Test Planning and Design Space Exploration in a Core-based Environment*

Érika Cota † Luigi Carro † Alex Orailoglu ‡ Marcelo Lubaszewski †

†PPGC - Instituto de Informática
Universidade Federal do Rio Grande do Sul
PO Box 15064, Porto Alegre - Brazil
erika@inf.ufrgs.br, {carro,luba}@iee.ufrgs.br

‡Computer Science and Engineering Dept.
University of California, San Diego
La Jolla, CA 92093 - USA
alex@cs.ucsd.edu

Abstract

This paper proposes a comprehensive model for test planning in a core-based environment. The main contribution of this work is the use of several types of TAMs and the consideration of different optimization factors (area, pins and test time) during the global TAM and test schedule definition. This expansion of concerns makes possible an efficient yet fine-grained search in the huge design space of a reuse-based environment. Experimental results clearly show the variety of trade-offs that can be explored using the proposed model, and its effectiveness on optimizing the system test design.

1. Introduction

The design of a system is a process that involves several constraints, features and trade-offs of different cost factors, such as area, performance, power and test time. Nowadays, test planning has become one of the most expensive steps during the design flow of an electronic circuit built in a core-based design paradigm. Access to embedded cores, the integration of several test methods (not to mention the diversity among the cores themselves) and the optimization of cost factors such as area overhead, test time, and number of test pins in the interface, are just a few of the several problems that need to be tackled during test planning. If extra hardware is ultimately required it enormously complicates the synthesis equations for balancing system constraints.

A great deal of effort has been expended in the last few years, towards the development of suitable solutions for the test of core-based systems [1]-[15]. The main drawback of the current solutions is that the test planning task is conceived as an "after the design" step. Although the definition of the test solution is performed before synthesis and although some approaches use partial information about connectivity and placement of the system, none of the solutions presented so far considers test planning as early as during the design of the system itself. This means that a good solution from the logical point of view is accomplished, although its physical implementation according to the system characteristics is rarely considered. This kind of approach limits the quality of the solutions basically to the set of cores being used in the system, instead of tackling the system as an entity. Usually, the presented results do not provide to the system designer an accurate information about critical points of the system that could be improved, either in conjunction with the core provider, or during the design and synthesis process. Besides, current approaches usually have a restricted model for the test access mechanisms (mostly bus-centered) and based on this model the remaining parameters are optimized.

This work proposes a comprehensive model for the system, representing different aspects of each core, as well as the different conditions and restrictions of the system, in a unified structure. The main contributions of this model are fourfold: 1) we do not assume a single type of connection for the internal TAMs in the system. Partial busses are considered along with functional connections, transparency, and other bypass modes available through the wrapper or the core configuration; 2) the diversity of the test requirements among the embedded cores is used as an advantage to find the global test solution, by privileging critical cores with more test resources. This way, the global solution does not have to optimize fully every single core in the system; 3) both the schedule and the global TAM are defined together, and not as independent tasks as in other approaches; 4) the model represents a step closer in the integration of DFT planning early in the design process, and to a fruitful relationship between core designers and users. The iterative search method provides the system designer with accurate information about critical points in the system being

*This work was partially supported by CAPES under the process BEX0752/00-2

developed. The designer can either suggest specific modifications to the core provider or use this information to guide the synthesis tools in order to find the best solution for the whole system, not only in terms of testing, but with respect to all parameters of the design. The multi-TAM model and the definition of the scheduling and the global TAM in parallel are the key for the fine-grained exploration of the design space so that good compromises among the various trade-offs being sought in the design as a whole can be found.

The paper is organized as follows: in section 2 we review some related works. Section 3 briefly presents the trade-offs involved when using the multiple TAMs model. Then, the algorithm for test planning is described in section 4, while experimental results are discussed in section 5. Section 6 presents the conclusions and possible expansions of the proposed model.

2. Related Works

In terms of test access mechanism definition, several approaches have been proposed [1]-[7]. These approaches range from adaptations of traditional test techniques, such as the 1149.1 standard, to the development of hierarchical and generic methods for test access. The P1500 standard proposal [8, 15] has emerged as a conciliatory proposal for the easier integration of the several possible solutions in a single step. The CTL language also proposed by the P1500 group is another feature to catalyze an accurate communication between core designers and core users. As for the performance of the test solution, several approaches have been proposed to tackle the TAM definition considering test time issues [9]-[14], [16].

Recently, a number of frameworks and increasingly more comprehensive models have been proposed to cope with global optimizations for the final solution. These frameworks differ by the type of core test methods addressed and the TAM definition, but they all manage the distribution of test resources considering a variety of cost factors.

Benso *et al* present in [17] a tool for integration of cores with different test requirements (full scan, partial scan and BIST ready cores). The TAM follows a bus-based model that connects a group of cores to a BIST controller. Scheduling of BIST resources and data pattern delivery are also considered in the test solution.

Nourani and Papachristou propose in [18] the definition of the test access architecture by taking advantage of the connections already present in the system. The method is well suited for structural testing, but it can be quite limited for other test methods, such as scan-based ones, where no previous functional connections exist for the scan pins in the cores. This model has been improved and presented in [12], where other structures in the system, such as busses and tri-state ports, are also considered for the test path definition. After defining the TAM for each core, the tool defines the test scheduling so that a minimum test time for the system is achieved.

Larsson and Peng propose in [19] a framework for SOC testing which considers test time minimization, TAM optimization, test set selection and test resource placement, along with test resources and power consumption constraints. The tool is based on the fact that different test sets can be used to test a core. This way, each test set is evaluated under power, time, memory requirements, and so on, and the best test set is chosen according to the system constraints. They further assume that scan chains can be divided into smaller ones, to accelerate test time. BIST resources are then placed in the system according to their usage by the cores. After the TAM definition, the test schedule is generated so that the minimum test time for that specific TAM is achieved.

Each method considers a subset of the system constraints and assumes restricted TAM models on top of which the solution is searched. Even the methods that contemplate several cost factors at the same time still define only the TAM based on the restricted set of test resources, and then optimize the schedule to achieve smaller test times. Then, it is up to the designer to provide different amounts of test resources (as BIST controllers, TAM bitwidth, number of TAMs or test sets of the cores) and perform a number of searches under a variety of constraints.

However, the design space for the system integrator is huge and fine-grained and the consideration of DFT inclusion in the earlier steps of the design process becomes crucial for a good test strategy. Thus, an automatic test planning scheme (model and search algorithm) that can cope with the various trade-offs being sought for each system is the key for solving system test requirements within specific design constraints.

3. TAM Definition and Cost Factors

In a model that aims at optimizing several parameters at the same time and at finding the best global solution, different options for internal connections must be available, so that the best one can be selected among all.

We have defined a multiple yet not exhaustive model for connecting a core under test (CUT) to another core in the system, called here a *neighbor core*. Each type of connection implies distinct costs for the area overhead of the connection, the number of pins for the CUT in the interface, and CUT test time. The area overhead includes the values for the wrapper (basic mandatory operations and a 1-flip-flop boundary cell model) and extra wiring (connections and bypasses). The number of extra pins for each core

is the number of new pins created in the system interface when the TAM for that core is completely defined.

Each pair of cores in the TAM is connected by one of the methods defined below. The algorithm that selects the type of connection and combines the pairs in a global solution is explained in section 4. The TAMs were defined for scan, external (non-scan) and BIST testing schemes.

3.1. Direct External Access

In this mode, no previously existent connection is reused. A direct connection between the CUT interface and the system interface is established. The bitwidth of this connection is assumed to be as large as the number of bits that need to be propagated to the interface. This implies an overhead of *n* in the number of pins, for *n* the number of test signals being propagated. The area overhead is proportional to *n* and to the routing distance from CUT to the system interface. The impact on CUT test time, on the other hand, is the smallest possible, since only one cycle delay is required for the test application or observation.

3.2. Reuse of Functional Connections

In this TAM, functional connections already present between CUT and another core are reused. If the bitwidth of the connection is smaller than the number of bits that must be propagated, extra wires are created between a CUT and the neighbor core. However, this is the only point in the system where a full bitwidth connection is created. A parallel bypass is established between the full bitwidth connection (neighbor-CUT) and one of the functional inputs of the neighbor core. Such functional input is the one that minimizes the test time of CUT at this point, that is, the one that presents the smallest number of cycles for the serial/parallel conversion operation. The test time of CUT using this connection is $\lceil \frac{n}{m} \rceil + 1$ cycles per test vector, where *m* is the number of bits of the selected input of the wrapper of the neighbor core.

Figure 1 shows an example of a connection using this TAM for a core with external testing. For scan-based cores, when there is no previous functional connection between scan pins of two cores, this TAM can be similarly used. However, the area overhead will be higher than for the functional pins, since the whole connection must be added into the system. The cost for extra pins is zero, as in the functional case, and the test time for a CUT is given by $(\lceil \frac{sc}{sn} \rceil + 1)L$, where *sc* is the number of scan chains in CUT, *sn* is the number of scan chains in the neighbor core and *L* denotes the maximum length of the scan chains of CUT.

It is important to state at this point that our model assumes, in this first prototype, internal scan chains of identical length. This precludes better test time optimizations

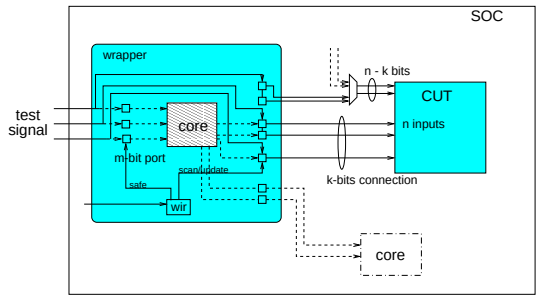

Figure 1. Access via functional connections

for scan pins, but it simplifies the algorithm by avoiding the problem of finding the best association between available and required pins, as shown in [20]. Thus, for example, if 32 test pins are being propagated and the neighbor core has 27 pins to be used, only 16 out of 27 pins will be considered for the serial/parallel conversion (chains are concatenated in groups of two). If the internal scan chains have similar lengths, test time is not deeply affected and total area overhead is reduced.

3.3. Use of Serial Bypass

The use of this TAM implies the transformation of the inputs (outputs) of CUT in an external scan-chain. A test vector is loaded serially, and an "update" cycle is required for its application to the core. This TAM is very inexpensive in terms of area, since a single wire traverses between the two cores being connected. On the other hand, it requires the largest number of cycles for test application. The new test time for CUT is $n + 1$ (external testing) or $L(sc + 1)$. Once this TAM is chosen, only serial bypass is considered from this point on, until the system interface is reached. The pin overhead for this TAM is 1.

3.4. Use of Transparency Functions

This TAM is considered for test pattern justification in the system when all inputs of the core under test are connected to the transparent core. The penalty on CUT test time is the number of cycles required for the propagation of the test vectors from the inputs of the neighbor core to its outputs. Transparency is usually the cheapest solution, since no extra hardware is required at the system level. The number of bits to be justified from this core to the system interface is the number of bits that control the transparency function.

3.5. Parallel Bypass

This TAM represents a bus-based access mechanism, where functional connections are reused only in the first level (from CUT to the first chosen neighbor). From this

point on, extra connections of bitwidth *n* are created accordingly. This is a compromise between the direct access TAM and the TAM presented in figure 1. Although the area cost for the extra connections may be higher for a single core, the possibility of reusing the bus is high, and the cost is attenuated among all cores using the bus. The pin overhead is *n* for the first core using this TAM, but is zero for any other core reusing it. In terms of test time, on the other hand, not only does the CUT manifest only a small increase on its test time (1 cycle), but also all other cores reusing the bus gain.

Although the TAMs were presented from the perspective of justification of test patterns, a similar model can be defined for the propagation of test results as well.

The defined TAMs are not the only possible ones. Different associations, as the one presented in [13], or the use of all functional connections in a multiple path search can also be modeled. However, as one can see in the experimental results, even this slightly expanded model allows a smoother search for the different possible solutions.

4. Global TAM and Scheduling Definition

Consider the arbitrary system shown in figure 2. It is composed of 10 cores from the ISCAS'85 and ISCAS'89 benchmarks. This system was proposed in [13], but no functional connection or placement is assumed on that work for this set of cores. The internal connections and placement were randomly assigned. Information about the number of test cycles for each core is also presented in the figure, and was calculated assuming a full bitwidth test access for each one. Dashed lines on some cores represent the number of scan chains defined for them. Internal scan chains are assumed to have equal or near equal lengths.

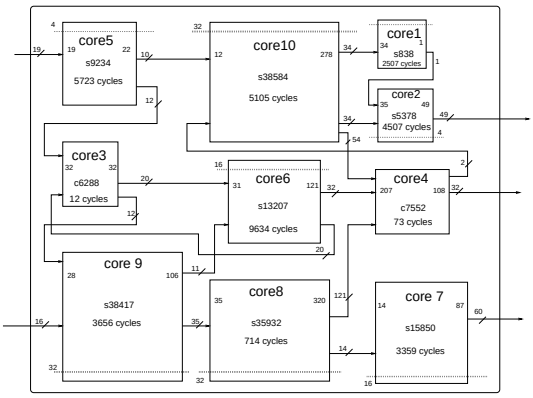

Figure 2. Synthetic system used as example

For a system with *N* cores, with t_i as the number of test cycles for each core $1 \leq i \leq N$, the minimum system test time possible is given by $max(t_i), i = 1..N$. To accomplish this minimum test time, the core *k* with $t_k = max(t_i), i = 1..N$ must have a full bitwidth and exclusive TAM. In the system of figure 2, core 6 is the core with the largest test time. The other cores with smaller test time can be accommodated in a scheduling as shown in figure 3, for example.

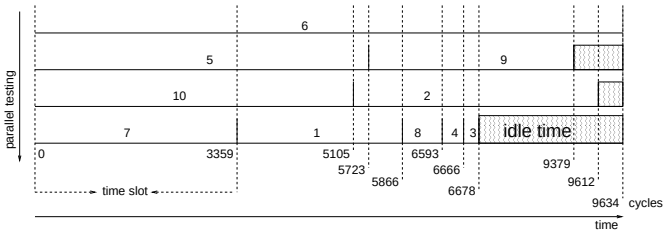

Figure 3. Test schedule for minimum test time

The schedule is represented as a series of time slots. Each time slot contains a group of cores that can be tested at the same time. Two cores can be tested in parallel if they do not share any test resource.

If a global TAM is defined based only on the availability of test resources, the test schedule must be defined so that cores sharing the same test resource do not overlap. This can increase the total test time enormously, depending on the set of resources available. On the other hand, if the schedule is defined to optimize only the global test time, the cost in terms of pins and area in the system interface may be huge, since several cores may need to have exclusive test access mechanisms.

The combined definition of the access mechanism and the test scheduling can provide composite solutions with affordable requirements both in terms of test resources and test time. For cost reduction, for example, we can use the idle time in any row of the schedule to increase the test time of a given core in that row. With the different models of TAMs proposed in section 3, every solution that increases the test time of a core up to the time limit defined by the idle time can be used. Then, the cheapest solution can be selected.

4.1. System Modeling

A system is defined as $S = 1, 2, ..., N$, a set of *N* cores. For each core, the following information is given:

- dimensions (approximate length and width);
- number of test inputs and outputs (for BIST, scan and external test);
- number of test vectors;
- maximum length in the scan chain;
- neighborhood: a core "neighborhood" is defined based on the placement information. The fact that a core *b* is a neighbor of core *a*, on either direction, means that either the wrapper of core *b* or the transparent mode of core *b* can be used to shortcut the path to/from core *a*

from/to the system interface. This information is intimately related to chip placement, since the location of the pins in the interface may have some restrictions. A core *b* may be a neighbor of core *a* in both directions, if no special rule needs to be observed in test pin placement.

The minimum test time for each core is calculated assuming an exclusive and full bitwidth TAM for each one. With this assumption, the test time is given by $t_i = (p_i + 1)L + p_i$, for $1 \leq i \leq N$, where p_i is the number of test patterns of core *i* [13]. If a core uses only external test ($L = 0$), the minimum test time reduces to p_i cycles.

For the system, the following information is required:

- routing distances between each core and from the core to the system boundaries.
- functional connectivity among the cores.
- cost and time constraints (number of extra pins allowed, test time limit, etc).
- optimization factors to be considered: area, pins, test time, power, etc.

4.2. The Algorithm

For the general case, where several cost factors are being considered at once, the algorithm is a search process that starts with the optimum solution for time (minimum test time for the system), with an exclusive TAM provided to the core that sets the time limit for the system. Then, every other core is placed in the schedule in such a way that the best solution for the defined time limit is found. If, after all cores are scheduled, the solution does not satisfy the system constraints, a small perturbation is done in the solution, so that the time limit is slightly increased and a new solution is sought. This process continues until the constraints are attained or until a preset number of iterations is completed.

Figure 4 shows the pseudocode for the implemented algorithm. A main loop controls the TAM definition and evaluates the cost of each global solution found, in steps 5 to 21.

The TAM for each core is represented by a tree on which the search algorithm for the shortest path takes place. The main difference of this search compared to similar approaches is that the tree is built on the fly (steps 11 to 14 in figure 4), according to the current test scheduling restrictions and available TAMs in the system. The cost of each arc in the tree is actually a set of costs, representing the optimization factors being considered. Each core has two subtrees representing the input and output paths, respectively, and each one is traversed independently. Since the tree represents a core-to-core connection, loops are prevented by the correct definition of CUT neighborhood so that a path to the system interface is always found.

One can observe that the kernel of the algorithm, presented at steps 11 to 16 in figure 4, is the possibility of

```
 1. Find core k with  t_k = max(t_i), i=1..n
 2. Schedule core k
 3. Set test time limit to t_k
 4. While (systems constraints not satisfied)
 5.   While (there are unscheduled cores)
 6.     select CUT = critical core for cost factors
 7.     For each direction of CUT ports
 8.       node = CUT
 9.       While (not in the interface) AND (accumulated cost <= current cost)
10.         neighbors = retrieve ( CUT neighborhood) AND (node neighborhood)
11.         For (each core in neighbors)
12.           time_slot = available time slot in the current schedule
13.           For (each modelled TAM)
14.             evaluate costs of TAM connection CUT−neighbor
15.           Select best TAM CUT−neighbor
16.         Select best neighbor for CUT
17.         accumulated cost += cost CUT−selected neighbor
18.         node = selected neighbor
19.         GO TO step 9.
20.     Insert CUT in the schedule
21.     Update conflict list
22.   IF (constraints not satisfied)
23.     set test time limit = infinity
24.     select CUT = critical core for cost factors
25.     Find a less expensive TAM to CUT
26.     Schedule CUT
27.     Set new test time limit for the system
28.     GO TO Step 4.
```

Figure 4. Pseudo-code of proposed model

choosing the best connection (step 15) among the modeled TAMs described in section 3, and the best point to advance the search towards the system interface (step 16). This range of possible decisions that can be evaluated and compared is the key for the fine grain search proposed. On the other hand, placement information can accelerate the search by indicating the most promising eligible neighbors for a core according to the location of the pins in the core periphery.

For each eligible neighbor retrieved in step 10 of the algorithm, the schedule is checked to define the amount of time available for the core under test, according to a list of conflicts over test resources. Cores that share a test resource cannot be placed in the same time slot in the schedule. Notice that, if an eligible core is not scheduled yet, a search in the schedule must ensure that a time slot remains for this neighbor to be placed subsequently. Since the cores are scheduled in a decreasing order of cost, less critical cores will have less impact on the global solution. Thus, they can keep a more expensive TAM (direct access, for example), while other cores are privileged.

The selection of which core is scheduled first is defined according to the cost of the current solution in terms of pins in the interface, and area and test time for each core. For different combinations of optimization factors, different costs are considered when selecting a core to be placed in the schedule. The most critical core (the most expensive one according to the cost factors being optimized) is selected from the list of unscheduled cores. Similarly, a TAM is defined first for the direction (input/output) that is using more

resources in the current solution.

The small perturbation in a current solution that allows a refined search for the global minimum, is shown at steps 22 to 27 of figure 4. It consists of selecting the most critical core using the same criteria used in step 6 of the algorithm. For this core, a new solution is searched as described at steps 7 to 19. However, at this point no scheduling checking is done (step 12). Any possible solution that reduces the cost of the available TAM for the critical core is considered. For test time minimization, the selection of the best neighbor at any level of the tree (step 16) is performed so that the neighbor that implies the smallest increment on the core test time is selected. Then, based on the conflicts of the current global solution, a new test time limit is calculated (step 27) such that current optimizations can be retained if necessary. The schedule is reinitialized and the search starts from the beginning (step 4).

5. Experimental Results

A prototype tool based on the described model was designed using Matlab version 6.0 [21]. Around 12500 lines of code implement the data structure and the search algorithm described in sections 3 and 4.2. In this paper, we describe the results provided by this tool for two representative examples.

5.1. Synthetic Benchmark

The first example considered here is the system presented in figure 2, from [13]. The main advantage of this example is its diversity in terms of number of test vectors and test requirements present among the cores. The availability of the test information in regards to these circuits is another factor that makes its use easier as a first benchmark for this kind of tool. On the other hand, the fact that no special function is realized by the defined system, and that the connections among the cores have been randomly assigned, is burdensome for tools that use this information.

The set of experiments performed for this example include a number of combinations of optimization factors, variations in the limit of extra test pins, and small variations in the system characteristics. Due to space limitations, only the more significant results are shown in table 1. The placement presented in figure 2 was used in all experiments. The first three rows represent the optimization of a single factor during the search, that is, only pin minimization, only area minimization and only time optimization. Even for this limited sample of the results obtained, one can see how this kind of comprehensive tool is suitable for a smooth design space exploration. From the first three experiments, we can note that the range for optimization is very large. Experiments 4, 5, 6 and 7 show the real trade-offs found by the tool.

The fourth experiment is the solution for the optimization of all cost factors with a limit as to the number of extra pins set to 48. These extra pins are used only for testing and do not include the pins for wrapper configuration.

If the system is such that more test resources are available, this information is used to improve the previous solution. For example, when increasing the limit of extra pins to 108, as shown in experiment 5, the area overhead is minimized. In this experiment, there are no restrictions regarding the location of the extra pins in the interface. Since less available pins need to be reused, TAMs could be defined by creating new pins closer to CUT than the available ones, reducing wiring area. Besides, test time reduces 56.2% for this experiment, as compared to the initial case (experiment 4).

In experiment 6 the pin limit is also set to 108, but area is not optimized. Note that a solution with fewer pins than the limit (105) is achieved, but the area increase from experiment 5 to experiment 6 is considerable, showing that this factor cannot be neglected during the search.

Experiment 7 shows how internal connections can influence the test decisions. For this experiment, the number of pins and internal connections for each core in figure 2 was reduced by half, making the system somewhat more realistic, since fewer internal pins remained unconnected. Then, for the same limit of extra pins (48), a solution with optimization of all factors was sought. The results for this experiment, in terms of area, are much better than in experiment 4. This shows that, in real systems, where internal connections are more regular, the area overhead is expected to be acceptable, as we show in the next example. Additionally, the number of pins used in this solution was reduced to 45 (4% less than the established limit).

Table 1. Test planning results for synthetic system

Experiment	Cost Factors	Test Time	No. of Pins	Area Overhead
1	only pins	544234	3	22%
2	only area	43663	1098	0.19%
3	only time	9634	1669	3%
4	all factors	42053	47	21%
5	all factors	18435	108	14%
6	no area	19714	105	48%
7	all factors	42053	45	15%

Comparing the results of experiment 6 with the results presented in [13], where only the test time and the number of pins are considered, the tool proposed here returns a test time only 4% higher than the test time achieved by the method proposed in [13]. However, we can see that, for the configuration of the cores presented in figure 2, the area overhead can be prohibitive when it is not considered.

We cannot compare the area of our solution with the one of [13], since no such results are presented therein. On the other hand, the area of experiment 6 is drastically reduced when this factor is also optimized, as shown in experiment 5. The penalty is a slight increase in the number of pins. Experiment 5 also shows that the proposed tool finds the same test time presented in [13] for the same number of pins, even when area is also optimized.

Despite the quality of the final test solution found by the proposed model, the iterativity of the method provides the system designer with partial solutions found during the search. With this information, it is possible to detect which cores are using more test resources and are the bottlenecks for the system. In this example, core 6 and core 10 appear as the two most critical cores in 54% and 46% of the partial solutions, respectively. Since core 6 is placed in the middle of the system in this example, the displacement of this core to the boundary could help reduce the area overhead. In terms of time, core 6 uses 37% of the total test time. This forbids its wrapper to be used by any other core during this time. However, if the test time of core 6 is internally optimized by reducing the number of test vectors or by using BIST, for example, the global solution can be highly improved.

5.2. Industrial Example

Figure 5 shows an instrumentation system for power measurement in industrial plants. This system was developed at UFRGS and is fully operational in the local industry.

The micro-processor is tested by 20 scan chains of length 52 each. Digital filters use around 3000 ATPG based vectors for testing. In the system configuration, the input is serial from the A/D converter and it is the same during test, since the filter assembles the data internally. A/D converters are assumed to have a BIST method for test. The memory has also an embedded BIST method that takes 262,144 cycles to execute. Core 9, a small watchdog implemented as a FSM, needs only 15 test vectors to be checked.

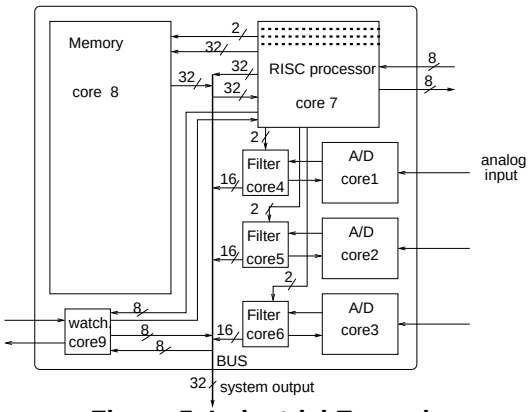

Figure 5. Industrial Example

The experiments with this example were performed to show how the system designer can use the test solution for making design decisions as well. In addition to the bus-based connection shown in figure 5, another configuration, without the bus and with direct connection between the cores was evaluated. The results are presented in table 2. All cost factors were set to be optimized, and the limit on the number of pins was set to 4 (less than 10% of system functional pins).

Table 2. Test planning results for industrial system

Experiment	Test Time	No. of Pins	Area Overhead
1 (bus)	262144	4	0.16%
2 (no bus)	262144	4	0.25%

The first conclusion from this experiment is that the system test time is defined mainly by the test of the memory. If the test of the A/D converters were considered, the system test time would be defined by them, since the sample ratio of each converter is much smaller than the system clock. If we consider, for example, a 10-bit converter running at 100Khz, the test time for 1024 test vectors (samples) would be 675,840 system cycles. The converters used on the system of figure 5 have 16 bits, which implies even higher test times. The test time for all other cores is 128,204 cycles. From the four pins included in the system, two are used by the scan chains of the micro-processor and the other two are used for the memory. Thus, all other cores could manage to use functional pins.

From the point of view of testing, this system is very limited by the test of the memory and probably by the test of the A/D converters. On the other hand, as an embedded application, area overhead is a very important factor for the final product. As the results show, the designer can decide for the cheapest solution without impacting the number of pins and test time.

The execution time for all experiments performed is around 20 minutes for the first example and 1 minute for the second one. However, since the prototype was developed in an interactive system (Matlab) where each function is interpreted before being executed, these times can be overestimating the real execution time of the program. Indeed, according to the profile run on the code, most of the execution time (39.6%) was spent by Matlab built-in functions used to access the data structure, while the search function itself used 32.6% of the running time. A much better performance in a tool implemented in a non-interpreted language can be easily expected.

6. Final Remarks

This paper presented an innovative model for test planning of core-based systems. The model is based on the definition of multiple TAMs inside the chip and on a fine-grained search algorithm for exploration of the design space. Experimental results have shown how different trade-offs can be achieved for a variety of system constraints, even when using the same set of embedded cores.

Although the proposed model is quite generic, contemplating different test methods (BIST, scan, external) and five types of TAM connections, some other extensions can still be studied. The possibility of exploring an independent path for each port of a core, for example, can optimize and take advantage of built-in features of each embedded block in the system, such as transparency and compaction modes. Besides, the reuse of available functional connections will also be intensified.

In addition to the current cost factors being considered (area, pins and test time), other ones can be easily inserted. Power consumption during test, for example, can be considered when searching a time slot in the available schedule. A core can only be part of a time slot if the power limit at that time is not exceeded.

In conclusion, the proposed model can be used as a powerful tool in the design process, by providing the designer with detailed information about test bottlenecks in the system. Moreover, the use of the proposed model in the core design process makes possible the evaluation of the impact of different test and design decisions in the core on a future system.

References

[1] L. Whetsel. An IEEE 1149.1 Based Test Access Architecture for ICs with Embedded Cores. In *International Test Conference*, pages 69–78, November 1997.

[2] D. Bhattacharya. Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit. In *16th IEEE VLSI Test Symposium*, pages 8–14, April 1998.

[3] P. Varma and S. Bhatia. A Structured Test Re-use Methodology for Core-based System Chips. In *International Test Conference*, pages 294–302, October 1998.

[4] E. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters. A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. In *International Test Conference*, pages 284–293, October 1998.

[5] Y. Makris and A. Orailoglu. RTL Test Justification and Propagation Analysis for Modular Designs. *Journal of Electronic Testing: Theory and Applications*, 13(2):105–120, October 1998.

[6] I. Ghosh, N. Jha, and S. Dey. A Low Overhead Design for Testability and Test Generation Technique for Core-based Systems. In *International Test Conference*, pages 50–59, November 1997.

[7] M. Benabdenebi, W. Maroufi, and M. Marzouki. CAS-BUS: a Scalable and Reconfigurable Test Access Mechanism for Systems on a Chip. In *Design, Automation and Test in Europe Conference*, pages 141–145, March 2001.

[8] E. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel. Towards a Standard for Embedded Core Test: an Example. In *International Test Conference*, pages 616–627, October 1999.

[9] J. Aerts and E. Marinissen. Scan Chain Design for Test Time Reduction in Core-based ICs. In *International Test Conference*, pages 448–457, October 1998.

[10] M. Sugihara, H. Date, and H. Yasuura. Analysis and Minimization of Test Time in a Combined BIST and External Test Approach. In *Design, Automation and Test in Europe Conference*, pages 134–140, March 2000.

[11] K. Chakrabarty. Design of System-on-a-chip Test Access Architectures Using Integer Linear Programming. In *18th IEEE VLSI Test Symposium*, pages 127–134, May 2000.

[12] M. Nourani and C. Papachristou. An ILP Formulation to Optimize Test Access Mechanism in System-on-chip Testing. In *International Test Conference*, pages 902–910, October 2000.

[13] V. Iyengar and K. Chakrabarty. Iterative Test Wrapper and Test Access Mechanism Co-optimization. In *5th IEEE Workshop on Testing Embedded Core-based Systems*, May 2001.

[14] Érika Cota, L. Brisolara, L. Carro, A. Susin, and M. Lubaszewski. MET: A Microprocessor for Embedded Test. In *5th IEEE Workshop on Testing Embedded Core-based Systems*, May 2001.

[15] R. Kapur, E. Marinissen, N. Mukherjee, M. Riccheti, T. Taylor, and J. Udell. P1500/D0.3. Technical report, IEEE P1500 Documentation Task Force, January 2001.

[16] S. K. Goel and E. J. Marinissen. TAM Architectures and Their Implication on Test Application Time. In *5th IEEE Workshop on Testing Embedded Core-based Systems*, May 2001.

[17] A. Benso, S. Chiusano, S. Di Carlo, P. Prinetto, F. Ricciato, M. Spadari, and Y. Zorian. HD2BIST: a Hierarchical Framework for BIST Scheduling, Data Patterns Delivering and Diagnosis in SoCs. In *International Test Conference*, pages 892–901, October 2000.

[18] M. Nourani and C. Papachristou. Structural Fault Testing of Embedded Cores Using Pipelining. *Journal of Electronic Testing: Theory and Applications*, 15(1-2):129–144, Aug-Oct 1999.

[19] E. Larsson and Z. Peng. An Integrated System-on-Chip Test Framework. In *Design, Automation and Test in Europe Conference*, pages 138–144, March 2001.

[20] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Iterative Test Wrapper and Test Access Mechanism Co-optimization. In *International Test Conference*, pages 1023–1032, October 2001.

[21] MathWorks. *MATLAB: the Language of Technical Computing*. MathWorks, Natick, MA, 1997.