Configurable BISR Chain For Fast Repair Data Loading

Wei Zou
Siemens Digital Industries Software
Wilsonville, Oregon, United States
zou@siemens.com

Benoit Nadeau-Dostie
Siemens Digital Industries Software
Ottawa, Ontario, Canada
benoit.nadeau-dostie@siemens.com

***Abstract*—Embedded memories take a significant portion of current system-on-chip (SOC) designs. Built-in self-repair (BISR) is widely used to improve the design yield by replacing the faulty elements of a memory with spare ones. BISR programs the repair data obtained by a memory built-in self-test (BIST) circuit to a fuse box and transfers the repair data to the memory when the chip is powered up. However, loading the repair data to the memories using a serial scan chain is very time consuming due to the increased number of memories. To reduce the long repair data loading time, we propose a configurable BISR chain repair system. In the proposed solution, the serial scan chain is partitioned into an optimal number of segments. Only the segments connected to defective memories are scanned. Experimental results show that the proposed method can significantly reduce the repair data loading time compared to two baseline methods.**

Keywords—BISR, BIRA, BIST, Embedded Memory, Yield

I. INTRODUCTION

Current high-density system-on-chip (SOC) designs include a large number of embedded memories. The number of memories can be over thousands and even tens of thousands in some applications. For example, in [1], a Graphcore MK2 IPU has 1472 powerful processor cores, and each core has an unprecedented 900MB memory. Designed tightly to the technology limits, memories are more prone to failures than other circuits, which can affect not only manufacture yield but also circuit reliability adversely. Built-in self-test (BIST) techniques [2] are typically employed to identify defects and problems in the memories. Moreover, a circuit having embedded memories usually includes built-in self-repair (BISR) circuitry [3] for performing a repair analysis (built-in repair analysis or BIRA) and for replacing faulty elements with spare ones. Repair information obtained by the built-in self-repair (BISR) circuitry can be stored in a non-volatile storage device such as a fuse box. When the circuit is powered up, the repair information can be retrieved and loaded. The transportation of the repair information between the memories and the fuse box involves a scan network, which is often referred to as a BISR chain. Data delivery via a single scan network is serial in nature. A circuit can have hundreds or thousands of memories. Using a conventional scan network can take too long time for manufacture and in-system test and repair.

To reduce the repair data loading time when a chip is powered up, two methods were proposed in [4] and [5]. In [4], the failure information coming from different memories or segments of memories tested is merged to calculate a common repair solution which can be broadcasted. Although the proposed method can reduce both the repair hardware overhead and the repair data loading time, it is difficult to achieve more than an order of magnitude due to routing issues. In [5], bypass logic is added to exclude a memory from the BISR chain if a memory is fault free and doesn't need a repair. There is one data bit which controls the bypass logic. If the data bit is '0', the memory will be included in the BISR chain, otherwise it will be excluded from the BISR chain. All the data bits controlling the bypass logic are concatenated into a scan chain dedicated for the chain configuration data shifting. The chain configuration data is programmed to the fuse box during the repair data programming, and it is loaded to the configuration chain before the repair data loading. A similar method is also proposed in [6]. For a mature process, only a very small number of memories need repair. Most of the memories are bypassed during the repair data loading. The repair data loading time can only be reduced by an order of magnitude with this method as the majority of memories only use one or two spares and the length of each repair register associated to a memory is about 10 bits on average. In this paper, we propose a new method which divides the BISR chain into an optimal number of segments and each segment contains a number of repair registers which may or may not be shared by several memories. A segment selection circuit is added to each segment to exclude the segment from the BISR chain if the memories of the segment are fault free. Different from the method of [5], the data bits controlling the segment selection circuits are part of the BISR chain, which reduces the routing overhead by not introducing a new scan chain. The proposed method can further reduce the repair data loading time compared to the method in [5].

The rest of this paper is organized as follows. Section II describes the general repair system widely used in the industry and the prior work on fast repair data loading. Section III describes the proposed configurable BISR chain repair system. Section IV summarizes experimental results. Section V concludes this paper.

II. THE GENERIC REPAIR SYSTEM AND PRIOR WORK OF FAST REPAIR DATA LOADING

The generic repair system widely used in the industry is shown in Fig. 1. For illustration purpose and easy understanding, we only show six memories which are tested by two memory BIST controllers. For each memory, there is a BISR register connected to it. The data stored in the BISR register is used for memory repair. The size of the BISR register is determined by the memory size and the number of spare elements that the memory has. Assume all the memories in Fig. 1 have 128 rows and only one spare row for repair. Any failing row detected during the test can be replaced with the spare row during the repair. For this memory, the BISR

2022 IEEE International Test Conference (ITC) | 978-1-6654-6270-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITC50671.2022.00012

2378-2250/22/$31.00 ©2022 IEEE
DOI 10.1109/ITC50671.2022.00012

Regular Paper

Authorized licensed use limited to: Siemens AG GBS B&S SOL GMS. Downloaded on September 13,2024 at 14:01:21 UTC from IEEE Xplore. Restrictions apply.

register has 8 bits and the composition of the BISR register is shown in Fig. 2. Seven bits (6 to 0 in this example) represent the row address of the faulty row to be repaired. One bit (bit 7) is the repair enable bit which activates the repair. If row 7 of the memory has a failure, the repair data in the BISR register is "10000111". If a memory doesn't have a failure, the repair data in the BISR register is "0000000". So, by checking if the BISR register contains "1" or not, we can determine whether the memory needs repair or not. All the BISR registers are concatenated into a scan chain which is referred as a BISR chain. The BISR chain is connected to a fuse box controller. During the memory test, the BIST controller analyzes each memory failure and generates the repair data for the memories which fail the test. After the memory test is finished, the repair data is transferred from the BIST controller to the BISR chain. The fuse box controller programs the repair data into a fuse box while shifting the BISR chain. When a chip is powered up, the fuse box controller reads the repair data from the fuse box and shifts the repair data to the BISR registers on the BISR chain. After that, the repair data is applied to the memory from the BISR registers. The number of repair data loading cycles when a chip is powered up equals the BISR chain length. For the design of Fig. 1, there are 6 BISR registers and each BISR register is 8 bits wide. The number of repair data loading cycles is 48.

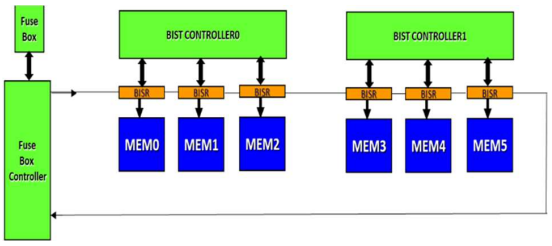


Fig. 1. The general memory repair system

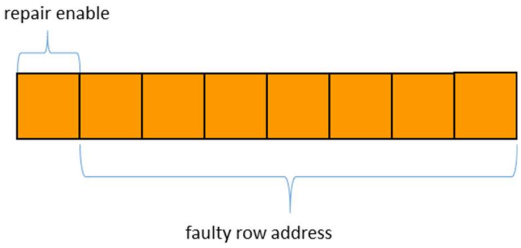


Fig. 2. The BISR register composition

In [4], instead of having a dedicated BISR register for each memory, several memories share one BISR register. By doing this, both the hardware overhead and the repair data loading time can be reduced. Fig. 3 shows the shared BISR repair system proposed in [4]. In Fig. 3, every three memories share one BISR register. The repair data in the BISR register will be broadcasted to three memories. Assuming the BISR registers in Fig. 3 have the same size as the ones in Fig. 1, the number of repair data loading cycles is one third of the generic repair system of Fig. 1. The method reduces both the BISR and BIRA hardware overhead and the repair data loading time. Best results are obtained for memories accessed through a shared bus architecture or using row repair as it is easier to share the BIRA circuits in these cases. However, IO/column repair is widely used and it is difficult to achieve more than an order of magnitude reduction for memories in that case due to the large number of connections required between the memories and the BIST controller.

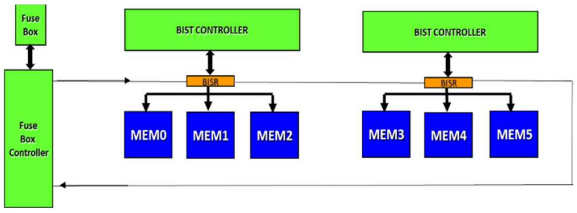


Fig. 3. BISR sharing method of [4]

For a mature process, typically only a very small number of memories on a chip are defective. It is wasting time shifting the repair data through the BISR registers of the memories which are not defective. In [5], a MUX is added at the end of each BISR register. The MUX select pin is controlled by a bypass register. An example hardware implementation of [5] is shown in Fig. 4. If a memory is not defective, the BISR register of that memory is bypassed by setting the MUX select pin to 1. FF is a pipeline flip-flop used for facilitating timing closure. Without pipelining, very long paths could be created since most BISR registers are bypassed. The configuration registers controlling the MUX select pin are concatenated into one scan chain which is dedicated for the chain configuration data shifting. The chain configuration data is programmed to the fuse box after the memory test. When a chip is powered up, the chain configuration data is loaded to the configuration data chain. The BISR registers are bypassed if the corresponding configuration register is set to 1. After that, the repair data is loaded into the BISR chain. The number of repair data loading cycles include two parts. The first part is the number of configuration chain loading cycles, and the second part is the number of BISR chain loading cycles. The total number of repair data loading cycles can be represented by equation (1).

$$Nmem + (Nmem - K) + \sum_{i=1}^{k} BISRi \qquad (1)$$

Where *Nmem* is the number of the memories, *K* is the number of memories which are defective and *BISRi* is the width of the *ith* BISR register. For example, if MEM2 in Fig. 4 is defective, the number of repair data loading cycles is 6+5+8=19 assuming all the BISR registers are 8 bits wide.

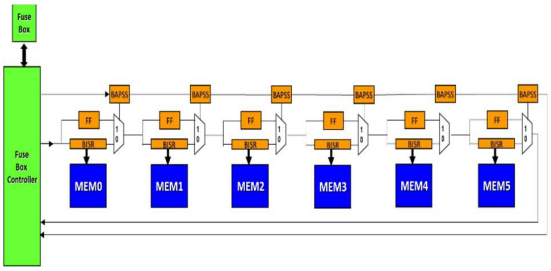


Fig. 4. Memory bypass repair system of [5]

III. PROPOSED CONFIGURABLE BISR CHAIN

Equation [1] shows that the number of repair data loading cycles of the method in [5] is roughly two times the number

Authorized licensed use limited to: Siemens AG GBS B&S SOL GMS. Downloaded on September 13,2024 at 14:01:21 UTC from IEEE Xplore. Restrictions apply.

of memories. Therefore, the number of repair data loading cycles of [5] is dominated by the number of memories. If a design has thousands of memories, the repair data loading time will be thousands of clock cycles. So, it is still time consuming for a design with a large number of memories. In this section, we propose a new configurable BISR chain repair system which can further reduce the repair loading time of [5]. The proposed repair system is shown in Fig. 5. The BISR chain is divided into a certain number of segments. For example, the BISR chain in Fig. 5 is divided into two segments and each segment has three BISR registers. A segment selection circuit (SSC) is added at the end of each segment. If a segment contains one or more memories which are defective and need repair, SSC includes this segment in the BISR chain, otherwise SSC excludes this segment from the BISR chain. The number of segments is decided by the BISR chain length and the chip defect density. The algorithm partitioning the BISR chain will be described later in Section 4. The primary function of SSC is to include a segment in the BISR chain or exclude a segment from the BISR chain. The hardware diagram of SSC is shown in Fig. 6. The definitions of the signals shown in Fig. 5 and Fig. 6 are described in Table 1. The control signals 1D, CF, UR, SR, SE and UE are generated by the fuse box controller. The signal SEL is the output of a segment selection register reg1. There are two options to include a segment into the BISR chain. Signals SEL and CF can be set to 1 and 0, respectively, or signals 1D and CF can be set to 1 and 0, respectively. A segment can be excluded from the BISR chain by setting SEL and 1D to 0 or set CF to 1. In Fig. 5, assume MEM1 is defective, and all other memories are fault free. The segment on the left-hand side will be included into the BISR chain by setting SEL = 1 and CF = 0 in the SSC on the left-hand side. The segment on the right-hand side will be excluded from the BISR chain by setting 1D = 0 and SEL= 0 in the SSC on the right-hand side. The active scan path of the BISR chain using these settings is shown in Fig. 7. The BISR chain length under this setting is 26 assuming each BISR register is 8 bits wide. So, 26 cycles are required to load the repair data.

In SSC, reg1 is a copy register of reg0, and the data shifted into reg0 is also shifted into reg1 when UE = SE = 1. Reg0 and reg1 are automatically set during the repair data programming phase and the chip power up phase. During the repair data programming phase, the repair data is transferred from the BIST controller to the BISR registers after memory BIST is finished. Reg0 is automatically set based on the repair data stored in the BISR registers by performing a "1" detection operation. When a memory needs repair, the BISR register of that memory contains at least one bit with value "1". So, if a segment contains one or more bits with a "1", at least one memory in this segment needs repair, and the segment is included into the BISR chain. The "1" detection operation is basically checking each bit of a segment to see whether it is "1" or not while shifting the segment. If a segment contains "1", reg0 of that segment is set to 1. Otherwise it is set to 0. The "1" detection is performed by first resetting reg0 and reg1 by pulsing SR and UR from "1" to "0" and back to "1". The BISR chain is then shifted with 1D=1, SE=1, CF=0 and UE=0. During the shifting, "0" is injected into the scan input of each segment because 1D = 1 and the output of the AND gate is 0. The purpose of injecting 0 is to prevent the repair data of the previous segment from entering the current segment. The total number of shift clock cycles of the "1" detection is the length of the segment. If the segments have different lengths, the number of shift clock cycles is the length of the longest segment. During the "1" detection operation, the input of reg0 is the result of ORing the output of reg0 and the data shifted out from the segment. So, reg0 is set to 1 if a segment contains "1", otherwise it holds the original value 0 during the shift. After the "1" detection is done, the segment selection information is stored in reg0 of each segment. The segment selection information is then programmed in the fuse box by rotating the BISR chain and setting the signals CF=1, SE=1 and UE=1. Since CF=1, all BISR registers are excluded from the BISR chain. The scan path of the BISR chain only includes reg0 of each segment. After the rotation is done, the segment selection information is written to the fuse box, and it is also copied from reg0 to reg1. By setting CF=0 and 1D=0, reg1 is controlling the select pin of the bypass MUX in SSC, and the segment which doesn't need repair is excluded from the BISR chain. Since the repair data in the BISR chain is lost during the "1" detection operation, we need to transfer the repair data from the BIST controller to the BISR register one more time before shifting the BISR chain to program the repair information in the fuse box.

When a chip is powered up, CF=1 is set first to only include reg0 of each segment in the BISR chain. Second, the segment selection information is loaded from the fuse box into reg0 and reg1 by setting SE=1 and UE=1. Third, 1D=0 and CF=0 are set to configure the BISR chain. Finally, the configured BISR chain is shifted to load the repair data from the fuse box.

The complete repair data programming sequence and repair data loading sequence are shown below.

Programming sequence:

1. Apply power-up sequence
2. Perform memory BIST test, analyze the memory failures and store the repair information in the controller.
3. Transfer the repair data from the BIST controller to the BISR registers.
4. Perform 1-detection operation to generate the segment selection information by setting CF=0, 1D=1, UR=1, UE=0, SR=1, SE=1.
5. Program the segment selection information into the fuse box while performing a BISR chain rotation with UR=1, UE=1, SR=1, 1D=0, SE=1. The number of rotation cycles is the number of segments.
6. Calculate the new BISR chain length. Reset all segments (SR=1→0→0), inject a leading 1 and shift the chain until the leading 1 appears at the scan out port SO of the BISR chain.
7. Transfer the repair data from the BIST controller to the BISR registers.
8. Include the BISR segments which need repair into the BISR chain by setting 1D=0, CF=0.
9. Program the repair data in the fuse box while performing a rotation with UR=1, SR=1, UE=0, 1D=0, SE=1, CF = 0, 1D = 0.

Authorized licensed use limited to: Siemens AG GBS B&S SOL GMS. Downloaded on September 13,2024 at 14:01:21 UTC from IEEE Xplore. Restrictions apply.

Power-up sequence:

1. Reset the BISR chain as well as reg0 and reg1 inside the SSC of each segment by changing UR and SR to 0 and then to 1. All the BISR registers contain "0" after this step.
2. Inject a leading 1 in the BISR chain. The leading 1 is used to measure and confirm the length of the configuration chain in the next step.
3. Load the segment selection information into the BISR chain until the leading 1 appears at the scan out port SO of the BISR chain. This is done by setting 1D = 0, CF=1, SE =1, UE=1. Since CF=1, all BISR registers are excluded from the BISR chain.
4. Update the segment selection chain length.
5. Reset the BISR chain and reset reg0 inside SSC of each segment by changing the signal SR to 0 and then to 1.
6. Include the BISR segments which need repair into the BISR chain by setting 1D=0, CF=0.
7. Inject a leading 1 in the BISR chain. The leading 1 is used to measure and confirm the length of the BISR chain in the next step.
8. Load the repair data into the BISR chain until the leading 1 appears at the scan out port SO of the BISR chain.
9. Update the repair chain length.

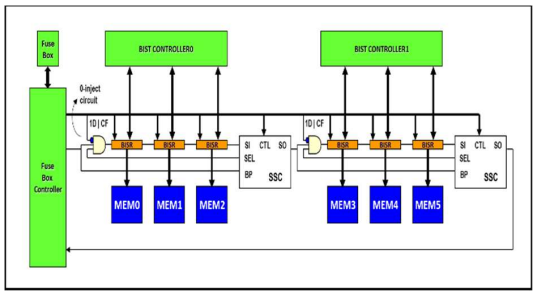


Fig. 5. Example circuit with two segments

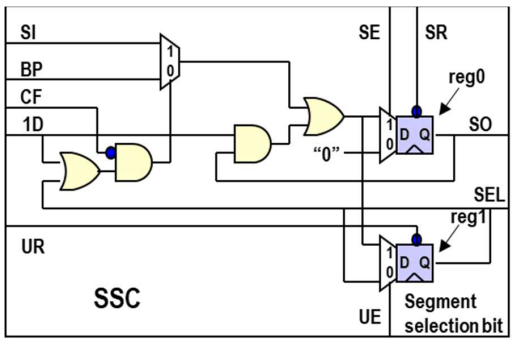


Fig. 6. SSC circuit logic

Table 1 SSC signals definitions

Name	Function
SI	scan in
SO	scan out
BP	segment bypass
1D	enable "1" detection
SE	scan enable
SR	scan reset
UR	update reset
UE	update enable
SEL	segment select
CF	configuration chain select

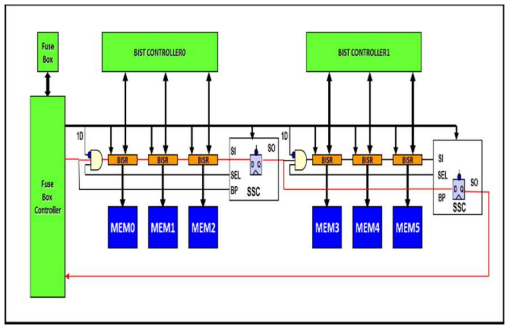


Fig. 7. Example circuit with bypassed segment

IV. BISR Chain Partitioning

The number of segments on the BISR chain is critical in determining the repair data loading time. In the proposed repair system, the repair data loading time not only includes the time for shifting the repair data, but also the time for shifting the segment selection bits. Eq. (2) can be used to represent the number of clock cycles required for loading the repair information:

$$T = (Nrepair * \frac{L}{Nseg} + Nseg) + Nseg \qquad (2)$$

where *Nrepair* is the number of segments where one or more memories need repair, *Nseg* is the total number of segments on the BISR chain, and *L* is the total length of all the BISR registers on the BISR chain. The first term in Eq. (2) represents the time for shifting the repair information through the scan cells in the segments where memory repair is needed and the scan elements in each SSC. The second term in Eq. (2) represents the time for shifting the segment selection bits into the segment selection registers in each SSC. For the loading time of the repair information *T* to reach a minimum, the derivative of *T* in Eq. (2) with respect to *Nseg* needs to be zero:

$$-Nrepair * \frac{L}{Nseg^2} + 2 = 0 \qquad (3)$$

Solving Eq. (3):

$$Nseg = \sqrt[2]{Nrepair * L/2} \qquad (4)$$

Authorized licensed use limited to: Siemens AG GBS B&S SOL GMS. Downloaded on September 13,2024 at 14:01:21 UTC from IEEE Xplore. Restrictions apply.

According to Eq. (4), the optimal value of *Nseg* depends on *Nrepair* which represents the chip defect density. For a mature process, the average value of *Nrepair* is between 1 or 2. However, the maximum number can be a few tens.

Assume that a circuit has 800 BISR registers, the average number of bits in a BISR register is 8, and *Nrepair* is 2. Using Eq. (4), an optimal number of repair register segments is computed to be 80. However, it may not be trivial to partition the BISR chain into 80 segments of about equal length. Not all BISR registers have the same length. Moreover, some memories may be inside a reused circuit block and the corresponding BISR registers need to be kept in one repair register segment. A greedy algorithm is chosen for the BISR chain partition. The pseudo code of the greedy algorithm is shown below. *Nbisr* is the total number of BISR registers on the BISR chain, and *len_bisr_i* is the length of the *ith* BISR register. We loop all the BISR registers on the BISR chain from the chain scan in port SI to the chain scan out port SO, and assign each BISR register to different segments based on the following rule. If the difference between the optimal segment length and the current segment length with the *ith* BISR register included is bigger than the difference between the optimal segment length and the length of the current segment without the *ith* BISR register included, the *ith* BISR will be assigned to the next segment, otherwise it will be assigned to the current segment. By doing this, we can make each segment length as close to the optimal segment length *L/Nseg* as possible.

The partition algorithm:

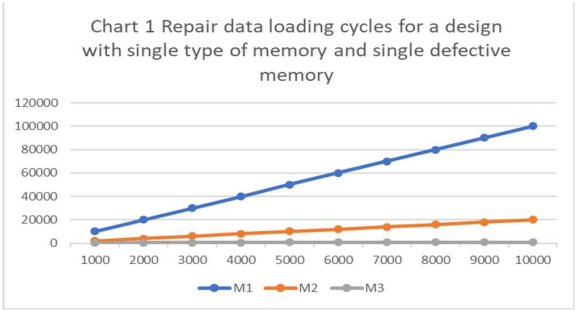
```
seg_len = 0;
seg_id = 0;
for (i =0; i<Nbisr; i++) {
  if { |seg_len+len_bisr_i -L/Nseg|> |L/Nseg -seg_len|} {
     assign the ith bisr to the next segment
     seg_len =  bisr_i_len;
     seg_id = seg_id+1;
   } else {
     assign the ith bisr to the current segment
     seg_len = seg_len + bisr_i_len;
   }
}
```

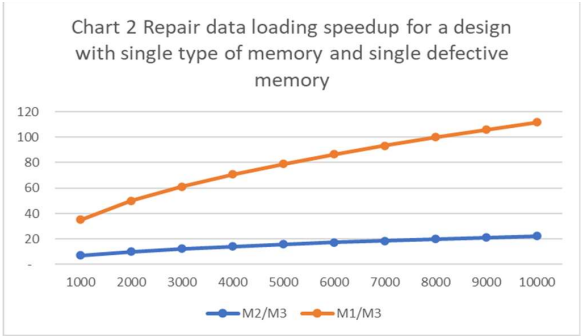
For the circuit of Fig. 5, the sum of all the BISR registers on the BISR chain is 48. Assuming that there is only one defective memory, the optimal segment length is 10 based on Equation (4). So, the BISR chain is divided into 6 segments and each segment has one memory using the greedy algorithm above. The repair data loading cycle of the BISR chain is 20, which is less than 28, the repair data loading cycle of the BISR chain with two equal sized segments. In Fig. 5, the BISR chain is divided into two segments to illustrate the concept of chain partitioning. Two is not the optimal number of segments in this case. Note that there is no advantage over [5] for such circuits with short BISR chains. The advantage of the proposed method will become apparent in the next section.

V. Experimental Results

Five experiments were conducted to compare the repair data loading time of the proposed method and two baseline methods which are the generic repair system and the method in [5]. In the experiments, the repair data loading time of all three methods is calculated as a function of the total number of memories, the BISR register size of each memory and the number of defective memories.

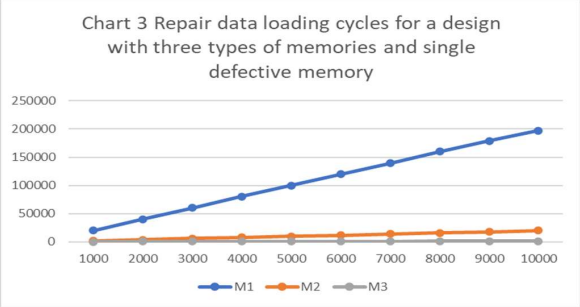
In the first experiment, the design has only one type of memory which has 10 bits of repair data. The number of memories in the design varies from 1000 to 10,000 with an increment of 1000. There is only one defective memory which is randomly selected. The proposed method assumes a value of 1 for *Nrepair* in Equation (4) when partitioning the BISR chain. The number of clock cycles for repair data loading is listed in chart 1. The X axis of chart 1 is the number of the memories in the design. The Y axis is the number of repair data loading cycles. The lines in color are the number of repair data loading cycles of the three methods. M1 is the generic repair system, M2 is the method in [5], and M3 is the proposed method. The repair data loading time speed up of M3 over M2 and M3 over M1 is shown in chart 2. Chart 2, shows that the proposed method can have up to 112X speed up compared to the generic repair system and up to 22 X speed up compared to the method in [5].

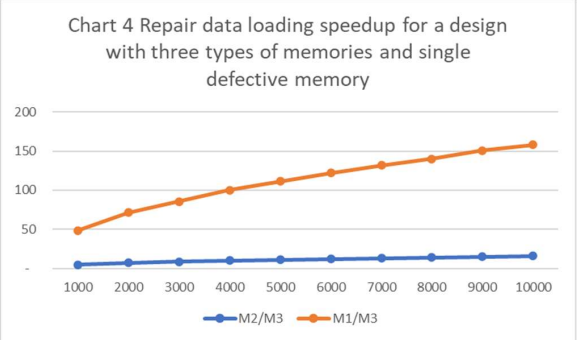




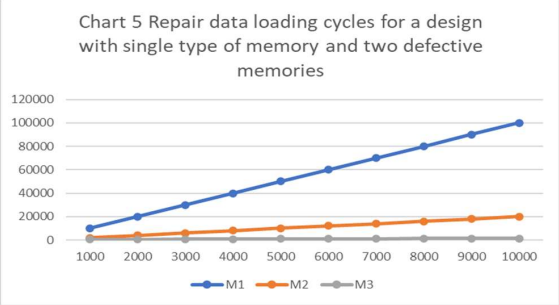
In the second experiment, the design has three different types of memories. The repair data bits of each memory type is 10, 20 and 30. The total number of memories in the design varies from 1000 to 10,000 with an increment of 1000. The number of each memory type is random. The design has only one defective memory which needs repair. A value of 1 for *Nrepair* in Equation (4) is used when partitioning the BISR chain. The results of the experiment are shown in chart 3 and chart 4. It can be seen that the proposed method can have up to 158X speed up compared to the regular repair system and up to 16 X speed up compared to the method in [5].

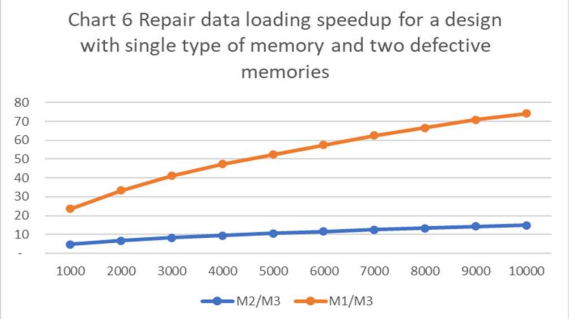
Authorized licensed use limited to: Siemens AG GBS B&S SOL GMS. Downloaded on September 13,2024 at 14:01:21 UTC from IEEE Xplore. Restrictions apply.

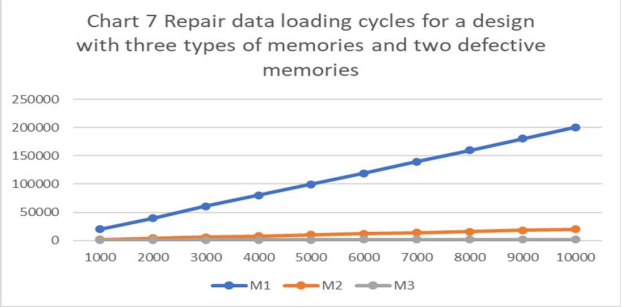


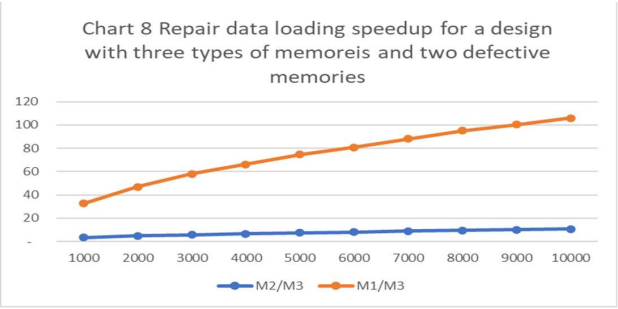


In the first two experiments, the design has only one defective memory, which matches the value used in Equation (4). However, the number of failing memories follows a negative binomial distribution where the average number is between 1 and 2 but can also be up to 10 or even more in some cases [7]. Experiments three and four repeat the first two experiments, but two defective memories are selected instead of one. *Nrepair* is still set to 1 during the BISR chain partitioning. The results of experiment three are shown in chart 5 and chart 6. The results of experiment four are shown in chart 7 and chart 8. Although the number of repairs is different than the value assumed in equation (4), the proposed method can still achieve a significant speed up compared to the two baseline methods.









In the last experiment, we characterize the repair data loading time of the proposed method with regards to the assumed number of defective segments. The design in the experiment has 5000 memories with three different types. The number of defective memories varies from 1 to 10 and the assumed number of defective segments varies from 1 to 20. The experimental results are shown in chart 9. In chart 9, the X axis is the assumed number of defective segments of the proposed method. The Y axis is the repair data loading cycles. The lines in color are the designs with different number of defective memories. From chart 9, it can be seen that the proposed method has a minimal repair data loading time when the number of defective memories approximately equals to the assumed number of defective segments. For example, if both the assumed and actual number of defective segments is 10, the repair data loading cycles is 2848, which is 35 times less cycles compared to 100110, the repair data loading cycles of the generic repair system, and 3.6 times less cycles compared to 10190, the repair data loading cycles of [5]. For the case where the actual number of defective memories is 1 but the assumed number is still 10, the number of cycles is 56 times less compared to the generic repair system and 6.4 times less compared to [5]. In practice, it is preferable to minimize the differences in the number of repair data loading cycles between chips requiring a different number of repairs. This can be achieved by setting the

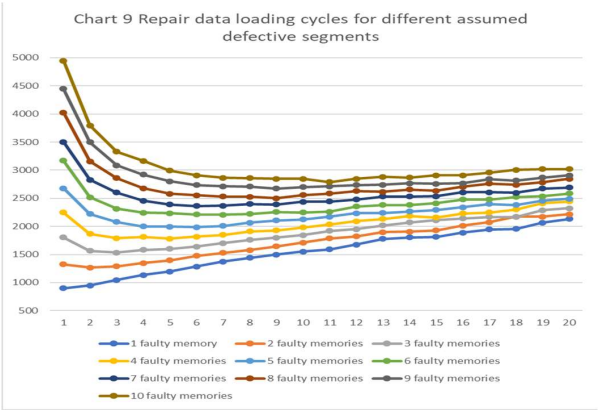
Authorized licensed use limited to: Siemens AG GBS B&S SOL GMS. Downloaded on September 13,2024 at 14:01:21 UTC from IEEE Xplore. Restrictions apply.

expected maximum number of defective memories (Nrepair in equation (4)) to a value between 10 and 20 when calculating the number of BISR chain segments. These values can be calculated using the equations of [6] or determined empirically based on previous production of circuits using a similar technology.

The active area increase of the BISR chain due to the addition of the SSC circuits can calculated as follows:

$$\Delta_{area} = (3 * Nseg)/L$$

One SSC circuit is roughly equivalent to 3 bits of the BISR chain hence the constant factor 3 in the equation. For the example of chart 9 with 20 expected repairs and a nominal BISR chain length of 100000 bits, the area increases amounts to about 3%. Four additional control signals (CF, 1D, UE, UR) need to be generated by the BISR controller to drive the SSC circuits distributed along the BISR chain. These signals are not as heavily loaded as the three original signals (SE, SR and clock) used to operate the BISR chain and significantly less buffering is required. In the example of chart 9, the number of SSC circuits is 1000 meaning that the load on the new control signals is only 1% of the load seen by the original signals.



VI. Conclusions

In this paper, we proposed a configurable BISR chain repair system for fast repair data loading. The proposed method partitions the serial BISR chain into an optimal number of segments and dynamically exclude the segments from the scan path of the BISR chain if the memories of the segments are not defective. Experimental results demonstrate that the number of repair data loading cycles can be reduced by one to two orders of magnitude compared to two baseline methods. An additional order of magnitude can be achieved by combining the proposed method with other methods, such as the one in [4], where BISR registers are shared by several memories.

References

[1] https://www.graphcore.ai/products/ipu

[2] Rochit Rajsuman, "Design and test of large embedded memories: an overview", IEEE Design & Test of Computers, vol.18, no. 3, pp. 16-27, May 2001.

[3] Yervant Zorian, "Embedded memory test & repair: Infrastructure IP for SOC yield", Proceedings of the International Test Conference, Baltmore, Oct. 2002, pp. 340-349.

[4] Benoit Nadeau-Dostie and Luc Romain, "Memory repair logic sharing techniques and their impact on yield", Proceedings of the International Test Conference, pp. 1-5, 2020.

[5] Devanathan VR, Harsharaj Ellur, Mohd Imran, Shivani Bathla., "Hierarchical, Physical-Aware, Built-In Self-Repair of Embedded Memories", Session11, Design Automation Conference 2013

[6] Akhil Garg, Prashant Dubey, "Fuse Area Reduction based on Quantitative Yield Analysis and Effective Chip Cost", Proceedings of the 21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'06), pp. 166 – 174, 2006

[7] Benoit Nadeau-Dostie, "Memory repair", Session 4D ET1 : Memory Testing Embedded Tutorial, International Test Conference 2021

Authorized licensed use limited to: Siemens AG GBS B&S SOL GMS. Downloaded on September 13,2024 at 14:01:21 UTC from IEEE Xplore. Restrictions apply.