of high-frequency symbols. The two stages are complementary: structural compression removes repeated long instruction fields and produces a compact symbol set, while Huffman coding exploits the non-uniform frequency of these symbols.

After instruction decomposition, the compression target changes from original long instructions to instruction templates and scheduling symbols. Since the occurrence frequencies of these symbols remain non-uniform, probability-based variable-length lossless coding can further reduce the average code length.

In this work, four classical lossless compression methods—Run-length, Golomb, Dictionary, and Huffman coding—are compared. They represent different coding strategies and are widely used in test data and instruction stream compression. Since BIST instruction set compression must be lossless, these methods are appropriate candidates. Figure 4 compares their compression ratios, where lower values indicate better performance. Although Huffman coding is the most complex, it achieves the highest compression efficiency because it exploits symbol occurrence frequencies, assigning shorter codewords to high-frequency symbols and longer codewords to low-frequency symbols. This makes it well suited to CP instruction streams with non-uniform frequency distributions. Moreover, Huffman coding is lossless and has moderate decoding complexity.

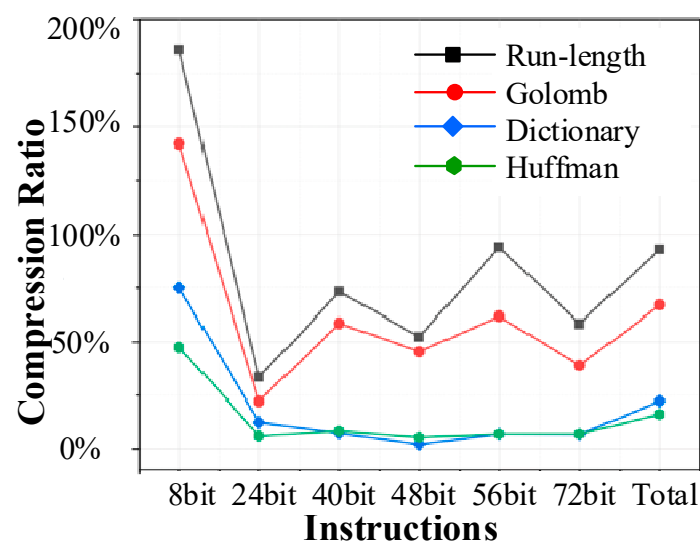


Figure 4. Compression ratios (defined as the compressed data volume divided by the original data volume) of the four compression methods.

As shown in Figure 5, the entropy-to-width ratio (R_{EW}) with two-stage compression is significantly improved compared with the case without two-stage compression. In Figure 5, $R_{EW} = H(C)/W_{orig}$ measures instruction stream redundancy; lower values indicate greater compression potential. The redundant encoding proportion is substantially reduced.

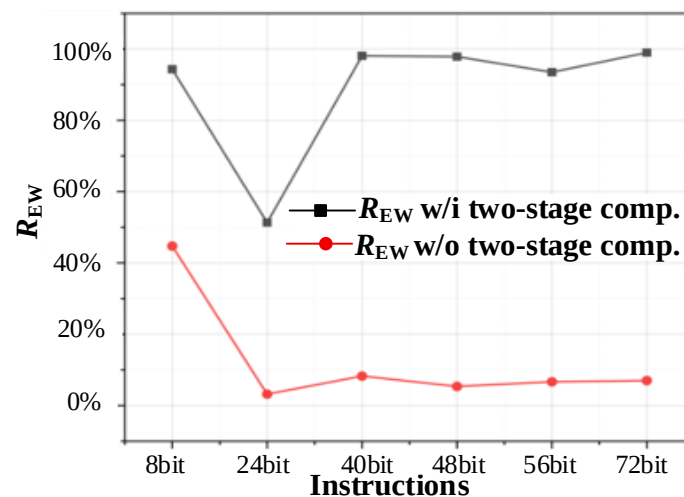


Figure 5. The entropy-to-width ratio $R_{EW} = H(C)/W_{orig}$ results with and without using two-stage compression.

5. Experimental Results

The proposed Auto-CP method is evaluated on a 40 nm 1 MB embedded Flash macro at 15 MHz using an Advantest 93,000 ATE. The test chip contains both the conventional eFlash BIST and the proposed Auto-CP BIST, as shown in Figure 6. The Auto-CP controller itself occupies $4213 \mu\text{m}^2$, which is approximately 0.35% of the 1 MB embedded Flash macro area.

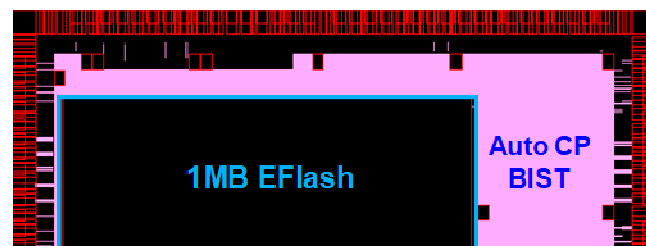


Figure 6. The layout of the proposed test chip.

The CP flow is divided into four segments according to the CP test flow and test item type, as summarized in Table X. Auto_0 contains 11 test items, mainly eFlash functional and stress tests. Auto_1 contains 19 test items, mainly eFlash stress tests. Auto_2 contains 101 test items, mainly eFlash functional margin, redundancy, and stress tests. Auto_3 contains 45 test items, mainly eFlash pre-program, Vread, and erase–retry tests. In the conventional BIST scheme, the ATE sends test vectors step by step. In the Auto-CP scheme, the CP flow is segmented, with compressed instructions and scheduling sequences loaded into the BIST circuit once for on-chip autonomous execution.

Figure 7 compares the ratio of the ATE interaction time to the total test time (T_{ATE}/T_{total}) for the conventional BIST scheme and the proposed Auto-CP scheme, across four test segments (Auto_0 to Auto_3) at 15 MHz working frequency. In the conventional BIST scheme, the ATE sends test vectors step by step, and the ATE overhead ratios are 27%, 21%, 21%, and 35% for Auto_0 to Auto_3, respectively, giving an average of 26%. For Auto_0, which contains the most test items and longest instruction sequence, the ATE overhead reaches 35%. In contrast, the proposed Auto-CP scheme reduces the ATE overhead ratios to 18%, 6%, 5%, and 23%, giving an average of 13%. The average ratio dropped significantly from 26% to 13%. This confirms that the one-time compressed program loading

and autonomous on-chip execution effectively eliminates the repeated step-by-step ATE interactions described in Section 2.

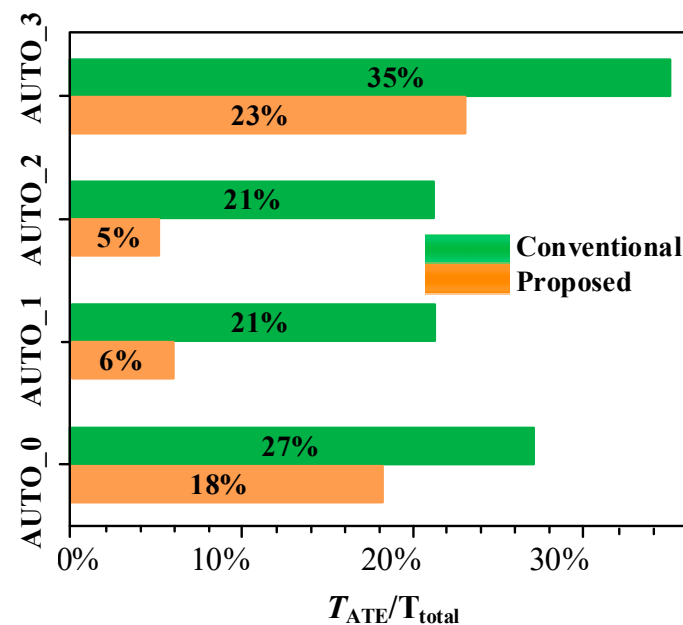


Figure 7. The ratio of the ATE interaction time to the total test time across the four segments.

Figure 8 presents the reduction in the total CP test time across the four segments. The segment-level reductions are 11%, 16%, 17%, and 16%, with a simple average of 15%. The time reduction ΔT is obtained by comparing the removed step-by-step ATE interaction overhead T_{STEP} with the added overhead $T_{\text{LOAD}} + T_{\text{onchip}}$. The measured total CP time is 42.65 s for the conventional BIST flow and 36.25 s for Auto-CP, giving an absolute reduction of $\Delta T = 6.4$ s. The results demonstrate that the proposed method achieves a net positive gain, confirming the validity of Equation (4). These results also have practical implications for the production cost and efficiency. For a 40 nm 1 MB embedded Flash, the CP test cost per 10,000 wafers is approximately 1,500,000 USD. Based on the 15% total time reduction (42.65 s $\rightarrow$ 36.25 s), the estimated saving is about 225,000 USD per 10,000 wafers.

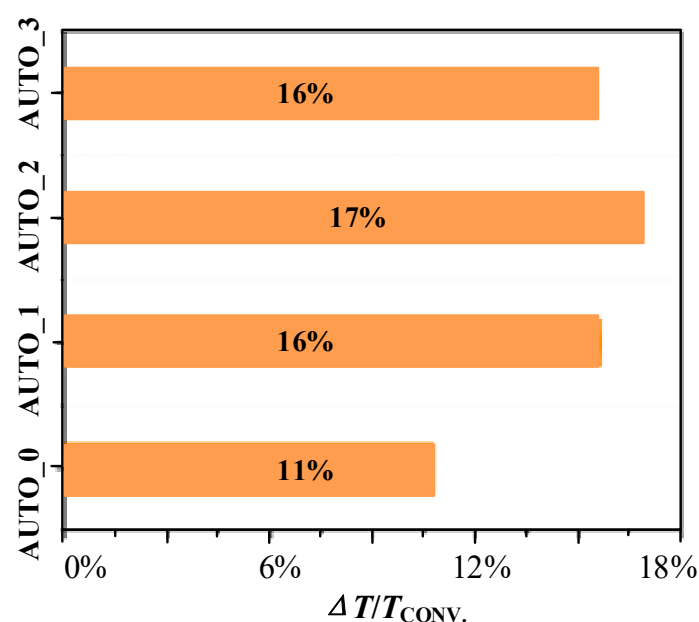


Figure 8. The reduction in total CP test time across the four segments. T_{conv} is defined as the total CP test time using the conventional BIST/ATE flow.

6. Conclusions

This paper presented an Auto-CP BIST method for embedded Flash CP testing in automotive MCUs, addressing the inefficiency of frequent ATE-BIST interactions in conventional test flows. The core innovation lies in an entropy-guided two-stage compression algorithm combined with on-chip autonomous scheduling, which fundamentally transforms the conventional step-by-step ATE-controlled test model into a one-time loading and autonomous execution model.

The proposed method comprises four key components: (1) information entropy-based redundancy analysis that quantifies the compression potential of the BIST instruction stream; (2) an instruction/sequence decomposition mechanism that separates reusable operation templates from flow control scheduling information; (3) Huffman lossless coding that further reduces the code length of high-frequency symbols; and (4) an on-chip Auto-CP controller that decodes and autonomously executes the compressed test program.

Experimental results on a 40 nm 1 MB embedded Flash demonstrated that the proposed method improves the total CP test time by 15%, and lowers the ATE overhead ratio from 26% to 13%. These results validate that the entropy-guided compression and on-chip autonomous scheduling approach effectively reduce the ATE interaction overhead. The proposed Auto-CP method provides a practical solution for improving the mass production CP test efficiency in automotive embedded Flash testing. Future work could extend this framework to multi-Flash test environments and explore adaptive compression strategies for different Flash technology nodes.

Future work can extend the proposed Auto-CP framework to a multi-Flash testing environment. In such an environment, each eFlash macro can be equipped with its own original eFlash BIST and Auto-CP controller. An Auto-CP BIST interconnector can be added to manage communication between the ATE and different Auto-CP controllers, while an SPI/JTAG controller handles communication between the ATE and the MCU BIST circuitry.

Author Contributions: Conceptualization, Q.L.; methodology, Q.L.; software, Q.L.; validation, Q.L.; formal analysis, Q.L.; investigation, Q.L.; resources, Q.L.; data curation, Q.L.; writing—original draft preparation, Q.L.; writing—review and editing, Q.L.; visualization, Q.L.; supervision, D.W.Z.; project administration, D.W.Z.; funding acquisition, D.W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Suzuki, J.; Yamashita, J.; Hanyu, M.; Ido, M.; Saito, T.; Nakashima, Y.; Hayashikoshi, M.; Kiyota, Y. A Cost-Effective High Accuracy Auto-Trimming System without Tester Constraint for Low-End Embedded Flash Memory. In Proceedings of the 2018 International SoC Design Conference (ISOCC), Daegu, Republic of Korea, 5–8 November 2018; pp. 48–49. [\[CrossRef\]](#)
2. Mittal, R.; Kawoosa, M.; Parekhji, R.A. Systematic Approach for Trim Test Time Optimization: Case Study on a Multi-Core RF SoC. In Proceedings of the 2014 International Test Conference, Seattle, WA, USA, 20–25 October 2014; pp. 1–9. [\[CrossRef\]](#)
3. Yu, X.; Li, B. Simulation SPI+FLASH System Based on FPGA. In Proceedings of the 2019 6th International Conference on Systems and Informatics (ICSAI), Shanghai, China, 2–4 November 2019; pp. 719–724. [\[CrossRef\]](#)
4. Shaila, C.K.; Manoj, G.; Divya, P.S.; Vijila, M. Functional Verification of SPI Protocol using UVM based on AMBA Architecture for Flash Memory Applications. In Proceedings of the 2023 4th International Conference on Signal Processing and Communication (ICSPPC), Coimbatore, India, 26–28 April 2023; pp. 311–315. [\[CrossRef\]](#)
5. Li, L.; Xiao, P.; Lu, Z.; Wang, Y.; Hu, Y.; Guan, Y.; Wang, K.; Li, D.; Zhang, F. Testing and Trimming Methods of High-Resolution and Large Swing for ADC Based on ATE. In Proceedings of the 2021 IEEE 14th International Conference on ASIC (ASICON), Kunming, China, 26–29 October 2021; pp. 1–4. [\[CrossRef\]](#)

6. Zhao, Z.; Cai, Z. A Review of Intelligent Design for Test Based on Machine Learning. In Proceedings of the 2023 International Symposium of Electronics Design Automation (ISED), Nanjing, China, 16–18 May 2023; pp. 116–120. [\[CrossRef\]](#)
7. Zhao, H.; Yang, F.; Shan, J. A Graph AutoEncoder Approach for Fault Prediction in Test Pattern Generation. In Proceedings of the 2024 2nd International Symposium of Electronics Design Automation (ISED), Xi'an, China, 10–12 May 2024; pp. 462–467. [\[CrossRef\]](#)
8. Zhang, G.; Yuan, Y.; Liang, F. Low Cost Test Pattern Generation in Scan-Based BIST Schemes. *Electronics* **2019**, *8*, 314. [\[CrossRef\]](#)
9. Vethamuthu Edward Alaises, R.S.; Sathasivam, S. Novel BIST Solution to Test the TSV Interconnects in 3D Stacked IC's. *Electronics* **2023**, *12*, 908. [\[CrossRef\]](#)
10. Maeda, Y.; Iwata, H.; Matsushima, J.; Wang, S.; Higami, Y.; Takahashi, H. Towards an ISO26262 Compliant DFT Architecture Enabling POST for Ultra-Large-Scale Automotive MCU. In Proceedings of the 2nd IEEE International Workshop on Automotive Reliability & Test, Fort Worth, TX, USA, 2–3 November 2017.
11. Wang, S.; Higami, Y.; Takahashi, H.; Iwata, H.; Maeda, Y.; Matsushima, J. Fault-detection-strengthened method to enable the POST for very-large automotive MCU in compliance with ISO26262. In Proceedings of the 2018 IEEE 23rd European Test Symposium (ETS), Bremen, Germany, 28–31 May 2018; pp. 1–2. [\[CrossRef\]](#)
12. Wu, W.-F.; Wu, K.-W.; Huang, H.-W. Reliability Analysis of a Vehicle Motor Control Unit in Consideration of Functional Safety of ISO 26262. *J. Chin. Inst. Eng.* **2026**, *49*, 816–833. [\[CrossRef\]](#)
13. Shivakumar, V.; Senthilpari, C.; Yusoff, Z. Test Power and Area Optimized Logic Built-In Self-Test with Higher Fault Coverage for Automobile SoCs. *Microelectron. J.* **2022**, *124*, 105430. [\[CrossRef\]](#)
14. Lee, J.; Jeong, Y.; Yeom, K.; Jeon, C.; Woo, J.; Lee, S.; Lee, G.Y.; Hwang, D.H.; Chung, Y.S.; Seo, M.; et al. Highly Reliable 28 nm Embedded Flash Process Development for High-Density and High-Speed Automotive Grade-1 Application. In Proceedings of the 2021 IEEE International Memory Workshop (IMW), Dresden, Germany, 16–19 May 2021; pp. 1–3.
15. Bernardi, P.; Insinga, G.; Paganini, G.; Cantoro, R.; Beer, P.; Coppetta, M.; Mautone, N.; Carnevale, G.; Scaramuzza, P.; Ullmann, R. Optimized diagnostic strategy for embedded memories of Automotive Systems-on-Chip. In Proceedings of the IEEE European Test Symposium (ETS), Barcelona, Spain, 23–27 May 2022; pp. 1–6.
16. Angione, F.; Bernardi, P.; Filippini, G.; Tempesta, C.; Reorda, M.S.; Appello, D.; Tancorre, V.; Ugioli, R. Online scheduling of concurrent Memory BISTs execution at Real-Time Operating-System level. In Proceedings of the 2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Austin, TX, USA, 19–21 October 2022; pp. 1–6. [\[CrossRef\]](#)
17. Nti Darko, E.; Bruce, I.; Oko-Odion, E.; Bhatheja, K.; Karimpour, S.; Chen, D. Low-Cost Direct Digital Synthesis-Based On-Chip Waveform Generation for Analog/Mixed Signal BIST Applications. *Metrology* **2025**, *5*, 8. [\[CrossRef\]](#)
18. Liu, Q.; Zhang, D.W. Auto-Trimming-Based Designs for the Test Optimization of Mass-Produced Automotive Microcontroller Unit Chip Probing Tests. *Electronics* **2024**, *13*, 4894. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.