2023（第五届）集成电路 EDA 设计精英挑战赛

赛题指南

一、赛题名称

MBIST 自动规划分组算法

二、命题企业

深圳国微芯科技有限公司

三、赛题 Chair

金洲 （中国石油大学（北京））

四、赛题背景

存储器单元的可测性问题是数字电路 DFT 设计中的重要内容。在各种规模的数字芯片中，特别是在大规模 SOC、CPU、GPU、FPGA 或 AI-ASIC 芯片中，都广泛分布着大大小小不同功能，不同端口的存储器单元。这些存储器单元在物理空间上也分散位于芯片各处。目前工业界通常采用 MBIST（Memory Build-in Self Test）存储器内建自测试的方式，对这些存储单元进行有效测试。

MBIST 的电路一般包含一个控制器和对应的存储器接口。控制器负责按照测试算法产生对应的测试地址、测试数据、读写控制

及其它存储器功能控制信号，并将这些信号传递给存储器接口；控制器也负责接收存储器的输出数据的反馈，对输出数据进行比对。现代工业界使用的 MBIST 控制器结构可实现一个控制器连接多个存储器并同时进行测试。由于DFT插入的电路（包括MBIST控制器以及与控制器链接的测试网络）属于辅助测试电路，所以理论上，在满足测试覆盖要求的前提下，这些电路在设计中使用得越少越好。使用越少，占用的额外芯片面积越少，并拥有更高效费比。因此共享式的MBIST方式是当前和未来的主流设计。图1、2均来自SIEMENS的官方网站，展示了一种芯片上MBIST架构的布局框图，以及多存储单元共享BIST控制器的情景：

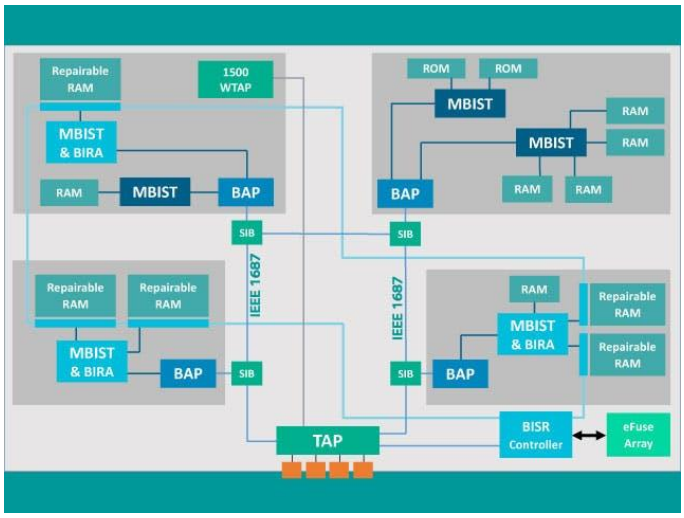

图1 一种芯片上MBIST的结构布局

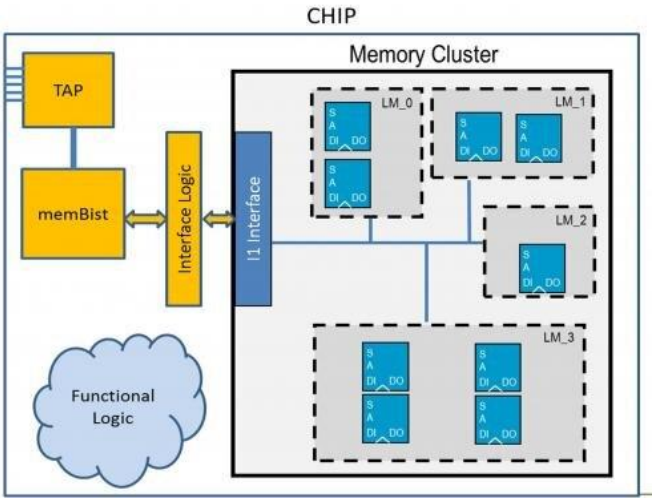

图 2 单个 MBIST 控制器控制一组存储单元

结构明确后，DFT 工程师在进行 MBIST 设计时思考的主要问题是：如何对分散在芯片上的这些存储单元进行有效规划和分组，并把它们指派给不同的 MBIST 控制器？是否存在一种最优的分组方式？

在实际的芯片设计工程项目中，有着众多的限制约束条件制约分组，例如：面积、测试功耗、时序、测试算法、BIST 结构、存储器结构等等。想要获得一个较优的分组结果需要从相当多的方面进行综合考量，这些约束中有些还是彼此制约的关系，需要在一定程度上进行折衷。最终，要求能达成所有的设计目标。

五、赛题描述

（一）请设计一个python或C++程序，该程序应能读取数据集中的存储器列表文件，并通过读取、分析数据集中的其它文件内

容对存储器集合进行分组。程序输出为存储器列表的一个分组。

（二）国微芯将提供程序需要读取的数据集文件。数据集文件为多组，每组代表一个芯片设计中所涉及到的各种数据。数据集分为编程时试用和评分用两部分，用于评分的数据集将不同于前期公共开放的试用数据集。

（三）每组数据集代表一个设计，意味着打分时，程序需经历多组数据的测试。在不同设计下的分组结果将以加权平均的方式得到最后得分，反映程序在各种设计上的综合表现能力。

（四）建议python采用3.10以上stable版本，C++使用兼容C++17版本，并最好在UNIX-like环境下编程。考虑数据集的使用便捷性，不接受windows环境下的可执行程序。推荐使用ubuntu 22.04 LTS操作系统。

六、数集说明

（一）存储器列表：

文本文件，以.f或.list后缀结尾。文件中，以存储器类型为分隔，列出了设计中包含的所有存储器。每行一个存储器，并包含了其完整的层次化名。例如：

```
spsram_t_1024x16m4s:
```

```
u_top/block2/wrapper_i2/spsram16_b2
```

（二）存储器datasheet：

文本文件，以.ds后缀结尾。该文件是存储器的数据说明文件，每种存储器对映一个数据表文件。其中包含有存储器的面积、功耗等重要信息。

（三）存储器物理布局文件：

存储器物理布局文件是物理设计的DEF（design exchange format）文件的节选，列出了所有存储单元在版图上的物理位置。文件格式为def文件格式。每一行包含有存储单元的instance名、module名和坐标信息。例如：

```
<File Partial>


- u_top/block2/wrapper_i1/spsram16_b2 spsram_t_1024x16m4s + FIXED ( 2926054 366624 )
S + WEIGHT 1
  + HALO 4000 4000 4000 4000
 ;
- u_top/block2/wrapper_i2/spsram16_b2 spsram_t_1024x16m4s + FIXED ( 3208354 366624 )
S + WEIGHT 1
  + HALO 4000 4000 4000 4000
```

（四）存储器BIST模型库文件：

存储器的BIST模型库文件可能以.lib，.masis，.lvlib或.memlib等后缀结尾。每种类型的存储器对映一个存储器BIST模型描述文件。不同后缀的文件格式可能有所不同，但都包含有存储器的重要逻辑功能信息，包括但不限于，测试算法，端口类型，有效信号值等。

（五）存储器的verilog描述文件：

存储器的verilog描述文件，文件格式为.v后缀的verilog语言程序文件。同样是每种类型的存储器对映一个存储器verilog描述文件。

（六）程序输出

参赛程序最终输出，为存储单元列表文件中，包含的所有存储单元实例的一个分隔列表。结构类似存储器单元列表，以controller_*控制器标签作为分割，例如：

```
Controller_1:
    u_top/block2/wrapper_i1/spsram16_b2

Controller_2:
    u_top/block2/wrapper_i2/spsram24_b0
    u_top/block2/wrapper_i2/spsram24_b1
```

```
Controller_3:
    u_top/block2/wrapper_i2/spsram32_b1
```

七、评分细则

（一）上述每个数据集代表一个具体设计。参赛程序需通过多组数据集的测试。在一个设计的数据集上，参赛程序的分值由两部分构成，基础分和比较分。每个数据集中有一个*.spec后缀结尾的文件，其中包含程序的分组结果应满足的设计指标约束。对设计约束的检查构成基础分部分。之后会按程序的输出结果进行排序，根据排序的相对分布给出比较分。不能给出满足设计指标约束的分组结果的程序，会相应扣减基础分。

（二）当参赛程序的设计约束和运行时间约束检查结束后，对输出的分组进行排序，按分组数量从少到多的方式进行排序。分组数量越少的程序被认为是性能更好的程序。因为较少的分组，意味着额外占用的测试电路的面积越少。对参赛程序的输出结果，基础分由DRC（设计规则检查）的通过率决定；比较分由程序的分组数和运行时间决定。这三项指标是决定程序优劣的指标。

（三）程序运行时间是EDA软件实现的算法复杂度的重要指标，因此，运行时间上限会作为一项特殊的设计约束要求给出，参赛程序应在指定时限耗尽前输出运行结果。为了准确评估程序性能，评分机制可能会给出几个档位的时间上限。每次打分时，所有参赛程序按同一档位的运行时间上限作统一约束。

（四）比较分的基准值由程序输出的分组数和程序运行时间共同决定。当出现多个参赛程序的分组数量相同的情况时，程序的运行时间将影响比较分的基准值。运行时间按从短到长进行排序，程序运行时间越短，被认为性能越好。

（五）不同设计的数据集按照其设计的复杂度设定不同的权重系数。在不同设计下的分组结果将以加权平均的方式得到最后得分，反映程序在各种设计上的综合表现能力。

（六）评分采用计算机程序自动评分的方式进行。

（七）评分公式说明：

设计约束项（整数）：$D_c = i,\ 0 \le i$

运行时间上限：t_{max}

运行时间：t

每项设计约束的基础分值：$b = 100/(i+1)$

比较分取值空间：$[1,\ b-1]$

DRC检查通过项（整数）：$p \in [0,\ i]$

当前程序分组数：g

比较分基准：$s = g + t/t_{max}$

基础分得分计算：

$$S_{base} = p * b$$

比较分计算：

$$S_{comp} = 1 + \big((b-1)-1\big) * (\frac{s_{max} - s}{s_{max} - s_{min}})$$

s_{max}, s_{min}分别是所有参赛队在该数据集上计算比较分基准后的最大值与最小值。

在单个数据集上的得分：$S_k = S_{base} + S_{comp}$

最终得分为在所有测试数据集上的加权平均分，设数据集的数量为n：

$$\sum_{k=1}^{n} w_k = 1$$

$$S_{final} = \sum_{k=1}^{n} S_k w_k$$

八、参考资料

[1]. A. B. Kahng and I. Kang, "Co-optimization of memory BIST grouping, test scheduling, and logic placement," 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2014, pp.1-6

[2]. C. -H. Yeh, C. -H. Cheng and S. -H. Huang, "Grouping and placement of memory BIST controllers for test application time minimization," 2016 5th International Symposium on Next-Generation Electronics (ISNE), Hsinchu, Taiwan, 2016, pp. 1-2

[3]. K. S. Das and P. Prakash, "Automatic MBIST Scheduling Engine," 2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2019, pp. 1-6

[4]. Tessent MemoryBIST User's Manual， Siemens Industry Software, Inc., Wilsonville,USA,2021.

[5]. L. Martirosyan, G. Harutyunyan, S. Shoukourian and Y. Zorian, "A power based memory BIST grouping methodology," 2015 IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 2015, pp. 1-4

[6]. M. Miyazaki, T. Yoneda and H. Fujiwara, "A memory grouping method for sharing memory BIST logic," Asia and South Pacific Conference on Design Automation, 2006., Yokohama, Japan, 2006, pp. 6 pp.-

[7]. 陈佳楠,马永涛,李松,等. 多目标优化的多存储器内建自测试[J]. 电子测量与仪器学报,2020,34(1):193-199

[8]. B. Ghoshal and I. Sengupta, "A Distributed BIST Scheme for NoC-Based Memory Cores," 2013 Euromicro Conference on Digital System Design, Los Alamitos, CA, USA, 2013, pp. 567-574

[9]. Hu C, Li X, Fu Z, et al. The Implementation of a Configurable MBIST Controller for Multi-core SoC[C]//CCF National Conference on Computer Engineering and Technology. Springer, Singapore, 2019: 91-100.

[10]. 姜爽,刘诗斌,郭晨光,等. 嵌入式 SRAM MBIST 优化设计研究[J]. 微电子学与计算机,2020,37(8):37-42.

[11]. R. Silveira, Q. Qureshi and R. Zeli, "Flexible architecture of memory BISTs," 2018 IEEE 19th Latin-American Test Symposium (LATS), 2018, pp. 1-6.