ABSTRACT

Increasing embedded systems functionality causes a steep increase in code size. For instance, more than 60MB of software is installed in current state-of-the-art cars. It is often challenging and cumbersome to host vast amount of software in an efficient way within a given hardware resource budget of an embedded system. This may be done by using code compression techniques, which compress the program code off-line (i.e. at design time) and decompress it on-line (i.e. at run time). The overview of a traditional test compression framework is shown in Fig. 1. The original test data is compressed and stored in the memory. Thus, the memory size is significantly reduced. An on-chip decoder decodes the compressed test data from the memory and delivers the original uncompressed set of test vectors to the design-under-test (DUT). The major contributions of this paper are as follows: 1) it develops an efficient bitmask selection technique for test data in order to create maximum matching patterns; 2) it develops an efficient dictionary selection method which takes into account the bitmask based compression; and 3) it proposes a test compression technique using efficient dictionary and bitmask selection to significantly reduce the testing time and memory requirements.

Key words: SOC, BIST, ILS, ATE, LAT, DCE, DUT, Compression, Dictionary
1. INTRODUCTION

Embedded systems are everywhere from household appliances to biomedical, military, geological and space equipments. The complexity of such systems is increasing at an exponential rate due to both advancements in technology and demands for sophisticated applications, in the areas of communication, multimedia, networking and entertainment. Memory is one of the key driving factors in embedded system design since a larger memory indicates an increased chip area and a higher cost. As a result, memory imposes constraints on the size of the application programs. Test Data compression techniques address the problem by reducing the program size.

IN SYSTEM-ON-CHIP (SOC) designs, higher circuit densities have led to larger volume of test data, which demands larger memory requirement in addition to an increased testing time. Test data compression plays a crucial role, reducing the testing time and memory requirements. It also overcomes the automatic test equipment (ATE) bandwidth limitation. Alternatives to external testing include built-in self-test (BIST). However, BIST is not appropriate for logic testing because of its random-resistant fault and bus contention during test application, which leads to inadequate test coverage. Other alternatives like bit-flipping and bit fixing provide greater fault coverage, with the disadvantage that structural information has to be provided. Reduction of test data using structural methods like Illinois Scan Architecture (ILS) demands modification of the design. Test data compression algorithms can reduce the test data to a larger degree without facing any of the aforementioned disadvantages.

![Test Data Compression Methodology](image)

The overview of a traditional test compression framework is shown in Fig. 1.1 The original test data is compressed and stored in the memory. Thus, the memory size is significantly reduced. An on-chip decoder decodes the compressed test data from the memory and delivers the original uncompressed set of test vectors to the design-under-test (DUT). Compression ratio, widely accepted as a primary metric for measuring the efficiency of test data compression, is defined as:

\[
\text{Compression Ratio} = \frac{\text{Compressed Program Size}}{\text{Original Program Size}}
\]

Dictionary-based test data compression is a promising approach. Dictionary-based compression techniques are also popular in embedded systems domain since they provide a dual advantage of good compression efficiency as well as fast decompression mechanism.

Many recently proposed techniques have tried to improve the dictionary-based compression techniques by considering mismatches. However, the efficiency of these techniques depends on the number of bits allowed to mismatch. It is obvious that if more number of bit changes is allowed, more matching patterns will be generated. However, remembering more bit Positions may lead to unprofitable compression. Bitmap-based compression addresses this issue by creating more
matching patterns with the aid of bitmasks, while taking care of the size of the compressed code. This paper tries to combine the advantages of dictionary based test compression as well as bitmask-based test data compression.

Bitmask based compression of test data may seem attractive, but it presents various challenges. The primary concern is the presence of don’t cares (‘X’”) in the test data set. Since bitmask based compression technique was not designed for data with don’t care values, direct application of this technique on test data does not result in a good compression efficiency. We have to determine not only the effective bitmasks, but also a profitable dictionary that produces optimal results. We demonstrate in Section III that selection of bitmasks and dictionary using existing techniques are not appropriate in case of test data compression using bitmasks. Our approach solves these problems by selecting profitable bitmasks as well as proposing efficient dictionary selection algorithms, to improve the compression efficiency without introducing any additional decompression penalty.

2. DICTIONARY-BASED COMPRESSION

This chapter describes existing dictionary-based approaches and analyzes their limitations. First, a standard dictionary-based approach is discussed. Next, we describe a recently proposed technique that improves the standard approach by considering mismatches (hamming distance). Finally, we present a detailed cost-benefit analysis of the recent approaches in terms of how much repeating patterns they can generate from the mismatches. This analysis forms the basis of my thesis work to maximize the repeating patterns using bitmasks.

2.1 Dictionary-based Approach

Dictionary-based test data compression techniques provide compression efficiency as well as fast decompression mechanism. The basic idea is to take the advantage of commonly occurring instruction sequences by using a dictionary. The repeating occurrences are replaced by a word that points to the index of the dictionary that contains the pattern. Figure 2–1 shows an example of dictionary based test data compression using a simple program binary. The binary consists of ten 8-bit patterns i.e., total 80 bits. The dictionary has two 8-bit entries. The compressed program requires 62 bits and the dictionary requires 16 bits. In this case, the compression ratio is 97.5% (using Equation (1.1)). This example shows a variable length encoding. As a result, there are many factors that also need to be included in the computation of the compression ratio including the size of the Line Address Table (LAT) as well as byte alignment for branch targets.

![Fig 2.1 Dictionary Based Compression Approach](Image)

**Fig 2.1 Dictionary Based Compression Approach**
2.2 Improved Dictionary-based Approach

Recently proposed techniques improve the dictionary based compression technique by considering mismatches. The basic idea is to determine the instruction sequences that are different in few bit positions (hamming distance) and store that information in the compressed program and update the dictionary (if necessary). The compression ratio will depend on how many bit changes are considered during compression. Figure 2–2 shows the encoding format used by these techniques for 32-bit program code.

![Encoding Scheme for Incorporating Mismatches](image)

Figure 2.2: Encoding Scheme for Incorporating Mismatches

It is obvious that if more bit changes are allowed, more matching sequences will be generated. However, the size of the compressed program will increase depending on the number of bit positions. Section 2.3 describes this topic in detail. Prakash et al considered only one-bit change for 16-bit patterns (vectors). Ros et al. considered a general scheme of up to 7 bit changes for 32-bit patterns and concluded that a 3-bit change provides best compression ratio.

![Improved Dictionary Based Compression Technique](image)

Fig 2.3: Improved Dictionary Based Compression Technique

Figure 2–3 shows the improved dictionary based scheme using the same example (shown in Figure 2–1). This example considers only 1-bit change. An extra field is necessary to indicate whether mismatches are considered or not. In case a mismatch is considered, another field is necessary to indicate the bit position that is different from an entry in the dictionary. For example, the third pattern (from top) in the original program is different from the first dictionary entry (index 0) on the sixth bit position (from left). The compression ratio for this example is 95%.

3. TEST DATA COMPRESSION USING BIT-MASKS

The proposed approach tries to incorporate maximum bit changes using mask patterns without adding significant cost (extra bits) such that the compression ratio is improved. Our compression technique also ensures that the decompression efficiency is improved or remains the same compared to the existing techniques.
Figure 3–1: Encoding Format for Our Compression Technique

Figure 3–1 shows the generic encoding scheme used by the compression technique. This scheme is similar to the 32-bit format shown in Figure 2–2 where individual bit changes are recorded. However, as described in Section 2.3, storing individual bit changes limits the number of matches. Our encoding format (Figure 3–1) can store information regarding multiple mask patterns. For each pattern, it stores the mask type, the location where mask needs to be applied, and the mask pattern. The generic encoding scheme can be optimized once the types and sizes of the bitmask combinations are determined.

Test Data compression using bit-masks is a promising approach. However, it introduces two major challenges. First, how to select a set of mask-patterns that will maximize the matching sequences while minimize the cost of using bit-masks? Second, how to perform efficient dictionary selection using the selected bit-masks? We have developed efficient mask selection and dictionary selection technique to improve compression ratio with introducing any decompression penalty. This chapter is organized as follows. Section 3.1 describes our mask selection procedure. Section 3.2 outlines our dictionary selection technique. Section 3.3 presents our test data compression algorithm using the mask selection and the dictionary selection procedures. Finally, Section 3.4 describes the decompression framework to support our bit-mask based test data compression technique.

3.1. Decompression Technique

The design of a decompression engine (DCE), shown in Figure3.3, that can easily handle bitmasks and provide fast decompression. The design of our engine is based on the one cycle decompression engine proposed by Seong et al. The most important feature is the introduction of XOR gate in addition to the decompression scheme for dictionary based compression. The decompression engine generates a test data length bitmask, which is then XORRed with the dictionary entry. The test data length bitmask is created by applying the bitmask on the specified position in the encoding.

Figure 3.3: Decompression engine for bitmask-based encoding
The generation of bitmask is done in parallel with dictionary access, thus reducing additional penalty. The DCE can decode more than one compressed data in one cycle. Algorithm 2 provides an overview of our decompression procedure. The decompression engine takes the compressed vector as input. It checks the first bit to see whether the data is compressed. If the first bit is “1” (implies uncompressed), it directly sends the uncompressed data to the output buffer. On the other hand, if the first bit is a “0”, it implies this is a compressed data. Now, there are two possibilities in this scenario. The data may be compressed directly using dictionary entry or may have use bitmasks. The decompression engine will operate differently in these two cases.

4. SIMULATION RESULTS

4.1. Compression

This module is used to integrate all the modules. This is the top module for compression. Test vectors and bit masking type are given to this module and we will get dictionary table and compressed data as outputs.

![Block Diagram of Compression](image1)

Figure 4.11: Block Diagram of Compression

![Waveform of Compression](image2)

Figure 4.12: Waveform of Compression
4.2. Decompression

This module is used to decompress the data. Compressed data is decompressed using dictionary table and we will get original uncompressed data.

![Block Diagram of Decompression](image1)

![Waveform of Decompression](image2)

5. CONCLUSIONS

Embedded systems are constrained by the memory size. Test Data compression techniques address this problem by reducing the code size of the application programs. Dictionary-based code compression techniques are very popular since they generate good compression by exploiting repeating patterns. Recent techniques uses bit toggle information to create matching patterns and thereby improve compression ratio. However, due to lack of efficiency in matching scheme, the existing techniques can match up to three bit differences.
In the thesis, I presented an efficient test data compression scheme using bit-masks that can significantly increase the number of matching patterns. This approach can handle multiple bit mismatches without incurring any compression or decompression penalty. Moreover, I examined and presented the approaches for the two challenges that were introduced by the proposed technique: the choice of the mask combination and the dictionary entry selection. I applied the technique using applications from various domains and compiled them for different architectures to demonstrate the usefulness of the approach. The experimental results show that the proposed technique reduces the original program size up to 45%. This technique outperforms all the existing dictionary based techniques by an average of 15%, giving compression ratios of 55%-65%. It also enables parallel decompression that is suitable for VLIW processors.

REFERENCES

ABOUT THE AUTHORS

AJMAL SADIQ. MAHAMMAD, Asst. Professor

Received B.Tech degree in Electronics and Communication Engineering from the University of JNTU and M.E degree in Digital Systems from the Osmania University, Hyderabad. He is currently working as an Asst. Professor in ECE Department at RVR Institute of Engineering & Technology, Hyderabad. Up to now he was attended several National and International Conferences, Workshops. His currently research interests include VLSI Design, Image Processing, Antennas &Microwave, Digital Signal Processing.

T. NAGA RAJU, Asst. Professor

Received B.Tech degree in Electronics and Communication Engineering from the University of JNTU Hyderabad and M.Tech degree in VLSI System Design from the JNTU-Hyderabad. He is currently working as an Asst. Professor in ECE Department at RVR Institute of Engineering & Technology, Hyderabad. Up to now he was attended several National and International Conferences, Workshops. His currently research interests include VLSI Design, Image Processing, Antennas &Microwave, Digital Signal Processing.

KUMAR.KESHAMONI, Asst. Professor

Kumar. Keshamoni obtained his B.Tech in Electronics & Communication Engineering from the University of JNTU, his M.Tech in VLSI Design from the University of JNTU-Hyderabad. He is currently the Assistant Professor in Department of Electronics & Communication Engineering, RVR Institute of Engineering in University of the JNTU-Hyderabad. Kumar. Keshamoni is Chair of World Committees Associate Member of UACEE (Universal Association of Computer & Electronics Engineers), Member of IAENG (International Association of Engineers) and an Advisory Board of WATT (World Association of Technology Teachers). He is also on the Advisory & Editorial Board of IJETAE, an Associate Editor of IJPRET, Editor of IJOART, Editorial Board & Review Committee of IJSER, an Editorial & Review Board of IJTRA, Editor, writer & Peer Reviewer of edition International Publisher, Peer Reviewer of AJET, Review Committee of IJERT, Editorial & Review Board of IJOAR, Review Board of IJASCSE, Advisory& Editorial Board of IURDET, Member of IAENG, Writer& Review Board of IJEVS, Advisory Board of WRJET, Scientific Committee & Editorial Review Board on Engineering and Physical Sciences of WASET and Editorial Board of IAASE. He is also working as a member for 5 International conference committees. He has published about 13 research articles in International Journals and 5 papers in National and International Conferences and attended several National, International Conferences, and Workshops & Faculty Development Programs in different organizations. His recent research interests lie in the area of Embedded, VLSI Design, Digital Signal Processing and Nanotechnology.