

# Test scheduling for built-in self-tested embedded SRAMs with data retention faults

Q. Xu, B. Wang, A. Ivanov and F.Y. Young

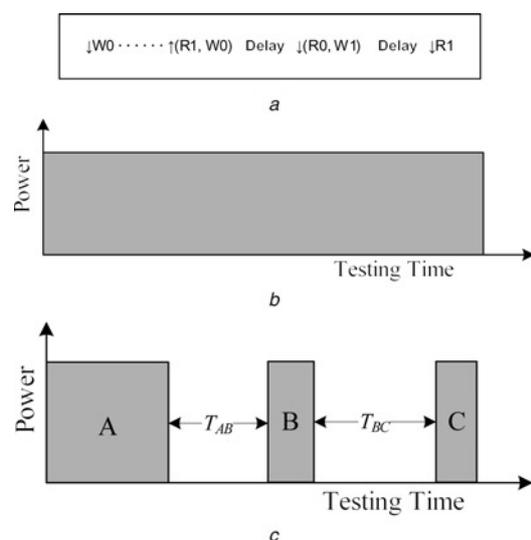
**Abstract:** The test scheduling problem for built-in self-tested embedded SRAMs (e-SRAMs) when data retention faults (DRFs) are considered is addressed here. We proposed a ‘retention-aware’ test power model by taking advantage of the fact that there is near-zero test power during the pause time for testing DRFs. The proposed test scheduling algorithm then utilises this new test power model to minimise the total testing time of e-SRAMs while not violating given power constraints, by scheduling some e-SRAM tests during the pause time of DRF tests. Without losing generality, we consider both cases where the pause time for DRFs is fixed and cases where it can be varied. Experimental results show that the proposed ‘retention-aware’ test power model and the corresponding test scheduling algorithm can reduce the testing time of e-SRAMs significantly with negligible computational time.

## 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 64 bits to larger caches with sizes of hundreds of kilobits or even megabits [1]. 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 for all possible faults becomes a major challenge for the SoC system integrators [2]. 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 [3]. Therefore efficient power-constrained test scheduling techniques (e.g. [4]) play a key role in reducing e-SRAM test cost.

Most prior work in test scheduling assumes a constant power consumption during the entire test process. As shown in Fig. 1b, an e-SRAM test can be represented by a rectangle, where its width denotes the testing time and its height denotes the test power. Although simple and effective for logic testing, this model is overly pessimistic for e-SRAM testing when data retention faults (DRFs) are considered [5]. DRFs model the defects in SRAM bit cells that fails to retain a stored logic value. The most common test method for DRFs is simply loading a known value

into the cell and waiting for a period of time (up to hundreds of milliseconds [1]), and then reading it out, as shown in Fig. 1a. 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, we propose a ‘retention-aware’ test power model for 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  $T_{BC}$  corresponding to the DRF pause times, as shown in Fig. 1c. Based on this new test power model, we present an efficient and effective test scheduling algorithm that minimises the total testing time of e-SRAMs under given power constraints, by scheduling some e-SRAM tests during the pause time of DRF tests. Without loss of generality, we consider both cases where the pause time for DRF tests is fixed and cases where it can be varied.



**Fig. 1** Test power model for e-SRAMs with data retention faults  
 a March algorithm for testing DRFs  
 b Traditional test power model  
 c Retention-aware test power model

© The Institution of Engineering and Technology 2007  
 doi:10.1049/iet-cdt:20060128

Paper first received 28th August 2006 and in revised form 3rd January 2007  
 Q. Xu and F.Y. Young are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong  
 B. Wang is with AMD, AMD One Place, Sunnyvale, CA, USA  
 A. Ivanov is with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, B.C., Canada, V6T 1Z4  
 E-mail: qxu@cse.cuhk.edu.hk

Experimental results show that our approach significantly reduces the total testing time under given power constraints.

## 2 Prior work and motivation

### 2.1 Related work in DRF tests

From the functional point of view, e-SRAM data retention faults behave as that the e-SRAM cell cannot retain a logic 1/0 after a certain amount of time [6]. From the defect point of view, DRFs are usually caused by a defective source, drain or gate open of the pull-up transistor of the e-SRAM cell or by a defective power or ground path. Based on the above, there are mainly two types of DRF testing methodologies: (i) functional-based, that is, introducing pass time in March tests [7, 8] and (ii) defect-based, that is embedding various design for test (DFT) circuitries to identify DRFs in a short time [1, 9–17].

DFT-based DRF testing methods embed dedicated circuitries in e-SRAM cells and/or their peripherals and detect DRF-related defects with specially designed operations. Among the previous work [1, 9–17] weak-write method [15] has excellent DRF detectability due to the fact that the weak write value can be programmable on the fly, while pre-discharge write method proposed in [16] leads to the most significant test time savings (close to zero) and has the additional benefit of at-speed testability that is more important for deep sub-micron technology [18]. Although effective on detecting DRFs, the above DFT techniques require more design efforts and also often come with high hardware and/or performance overhead. Moreover, since the DFT circuitries are implemented at transistor level, these techniques are technology-dependent and hence requires verification at every technology node for all corner cases, which may significantly increase time-to-market. Because of the above reasons, most memory compilers supplied by memory vendors today do not provide the feature to apply the above DFT techniques.

Therefore we consider the case that DRF tests are applied in the traditional functional-based methods. As shown in Fig. 1a, all the e-SRAM cells are firstly initialised as a logic value 1/0. After that, the e-SRAM under test is disabled, that is, no read or write operation is conducted, for a pre-defined pause time (up to several hundred milliseconds) before reading the values out. To reduce DRF testing time, Wang *et al.* [8] proposed to reuse the initialisation time of the neighbourhood cells which are not on the same row as the cells under test as part of the pause time. This technique, however, is only effective for large e-SRAMs. As discussed in [1], retention testing needs to consider the slow process corner case, whose leakage (responsible for the loss of the stored logic value) actually slows from 130 to 90 nm. 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 when applying pause test, especially for small e-SRAMs. In fact it is the above observation that motivates this work on how to effectively and efficiently utilise the pause time for DRF tests in test scheduling process.

### 2.2 Related work in test scheduling

Test scheduling is the process that allocates test resources (e.g. test bus lines or BIST engines) to cores at different time in order to minimise the overall testing time, while at the same time satisfying the given constraints [19]. Various

constraints need to be considered during test scheduling, but probably the most important one is the test power constraint. That is, testing more cores in parallel usually result in reduced testing time; however, it will also increase the test power, which may lead to destructive testing [20].

Many test scheduling techniques have been proposed in the literature [3, 21–25] (only name a few). In particular, [21, 23] 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 optimisation 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.* [4] 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 testing process and the computational time is quite high when the number of memory cores is large. Fang *et al.* [26] presented an effective and efficient power-constrained test scheduling heuristic for their hardware/software co-testing methodology. None of the above work considered the special features of DRF pause tests.

### 2.3 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 [27] or may not [28] be able to be tested in parallel. Since at-speed testability for e-SRAMs becomes more important with the ever increasing operational frequency, most of the current system integrators prefer to design unique BIST engine for each and every e-SRAM, at least for the timing-critical portion of the BIST engine, for example, the address generator, the control signal generator and the comparator [29]. As a result, 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 easily be generalised to the 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’, that is, 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 hard-wired, however, the pause time is a pre-determined fixed value. Without loss of generality, we consider both cases.

## 3 Retention-aware test scheduling

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

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

- the total number of e-SRAMs  $N_m$ ;
- the maximum allowed test power  $P_{\text{max}}$ ;

- 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 blocks A, B and C;
- the minimum pause time for testing DRFs  $T_{\text{pause}}$ ;

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

### 3.1 Scheduling with flexible DRF pause time

**3.1.1 Packing-based scheduling strategy:** Since each e-SRAM test  $i$  can be modelled by three rectangular blocks  $A_i$ ,  $B_i$  and  $C_i$  (see Fig. 1c), our objective can be seen as to pack all the rectangles  $A_i$ ,  $B_i$ , and  $C_i$  ( $i = 1, \dots, N_m$ ) into a rectangular region of height not exceeding  $P_{\text{max}}$  and of a minimised width such that for every e-SRAM  $i$ , the separation between  $A_i$  and  $B_i$  and the separation between  $B_i$  and  $C_i$  are at least  $T_{\text{pause}}$ . This is a typical constrained rectangle packing problem and can be modelled and solved by using a SA approach as described in [30], borrowed from the floorplanning literature.

In this approach, SA is used to search for a good packing satisfying a given set of general placement constraints. In each annealing step, a candidate packing solution  $S$  represented by a sequence pair [31], is evaluated. A pair of constraint graphs,  $G_h$  and  $G_v$ , are constructed according to the sequence pair to realise a packing from its representation. To impose a ‘minimum separation’ constraint between two blocks, for example, between  $A_i$  and  $B_i$  (or between  $B_i$  and  $C_i$ ), an edge of weight  $T_{\text{pause}}$  will be inserted into the horizontal constraint graph from  $A_i$  to  $B_i$  (from  $B_i$  to  $C_i$ , respectively). According to the definition of horizontal constraint graph, an edge  $e(v_i, v_j)$  from  $v_i$  to  $v_j$  of weight  $w$  means that the block represented by  $v_j$  must be placed at a distance of at least  $w$  units on the right of the block represented by  $v_i$ . After adding all these additional constraint edges, a single source shortest path algorithm can be performed on the constraint graphs to find out the location of each block. The resulting packing will automatically have all the minimum separation constraints satisfied. It may happen that a positive cycle is formed in the horizontal constraint graph after adding those additional constraint edges and the single source shortest path algorithm will be failed, implying that the current candidate floorplan solution is infeasible to satisfy all the minimum separation constraints. In this case, we will remove all the additional constraint edges and simply pack the blocks according to the sequence pair. A penalty term will be included in the cost function to penalise the violated constraints. The cost function of a candidate solution  $S$  used in the annealing process is as follows

$$\text{cost}(S) = \text{area}(S) + \alpha \times \text{Penalty}_1(S) + \beta \times \text{Penalty}_2(S)$$

where  $\alpha$  and  $\beta$  are weights,  $\text{area}(S)$  is the area of  $S$  and is computed as  $P_{\text{max}} \times \text{width}(S)$ ,  $\text{Penalty}_1(S)$  is the penalty for exceeding the maximum allowed test power  $P_{\text{max}}$  and  $\text{Penalty}_2(S)$  is the penalty for violating the minimum separation constraints.  $\text{Penalty}_1(S)$  and  $\text{Penalty}_2(S)$  are computed as

$$\begin{aligned} \text{Penalty}_1(S) &= (\max\{0, \text{height}(S) - P_{\text{max}}\})^2 \\ \text{Penalty}_2(S) &= \sum_{i=1}^n (\max\{0, T_{\text{pause}} - (x(B_i) - x(A_i))\})^2 \\ &\quad + \sum_{i=1}^n (\max\{0, T_{\text{pause}} - (x(C_i) - x(B_i))\})^2 \end{aligned}$$

where  $x(R)$  of a rectangular block  $R$  is the  $x$ -coordinate of the lower left corner of  $R$ . The SA engine provides a very flexible framework to solve this constrained block packing problem. However, its runtime is very long for problem instances with a large number of blocks and constraints. To make use of this packing-based approach, some groupings between the memories will be done as a pre-processing step. First of all, some memories of the same type and belonging to the same testing period, that is, period A, B or C, 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 of square-shaped rectangles are relatively easier). For example, if there are 12 BISTed e-SRAM  $i$ , with test power consumption  $P^i = 18$ , testing time  $T_A^i = 60$ ,  $T_B^i = 12$  and  $T_C^i = 12$ , these 12  $A$  blocks will be grouped together in the form of  $6 \times 2$  since the dimensions of this  $6 \times 2$  combined block will be  $108 \times 120$  ( $6 \times P^i = 108$ ), which is a possible shape closest to a square. Similarly, we do such pre-processing for the A, B and C blocks of each BISTed e-SRAM  $i$  to reduce the problem size. Memories of the same type and belonging to the same testing period will be grouped together if their total area does not exceed a certain threshold of the total area of all the memory blocks. This threshold is set by the user to control the trade-off between the optimality of the solution and the runtime. The smaller the threshold, less grouping will be done, and the solution quality will be higher but the runtime will be longer. In all the following experiments, a threshold of 0.01 is used, that means there will be at most 100 blocks in the problem instance after grouping.

**3.1.2 Fast scheduling heuristic:** The pre-processing step used in the above packing-based scheduling strategy significantly reduces computational complexity, but it also greatly restricts 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 [26]. In this heuristic (as shown in Fig. 2), every e-SRAM test block (i.e.  $A_i$ ,  $B_i$  or  $C_i$ ) is treated as a scheduling unit, and its data structure is as follows

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

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

The algorithm *DRF\_Flexible\_Schedule* takes the set of memory test blocks MB,  $T_{\text{pause}}$  and  $P_{\text{max}}$  as inputs and outputs the test schedule of all e-SRAMs. It starts by initialising the *lowerLimit* for every memory test block in MB. For the blocks whose *type* is ‘A’, *lowerLimit* is initialised to be zero; while for the other memory blocks whose *type* is ‘B’ or ‘C’, they are initialised to be  $\infty$ . As a result,



**Table 1: e-SRAM configurations of the experimental test cases**

Test case	e-SRAM	$N$	$P, \mu W$	$T_A, cc$	$T_B, cc$	$T_C, cc$
1	$64 \times 256$	500	5914	16 896	1408	806
	$512 \times 8$	500	1475	135 168	11 264	5734
2	$16 k \times 32$	10	12 894	4 325 376	360 448	180 326
	$64 k \times 16$	5	46 224	17 301 504	1 441 792	720 998
3	$32 k \times 16$	1	23 904	8 650 752	720 8963	360 550
	$8 k \times 32$	2	7666	2 162 688	180 224	90 214
	$8 k \times 8$	3	6930	2 162 688	180 224	90 214
	$4 k \times 16$	3	4374	1 081 344	90 112	45 158
	...	...	...	...	...	...
	$128 \times 66$	1	3215	33 792	2816	1510
	$16 \times 8$	10	545	4224	352	278
	$8 \times 8$	8	503	2112	176	190

cases that the BIST engines are hard-wired and the pause time simply cannot be changed. As a result, in this section, we consider how to schedule e-SRAM tests with fixed DRF pause time

$$T_{AB} = T_{BC} = T_{\text{pause}}$$

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 already. Therefore the three blocks cannot be treated as independent scheduling units and have to be considered as a whole. At the same time, it is fairly difficult to keep track of the power profile during the scheduling process. For example, as can be observed from Fig. 3, the power profile after scheduling only three e-SRAMs is already quite complex. 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

**Table 2: Testing time comparison for test case 1**

Test case 1						
$P_{\text{max}} = 60 \text{ mW}$						
$T_{\text{pause}}, cc$	$T_{\text{reg}}, cc$	$T_{\text{packing}}, cc$	$T_{\text{flex}}, cc$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, cc$	$\Delta T_{\text{fixed}}, \%$
50 k	9 114 548	3 151 270	2 868 234	-68.53	5 778 228	-36.60
100 k	15 400 602	3 201 270	2 918 836	-81.05	7 006 878	-54.50
500 k	65 800 602	3 960 890	3 536 614	-94.63	7 691 198	-88.31
1 M	128 800 602	4 960 890	4 536 614	-96.48	9 130 290	-92.91
5 M	632 800 602	12 960 900	12 536 614	-98.02	15 018 214	-97.63
10 M	1 262 800 602	22 960 900	22 536 614	-98.22	30 019 430	-97.62
$P_{\text{max}} = 100 \text{ mW}$						
$T_{\text{pause}}, cc$	$T_{\text{reg}}, cc$	$T_{\text{packing}}, cc$	$T_{\text{flex}}, cc$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, cc$	$\Delta T_{\text{fixed}}, \%$
50 k	5 695 792	2 340 260	1 735 692	-69.53	3 546 348	-37.74
100 k	9 609 738	2 390 260	1 759 736	-81.69	4 335 278	-54.89
500 k	40 809 738	3 149 890	2 539 852	-93.78	4 624 434	-88.67
1 M	79 809 738	4 149 890	3 539 852	-95.56	6 114 316	-92.34
5 M	391 809 738	12 149 900	11 539 750	-97.05	15 018 214	-96.17
10 M	781 809 738	22 149 900	21 539 750	-97.24	30 019 430	-96.16
$P_{\text{max}} = 200 \text{ mW}$						
$T_{\text{pause}}, cc$	$T_{\text{reg}}, cc$	$T_{\text{packing}}, cc$	$T_{\text{flex}}, cc$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, cc$	$\Delta T_{\text{fixed}}, \%$
50 k	2 795 314	1 954 380	8 69 660	-68.89	1 773 174	-36.57
100 k	4 695 314	1 645 460	9 62 636	-79.50	2 270 858	-51.64
500 k	19 895 314	2 445 460	1 762 636	-91.14	3 066 764	-84.59
1 M	38 895 314	3 445 460	2 762 534	-92.90	3 098 342	-92.03
5 M	190 895 314	11 445 500	10 762 534	-94.36	15 018 214	-92.13
10 M	380 895 314	21 445 500	20 762 534	-94.55	30 019 430	-92.12
$P_{\text{max}} = 500 \text{ mW}$						
$T_{\text{pause}}, cc$	$T_{\text{reg}}, cc$	$T_{\text{packing}}, cc$	$T_{\text{flex}}, cc$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, cc$	$\Delta T_{\text{fixed}}, \%$
50 k	1 099 882	1 545 460	4 06 546	-63.04	7 10 872	-35.37
100 k	1 885 936	1 645 460	5 06 546	-73.14	9 59 346	-49.13
500 k	8 285 936	2 445 460	1 306 342	-84.23	1 557 670	-81.20
1 M	16 285 936	3 445 460	2 306 342	-85.84	3 098 342	-80.98
5 M	80 285 936	11 445 500	10 306 342	-87.16	15 018 214	-81.29
10 M	160 285 936	21 445 500	20 306 342	-87.33	30 019 430	-81.27

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 Fig. 4.

The procedure *Group\_Tests* takes the set of e-SRAMs  $M$ ,  $T_{\text{pause}}$  and  $P_{\text{max}}$  as inputs and outputs the e-SRAM test groups MG. It starts by initialising the set of ungrouped e-SRAMs  $M_{\text{ungrouped}}$ , and the index  $i$  of the current memory group  $mg_i$ . Then we sort the memory tests in non-increasing order in terms of their power consumption (line 2). Inside the outer loop of the procedure, the first e-SRAM test (i.e. the memory test in  $M_{\text{ungrouped}}$  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  $T_{\text{AB}_1}$  and  $T_{\text{BC}_1}$ . To check the feasibility, we define terms  $\text{Range}_A$ ,  $\text{Range}_B$ ,  $T_{\text{AB\_occupied}}$ , and  $T_{\text{BC\_occupied}}$ , 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  $T_{\text{AB}}$ , and the already occupied DRF pause time  $T_{\text{BC}}$ , respectively. The physical meaning of the above terms can easily be observed from Fig. 5. Whenever an e-SRAM test is grouped into  $mg_i$ , these values are updated (lines 8–9, 22–23). When  $T_{\text{AB\_occupied}} > T_{\text{pause}}$  or  $T_{\text{BC\_occupied}} > T_{\text{pause}}$ , no more e-SRAM tests can be grouped into  $mg_i$ , and hence we proceed to generate a new memory test group (lines 11–13). Otherwise, we first try to find a compatible e-SRAM test with maximum power consumption that is able to fit in without conflicts (see Fig. 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. Fig. 5 shows an example grouping process with four e-SRAM tests.

After the e-SRAM test groups are generated with the above procedure, each group is treated as a single unit during the test scheduling process, which, again, can be modelled as a rectangle (i.e. the dashed-rectangle as

**Table 3: Testing time comparison for test case 2**

Test case 2						
$P_{\text{max}} = 60 \text{ mW}$						
$T_{\text{pause, CC}}$	$T_{\text{reg, CC}}$	$T_{\text{packing, CC}}$	$T_{\text{flex, CC}}$	$\Delta T_{\text{flex, \%}}$	$T_{\text{fixed, CC}}$	$\Delta T_{\text{fixed, \%}}$
50 k	97 821 470	1 12 821 000	97 321 470	−0.51	97 821 470	0
100 k	98 321 470	112 821 000	97321470	−1.02	98 321 470	0
500 k	1 02 321 470	1 12 821 000	97 321 470	−4.89	1 02 321 470	0
1 M	1 07 321 470	1 12 821 000	97 321 470	−9.32	1 07 321 470	0
5 M	1 48 661 500	1 16 379 000	99 437 478	−33.11	1 60 838 270	8.19
10 M	2 22 187 620	1 22 007 000	1 08 670 310	−51.09	2 32 465 150	4.63
$P_{\text{max}} = 100 \text{ mW}$						
$T_{\text{pause, CC}}$	$T_{\text{reg, CC}}$	$T_{\text{packing, CC}}$	$T_{\text{flex, CC}}$	$\Delta T_{\text{flex, \%}}$	$T_{\text{fixed, CC}}$	$\Delta T_{\text{fixed, \%}}$
50 k	58 692 882	70 288 300	55 689 624	−5.12	58 692 882	0
100 k	58 992 882	70 288 300	55689 624	−5.60	58 992 882	0
500 k	61 392 882	70 288 300	55 829 074	−9.06	61 392 882	0
1 M	64 392 882	70 288 300	56 689 420	−11.96	64 392 882	0
5 M	1 03 259 032	73 846 500	64 067 302	−37.95	96 502 962	−6.54
10 M	1 43 259 032	83 079 000	74 067 302	−48.30	1 39 479 090	−2.64
$P_{\text{max}} = 200 \text{ mW}$						
$T_{\text{pause, CC}}$	$T_{\text{reg, CC}}$	$T_{\text{packing, CC}}$	$T_{\text{flex, CC}}$	$\Delta T_{\text{flex, \%}}$	$T_{\text{fixed, CC}}$	$\Delta T_{\text{fixed, \%}}$
50 k	39 128 588	46 187 700	36 865 798	−5.78	39 128 588	0
100 k	39 328 588	46 237 700	36 965 798	−6.01	39 328 588	0
500 k	40 928 588	46 637 700	37 765 798	−7.73	40 928 588	0
1 M	42 928 588	47 137 700	38 765 798	−9.70	42 928 588	0
5 M	58 928 588	51 812 400	46 765 798	−20.64	64 335 308	9.18
10 M	78 928 588	61 812 400	56 765 798	−28.08	92 986 060	17.81
$P_{\text{max}} = 500 \text{ mW}$						
$T_{\text{pause, CC}}$	$T_{\text{reg, CC}}$	$T_{\text{packing, CC}}$	$T_{\text{flex, CC}}$	$\Delta T_{\text{flex, \%}}$	$T_{\text{fixed, CC}}$	$\Delta T_{\text{fixed, \%}}$
50 k	19 564 294	24 610 900	20 005 068	2.25	19 564 294	0
100 k	19 664 294	24 710 900	20 005 068	1.73	19 664 294	0
500 k	20 464 294	25 510 900	20 464 294	0	20 464 294	0
1 M	21 464 294	26 510 900	21 464 294	0	21 464 294	0
5 M	29 464 294	34 510 900	29 464 294	0	32 167 654	9.18
10 M	39 464 294	44 510 900	39 464 294	0	46 493 030	17.81

shown in Fig. 5). A heuristic similar to Algorithm 1 without constraints is then utilised for this problem to minimise testing time.

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

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 blocks A, B and C, respectively. Note that we assume all e-SRAMs are tested in 100 MHz 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  $T_{\text{pause}}$  and the given power constraint  $P_{\text{max}}$ .  $T_{\text{reg}}$ ,  $T_{\text{packing}}$ ,  $T_{\text{flex}}$  and  $T_{\text{fixed}}$  represent 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  $T_{\text{pause}}$  can be varied, the testing time using the fast heuristic shown in Section 3.1.2 when  $T_{\text{pause}}$  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 100 MHz,  $T_{\text{pause}}$  varies from 500  $\mu\text{s}$  to 100 ms in our experiments.  $\Delta T_{\text{flex}}$  and  $\Delta T_{\text{fixed}}$  are calculated as  $\Delta T_{\text{flex}} = T_{\text{flex}} - T_{\text{reg}}/T_{\text{reg}} \times 100\%$  and  $\Delta T_{\text{fixed}} = T_{\text{fixed}} - T_{\text{reg}}/T_{\text{reg}} \times 100\%$ .

**Table 4: Testing time comparison for test case 3**

Test case 3						
$P_{\text{max}} = 60 \text{ mW}$						
$T_{\text{pause}}, \text{CC}$	$T_{\text{reg}}, \text{CC}$	$T_{\text{packing}}, \text{CC}$	$T_{\text{flex}}, \text{CC}$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, \text{CC}$	$\Delta T_{\text{fixed}}, \%$
50 k	9 832 198	10 257 360	9 832 198	0	9 832 198	0
100 k	9 932 198	10 168 740	9 932 198	0	9 932 198	0
500 k	19 845 932	10 732 200	10 732 198	-45.92	10 732 198	-45.92
1 M	32 374 618	12 392 580	11 732 198	-63.76	12 277 186	-62.08
5 M	1 34 827 711	23 252 400	19 732 198	-85.36	23 650 983	-82.46
10 M	2 64 827 711	32 190 600	29 732 198	-88.77	38 655 739	-85.40
$P_{\text{max}} = 100 \text{ mW}$						
$T_{\text{pause}}, \text{CC}$	$T_{\text{reg}}, \text{CC}$	$T_{\text{packing}}, \text{CC}$	$T_{\text{flex}}, \text{CC}$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, \text{CC}$	$\Delta T_{\text{fixed}}, \%$
50 k	9 832 198	9 832 200	9 872 412	0.41	9 832 198	0
100 k	9 932 198	10 610 640	9 932 198	0	9 932 198	0
500 k	11 946 457	11 654 380	10 732 198	-10.16	10 732 198	-10.16
1 M	19 835 346	12 344 320	11 732 198	-40.85	11 732 198	-40.85
5 M	82 783 700	21 061 200	19 732 198	-76.16	23 650 983	-71.43
10 M	1 62 783 700	32 850 000	29 732 198	-81.74	38 655 739	-76.25
$P_{\text{max}} = 200 \text{ mW}$						
$T_{\text{pause}}, \text{CC}$	$T_{\text{reg}}, \text{CC}$	$T_{\text{packing}}, \text{CC}$	$T_{\text{flex}}, \text{CC}$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, \text{CC}$	$\Delta T_{\text{fixed}}, \%$
50 k	9 832 198	10 340 040	10 002 636	1.73	9 832 198	0
100 k	9 932 198	10 168 740	10 012 422	0.81	9 932 198	0
500 k	10 732 198	11 002 540	10 732 198	0	10 732 198	0
1 M	11 732 198	11 744 900	11 732 198	0	11 732 198	0
5 M	42 463 843	19 732 200	19 732 198	-53.53	23 650 983	-44.30
10 M	82 463 843	30 002 600	29 732 198	-63.95	38 655 739	-53.12
$P_{\text{max}} = 500 \text{ mW}$						
$T_{\text{pause}}, \text{CC}$	$T_{\text{reg}}, \text{CC}$	$T_{\text{packing}}, \text{CC}$	$T_{\text{flex}}, \text{CC}$	$\Delta T_{\text{flex}}, \%$	$T_{\text{fixed}}, \text{CC}$	$\Delta T_{\text{fixed}}, \%$
50 k	9 832 198	10 121 620	10 002 636	1.73	9 832 198	0
100 k	9 932 198	10 132 160	10 012 422	0.81	9 932 198	0
500 k	10 732 198	11 002 540	10 732 198	0	10 732 198	0
1 M	11 732 198	11 833 860	11 732 198	0	11 732 198	0
5 M	20 076 869	19 732 200	19 732 198	-1.72	23 650 983	17.80
10 M	40 076 869	29 732 200	29 732 198	-25.81	38 655 739	-3.55

$t_{reg}/T_{reg} \times 100\%$ , 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 Section 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 are 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 800 mW. When  $P_{max} = 500$  mW, 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 are 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,  $P_{max} = 200$  mW and  $T_{pause} \geq 5$  M in test case 2). This is due to the fact that test case 2 has only 15 large e-SRAMs, and when  $T_{pause} \geq 5$  M several e-SRAMs can be grouped into one scheduling unit (when  $T_{pause} < 5$  M, e-SRAMs in test case 2 cannot be grouped and the scheduling process is exactly the same as the single-rectangle power model). As shown in Fig. 3, the grouping happens in the horizontal direction and the testing time of the group becomes larger than the testing time of each individual e-SRAM. By using the single-rectangle

**Table 5: Testing time comparison for test case 4**

Test case 4						
$P_{max} = 60$ mW						
$T_{pauser, CC}$	$T_{reg, CC}$	$T_{packing, CC}$	$T_{flex, CC}$	$\Delta T_{flex, \%}$	$T_{fixed, CC}$	$\Delta T_{fixed, \%}$
50 k	1 07 653 668	1 29 819 000	1 07 103 668	-0.51	1 07 653 668	0
100 k	112 463 987	1 30 037 000	1 07 153 668	-4.72	1 08 253 668	-3.74
500 k	1 76 764 981	1 29 961 000	1 07 553 668	-39.15	1 13 053 668	-36.04
1 M	2 58 026 656	1 30 542 000	1 08 053 668	-58.12	1 19 053 668	-53.86
5 M	9 09 248 192	1 30 842 000	1 09 890 674	-87.91	1 85 165 072	-79.64
10 M	1 725 540 774	1 37 209 000	116 239 718	-93.26	2 36 508 457	-86.29
$P_{max} = 100$ mW						
$T_{pauser, CC}$	$T_{reg, CC}$	$T_{packing, CC}$	$T_{flex, CC}$	$\Delta T_{flex, \%}$	$T_{fixed, CC}$	$\Delta T_{fixed, \%}$
50 k	63 659 032	79 170 800	58 392 882	-8.27	63 659 032	0
100 k	67 345 682	78 933 800	58 394 209	-13.29	64 059 032	-4.88
500 k	1 07 396 086	80 139 300	58 355 585	-45.66	67 259 032	-37.37
1 M	1 56 979 271	80 120 800	58 864 114	-62.50	71 718 391	-54.31
5 M	5 60 096 335	89 052 600	64 067 302	-88.56	1 03 260 488	-81.56
10 M	1 060 515 913	90 178 300	74 067 302	-93.02	1 41 904 895	-86.62
$P_{max} = 200$ mW						
$T_{pauser, CC}$	$T_{reg, CC}$	$T_{packing, CC}$	$T_{flex, CC}$	$\Delta T_{flex, \%}$	$T_{fixed, CC}$	$\Delta T_{fixed, \%}$
50 k	39 128 588	49 206 700	37 847 244	-3.27	39 128 588	0
100 k	39 328 588	50 167 900	37 847 244	-3.77	39 328 588	0
500 k	53 371 616	50 444 000	37 986 694	-28.83	40 928 588	-23.31
1 M	78 037 504	51 967 900	38 765 798	-50.32	42 928 588	-44.99
5 M	2 76 225 151	57 333 200	46 765 798	-83.07	64 603 720	-76.61
10 M	5 25 644 342	67 333 200	56 765 798	-89.20	94 603 105	-82.00
$P_{max} = 500$ mW						
$T_{pauser, CC}$	$T_{reg, CC}$	$T_{packing, CC}$	$T_{flex, CC}$	$\Delta T_{flex, \%}$	$T_{fixed, CC}$	$\Delta T_{fixed, \%}$
50 k	19 564 294	27 494 700	19 564 294	0	19 564 294	0
100 k	19 664 294	27 594 700	19 664 294	0	19 664 294	0
500 k	21 483 404	28 394 700	20 685 190	-3.72	20 464 294	-4.74
1 M	31 559 844	29 394 700	21 464 294	-31.99	21 464 294	-31.99
5 M	1 11 159 194	37 394 700	29 464 294	-73.49	32 303 628	-70.94
10 M	2 19 778 181	47 394 700	39 464 294	-82.04	47 301 790	-78.48

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  $T_{\text{reg}} < T_{\text{flex}}$  and we attribute them to the fact that the fast heuristic explores only part of the solution space.

## 5 Conclusion

Traditionally, test power modelling treats e-SRAMs the same as logic cores and represents the test using a 'single-rectangle' model. This paper showed that this model is overly conservative because of the near-zero power delay cycles used to detect data retention faults. By taking advantage of this property, we proposed a retention-aware test power model and the associated test scheduling techniques. We considered 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 [1], the DRF pause time can be as large as up to hundreds of *ms* even for the future technologies, the proposed approach is able to greatly reduce the e-SRAM test cost.

## 6 Acknowledgment

This work was supported in part by the Hong Kong SAR UGC Direct Grant 2050366.

## 7 References

- Aitken, R., Dogra, N., Gandhi, D., and Becker, S.: 'Redundancy, repair, and test features of a 90 nm Embedded SRAM generator'. Proc. IEEE Int. Sym. on Defect and Fault Tolerance in VLSI Systems (DFT), 2003, pp. 467–474
- Bommireddy, A., Khare, J., Shaikh, S., and Su, S.-T.: 'Test and debug of networking SoCs - a case study'. Proc. IEEE VLSI Test Symp. (VTS), 2000, pp. 121–126
- Zorian, Y.: 'A distributed BIST control scheme for complex VLSI devices'. Proc. IEEE VLSI Test Symp. (VTS), Princeton, NJ, 1993, pp. 6–11
- Wang, C.-W. *et al.*: 'Test scheduling of BISTed memory cores for SOC'. Proc. IEEE Asian Test Symp. (ATS), Tamuning, Guam, USA, 2002, pp. 356–361
- Wang, B., Yang, J., Wu, Y., and Ivanov, A.: 'A retention-aware test power model for embedded SRAM'. Proc. IEEE Asia South Pacific Design Automation Conf. (ASP-DAC), 2005, pp. 1180–1183
- Dekker, R., Beenker, F., and Thijssen, L.: 'A realistic fault model and test algorithms for static random access memories', *IEEE Trans. Computer-Aided Design*, 1990, **9**, (6), pp. 567–572
- Rajsuman, R.: 'An algorithm and design to test random access memories'. Proc. Int. Symp. on Circuits and Systems (ISCAS), 1992, pp. 439–442
- Wang, B., Yang, J., and Ivanov, A.: 'Reducing test time of embedded SRAMs'. Proc. IEEE Int. Workshop on Memory Technol., Design and Testing (MTDT), 2003, pp. 47–52
- Adams, R.D., Deo, A.P., and Zarrineh, K.: 'Method and apparatus for testing memory cells for data retention faults', U.S. Patent 6,681,350, Cadence Design Systems, Inc., 20 January 2004
- Brauch, J., and Fleischman, J.: 'Design of cache test hardware on the HP PA8500'. Proc. IEEE Int. Test Conf. (ITC), 1997, pp. 286–293
- Champac, V.H., and Avendano, V.: 'Test of data retention faults in CMOS SRAMs using special DFT circuitries', *IEE Proc., Circuits, Devices and Systems*, 2004, **151**, (2), pp. 78–82
- Champac, V.H., Avendano, V., and Linares, M.: 'Bit line sensing strategy for testing for data retention faults in CMOS SRAMs', *IEE Electron. Lett.*, 2000, **36**, (14), pp. 1182–1183
- Champac, V.H., Castillejos, J., and Figueras, J.: 'IDDQ testing of opens in CMOS SRAMs'. Proc. IEEE VLSI Test Symp. (VTS), 1998, pp. 106–111
- Kuo, C., Toms, T., Neel, B.T., Jelemsky, J., Carter, E.A., and Smith, P.: 'Soft-defect detection (SDD) technique for a high-reliability CMOS SRAM', *IEEE J. of Solid-State Circ.*, 1990, **25**, (1), pp. 61–67
- Meixner, A., and Banik, J.: 'Weak write test mode: an SRAM cell stability design for test technique'. Proc. IEEE Int. Test Conf. (ITC), 1997, pp. 1043–1052
- Yang, J., Wang, B., Wu, Y., and Ivanov, A.: 'Fast detection of data retention faults and other SRAM cell open defects', *IEEE Trans. Computer-Aided Design*, 2006, **25**, (1), pp. 167–180
- Yoon, D.H., Kim, H.S., and Kang, S.: 'Dynamic power supply current test for CMOS SRAM'. Proc. Int. Conf. on Computer-Aided Design (ICCAD), 2001, pp. 399–402
- Powell, T.J., Cheng, W.T., Rayhawk, J., Samman, O., Policke, P., and Lai, S.: 'BIST for deep submicron ASIC memories with high performance application'. Proc. IEEE Int. Test Conf. (ITC), 2003, pp. 386–392
- Xu, Q., and Nicolici, N.: 'Resource-constrained system-on-a-chip test: a survey', *IEE Proc., Computers Digital Tech.*, 2005, **152**, (1), pp. 67–81
- Girard, P.: 'Survey of low-power testing of VLSI circuits', *IEEE Design & Test of Computers*, 2002, **19**, (3), pp. 80–90
- Chou, R.M., Saluja, K.K., and Agrawal, V.D.: 'Scheduling tests for VLSI systems under power constraints', *IEEE Trans. VLSI Syst.*, 1997, **5**, (2), pp. 175–184
- Goel, S.K., and Marinissen, E.J.: 'Effective and efficient test architecture design for SOCs'. Proc. IEEE Int. Test Conf. (ITC), Baltimore, MD, 2002, pp. 529–538
- Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'Integrated wrapper/TAM co-optimization, constraint-driven test scheduling, and tester data volume reduction for SOCs'. Proc. ACM/IEEE Design Automation Conf. (DAC), 2002, pp. 685–690
- Larsson, E., Pouget, J., and Peng, Z.: 'Defect-aware SOC test scheduling'. Proc. IEEE European Test Workshop (ETW), 2004, pp. 359–364
- Xu, Q., and Nicolici, N.: 'Multi-frequency test access mechanism design for modular SOC testing'. Proc. IEEE Asian Test Symp. (ATS), 2004, pp. 2–7
- Fang, B.H., Xu, Q., and Nicolici, N.: 'Hardware/software co-testing of embedded memories in complex SOCs'. Proc. Int. Conf. on Computer-Aided Design (ICCAD), 2003, pp. 599–605
- Jone, W.B., Huang, D.C., Wu, S.C., and Lee, K.J.: 'An efficient BIST method for distributed small buffers', *IEEE Trans. VLSI Syst.*, 2002, **10**, (4), pp. 512–515
- Nadeau-Dostie, B., Silburt, A., and Agrawal, V.K.: 'Serial interfacing technique for embedded memory testing', *IEEE Design & Test of Computers*, 1990, **7**, (2), pp. 52–63
- Wang, B., and Xu, Q.: 'Test/repair area overhead reduction for small embedded SRAMs'. Proc. IEEE Asian Test Symp. (ATS), 2006, pp. 37–44
- Young, F.Y., Chu, C.N., and Ho, M.L.: 'Placement constraints in floorplan design', *IEEE Trans. VLSI Syst.*, 2004, **12**, (7), pp. 735–745
- Murata, H., Fujiyoshi, K., Nakatake, S., and Kajitani, Y.: 'Rectangle-packing-based module placement'. Proc. Int. Conf. on Computer-Aided Design (ICCAD), 1995, pp. 472–479