

# A Heuristic for Test Scheduling at System Level

Marie-Lise Flottes, Julien Pouget, Bruno Rouzeyre

LIRMM – UMR CNRS / University of Montpellier, 161 Rue Ada – 34392 Montpellier Cedex 5

Tel: +33 467 418 577 – Fax: +33 467 418 500 – Email: [rouzeyre@lirmm.fr](mailto:rouzeyre@lirmm.fr)

## Abstract

*This paper considers the test-scheduling problem of a SoC. The proposed approach is based on a "sessionless" test scheme. It minimizes the system test time while respecting a power dissipation limit and test resource sharing constraints. Experimental results show that our approach outperforms other related test scheduling solutions.*

## 1. Test schemes

The basic approach in test scheduling is to organize tests for the cores into so-called test sessions. The test session ends after completion of the longest test in that session (variable length test sessions) or in the system (fixed length test sessions). We can easily foresee that these approaches are not optimal in terms of test time. Thus, another approach consists in starting the test of a core as soon as possible without regards for test completion on other cores except if some resource or power dissipation constraints prevent their parallelism (sessionless scheduling). As an example, we implemented three controllers with reference to the three scheduling methods for the test control of a SoC example. The result shows that the extra costs attached to the sessionless scheme are insignificant compared to global system cost. This approach is thus conceivable.

## 2. The heuristic

We propose an algorithm that solves the sessionless based scheduling problem in relatively short time. The complexity of the proposed algorithm is  $O(n^3)$  where  $n$  is the number of cores in the system under test. The proposed algorithm returns the test start date  $T_i$  for every core  $i$ , and the total test time for the system:  $T_{max}$ :

$L1$  = list of cores sorted by decreasing  $D_i$  values

$L2 = \emptyset$

$T_{max} = 0$

While  $L1 \neq \emptyset$

$T_{max} = D_i + \text{Place}(\text{first core in } L1)$

    For all other cores  $i$  in  $L1$

$T_i = \text{Place}(i)$

        if  $T_i + D_i > T_{max}$

            remove  $i$  from the placed cores

$L2 = L2 \cup \{i\}$

$L1 = L2$

$D_i$  is the test-time for core  $i$ , it is expressed in terms of clock cycles. List  $L1$  keeps track of cores not already tested in the present partial scheduling solution. The

function  $\text{Place}(i)$  positions  $i$  as soon as possible i.e. it computes the earliest date  $T_i$  to start the test on core  $i$  taking into account that for any other already placed core  $j$  such that  $i$  and  $j$  tests overlap (  $(T_j \leq T_i$  and  $T_j + D_j > T_i)$  or  $(T_i \leq T_j$  and  $T_i + D_i > T_j)$  ), first, the power limit is respected ( $\sum P_j + P_i < P_{max}$ ), and second, the resources involved in  $i$  and  $j$  tests do not conflict. At every iteration, the algorithm differs the placement of core  $i$  in the partial solution if its test increases the total system test time ( $T_i + D_i > T_{max}$ ).

## 3. Experimental results

We compare our technique with the solution proposed by [1]. The authors report a total test time of 7985 cycles for the SoC example while our heuristic leads to a total test time of 6809 clock cycles. We obtain this result with one second of computation time while an exhaustive solution would lead to excessive CPU time if the number of cores on the system exceeds 10. Moreover, the CPU time is in the order of the second for about 100 cores in a SoC. Thus, we can foresee using our tool to explore different solutions.

The proposed algorithm can also with precedence constraints. It leads the algorithm to postpone a test if the ones that must be applied first have not been yet scheduled. This new constraint may lead to postpone the test of the first element in  $L1$ . Consequently, the first test in  $L1$  that can be scheduled (not necessarily the first element in list  $L1$ ) is used to increase  $T_{max}$ .

## 4. Conclusion

We have presented an efficient scheme for organizing the test at system level and a corresponding power constrained test scheduling algorithm. This approach outperforms classical ones, which are based on test sessions. Finally, present work assumes that the test architecture is fixed before test scheduling. We expect a better test area and test time tradeoff by assigning dynamically the test resources when needed.

## 5. References

[1] K. Chakrabarty, "Design of system-on-a-chip test access architectures using integer linear programming", Proc. VLSI Test Symp., pp 127-134, 2000.