Cost Model-Driven Test Resource Partitioning for SoCs

A. Würtenberger[†], P. Rosinger[†], B. M. Al-Hashimi[†] and K. Chakrabarty[‡]

[†]School of ECS, University of Southampton
SO17 1BJ, United Kingdom
{pmr,bmah}@ecs.soton.ac.uk

[‡]Dept. of Electrical & Computer Engineering
Duke University, Durnham, NC 27708, USA
krish@ee.duke.edu

Abstract

The main factors contributing to the cost of system-on-a-chip (SOC) manufacturing test are the required number of tester pins, the test application time, the tester memory requirements and the area overhead required by the test resources. These factors contribute with different weights, depending on the cost model of each SOC. Several methods have been recently proposed to optimize each of these factors, however none of the existing methods employs an objective function derived from the actual cost model of each product. This paper proposes a Genetic Algorithm-based test cost optimisation method, which enables the test designer to target directly the solutions matching the test cost model specific to each SOC.

1 Introduction

A significant fraction of the manufacturing costs of high-end integrated circuits goes towards the testing, necessary for satisfying the high quality requirements imposed by today's extremely competitive market. The main factors contributing to the cost of test are: test application time (TAT), tester channel count (TCC) [1] and area overhead (AO) required by the rest resources [2]. Several test resource partitioning (TRP) methods aiming to deliver low cost test solutions have been proposed and summarised recently [3]. These methods include various test access mechanism (TAM) design and test scheduling (order of core test execution) solutions, which facilitate on-chip test data transport and shorten test times in core-based SOCs by sharing tester channels between several embedded cores of the SOC [4, 5], as well as test data compression (TDC) methods [3], which reduce the amount of data to be transferred between the tester and the chip under test and the tester memory requirements at the cost of additional hardware needed for compressing/decompressing the test data. TDC can be carried out at

chip level (one compressor/decompressor pair per chip), at core level (each embedded core has it's own compressor/decompressor core), or at TAM level (a compressor/decompressor pair can be shared by multiple embedded cores, thus reducing the area overhead [6], see Fig. 1). A common characteristic of the existing TRP methods [3, 4, 5] is that they assume a fixed cost model for all SOCs in terms of TAT, AO, TCC. However, different SOC are produced in different volumes, have different complexities, hence requiring different amounts of test data and tester channels. This means that the relative contribution of each of these factors to the overall cost of test varies from one SOC to another, as reported recently [7]. Consequently, methods assuming a fixed cost model cannot guarantee the best solution in terms of the overal cost of test for every SOC, as shown in this paper. This is important because reducing cost of test is essentially the ultimate goal for DFT. This paper proposes a Genetic Algorithm-based test cost optimisation method, which enables the test designer to target directly the solutions matching the test cost model specific to each product.

2 Proposed Genetic Algorithm for Test Resource Partitioning

The proposed cost model driven TRP method is implemented using genetic algorithms (GA). For background information on GAs see [8]. The basic idea of the proposed method is to derive the fitness–function driving the GA from the target cost model. To make this possible, we propose the phenotype (or potential solution): $\mathcal{P}(\mathcal{C}, \mathcal{D}) =: \mathcal{P}$, where $\mathcal{C}$, $\mathcal{D}$ are two **chromosomes**. Each gene in $\mathcal{C} := (C_i)_{i=1,\cdots,n_{cores}}$ (n_{cores} equals the number of cores) represents exactly one core, whereas the genes in $\mathcal{D} := (D_i)_{i=1,\cdots,m}$ point to available compression methods. Rectangle packing formulations have been commonly used for solving TAM design problems [4, 5]. The TAM is represented as a bin with given height (equal to the number of TAM-wires) and infinite width (representing the TAT). A test of a core can be represented as a rectangle with the height is given by the necessary test data inputs, and the length of the rectangle is given by the testing time of the specific core. In order to reduce area overhead, the proposed method allows multiple cores to share one compatible decompressor (for example cores C2,C4,C6,C7 and C8 shown in Fig. 2 share a 25X decompressor, while each of the other cores has its own decompressor). This way, the time for testing all cores and the the number of tester channels connected to the decompressor define a decompressor-rectangle. Algorithm 1 shows how a given phenotype $\mathcal{P}(\mathcal{C},\mathcal{D})$ is mapped to its corresponding TRP solution. To obtain a test schedule, the test rectangles are added to the test schedule according to the following two rules: (i) place the rectangle as early (i.e. as left) as possible, without overlapping already placed rectangles. (ii) place the rectangle as far down (i.e. close to TAM wire 0) as possible.

The generic cost model description used in the proposed method has the following form: $C_{test} = F(TAT, AO, TCC)$, where F can be any function based on these parameters. Since the GA fitness function has inverse relationship to a cost function (the higher the value of the fitness function the better the solution), the fitness function is defined

as $1/C_{test}$. This way, a high value of the fitness function will correspond to a low cost value.

3 Experimental Results

The proposed cost model driven TRP method was experimentally validated using 10 hypothetical SoCs, containing between 8 and 15 cores and TAM widths w ranging between 8 and 15 lines. The following parameters were used in the GA. For each setup, the length of $\mathcal{C} := (C_i)_{i=1,\cdots,n_{cores}}$ equals the number of cores and the length of $\mathcal{D}$ was set to 40. 60 generations are processed to obtain the desired solution. Each generation contains a population of 100 phenotypes. The crossover-rates were $\mathfrak{P}_{\times}^{C} := 0.9$ and $\mathfrak{P}_{\times}^{D} := 0.3$, and the mutation-rates $\mathfrak{M}^{C} := 0.07$ and $\mathfrak{M}^{D} := 0.01$. Table 2 shows the comparison between the test costs produced by the proposed method and the method presented in [5] combined with core-level TDC for three different cost models. The TAM design method proposed in [5] combined with core-level TDC leads to the shortest TAT since each core has it's own compressor/decompressor pair and therefore, the corresponding tests are compressed to the maximum achievable. This represents a typical fixed objective test design flow, in this case the optimization objective being the TAT. The following cost function was used for the proposed method: $Cost = TAT * Weight_{TAT} + AO * Weight_{AO} + TCC * Weight_{TCC}$ where the weights (see Table 1 for three examples) are specified by the test designer as the cost per second of test ($Weight_{TAT}$), the cost per mm^2 of silicon ($Weight_{AO}$) and the cost per tester channel ($Weight_{TCC}$) respectively. For example, cost model 1 depicts a scenario where TCC is the main contributor to the cost of test (50 cents per test pin), while AO and TAT have much lower contributions (0.1 cent per mm^2 and 1 cent per second of test respectively). This can happen for example for chips produced in very low volumes and when the number of available pins is very limited, and extra test pins may require upgrading to a higher pin count, and hence more expensive, chip package. The results reported in columns 2 to 4 of Table 2 show cost improvements over the fixed objective approach of up to nearly 35% for SOC s9. Cost model 2 illustrates a case where area overhead is the most expensive component of the cost of test (10 cents per mm^2), while TAT and TCC contribute with only 1 cent per second of test and 0.05 cents per tester channel. Cost reductions in this case are even higher, going up to 73% (SOC s3). Cost model 3 uses the TAT as the only optimization objective, just like the TAM-design from [5] combined with core-level TDC. In this case, the cost model driven method, due to its heuristic nature, under-performs the fixed objective method, the average cost increase is $< 10\%$. This is however a very unrealistic cost model since it does not take into account other factors which contribute to the overall cost of test, such as TCC and AO. These experiments show that by driving the TRP optimization using a customizable objective function, as the case with the proposed method, derived from the actual cost model of the SOC can lead to significant reductions in the overall cost of test.

4 Conclusion

A new test cost optimisation method for SOC manufacturing test with user specified cost model has been proposed. This method provides better solution in terms of overall cost of test when compared with existing fixed objective methods. The proposed method makes a contribution towards the need to develop low cost SOC solution, as highlited by the ITRS'03.

5 Acknowledgment

This work was supported by the EPSRC UK grant no. GR/S41135/01. The authors would like to thank Dr. Gonciari and Prof. Reddy for the fruitful discussions.

References

[1] Ajay Khoche and Jochen Rivoir. I/O bandwidth bottleneck for test: Is it real? In *Workshop on TRP*, pages 2.3–1–2.3.6, Atlantic City, NJ, USA, November 2000.

[2] Janusz Rajski, Jerzy Tyzer, Mark Kassab, Nilanjan Mukherjee, Rob Thompson, Kun-Han Tsai, Andre Hertwig, Nagesh Tamarapalli, Grzegorz Mrugalski, Geir Eide, and Jun Qian. Embedded deterministic test for low-cost manufacturing test. In *ITC*, pages 301–310, Baltimore, MD, USA, October 2002.

[3] Bashir M.Al-Hashimi (Ed.). *System-on-Chip: Next Generation Electronics*, chapter 22. Efficient modular testing and test resource partitioning for core-based SoCs, pages 751–783. IEE, 2006.

[4] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. On using rectangle packing for SOC Wrapper/TAM co-optimization. In *VTS*, pages 253–258, Monterey, CA, USA, April 2002.

[5] Wei Zou, S.M. Reddy, I. Pomeranz, and Yu Huang. SOC test scheduling using simulated annealing. In *VTS*, pages 325–330, Napa Valley, CA, USA, April 2003.

[6] Paul Theo Gonciari and Bashir M Al-Hashimi. A compression-driven test access mechanism design approach. In *ETS*, pages 100–105, Corsica, France, May 2004.

[7] Rohit Kapur, T.W. Williams, Jennifer Dworak, and M. Ray Mercer. How to evaluate test compression methods. `http://www.us.design-reuse.com/articles/article8889.html`, October 2004. (8. Dec. 2005).

[8] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Pub. Co., 1989.

Figures and tables

List of Figures

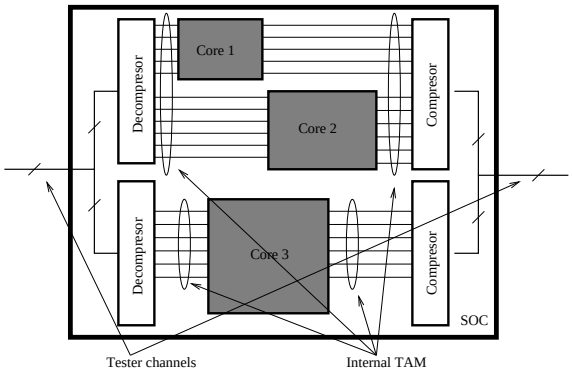

Figure 1. TRP using test data compression

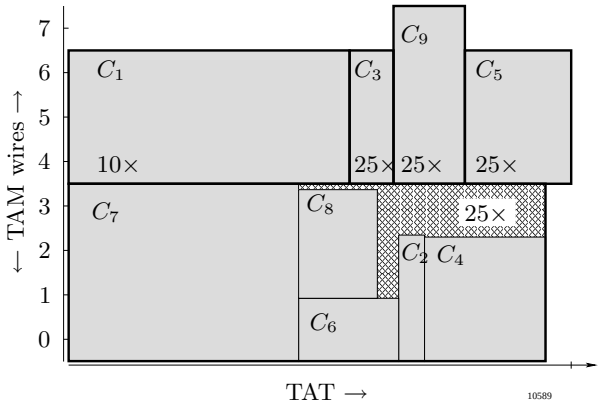

Figure 2. Potential TRP solution for a hypothetical SOC

List of Algorithms

```
Data: (C_i)_{i=1,...,n_cores}, (D_i)_{i=1,...,m}
Result: P
foreach Core in (C_i)_{i=1,...,n_cores} do
    Decompressor ← GetNext((D_i)_{i=1,...,m})
    if Decompressor = None then
        break
    else if Compatible(Core, Decompressor) then
        Assign(Core → Decompressor)
        if not Decompressor in TAM then
            Assign(Decompressor → TAM)
        end
    end
end
foreach D in (D_j)_{j=1,...,k} do
    foreach C in D do
        Place(C, TAM_D)
    end
    Place(D, TAM)
end
```

Algorithm 1: **Mapping a GA phenotype to a TRP solution**

List of Tables

Cost model	Weights		
	TAT (cents/s)	AO (cents/mm^2)	TCC(cents/pin)
Cost model 1	1	0.1	50
Cost model 2	1	10	0.5
Cost model 3	1	0	0

Table 1. Cost model examples

SoC	Cost model 1(TCC)			Cost model 2(AO)			Cost model 3(TAT)		
	[5]	Proposed	Improvement(%)	[5]	Proposed	Improvement(%)	[5]	Proposed	Improvement(%)
"s1"	79.92	63.69	20.30	156.35	75.47	51.72	8.45	8.45	0
"s2"	86.91	72.62	16.43	164.58	70.95	56.89	10.37	12.31	-18.72
"s3"	57.79	50.99	11.76	179.59	47.64	73.46	11.43	12.74	-11.48
"s4"	86.64	74.47	14.04	167.16	66.72	60.08	9.81	11.43	-16.53
"s5"	70.93	66.52	6.21	202.99	62.93	68.99	13.95	13.99	-0.25
"s6"	89.13	80.70	9.45	233.22	83.20	64.32	16.98	21.46	-26.38
"s7"	51.22	46.51	9.17	119.33	37.27	68.76	10.13	10.20	-0.75
"s8"	78.59	74.24	5.53	198.40	83.48	57.92	11.86	13.18	-11.14
"s9"	75.79	49.46	34.74	100.74	51.32	49.05	4.84	4.84	0
"s10"	49.814	44.06	11.53	111.39	42.10	62.19	8.79	9.52	-8.29
"Average improvement"			13.92			61.34			-9.35

Table 2. Comparison between the cost model driven method and a fixed objective method [5]