Retention-Aware Test Scheduling for BISTed Embedded SRAMs

Qiang Xu[†], Baosheng Wang[‡] and F. Y. Young[†]

[†] Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: {qxu,fyyoung}@cse.cuhk.edu.hk

[‡] ATI Technologies Inc.
Commerce Valley Drive East, Markham, Ontario, Canada L3T 7X6
Email: bawang@ati.com

Abstract

In this paper we address the test scheduling problem for Built-in Self-tested (BISTed) embedded SRAMs (e-SRAMs) when Data Retention Faults (DRFs) are considered. The proposed test scheduling algorithm utilizes the "retention-aware" test power model [1] to minimize the total testing time of e-SRAMs while not violating given power constraints. Without losing generality, we consider both cases where the pause time for data retention faults is fixed and cases where it can be varied. Experimental results show that the "retention-aware" test scheduling algorithm can reduce the testing time of e-SRAMs up to more than 98 percent at the computational time within a second.

1 Introduction

Embedded memories, in particular embedded SRAMs (e-SRAMs), tend to consume most of the silicon area in today's system-on-a-chips (SoCs), ranging from register files as small as 64bits to larger caches with sizes of hundreds of kilobits or even megabits [2]. Because of their extreme density, e-SRAMs are more prone to manufacturing defects than the other types of on-chip circuitry (e.g., standard cells) and it is important to test them thoroughly to ensure an acceptable SoC yield. Therefore, how to efficiently and effectively test these hundreds of instances of e-SRAMs on-chip becomes a major challenge for the SoC system integrators [3]. On the one hand, we would like to let more e-SRAMs be tested in parallel to reduce the total testing time and hence the SoC test cost. On the other hand, however, the test power constraint becomes a major concern because power consumption in test mode is usually higher than the one in functional mode [17]. Therefore, efficient power-constrained test scheduling techniques (e.g., [12]) play a key role on reducing e-SRAM test cost.

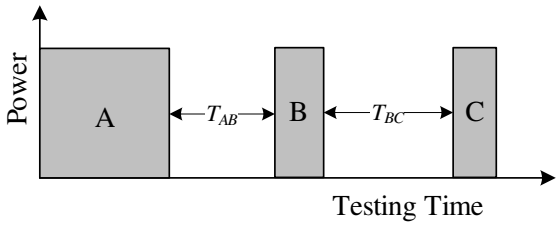

Figure 1. The retention-aware test model.

Most prior work in test scheduling assumes a constant power consumption during the entire test process. Although simple and effective for logic testing, this model is overly pessimistic for e-SRAM testing when data retention faults (DRFs) are considered [1]. DRFs model the defects in SRAM bit cells that fails to retain a stored logic value. The most common retention test method is simply to load a known value into the cell and wait for a period of time (up to hundreds of milliseconds [2]), and read it out. During the two DRF pause time (for retention test of both logic '0' and logic '1'), no read/write operations are performed and hence it consumes near-zero test power. By taking this property into account, Wang et al. [1] proposed a "retention-aware" test power model for testing built-in self-tested (BISTed) e-SRAMs, in which each e-SRAM test is represented by three rectangles A, B, and C with interval T_{AB} and T_{BC}, as shown in Figure 1. Empirical analysis is given in [1] that shows this model may significantly reduce the total e-SRAM testing time. However, automatic test scheduling algorithms are not provided in their work. Therefore, in this paper, we propose "retention-aware" test scheduling techniques for e-SRAMs based on the test power model presented in [1]. Experimental results show that our approach significantly reduces the total testing time under given power constraints.

The remainder of this paper is organized as follows. Section 2 reviews the related work in this domain. The detailed "retention-aware" test scheduling algorithms are presented in Section 3. Next, Section 4 presents our experiments on four test cases. Finally, Section 5 concludes this paper.

2 Prior Work

Related Work in DRF Tests: As discussed in [2], retention testing needs to consider the slow process corner case, whose leakage (responsible for the loss of the stored logic value) actually slows from 130nm to 90nm. Because of this, the pause time for testing DRFs does not decrease significantly with the increasing chip operational frequency, and hence the testing time for DRFs dominates the total e-SRAM testing time. To address this problem, many design for test (DFT) techniques have been proposed to completely remove the pause time from the test flow, by introducing customized circuitry [2, 10, 14, 15]. Although effective, the above DFT techniques often come with high cost in terms of hardware/performance overhead and design efforts. Moreover, the memory compiler supplied by memory vendors often does not provide the feature to apply the above DFT techniques. As a result, in this paper we assume the DRF tests are applied in the traditional way with extended wait period.

Related Work in Test Scheduling: Many test scheduling techniques have been proposed in the literature [4, 6, 7, 9, 13, 17] (only name a few). In particular, [4, 7] considered power constraint in their work. The above work, however, mainly targets on the test scheduling of logic cores (usually scanned), and one of the design aims is to design an efficient test access mechanism (TAM) architecture to link the test source/sink to the core under test. e-SRAM tests, however, are usually conducted by BIST engines, without involving TAM design and optimization issues. Another major difference of e-SRAM test from logic test is that the testing time for an e-SRAM is a fixed constant with its size given, while the testing time for a logic core usually varies with the assigned TAM width. Wang et al. [12] proposed a simulated annealing (SA) algorithm for the test scheduling of BISTed memory cores. Test power for each memory is assumed to be constant during its entire test process and the computational time is quite high when the number of memory cores is large.

The Impact of e-SRAM BIST Architecture: How the e-SRAM BIST architectures are designed affects the test scheduling process. For example, when many different e-SRAMs share the same BIST engine to save silicon area, depending on the BIST scheme, they may [8] or may not [11] be able to be tested in parallel. In this paper, we consider the case that each e-SRAM is supplied with its own BIST engine. It is important to note that, however, the proposed approach can be generalized to the above BIST-sharing scenario by adding additional constraints into the test scheduling process. In addition, whether the BIST engine is "soft" or "hard" significantly affects the test scheduling process. When it is "soft", i.e., the system integrator is able to modify its architecture, the pause time for DRF tests (i.e., T_{AB} and T_{BC}) can be changed easily. When it is hardwired, the pause time is a pre-determined fixed value. Without losing generality, we consider both cases in this paper, as shown in the following section.

3 Retention-Aware Test Scheduling

The retention-aware test scheduling problem invested in this section can be stated as follows:

Problem $P_{drf-opt}$: Given the test parameters for the BISTed e-SRAMs, including

- the total number of e-SRAMs N_m;
- the maximum allowed test power P_{max};
- for each BISTed e-SRAM i, the test power consumption P_i, the testing time T_{A_i}, T_{B_i} and T_{C_i} for block A, B and C;
- the minimum pause time for testing DRFs T_{pause};

determine the test schedule of all e-SRAMs such that (i) the total testing time is minimized; (ii) the pause time for testing DRFs satisfies $T_{AB} \geq T_{pause}$ and $T_{BC} \geq T_{pause}$; and (iii) the test power consumption at any moment does not exceed P_{max}.

In this section, we first consider the case that the BIST architectures for e-SRAMs are "soft" and hence T_{AB} and T_{BC} can vary as long as they are greater than T_{pause}. Next, we consider the case that the BIST architectures for e-SRAMs are "hard" or the system integrator is not willing to make any changes. In this case, $T_{AB} = T_{BC} = T_{pause}$ holds for all e-SRAMs.

3.1 Scheduling with Flexible DRF Pause Time

3.1.1 Packing-based Scheduling Strategy

The retention-aware test scheduling with flexible DRF pause time problem can be modeled as a constrained rectangle packing problem. That is, we try to pack all the rectangles A_i, B_i and C_i ($i = 1 \ldots N_m$) into a rectangular region of height not exceeding P_{max} and of a minimized width such that for each BISTed e-SRAM i, the separation between A_i and B_i and the separation between B_i and C_i are at least T_{pause}. We first borrow a SA-based approach as described in [16] to solve this problem.

To impose a 'minimum separation' constraint between two blocks, an edge of weight T_{pause} is inserted into the horizontal constraint graph from A_i to B_i (from B_i to C_i respectively). The cost function of a candidate solution S used in the annealing process is as follows:

$$cost(S) = area(S) + \alpha \times Penalty_1(S) + \beta \times Penalty_2(S)$$

, where α and β are weights, $area(S)$ is the area of S and is computed as $P_{max} \times width(S)$, $Penalty_1(S)$ is the penalty for exceeding the maximum allowed test power P_{max} and $Penalty_2(S)$ is the penalty for violating the minimum separation constraints. $Penalty_1(S)$ and $Penalty_2(S)$ are computed as:

$$\begin{aligned} Penalty_1(S) &= (\max\{0, height(S) - P_{max}\})^2 \\ Penalty_2(S) &= \textstyle\sum_{i=1}^{n} (\max\{0, T_{pause} - (x(B_i) - x(A_i))\})^2 + \\ &\quad \textstyle\sum_{i=1}^{n} (\max\{0, T_{pause} - (x(C_i) - x(B_i))\})^2 \end{aligned}$$

, where $x(R)$ of a rectangular block R is the x-coordinate of the lower left corner of R. The simulated annealing engine provides a very flexible framework to solve the constrained packing problem. However, it's runtime is long for problem instances with a large number of blocks and constraints. Therefore, in order to make use of this packing based approach effectively, some groupings between the memories is done first as a pre-processing step. First of all, all memories of the same type and of the same testing time (all A's, all B's or all C's) will be grouped together as one block and they are grouped in such a way to form a square-shaped rectangle as much as possible (packing for square-shaped rectangles are relatively easier). Then different types of memories belonging to the same testing time may be grouped together if their total area does not exceed a certain given threshold of the total area of the memory blocks. This threshold can be used to control the tradeoff between the optimality of the solution and the runtime. The smaller the threshold, less grouping will be done, the solution quality will be higher but the runtime will be longer. In all the following experiments, a threshold of 0.01 is used such that there will be at most 100 blocks in the problem instance after grouping.

3.1.2 A Fast Scheduling Heuristic

The pre-processing step used in the above packing-based scheduling strategy significantly reduces computational complexity, but it also greatly restrict the available solution space and hence may lead to excessive testing time. In this section, we present another heuristic that is both efficient in terms of runtime and effective in terms of testing time, based on the algorithm presented in [5]. In this heuristic, each memory test block is treated as a scheduling unit, and its data structure is as follows:

Data structure memory block

```
1. index;        /* The memory index */
2. type;         /* Memory block type, i.e., A, B or C*/
3. power;        /* Testing power */
4. time;         /* Testing time */
5. lowerLimit;   /* The earliest possible schedule time */
6. begin;        /* Schedule begin time */
7. end;          /* Schedule end time */
8. isScheduled;  /* Scheduled or not */
```

While the other variables are self-explanatory, the variable *lowerLimit* is utilized to meet the DRF interval T_{pause} constraint and is discussed in detail in the following algorithm.

The Algorithm *DRF_Flexible_Schedule* starts by initializing the *lowerLimit* for every memory test block in *MB*. For the blocks whose *type* is 'A', *lowerLimit* is initialized to be zero; while for the other memory blocks whose *type* is 'B' or 'C', they are initialized to be ∞. As a result, in the very beginning of the test scheduling process, only 'A' type of memory test blocks can be scheduled. Next, the current schedule begin time is initialized to zero, the currently available power constraint P_{avl} is initialized to P_{max} and the number of unscheduled memory blocks is initialized to the size of *MB* (lines 2). As long as there exist unscheduled blocks, the algorithm first tries to find the maximum memory test block that can be scheduled at *thisTime* (line 5). If such m_i exists, it will be scheduled by updating its $begin_i$, end_i and $isScheduled_i$ (line 7). Line 8 updates P_{avl} and $N_{unscheduled}$ after scheduling m_i. If m_i is of 'A' or 'B' type, we need to update the *lowerLimit* of the corresponding 'B' or 'C' block (line 9). If no such blocks can be found and at the same time $P_{avl} = P_{max}$, which means all the unscheduled blocks are of type 'B' or 'C', and their *lowerLimit* all exceed *thisTime*. In this time, we have to insert idle time into the test schedule and update *thisTime* accordingly (lines 11-12). If no such blocks can be found but $P_{avl} < P_{max}$, which means the current available test power is not enough, we will record this idle power P_{idle} (line 14), and branch to finish some currently scheduling blocks to release more available test power (lines 15-20). The algorithm then repeats the loop (lines 4-24) and ends only when all memory test blocks are scheduled.

3.2 Scheduling with Fixed DRF Pause Time

This section considers how to schedule e-SRAM tests with fixed DRF pause time $T_{AB} = T_{BC} = T_{pause}$. Because of this fixed wait period, whenever an 'A' type of memory test block m_i^A is scheduled, the schedule of its corresponding m_i^B and m_i^C are determined. Therefore, the three blocks cannot be treated as independent scheduling units and it is fairly difficult to keep track of the power profile during the scheduling process. As can be observed from Figure 3, the power profile after scheduling only 3 e-SRAMs is already quite complex.

Algorithm 1. *DRF_Flexible_Schedule*

INPUT: MB, T_{pause}, P_{max}
OUTPUT: e-SRAM test schedule

```
1. Initialize lowerLimit for MB;
2. Initialize thisTime = 0; P_avl = P_max; N_unscheduled = |MB|;
3. while (N_unscheduled != 0) {
4.     if (P_avl > 0) {
5.         find m_i with maximum test length subject to
.                  lowerLimit_i <= thisTime and P_i < P_avl;
6.         if (found) {
7.             schedule m_i;
8.             P_avl -= P_i; N_unscheduled --;
9.             if (type_i != C)   lowerlimit_{i+1} = end_i + T_pause;
10.        } else if (P_avl = P_max) {
11.            find m_i with minimum lowerlimit_i subject to
.                  isScheduled_i = false;
12.            thisTime = lowerlimit_i;
13.        } else {
14.            P_idle = P_avl; P_avl = 0; }
.      } else {
15.        P_avl += P_idle;
16.        find m_i with minimum end_i subject to
.                  isScheduled = true and begin_i = thisTime;
17.        thisTime = end_i;
18.        for(all m_j with end_j = thisTime) {
19.            P_avl += power_j;
20.            N_unscheduled --; }
.  }}
```

Figure 2. Pseudocode for e-SRAM test scheduling with flexible DRF pause time

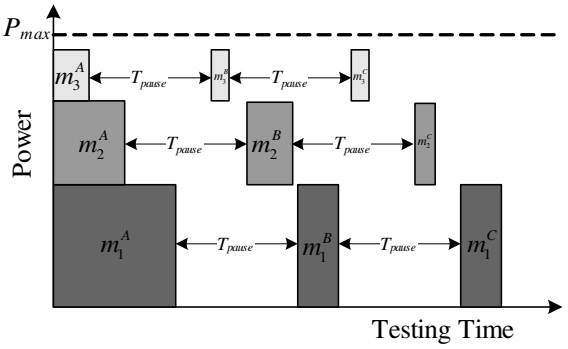

Figure 3. A scheduling example of 3 memory cores with fixed DRF pause time.

To reduce the complexity of this problem, instead of dynamically scheduling memory test blocks in between the DRF pause time T_{AB} and T_{BC}, we propose to group multiple e-SRAM tests statically before scheduling them. The main idea is to try to fill up the DRF pause time as much as possible during the initial grouping phase, and then treat the entire group of e-SRAM tests as a single scheduling unit. The pseudocode for this pre-processing procedure is shown in Figure 5.

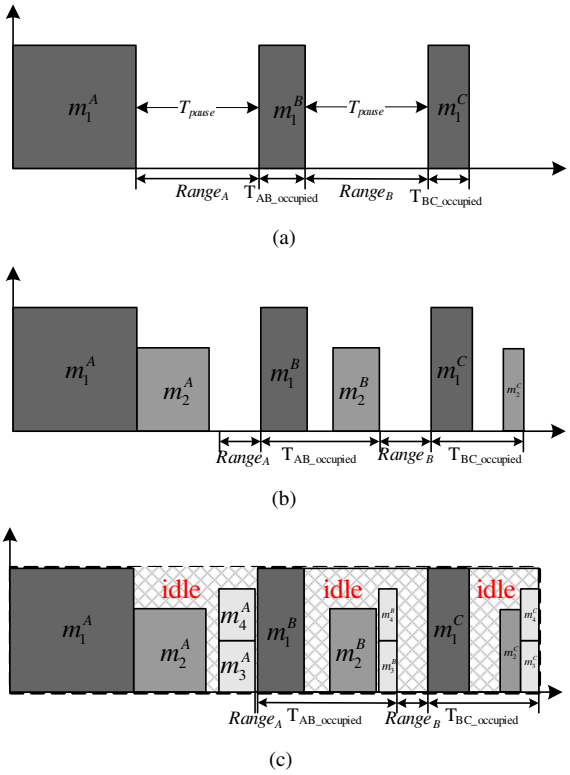

Figure 4. An e-SRAM tests grouping example.

The procedure starts by initializing the set of ungrouped e-SRAMs $M_{ungrouped}$, and the index i of the current memory group mg_i. Then we sort the memory tests in nonincreasing order (line 2). Inside the outer loop of the procedure, the first e-SRAM test (i.e., the memory test in $M_{ungrouped}$ with the maximum test power) is put in mg_i (line 4). When this is the last ungrouped e-SRAM, the procedure has already finished grouping and terminates (lines 5-7). Otherwise, we try to group other e-SRAM tests with their 'A' and 'B' blocks embedded in T_{AB_1} and T_{BC_1}. To check the feasibility, we define terms $Range_A$, $Range_B$, $T_{AB\_occupied}$, and $T_{BC\_occupied}$, which denotes the range to fit the e-SRAM's 'A' block, the range to fit the e-SRAM's 'B' block, the already occupied DRF pause time T_{AB}, and the already occupied DRF pause time T_{BC}, respectively. The physical meaning of the above terms can be easily observed from Figure 4. Whenever a e-SRAM test is grouped into mg_i, these values are updated (lines 8-9, 22-23). When $T_{AB\_occupied} > T_{pause}$ or $T_{BC\_occupied} > T_{pause}$, no more e-SRAM tests can be grouped into mg_i, and hence we proceed to a new memory test group (lines 11-13). Otherwise, we first try to find a compatible e-SRAM test with maximum P that is able to fit in without conflicts (see Figure 4). If such memory test exists, it is grouped (line 18). If the available test power allows and there are some other exactly the same type of memories, they are grouped with the same schedule (lines 19-21). The procedure halts when all e-SRAM tests are grouped. Figure 4 shows an example grouping process with four e-SRAM tests.

Each memory test group is treated as a single unit during the scheduling process, which, again, can be modeled as a rectangle (i.e., the dashed-rectangle as shown in Figure 4). A heuristic similar to Algorithm 1 without constraints can then be used for e-SRAM scheduling.

Algorithm 2. *Group_Tests*

INPUT: M, T_{pause}, P_{max}
OUTPUT: MG

1. initialize $M_{ungrouped} = M$; $i = 1$;
2. sort $M_{ungrouped}$ such that $P_1 \geq P_2 \geq ... \geq P_{|M|}$;
3. **while** $M_{ungrouped} \neq \phi$ {
4. $mg_i = \{m_1\}$;
5. **if** ($|M_{ungrouped}| = 1$) {
6. $M_{ungrouped} = M_{ungrouped} \backslash mg_i$;
7. break;
. }
8. $T_{AB\_occupied} = T_{B\_1}$; $T_{BC\_occupied} = T_{C_1}$;
9. determine $Range_A$ and $Range_B$;
10. **while** (**true**) {
11. **if** ($T_{AB\_occupied} > T_{pause}$ || $T_{BC\_occupied} > T_{pause}$) {
12. $M_{ungrouped} = M_{ungrouped} \backslash mg_i$;
13. i++; **break**;
14. } **else** {
15. $P_{avl} = P_1$;
16. find compatible m^j with maximum test power;
17. **if** (found) {
18. $mg_i = mg_i \cup \{m_j\}$;$P_{avl} -= P_j$;
19. **while** ($P_{avl} > P_j$ && $m_{j+1} = m_j$) {
20. $mg_i = mg_i \cup \{m_{j+1}\}$;
21. $P_{avl} -= P_j$; j++;
. }
22. update $Range_A$ and $Range_B$;
23. update $T_{AB\_occupied}$ and $T_{BC\_occupied}$;
24. } **else** {
25. $M_{ungrouped} = M_{ungrouped} \backslash mg_i$;
26. i++; **break**; }
. }}}

Figure 5. Procedure for grouping e-SRAM tests.

4 Experimental Results

To show the benefits of the proposed retention-aware test scheduling techniques, we constructed four test cases as follows:

1. 500 instances of 64*256 and 500 instances of 512*8 e-SRAMs, in total 1000 e-SRAMs and about 10Mb;
2. 10 instances of 16k*32 and 5 instances of 64k*16 e-SRAMs, in total 15 e-SRAMs and about 10Mb;
3. 37 mixed types of e-SRAMs, in total 418 e-SRAMs and about 5Mb;
4. a combination of the above, in total 1433 e-SRAMs and about 25Mb.

The detailed configurations for the test cases are shown in Table 1, in which N, P, T_A, T_B, and T_C denote the number of each type of e-SRAMs, the test power consumption, the testing time for block A, B, and C, respectively. Note, we assume all e-SRAMs are tested in 100MHz when we acquire P from our memory compiler. Although different e-SRAMs may be tested in distinct frequencies in practice, this would not affect the effectiveness of our approach.

Tables 2-5 compare the total e-SRAM testing time using different test scheduling schemes with the variation of the DRF pause time T_{pause} and the given power constraint P_{max}. T_{reg}, $T_{packing}$, T_{flex}, and

	e-SRAM	N	P (μW)	T_A (cc)	T_B (cc)	T_C (cc)
Test case 1	64×256	500	5914	16896	1408	806
	512×8	500	1475	135168	11264	5734
Test case 2	$16k \times 32$	10	12894	4325376	360448	180326
	$64k \times 16$	5	46224	17301504	1441792	720998
Test case 3	$32k \times 16$	1	23904	8650752	720896	360550
	$8k \times 32$	2	7666	2162688	180224	90214
	$8k \times 8$	3	6930	2162688	180224	90214
	$4k \times 16$	3	4374	1081344	90112	45158
	...	...	...	...	...	...
	128×66	1	3215	33792	2816	1510
	16×8	10	545	4224	352	278
	8×8	8	503	2112	176	190

Table 1. e-SRAM configurations of the experimental test cases.

T_{fixed} represents the testing time using the regular "single-rectangle" test power model, the testing time using packing-based algorithm shown in Section 3.1.1 when T_{pause} can be varied, the testing time using the fast heuristic shown in Section 3.1.2 when T_{pause} can be varied, and the testing time using the grouping-based strategy shown in Section 3.2, respectively. They are all in unit clock cycles. Since we assume the e-SRAMs are tested in $100MHz$, T_{pause} varies from $500\mu s$ to $100ms$ in our experiments. ΔT_{flex} and ΔT_{fixed} are calculated as $\Delta T_{flex} = \frac{T_{flex} - T_{reg}}{T_{reg}} \times 100\%$ and $\Delta T_{fixed} = \frac{T_{fixed} - T_{reg}}{T_{reg}} \times 100\%$, which shows the benefit of the proposed "retention-aware" test scheduling algorithms for variable and fixed DRF pause time, respectively.

From these tables, we can observe T_{flex} (with computational time within a second) is better than $T_{packing}$ (with computational time in minutes) in all cases. This is mainly because, to reduce runtime, the packing-based scheduling strategy group many e-SRAM tests first. This limits the solution space for Problem $P_{drf-opt}$, which, however, can be explored in the fast heuristic presented in 3.1.2.

It can be also seen from Tables 1-4 that the total e-SRAM testing time is reduced in most cases with the proposed "retention-aware" test scheduling techniques, for both cases with flexible DRF pause time and cases with fixed DRF pause time. The reduction is especially significant when T_{pause} is large. This is expected because more e-SRAMs can fit in the DRF pause time during the scheduling process in such cases. While these times with idle test power consumption are wasted in traditional single-rectangle model. We can also observe that the savings in testing time is usually larger when P_{max} is smaller. This is also expected because the "retention-aware" test power model is not very effective when the power constraint is relaxed. For example, the total test power consumption of all e-SRAMs in test case 3 is less than $800mW$. When $P_{max} = 500mW$, similar to using single-rectangle test power model, the retention-aware scheduling approach also wastes lots of idle power in the final schedule. Therefore, the savings is not very large.

In a few cases, the proposed method leads to a slightly longer testing time (e.g., when DRF pause time is fixed, $P_{max} = 200mW$ and $T_{pause} \geq 5M$ in test case 2). This is due to the fact that test case 2 has only 15 large e-SRAMs, and when $T_{pause} \geq 5M$ several e-SRAMs can be grouped into one scheduling unit[1]. As shown in Figure 3, the grouping happens in the horizonal direction and the testing time of the group becomes larger than the testing time of each individual e-SRAM. By using the single-rectangle power model, however, these e-SRAMs may be able to be scheduled in the vertical direction and hence reduced testing time can be achieved. Nevertheless, this situation rarely happens when the number of e-SRAMs is large and/or the sizes of e-SRAMs are small. There are also other few cases that $T_{reg} < T_{flex}$ and we attribute them to the fact that the fast heuristic explores only part of the solution space.

[1] when $T_{pause} < 5M$, e-SRAMs in test case 2 cannot be grouped and the scheduling process is exactly the same as the single-rectangle power model.

5 Conclusion

We proposed retention-aware test scheduling techniques in this paper for testing e-SRAMs when DRFs are considered, for both cases where the DRF pause time is fixed and cases where it can be varied. Experimental results show that the proposed approach can significantly reduce e-SRAM testing time, especially when the power constraint is tight and/or the DRF pause time is large. As stressed in [2], the DRF pause time can be as large as up to hundreds of *ms*, the proposed approach is able to greatly reduce the e-SRAM test cost.

References

[1] B. Wang, J. Yang, Y. Wu, and A. Ivanov. A Retention-Aware Test Power Model for Embedded SRAM. In *Proc. IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, pages 1180–1183, January 2005.

[2] R. Aitken, N. Dogra, D. Gandhi, and S. Becker. Redundancy, Repair, and Test Features of a 90nm Embedded SRAM Generator. In *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 467–474, 2003.

[3] A. Bommireddy, J. Khare, S. Shaikh, and S.-T. Su. Test and Debug of Networking SoCs - A Case Study. In *Proc. IEEE VLSI Test Symposium (VTS)*, pages 121–126, 2000.

[4] R. M. Chou, K. K. Saluja, and V. D. Agrawal. Scheduling tests for VLSI systems under power constraints. *IEEE Transactions on VLSI Systems*, 5(2):175–184, June 1997.

[5] B. H. Fang, Q. Xu, and N. Nicolici. Hardware/Software Co-Testing of Embedded Memories in Complex SOCs. In *Proc. International Conference on Computer-Aided Design (ICCAD)*, pages 599–605, 2003.

[6] S. K. Goel and E. J. Marinissen. Effective and Efficient Test Architecture Design for SOCs, Oct. 2002.

[7] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Integrated Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs, June 2002.

[8] W. B. Jone, D. C. Huang, S. C. Wu, and K. J. Lee. An Efficient BIST Method for Distributed Small Buffers. *IEEE Transactions on VLSI Systems*, 10(4):512–515, August 2002.

[9] E. Larsson, J. Pouget, and Z. Peng. Defect-Aware SOC Test Scheduling. In *Proc. IEEE European Test Workshop (ETW)*, pages 359–364, 2004.

[10] A. Meixner and J. Banik. Weak Write Test Mode: an SRAM Cell Stability Design for Test Technique. In *Proc. IEEE International Test Conference (ITC)*, pages 1043–1052, 1997.

[11] B. Nadeau-Dostie, A. Silburt, and V. K. Agrawal. Serial Interfacing Technique for Embedded Memory Testing. *IEEE Design & Test of Computers*, 7(2):52–63, April 1990.

[12] C.-W. Wang et al. Test Scheduling of BISTed Memory Cores for SoC. In *Proc. IEEE Asian Test Symposium (ATS)*, pages 356–361, November 2002.

[13] Q. Xu and N. Nicolici. Multi-Frequency Test Access Mechanism Design for Modular SOC Testing, Nov. 2004.

[14] J. Yang, B. Wang, Y. Wu, and A. Ivanov. Fast Detection of Data Retention Faults and Other SRAM Cell Open Defects. *IEEE Transactions on Computer-Aided Design*, to appear.

[15] D. H. Yoon, H. S. Kim, and S. Kang. Dynamic Power Supply Current Test for CMOS SRAM. In *Proc. International Conference on Computer-Aided Design (ICCAD)*, pages 399–402, 2001.

[16] E. F. Young, C. C. Chu, and M. Ho. Placement constraints in floorplan design. *IEEE Transactions on VLSI Systems*, 12(7):735–745, July 2004.

[17] Y. Zorian. A Distributed BIST Control Scheme for Complex VLSI Devices. In *Digest of Papers, Eleventh Annual 1993 IEEE VLSI Test Symposium*, pages 4–9, Apr. 1993.

	Test Case 1											
	$P_{max} = 60mW$						$P_{max} = 100mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	9114548	3151270	2868234	-68.53	5778228	-36.60	5695792	2340260	1735692	-69.53	3546348	-37.74
100k	15400602	3201270	2918836	-81.05	7006878	-54.50	9609738	2390260	1759736	-81.69	4335278	-54.89
500k	65800602	3960890	3536614	-94.63	7691198	-88.31	40809738	3149890	2539852	-93.78	4624434	-88.67
1M	128800602	4960890	4536614	-96.48	9130290	-92.91	79809738	4149890	3539852	-95.56	6114316	-92.34
5M	632800602	12960900	12536614	-98.02	15018214	-97.63	391809738	12149900	11539750	-97.05	15018214	-96.17
10M	1262800602	22960900	22536614	-98.22	30019430	-97.62	781809738	22149900	21539750	-97.24	30019430	-96.16
	$P_{max} = 200mW$						$P_{max} = 500mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	2795314	1954380	869660	-68.89	1773174	-36.57	1099882	1545460	406546	-63.04	710872	-35.37
100k	4695314	1645460	962636	-79.50	2270858	-51.64	1885936	1645460	506546	-73.14	959346	-49.13
500k	19895314	2445460	1762636	-91.14	3066764	-84.59	8285936	2445460	1306342	-84.23	1557670	-81.20
1M	38895314	3445460	2762534	-92.90	3098342	-92.03	16285936	3445460	2306342	-85.84	3098342	-80.98
5M	190895314	11445500	10762534	-94.36	15018214	-92.13	80285936	11445500	10306342	-87.16	15018214	-81.29
10M	380895314	21445500	20762534	-94.55	30019430	-92.12	160285936	21445500	20306342	-87.33	30019430	-81.27

Table 2. Testing time comparison for test case 1

	Test Case 2											
	$P_{max} = 60mW$						$P_{max} = 100mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	97821470	112821000	97321470	-0.51	97821470	0	58692882	70288300	55689624	-5.12	58692882	0
100k	98321470	112821000	97321470	-1.02	98321470	0	58992882	70288300	55689624	-5.60	58992882	0
500k	102321470	112821000	97321470	-4.89	102321470	0	61392882	70288300	55829074	-9.06	61392882	0
1M	107321470	112821000	97321470	-9.32	107321470	0	64392882	70288300	56689420	-11.96	64392882	0
5M	148661500	116379000	99437478	-33.11	160838270	8.19	103259032	73846500	64067302	-37.95	96502962	-6.54
10M	222187620	122007000	108670310	-51.09	232465150	4.63	143259032	83079000	74067302	-48.30	139479090	-2.64
	$P_{max} = 200mW$						$P_{max} = 500mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	39128588	46187700	36865798	-5.78	39128588	0	19564294	24610900	20005068	2.25	19564294	0
100k	39328588	46237700	36965798	-6.01	39328588	0	19664294	24710900	20005068	1.73	19664294	0
500k	40928588	46637700	37765798	-7.73	40928588	0	20464294	25510900	20464294	0	20464294	0
1M	42928588	47137700	38765798	-9.70	42928588	0	21464294	26510900	21464294	0	21464294	0
5M	58928588	51812400	46765798	-20.64	64335308	9.18	29464294	34510900	29464294	0	32167654	9.18
10M	78928588	61812400	56765798	-28.08	92986060	17.81	39464294	44510900	39464294	0	46493030	17.81

Table 3. Testing time comparison for test case 2

	Test Case 3											
	$P_{max} = 60mW$						$P_{max} = 100mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	9832198	10257360	9832198	0	9832198	0	9832198	9832200	9872412	0.41	9832198	0
100k	9932198	10168740	9932198	0	9932198	0	9932198	10610640	9932198	0	9932198	0
500k	19845932	10732200	10732198	-45.92	10732198	-45.92	11946457	11654380	10732198	-10.16	10732198	-10.16
1M	32374618	12392580	11732198	-63.76	12277186	-62.08	19835346	12344320	11732198	-40.85	11732198	-40.85
5M	134827711	23252400	19732198	-85.36	23650983	-82.46	82783700	21061200	19732198	-76.16	23650983	-71.43
10M	264827711	32190600	29732198	-88.77	38655739	-85.40	162783700	32850000	29732198	-81.74	38655739	-76.25
	$P_{max} = 200mW$						$P_{max} = 500mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	9832198	10340040	10002636	1.73	9832198	0	9832198	10121620	10002636	1.73	9832198	0
100k	9932198	10168740	10012422	0.81	9932198	0	9932198	10132160	10012422	0.81	9932198	0
500k	10732198	11002540	10732198	0	10732198	0	10732198	11002540	10732198	0	10732198	0
1M	11732198	11744900	11732198	0	11732198	0	11732198	11833860	11732198	0	11732198	0
5M	42463843	19732200	19732198	-53.53	23650983	-44.30	20076869	19732200	19732198	-1.72	23650983	17.80
10M	82463843	30002600	29732198	-63.95	38655739	-53.12	40076869	29732200	29732198	-25.81	38655739	-3.55

Table 4. Testing time comparison for test case 3

	Test Case 4											
	$P_{max} = 60mW$						$P_{max} = 100mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	107653668	129819000	107103668	-0.51	107653668	0	63659032	79170800	58392882	-8.27	63659032	0
100k	112463987	130037000	107153668	-4.72	108253668	-3.74	67345682	78933800	58394209	-13.29	64059032	-4.88
500k	176764981	129961000	107553668	-39.15	113053668	-36.04	107396086	80139300	58355585	-45.66	67259032	-37.37
1M	258026656	130542000	108053668	-58.12	119053668	-53.86	156979271	80120800	58864114	-62.50	71718391	-54.31
5M	909248192	130842000	109890674	-87.91	185165072	-79.64	560096335	89052600	64067302	-88.56	103260488	-81.56
10M	1725540774	137209000	116239718	-93.26	236508457	-86.29	1060515913	90178300	74067302	-93.02	141904895	-86.62
	$P_{max} = 200mW$						$P_{max} = 500mW$					
T_{pause}	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)	T_{reg} (cc)	$T_{packing}$ (cc)	T_{flex} (cc)	ΔT_{flex} (%)	T_{fixed} (cc)	ΔT_{fixed} (%)
50k	39128588	49206700	37847244	-3.27	39128588	0	19564294	27494700	19564294	0	19564294	0
100k	39328588	50167900	37847244	-3.77	39328588	0	19664294	27594700	19664294	0	19664294	0
500k	53371616	50444000	37986694	-28.83	40928588	-23.31	21483404	28394700	20685190	-3.72	20464294	-4.74
1M	78037504	51967900	38765798	-50.32	42928588	-44.99	31559844	29394700	21464294	-31.99	21464294	-31.99
5M	276225151	57333200	46765798	-83.07	64603720	-76.61	111159194	37394700	29464294	-73.49	32303628	-70.94
10M	525644342	67333200	56765798	-89.20	94603105	-82.00	219778181	47394700	39464294	-82.04	47301790	-78.48

Table 5. Testing time comparison for test case 4