X-Tracer: A Reconfigurable X-Tolerant Trace Compressor for Silicon Debug

Feng Yuan[†‡], Xiao Liu[†] and Qiang Xu[†‡]
[†]CUhk REliable Computing Laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[‡]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Email: {fyuan,xliu,qxu}@cse.cuhk.edu.hk

ABSTRACT

The effectiveness of at-speed silicon debug is constrained by the limited trace buffer size and/or trace port bandwidth, requiring highly-efficient trace data compression solutions. As it is usually inevitable to have unknown 'X' values during silicon debug, trace compressor should be equipped with X-tolerance feature in order not to significantly degrade error detection capability. To tackle this problem, this paper presents a novel reconfigurable X-tolerant trace compressor, namely X-Tracer, which is able to tolerate as many X-bits as possible in the trace streams while guaranteeing high compression ratio, at the cost of little extra design-for-debug hardware. Experimental results on benchmark circuits demonstrate the effectiveness of the proposed technique.

Categories and Subject Descriptors

*B.7.3 [**Integrated Circuits**]: Reliability and Testing*

General Terms

Design, Verification, Reliability, Algorithm.

Keywords

Silicon Debug, Trace Data Compression, X-Tolerance

1. INTRODUCTION

The ever-increasing design complexity of integrated circuits (ICs) and the inherent inaccuracy of circuit models at high abstraction levels significantly challenge the effectiveness of pre-silicon verification techniques, and it is not uncommon that IC products need to go through multiple re-spins to be error-free [1], despite the fact that more than half of the resources are devoted to verification tasks [2]. Consequently, to reduce expensive re-spins and time-to-market, silicon debug (also known as post-silicon validation) cannot be an afterthought and has become an essential step in today's IC design flow.

1.1 Related Work

Since the circuit under debug (CUD) is a piece of silicon that has already been fabricated, the main challenge in silicon debug is the limited visibility of internal signals. To tackle this problem, usually dedicated design-for-debug (DfD) circuitries are added to the design to improve its observability.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2012, June 3-7, 2012, San Francisco, California, USA.
Copyright 2012 ACM 978-1-4503-1199-1/12/06 ...$10.00.

Trace-based debug [3] that allows designers to real-time observe a set of signals in consecutive cycles, being non-intrusive to the circuit's normal operation, is one of the most effective silicon debug techniques and has been widely adopted by the industry [4, 5]. To be specific, in this technique, a set of "key" signals in the CUD are tapped and they can be traced after being triggered. The sampled data are then sent to internal trace buffers and/or external trace ports via trace interconnection fabric [6], for later analysis by debug software and physical probing tools to further root cause and fix the bug (e.g., [7, 8, 9]).

Once a bug is activated, it leaves its erroneous effects in one or more state elements of the circuit at some cycles. The objective of trace-based debug is to observe and localize such errors with as few debug runs as possible. Since it is not possible for us to trace all internal signals in the circuit, on one hand, the effectiveness of trace-based debug depends on the quality of the selected trace signals, which may include both manually-picked signals by experienced designers and signals selected via automated solutions guided by some visibility-enhancement metrics (e.g., [10, 11, 12, 13]). On the other hand, even with pre-determined trace signals that can capture a bug, it will only manifest itself at some specific time and it is crucial to ensure the signals at the "right" time are indeed traced.

Clearly, the more trace data that we can acquire, the higher possibility for us to catch a bug's erroneous effects in them and the less time and effort to identify the bug. Unfortunately, what we can trace in each debug run is usually quite limited. This is because, trace-based debug involves non-trivial overhead and we are only given limited trace buffer size and/or few external pins as trace ports.

Because of the above, it is not quite economical to store the"raw" trace data. In [14], Park and Mitra compressed the execution states of microprocessor into a small amount of *footprints*, taking advantage of the fact that the locality feature of instruction sequence and redundant information in monitored data that can be easily identified with the executed instructions. [15, 16] utilized the data locality feature when accessing cache and adopted dictionary-based compression to improve the compression ratio.

The above trace compression solutions focused on debugging microprocessors. Several compression techniques have also been presented for signal tracing in general logic circuits to improve their error detection capability, and they can be broadly classified into the following three types:

- *lossless trace compressors*, which take advantage of the locality of trace data for lossless compression. In [17], Anis and Nicolici presented several dictionary-based compressors to trace repeatable data. Based on the observation that toggling rate of state values is usually low, Prabhakar *et al.* [18] proposed to compress the differential data to achieve higher compression quality.

- *spatial lossy trace compressors*, which compacts a set of N signals into M parity signals ($N > M$) with XOR network before signal tracing starts [19]. To reduce routing overhead,

such spatial compressors are usually organized as a tree-like structure as part of the trace interconnection fabric.

- *temporal lossy trace compressors*, which compacts a number of cycles (e.g., 1k) of the raw data into a signature during signal tracing [20, 21], with the help of multiple-input signature register (MISR), originally used for test response compaction in VLSI testing domain. As shown in Fig. 1, with the assumption that the CUD behaves repeatable in different debug iterations, [20] consecutively zooms-in the failure signatures by reconfiguring the compaction ratio in their MISR-based compressor for each debug run to localize the error.

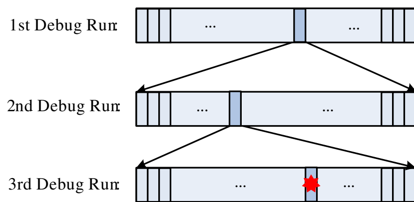


Figure 1: Iterative Debug Flow in [16].

In this work, we focus on the temporal trace compressor design, which provides significantly higher compression ratio when compared to the other types of compressors. Please note, at the same time, the above three trace compression methodologies are not contradictory, and in fact, they can be combined together to further enhance the real-time observability of the CUD.

1.2 Motivation and Summary of Contributions

From the above, it is clear that temporal lossy trace compressors are quite appealing due to their impressive compression ratio. However, the effectiveness of such MISR-based compressors relies on the existence of *clean* "golden vector" to generate reference signatures for comparison. This is usually not the case during silicon debug, rendering the lossy compression technique less effective on error detection. This is because: (i). it is often too time-consuming to run gate-level simulation for failed silicon test,and hence designers often resort to high-level simulator to generate "golden vectors" and many unknown bits (X-bits) are obtained when they are mapped onto gate-level vectors; (ii). asynchronous clock domains and uninitialized state elements also result in many X-bits in functional patterns.

In test response compaction techniques, X-bits will also corrupt the signature and hence result in fault coverage loss. Various techniques were presented to tackle this problem in the VLSI testing domain. X-masking hardware can be introduced to mask X-bits before they are fed into response compactor [22]. X-tolerant techniques are able to inherently tolerate X-bits by connecting each compactor input with redundant channels [23]. X-canceling solution identifies linear dependency of the signature with Gauss-Jordan elimination and cancels X-bits by XORing relevant signatures together [24].

The above techniques, however, cannot be effectively reused to tolerate X-bits in trace-based silicon debug. This is because: (i). signal tracing is conducted in functional mode and hence those solutions that rely on blocking X-bits in test mode cannot be used; (ii). unlike test patterns that are generated before the circuit is fabricated, the vectors used in silicon debug cannot be pre-calculated and hence we do not know the exact distribution of X-bits in advance; (iii). different from manufacturing test that only needs to guarantee fault detection capability, in silicon debug we are required to obtain as much information as possible to root-cause the bug.

Motivated by the above, in this paper, we propose a novel reconfigurable X-tolerant trace compressor, namely *X-Tracer*, which is able to tolerate as many X-bits as possible in the trace streams while guaranteeing high compression ratio, at the cost of little extra design-for-debug hardware. The major contributions of this work include:

- We propose a novel reconfigurable MISR-based trace compressor with redundancy that is able to effectively tolerate X-bits in trace-based debug;
- We develop a trace data extraction algorithm that is able to extract as much useful trace information as possible;

The remainder of this paper is organized as follows. Section 2 presents the proposed X-tracer design. The trace data extraction algorithm is then detailed in Section 3. Next, experimental results are shown in Section 4. Finally, Section 5 concludes this work.

2. PROPOSED X-TRACER DESIGN

The proposed X-Tracer design is constructed in two steps. First, we add redundancy into conventional MISR structure to equip it with dedicated X-tolerance feature. Then, we introduce reconfigurability into the compressor to further improve its X-tolerance capability. The details are given in the following.

2.1 MISR-Based X-Tolerant Trace Compressor with Explicit Redundancy

Conventional MISR design is typically constructed with a primitive polynomial feedback loop. Fig. 2(a) and Fig. 2(b) show two example MISR circuits with different polynomials. In this example, input data are compressed for five cycles and the corresponding MISR outputs are represented as linear combinations of inputs.

Clearly, with the above architecture, any unknown input bit will render the MISR output that utilize it to be unknown as well. However, as an input bit may present itself on multiple outputs, it is possible to cancel some X-bits with the technique shown in [25]. For example, given the MISR circuit in Fig. 2 (a), if only I_{03} appears to be an X-bit in the compression trace window, its effect can be canceled by XORing O_7 with O_5 (with external software support, detailed in Section 3), thus enabling us to preserve some of the traced information, i.e., I_{01}, I_{10}, I_{14}, I_{32} and I_{34} in this example.

The above X-canceling possibility is due to the implicit redundancy for certain trace bits, which, however, is not guaranteed. Let us consider the MISR circuit shown in Fig. 2 (a) again.

- It is inevitable that some information bits are only delivered to a single output bit (e.g., I_{00}). Consequently, if such bits are unknown, they will contaminate all the other information bits located in the same output (e.g., O_6) and results in significant trace information loss;
- Even for those information bits that are delivered to multiple outputs, the by-product of X-canceling operation may result in other trace information loss. For example, consider again the case of $I_{03} = X$ in O_7 and O_5, while I_{01}, I_{10}, I_{14}, I_{32} and I_{34} are reserved, the trace bits I_{12}, I_{21} and I_{30} are lost since they are also canceled being appeared in both O_7 and O_5 and at the same time cannot be recovered from other MISR outputs;
- Large number of X-bits in trace data will make it fail to find any X-canceling combination. As for this example, when I_{00}, I_{02}, I_{03} and I_{23} are all X-bits, there will be no uncontaminated information bits left by conducting X-canceling.

To overcome the above limitations, we propose to construct a MISR-based trace compressor with explicit redundancy, by combining multiple MISRs with different primitive polynomials and input mapping orders. An example is shown in Fig. 2, wherein we simply combine the two MISR circuits together to construct our X-tolerant trace compressor. With this design, all the traced bits will be included in at least two outputs. Then, when I_{03} is unknown, previously-lost information bits I_{12}, I_{21} and I_{30} can be recovered by $O_5 \oplus O_3$ and O_0. Even when we have many X-bits as the earlier example, i.e., when I_{00}, I_{02}, I_{03} and I_{23} are all X-bits, there are choices to cancel their effects with combinations such as $O_7 \oplus O_3 \oplus O_2$, $O_6 \oplus O_3 \oplus O_1$.

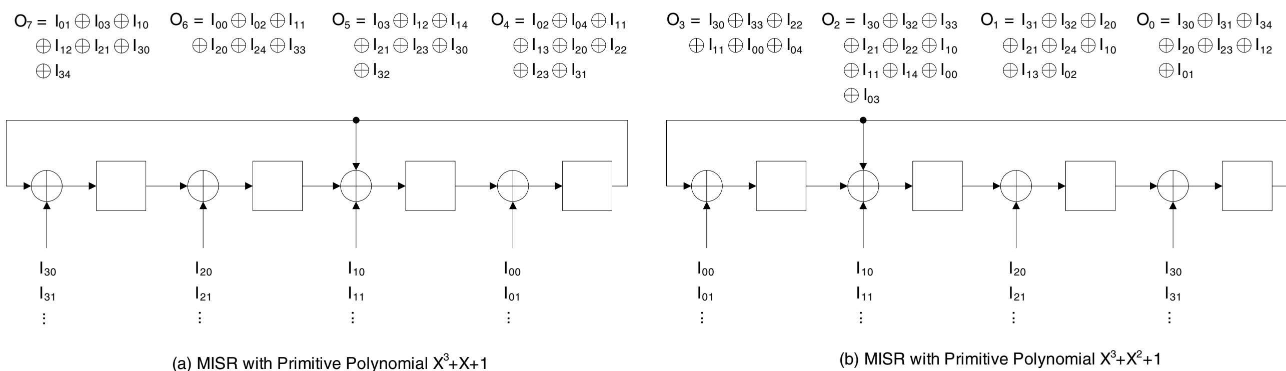


Figure 2: Example MISR-Based Trace Compressors.

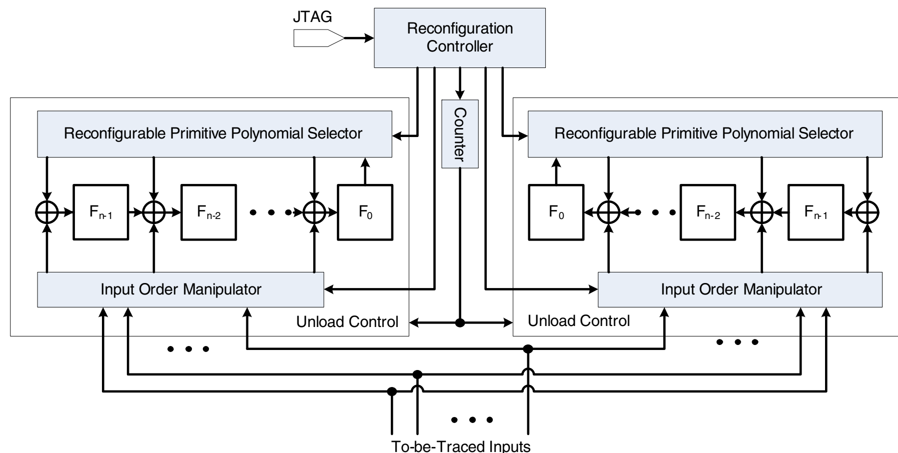


Figure 3: Reconfigurable X Tolerant Trace Compressor Architecture.

We attribute the benefits provided by our X-tolerant trace compressor structure to the following reasons:

1. The increased number of observing points for each and every traced bits provide more information redundancy and hence higher possibility to cancel X-bits.

2. As demonstrated by the symbolic expression of each MISR output (see Fig. 2), MISR with distinct primitive polynomial results in unique information bit distribution at the corresponding observing outputs. By combining MISRs with diversified information bit distributions, the proposed trace compressor thus provides additional opportunities to tolerate X-bits and reduce the possibility of useful information loss. For example, if two identical MISRs shown in Fig. 2(a) are used to compose the trace compressor, there will be only two observing outputs for I_{00} and it may cause X-canceling difficulty and information loss if it is an X-bit. Replacing one of them with the MISR shown in Fig. 2(b), however, gives us three observing outputs for I_{00} all together.

3. Distinct input orders in redundant MISRs facilitate to prevent some fixed combination of information bits. In the example design, $I_{01} \oplus I_{10}$ would be appearing in both MISRs if their input orders are kept the same, thus leading to information loss if one of them is unknown.

It is important to note that, while the proposed X-tolerant trace compressor design involves little area overhead, it does lead to significant compression ratio loss. With two MISRs to compose the proposed X-tolerant trace compressor, the compression ratio is only half of that of the single MISR design with the same trace buffer size. This effect has been taken into consideration for fair comparison in our experimental results shown in Section 4.

2.2 Reconfigurable Trace Compressor Design

As discussed earlier, while more redundancy is helpful to recover more trace information, the compression ratio is reduced and it may also involve high control complexity. Consequently, in our trace compressor design, we keep the redundancy ratio to be two. In order to further enhance the capability of the trace compressor, we introduce reconfigurability into our trace compressor design. By doing so, we are able to flexibly change the compressor's structure for each debug run, which enables us to extract more trace information.

Fig. 3 presents the overall structure of the proposed X-tracer design, which can be configured externally via JTAG interface. Three configuration options are provided to debug engineers. Firstly, the primitive polynomial can be reconfigured for each MISR, and it is implemented by selectively turning on/off the feedback loop from each output. Secondly, the module *Input Order Manipulator* is utilized to change MISR's input order, also individually-controllable. Finally, we introduce an internal *counter* to determine the cycle number to unload the MISR signatures. Adjustment of the counter value enables us to tradeoff compression ratio and X-tolerant capability, similar to [20]. Finally, our proposed trace compressor can be easily reconfigured to simply use a single MISR for trace compression as well, when there is nearly no X-bits and hence redundant tracing is not required.

3. TRACE INFORMATION EXTRACTION

After acquiring the X-contaminated signatures from the trace buffer in each debug run, we rely on an off-line processing step to extract as many useful trace bits as possible.

Given the X-contaminated signatures, we can construct the corresponding X-matrix and employ the X-canceling solution in [25] to extract useful trace data. An example X-matrix is shown in the following example, wherein each row corresponds to a MISR observing bit and each column represents a specific X-bit (entry '1' denotes that the corresponding X-bit contaminates the specific ob-

serving bit). Next, Gauss-Jordan elimination is utilized to identify all-zero rows, which represent X-canceling combinations. Assuming I_{00}, I_{02}, I_{03} and I_{23} in Fig. 2 are X-bits, one possible solution with [25] is shown as follows:

$$\begin{pmatrix} & I_{00} & I_{02} & I_{03} & I_{23} \\ O_7 & 0 & 0 & 1 & 0 \\ O_6 & 1 & 1 & 0 & 0 \\ O_5 & 0 & 0 & 1 & 1 \\ O_4 & 0 & 1 & 0 & 1 \\ O_3 & 1 & 0 & 0 & 0 \\ O_2 & 1 & 0 & 1 & 0 \\ O_1 & 0 & 1 & 0 & 0 \\ O_0 & 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} & I_{00} & I_{02} & I_{03} & I_{23} \\ O_3 & 1 & 0 & 0 & 0 \\ O_6+O_3 & 0 & 1 & 0 & 0 \\ O_7 & 0 & 0 & 1 & 0 \\ O_6+O_4+O_3 & 0 & 0 & 0 & 1 \\ O_7+O_6+O_5+O_4+O_3 & 0 & 0 & 0 & 0 \\ O_7+O_3+O_2 & 0 & 0 & 0 & 0 \\ O_6+O_3+O_1 & 0 & 0 & 0 & 0 \\ O_6+O_4+O_3+O_0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (1)$$

As a test response compaction technique, the objective of [25] is to find a solution that X-bits do not contaminate the bits used to fault detection. In silicon debug, however, our objective is to extract as many useful trace bits as possible. Consequently, we cannot directly use [25] to generate our solution.

Since different solutions lead to very different X-canceling combinations, the corresponding extracted trace data may vary significantly. We can in fact employ multiple solutions to extract more trace information for silicon debug. This, however, is a challenging problem because the number of solutions for an X-matrix can be huge (even with the rather constrained Gauss-Jordan elimination procedure). In this section, we propose novel techniques that are able to effectively explore X-canceling combination space and we present an algorithm to maximize the number of extracted trace bits.

3.1 X-Canceling Solution Space Exploration

Again, let us take the example shown in Eq. 1 to illustrate our proposed technique for X-canceling solution space exploration.

We first select a targeted MISR observing bit (e.g., O_6), and move the corresponding row down to the last position. By doing so, we aim at finding the combination of remaining observing bits to cancel the unknown targeted bit. Then, instead of conducting row operations, we perform column operations to reduce the modified X-matrix to *column echelon form* as follows:

$$\begin{pmatrix} O_7 & 0 & 0 & 1 & 0 \\ O_5 & 0 & 0 & 1 & 1 \\ O_4 & 0 & 1 & 0 & 1 \\ O_3 & 1 & 0 & 0 & 0 \\ O_2 & 1 & 0 & 1 & 0 \\ O_1 & 0 & 1 & 0 & 0 \\ O_0 & 0 & 0 & 0 & 1 \\ \hline O_6 & 1 & 1 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 \end{pmatrix} \quad (2)$$

With the *column echelon form* of the X-matrix, the first non-zero entry is called a pivot (in bold), and its corresponding row is *pivot row*, which is guaranteed to contain only one non-zero entry. In addition, we define all-zero row as the *free rows*, and the other rows as *stack rows*. According to linear algebra, if the last row (representing the targeted observing bit) is not a pivot row, there must exist at least one combination of the remaining bits to cancel the unknown targeted bit. For the sake of easy explanation, we use a vector S to denote a X-canceling combination, where 1 in S means that the corresponding observing bit is involved in the combination. Based on the above definition for the reduced X-matrix, we define the corresponding bits in the S as *pivot bits*, *stack bits* and *free bits*. Therefore, an initial solution S_{init} can be found in the following manner: we first identify non-zero entries on the last row of the reduced X-matrix; then find the pivots on the same column; finally, we fill the related pivot bits in S_{init} with 1s, and left the other as 0s. For the example in Eq. 2, the initial solution could be $S_{init} = \{0,1,1,1,0,0,0,1\}$.

Next, starting from the initial solution, we try to traverse the solution space and generate new solutions by transforming existing solutions. To guarantee the obtained solution is legal, the transformation obeys three well-defined bit flipping rules: (i). free bits can be freely flipped to generate a new valid solution; (ii). to flip stack bit, we need to further flip all the pivot bits whose corresponding pivots are on the same columns of non-zero entries of stack row correlated with to-be-flipped stack bit. For example, if we flip the fifth bit in S_{init}, the related stack row is $\{1,0,0,1\}^T$ and thus the first and forth pivots bit need be flipped at the same time. In this case, the new solution could be $S_{sec} = \{1,1,1,0,1,0,0,1\}$. (iii). the pivot bits and the last bit in each row cannot be flipped. With the above solution space exploration procedure, the X-canceling combinations can be acquired by simply changing different targeted observing bits.

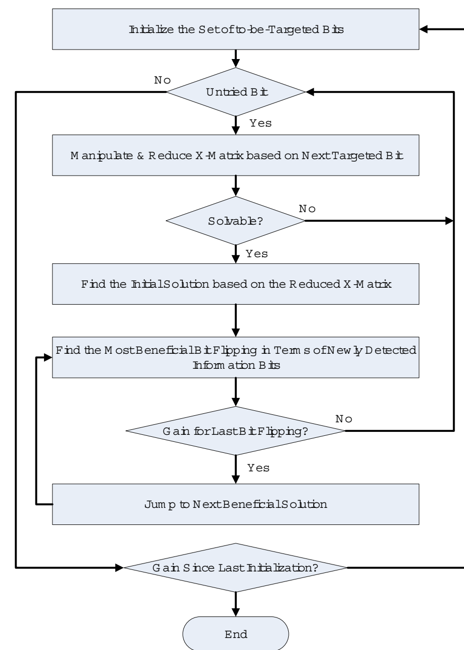


Figure 4: X-Free Information Extraction Algorithm.

3.2 X-Free Information Extraction

With the flexibly to explore the X-canceling combination solution space, our objective is to extract the maximum number of useful trace data (i.e., those X-free bits that are known).

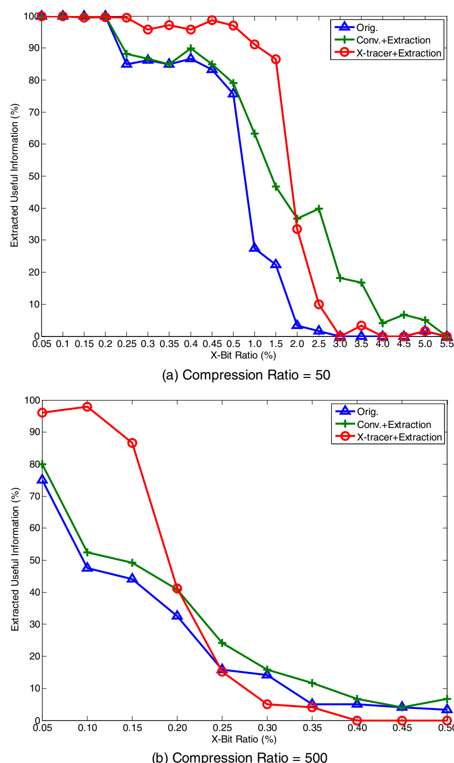
Our proposed algorithm is depicted in Fig. 4. It starts from putting all the observing outputs into a to-be-targeted bit set, from which we select one output as the targeted bit a time. Based on the original X-matrix, we first move the targeted bit column to the last position, and then perform row operations to reduce the matrix to *row echelon form*. Next, we try to find the next targeted bit if the reduced matrix is solvable, otherwise we try to restore the initial solution. Then, we search the solution space by greedily flipping the most beneficial bit, in which the *gain* is defined as the increased number of extracted information bits, and the search procedure is stopped when there is no more *gain*. After all the targeted bits have been tried, we check whether these is still *gain* since the last initialization. The whole procedure is re-initialized if it is not zero, otherwise it is terminated.

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

To evaluate the performance of our proposed X-tracer solution, in our experiments, we perform simulation studies under various X-bit distribution in the "raw" trace data, with X-bit ratio ranging from 0.05% to 5.5% injected randomly[1]. The trace buffer size is fixed as $32 \times 1k$ in our experiments, i.e., 32-bits are fed to the temporal compressor in every cycle and the number of trace entries is $1k$. Experiments are conducted under various compression ratio, ranging from 50 to 500. Since the proposed X-tracer architecture results in

[1]The simulation study is independent of the specific circuit under debug with random X-bit injection.



(a) Compression Ratio = 50

(b) Compression Ratio = 500

Figure 5: X-Tracer vs. Conventional MISR-Based Compressor.

compression ratio to be half of the conventional MISR-based compressor without redundancy when the compression window is the same, for fair comparison, as an example, this means when the compression ratio is 50, we compress 50 cycles of "raw" trace data into a signature with conventional compressor while we compress 100 cycles of data with X-tracer.

4.2 Results and Discussion

We first compare the extracted useful trace data with the proposed X-tracer against conventional MISR-based compressor without redundancy, as depicted in Fig. 5. Here, Y-axis represents the percentage of extracted information bits out of the useful information bits in the "raw" trace data (denoted as "Extracted Useful Information"), and we report the ratio from data obtained with X-tracer and the proposed trace information extraction algorithm (denoted as "X-tracer+Extraction"), the ratio from data obtained with conventional MISR-based compressor and the proposed extraction algorithm (denoted as "Conv.+Extraction") and the ratio from data obtained conventional MISR-based compressor and extracted by simply counting those signatures that are not contaminated (denoted as "Orig.").

The gap between "Conv.+Extraction" curve and "Orig." curve in Fig. 5 clearly shows the effectiveness of the proposed extraction algorithm, which does not involve any design overhead. In addition, as can be observed from the figure, in most cases, when the X-bit ratio is higher, the percentage of the extracted useful trace information gets smaller. We attribute the few exceptional cases to the fact that X-bits are injected randomly and hence it is possible to have some outliers.

As shown in Fig. 5(a), when the compression ratio is 50 and the X-bit ratio in the "raw" trace data is low (smaller than 0.25%), almost 100% useful information bits can be obtained by all the three solutions. With the increase of X-bit ratio and when it ranges between 0.25% and 2.0%, the proposed X-tracer design significantly outperforms conventional compressor, which proves the effectiveness of the proposed solution with the help of redundancy. When the X-bit ratio is even higher, none of the solutions can extract more than 50% of the useful trace information. From the results, under these circumstances, it can be seen that conventional compressor with the proposed extraction algorithm lead to more extracted trace information than X-tracer. This is due to the fact that the proposed X-tracer needs to compress twice cycles of the data when compared to the conventional compressor in order to keep the same compression ratio, the number of X-bits to be dealt with X-tracer in each compression window is thus also roughly twice. Consequently, when the X-bit ratio is very high, it is a lot harder to extract useful trace information with X-tracer.

When the compression ratio is 500, as shown in Fig. 5(b), most of the above conclusions hold true (for less X-bit ratio). We can observe the significant benefits provided with X-tracer when X-bit ratio is low. It is important to note that, what designers are more concerned about is the lost information during silicon debug in order not to miss the bug's erroneous effects. Hence, the proposed design, being able to recover more trace information, is highly desirable.

It is important to emphasize that, we usually would not try to debug a circuit under the situation when the extracted useful information is very limited (e.g., for the case when X-bit ratio is larger than 2.0% with compression ratio of 50). Instead, we would rather reduce the compression ratio and rely on more debug runs to have better visibility for the circuit under debug. Our next experiment is thus conducted to evaluate the impact of compression ratio on extracted trace information, as shown in Table 1. Generally speaking, when the compression ratio is low, X-bits will contaminate less useful information, and hence more information bits can be extracted by our algorithm for both structures. When X-bit ratio is relatively high, X-tracer can easily achieve high information bit ratio by slightly decreasing compression ratio. For example, when X-bit ratio is 0.2%, the extracted useful information grows from 41.18 to 86.89 with the drop of 20% compression ratio (i.e., from C.R.=500 to C.R.=400). Even for the case with X-bit ratio to be 0.5%, our solution is able to collect about 97.54% useful information data with 100$\times$ compression ratio, which can be very helpful for silicon debug. Meanwhile, reducing compression ratio with conventional compressor will also increase the extracted useful information, but the improvement is not as significant as that of X-tracer.

With repeatable errors, we can rely on more debug runs to incrementally obtain more useful information. By collecting all the data from multiple debug runs, we can again rely on our trace extraction algorithm to acquire more useful trace information. Fig. 6 describes the result when the compression ratio is fixed as 500 and we conduct 1, 2 and 3 debug runs to collect trace data. To be specific, we configure our X-tracer with different primitive polynomials and different input orders in each debug run. From the results, it is easy to observe that when X-bit ratio is not too high (i.e., less than 0.4%), our solution can achieve very high information ratio (more than 90%) by analyzing the combined trace data from just 3 debug runs. Even for the case with 0.45% and 0.5% X-bit ratios in "raw" trace data, we can extract useful information from barely 0% with one debug run to 73.8% and 57.3%, respectively, when two more debug runs are conducted. The above results prove the effectiveness of the reconfigurable trace compressor design.

The computational time of the proposed extraction algorithm is acceptable. It only takes up to hundreds of seconds to analyze the traced signatures with X-tracer. On the other hand, the extra hardware overhead of the proposed X-tracer design is negligible, because the extra MISR and some control circuitries are much smaller when compared to the trace buffer, which occupies most of the hardware cost in trace-based debug solution.

5. CONCLUSION

In this paper, we propose a reconfigurable trace data compressor design with explicit redundancy for silicon debug, in the presence of many X-bits during signal tracing. The proposed *X-tracer* design, together with the novel algorithms to extract useful trace data out of contaminated trace signatures, facilitates to obtain as much trace information as possible while guaranteeing high compression ratio, as demonstrated in our experimental results.

X-Bit Ratio (%)	Extracted Useful Information (%)									
	C.R.=500		C.R.=400		C.R.=300		C.R.=200		C.R.=100	
	Conv.	X-tracer	Conv.	X-tracer	Conv.	X-tracer	Conv.	X-tracer	Conv.	X-tracer
0.05	80.00	96.06	88.33	98.45	88.33	98.66	100.00	99.39	100.00	100.00
0.1	52.51	92.11	66.66	92.71	88.33	97.92	100.00	98.31	100.00	98.79
0.15	49.15	86.70	54.14	95.04	70.83	97.48	75.03	97.51	84.94	98.32
0.20	40.82	41.18	44.98	86.89	65.00	96.49	69.13	95.75	91.59	98.98
0.25	24.15	15.15	36.63	45.44	50.00	89.03	61.65	96.47	80.80	97.81
0.30	15.83	5.00	36.66	7.20	45.83	65.36	59.98	95.97	78.26	97.48
0.35	11.68	4.17	18.37	8.32	32.50	26.59	52.50	93.08	79.99	93.93
0.40	6.68	0.00	20.00	0.00	27.50	14.98	50.82	88.65	74.14	96.48
0.45	4.17	0.00	17.48	0.00	26.67	8.33	44.16	73.35	72.40	98.34
0.50	6.68	0.00	14.99	0.00	13.33	1.67	34.94	32.01	61.58	97.54

C.R.: Compression ratio;
Conv.: Useful information obtained with conventional MISR and the proposed extraction algorithm;
X-tracer: Useful information obtained with X-tracer and the proposed extraction algorithm;

Table 1: Experimental Results with Different Compression Ratio.

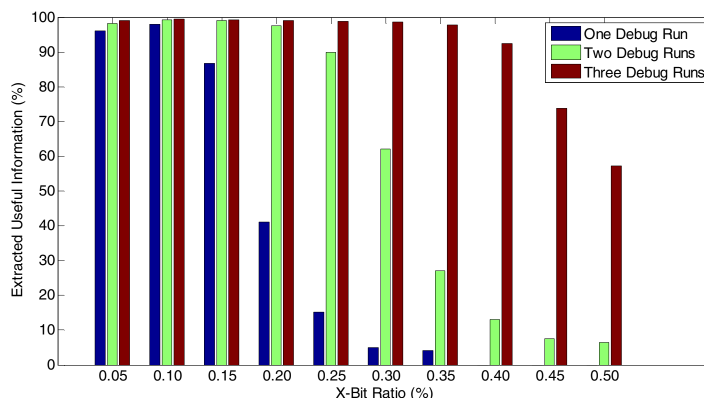


Figure 6: Experimental Results with Reconfigurable Compressors under Different Number of Debug Runs.

6. ACKNOWLEDGEMENTS

This work was supported in part by the General Research Fund CUHK418111 from Hong Kong SAR Research Grants Council (RGC), by National Science Foundation of China (NSFC) under grant No. 60901052, and in part by RGC Grant Direct Allocation under grant No. 2050488.

7. REFERENCES

[1] M. Abramovici. In-System Silicon Validation and Debug. *IEEE Design & Test of Computers*, 25(3):216–223, May-June 2008.

[2] Semiconductor Industry Association (SIA). *The International Technology Roadmap for Semiconductors (ITRS): 2003 Edition*. http://public.itrs.net/Files/2003ITRS/Home2003.htm, 2003.

[3] X. Liu and Q. Xu, "On signal tracing in post-silicon validation," in *Proceedings IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 262–267, 2010.

[4] ARM Ltd. How CoreSight Technology Gets Higher Performance, More Reliable Product to Market Quicker. http://www.arm.com.

[5] R. Leatherman and N. Stollon. An Embedded Debugging Architecture for SOCs. *IEEE Potentials*, Feb.-Mar. 2005.

[6] X. Liu and Q. Xu. Interconnection fabric design for tracing signals in post-silicon validation. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 352–357, 2009.

[7] R. H. Livengood and D. Medeiros. Design for (Physical) Debug for Silicon Microsurgery and Probing of Flip-Chip Packaged Integrated Circuits. In *Proc. IEEE International Test Conference (ITC)*, pp. 877–882, 2007.

[8] K. H. Chang, I. L. Markov, and V. Bertacco. Fixing Design Errors with Counterexamples and Resynthesis. In *Proc. IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, pp. 944–949, 2007.

[9] S. Tang and Q. Xu, "A multi-core debug platform for NoC-based systems," in *Proceedings IEEE/ACM Design, Automation, and Test in Europe (DATE)*, 2008, pp. 870–875.

[10] H. F. Ko and N. Nicolici. Automated Trace Signals Identification and State Restoration for Improving Observability in Post-Silicon Validation. In *Proc. Design, Automation, and Test in Europe (DATE)*, pp. 1298–1303, 2008.

[11] X. Liu and Q. Xu. Trace signal selection for visibility enhancement in post-silicon validation. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pp. 1338–1343, 2009.

[12] J.-S. Yang and N. A. Touba. Automated Selection of Signals to Observe for Efficient Silicon Debug. In *Proc. IEEE VLSI Test Symposium (VTS)*, pp. 79–84, 2009.

[13] X. Liu and Q. Xu, "On Multiplexed Signal Tracing for Post-Silicon Debug," in *Proceedings IEEE/ACM Design, Automation, and Test in Europe (DATE)*, pp. 1–6, 2011.

[14] S. B. Park and S. Mitra. IFRA: Instruction footprint recording and analysis for post-silicon bug localization in processors. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 373–378, 2008.

[15] C.H. Lai, et al. A trace-capable instruction cache for cost efficient real-time program trace compression in SoC. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 136 –141, 2009.

[16] A. Vishnoi, P.R. Panda, and M. Balakrishnan. Cache Aware Compression for Processor Debug Suppport. In *Proc. Design, Automation, and Test in Europe (DATE)*, 2009.

[17] E. Anis and N. Nicolici. On Using Lossless Compresstion of Debug Data in Embedded Logic Analysis. In *Proc. IEEE International Test Conference (ITC)*, pp. 1–10, October 2007.

[18] S. Prabhakar, R. Sethuram, and M.S. Hsiao. Trace Buffer-Based Silicon Debug with Lossless Compression. In *Proc. International Conference on VLSI Design*, pp. 358–363, 2011.

[19] J.-S. Yang and N. A. Touba. Enhancing Silicon Debug via Periodic Monitoring. In *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pp. 125–133, 2008.

[20] E. Anis and N. Nicolici. Low cost debug architecture using lossy compression for silicon debug. In *Proc. Design, Automation, and Test in Europe (DATE)*, pp. 225–230, 2007.

[21] J.S. Yang and N.A. Touba. Expanding Trace Buffer Observation Window for In-System Silicon Debug through Selective Capture. In *Proc. IEEE VLSI Test Symposium (VTS)*, pp. 345–351, 2008.

[22] M. C.-T. Chao, et al. Response Shaper: A Novel Technique to Enhance Unknown Tolerance for Output Response Compaction. In *Proc. International Conference on Computer-Aided Design (ICCAD)*, pp. 80–87, 2005.

[23] S. Mitra, M. Mitzenmacher, S. S. Lumetta, and N. Patil. X-Tolerant Test Response Compaction. *IEEE Design & Test of Computers*, 22(6):566–574, 2005.

[24] J.S. Yang, N.A. Touba, S.Y. Yang, and T.M. Mak. Industrial Case Study for X-Canceling MISR. In *Proc. IEEE International Test Conference (ITC)*, 2009.

[25] N. A. Touba. X-Canceling MISR-An X-Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR. In *Proc. IEEE International Test Conference (ITC)*, pp. 1–10, 2007.