Integrated Test Scheduling, Wrapper Design, and TAM Assignment for Hierarchical SOC

Haidar M. Harmanani and Rana Farah
Department of Computer Science and Mathematics
Lebanese American University
Byblos, 1401 2010, Lebanon

***Abstract*— System-On-Chip (SOCs) test minimization has received a lot of attention in the past few years. However, most recent work assumed flat hierarchy. This assumption is unrealistic especially in the case of *non-mergeable* legacy cores that have been placed and routed. This paper presents an efficient approach for test scheduling hierarchical core-based systems based on simulated annealing. The method minimizes the overall test application time while performing wrapper design and TAM assignment. We present experimental results for various SOC examples that demonstrate the effectiveness of our method.**

I. Introduction

The design and manufacturing methods of integrated circuits have known advances allowing the creation of a complete system on a single die. In order to keep pace with such advances, hardware engineers have adopted reuse methodologies where *predesigned* and *preverified* intellectual property blocks are embedded on a single chip (SOC) [10]. A core maybe *soft*, *firm*, or *hard*. Typically, soft and firm cores are *mergeable* with their surrounding logic while hard cores are usually placed and routed and are consequently *non-mergeable*. A core maybe designed in a hierarchical fashion and thus embeds other cores. Hierarchical cores have multiple levels of test hierarchy where a level corresponds to the depth in the test hierarchy tree. The top level is the SOC itself and consists of several *mega-cores* that have their own non-mergeable embedded cores. Cores at level n are called parent cores with respect to child cores at level $n+1$ [11]. A major challenge in the SOC design paradigm involves integrating IP blocks and developing a test methodology for post-manufacturing tests in addition to test time reduction by maximizing the simultaneous test of all cores. The problem, known as test scheduling, determines the order in which cores are tested and is equivalent to the $\mathcal{NP}$-complete *m-processor open shop scheduling* problem [1].

Simulated Annealing is a global stochastic method that is used to generate approximate solutions to combinatorial problems. The algorithm begins with an initial feasible configuration and generates a neighboring solution by perturbing the current one. The neighboring solution is accepted if its cost is less than that of the current solution; otherwise, it is accepted or rejected with a probability which is a function of the *temperature*, T. The temperature is decreased during the optimization process following a *cooling schedule*.

Several researchers addressed the test scheduling problem but mostly assumed flat cores with a single level TAM [15]; however, this assumption is only valid if the embedded cores are *mergeable*. Iyengar et al. [6] first formulated the integrated wrapper/TAM optimization problem using ILP and developed later several heuristic techniques using rectangle packaging [7]. Goel et al. [4] proposed an efficient heuristic for fixed-width architecture while Huang et al. [5] modeled the problem as a restricted 3-D bin packing problem and proposed a heuristic to solve it. Su et al. [12] formulated the problem using graph-based approach and solved it using tabu search. Zou et al. [16] used sequence pairs to represent bins placement and optimized the test schedule using simulated annealing; however, the approach is not constraint-driven. Pouget et al. [9] proposed a constraint-driven wrapper design and test scheduling. However, the method did not support hierarchical SOCs. Recently, Chakrabarty et al. [3] proposed a combination of integer linear programming, enumeration and efficient heuristics in order to solve the problem for hierarchical cores. Xu et al. [14] proposed a method for design space exploration of multi-level test access mechanism that facilitates test data reuse for hard mega-cores in hierarchical SOCs.

II. Problem Description

The test time of a SOC is based on the individual cores test time as well as on the determination of test start times. The cores test times are based on TAM assignments as well as on the wrapper design. The number of wrapper scan chains should be equal to the TAM width provided to the core. Typically, test adaptation is performed by serially connecting internal core scan-chains, the wrapper input cells and the wrapper output cells in order to form wrapper chains [2]. The length of the wrapper chains directly affects the core's test time since short wrapper chains lead to a short loading/unloading time. The number of wrapper chains, on the other hand, affects the number of TAM bits as each wrapper chain's input and output must be connected to a TAM wire. Thus, there is a clear trade-off between test time and the TAM capacity. However, increasing the number of TAM bits may not necessarily guarantee decreasing test time as the test time may hit a Pareto optimal point [7]. Furthermore, embedding cores adds an additional test conflict problem arising from the fact that it may not be possible to test the parent and the child cores concurrently due, for example, to a conflict in the use of the input wrapper cells. Therefore, an effective test scheduling approach must minimize the test time while

1-4244-1176-9/07/$25.00 ©2007 IEEE.

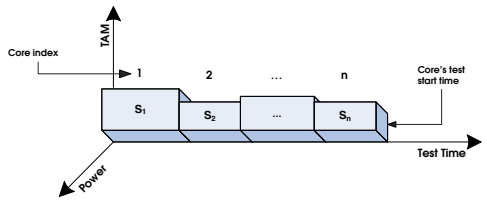

Fig. 1. Configuration representation

Assign_TAM(Core i)
{
TAM = number of scan-chains + $\frac{\max(p_i, p_o)}{\text{longest scan chain}}$
if (TAM $>$ maxTAM)
TAM = maxTAM
if the core is not level 0 and has less than two peers, then TAM = TAM$_{parent}$
if the core is at level n and has more than one peer then
TAM = 0.5 * TAM$_{parent}$
If a core is dominant, TAM = TAM + number of peer cores.
if a core is *test dominant* then TAM = TAM + number of peer cores
}

Fig. 2. Hierarchical TAM assignment algorithm

addressing test resources allocation and conflicts arising from the use of shared test access mechanism.

This paper presents an efficient method for integrated test minimization, TAM assignment and wrapper design for hierarchical SOCs using simulated annealing. Formally, given a SOC with N_C cores, a total TAM width $\mathcal{W}$, a set of design and test constraints, and a set of parameters for each mega-core, the problem we address in this paper is to minimize the overall test time such that, (i) the test schedule for the entire SOC is efficient, (ii) the TAM wires are optimally partitioned and assigned to cores (iii) the wrapper configuration for each core is determined, (iv) $\mathcal{W}$ is not exceeded, and (v) hierarchical cores receive at least their prespecified TAM widths.

The remainder of the paper is organized as follows. Section 3 formulates the *hierarchical SOC test scheduling problem* and describes the annealing configuration, the initial solution, the neighborhood functions, and the *test scheduling algorithm*. We conclude with *experimental results* in section 4.

III. Integrated Test Scheduling Algorithm

The proposed test scheduling method starts with a set of cores and generates, through a sequence of transformations, a set of compact and efficient test schedules. We explore design trade-offs by integrating test scheduling, wrapper design and TAM assignment into a single problem that we solve using a multi-objective simulated annealing algorithm that supports both flat and hierarchical SOC test scheduling.

A. Configuration Representation

The test minimization problem is mapped to an initial configuration based on a vector where every cell corresponds to a 3-D core as shown in Figure 1. The first dimension corresponds to a specific *test start time*, S_i, the second dimension corresponds to TAM width and the third dimension corresponds to the maximum power that the core may consume. The core *test finish time*, F_i, is equal to the test start time plus the core test time, T_i, that is, $F_i = T_i + S_i$. Note that the test start time, S_i, is not constant and it changes to the end times of other cores as the algorithm explores the neighborhood of the solution.

B. Initial Configuration

The initial configuration is chosen by assigning each core one bit TAM and computing the theoretical test time lower bound, LB_T, in order to guide the test minimization process. The wrapper input cells, the core scan-chains, and the wrapper output cells are serially connected in order to form the minimum number of balanced wrapper chains and are adapted to the number of assigned TAM bits using the BFD algorithm. The test time for core i is determined based on $t_i = (1 + \max(s_i, s_o)) \times p + \min(s_i, s_o)$ where, s_i and s_o denote respectively the scan-in and scan-out time for the core and p_i denotes the number of test patterns applied on the core i [8]. The algorithm ensures that all cores test times are less than the lower bound by allocating additional TAM bits. The initial test schedule is next chosen to be a serial schedule. However, when considering precedence constraints, the initial solution is chosen based on the topological sorting of all tests represented in the precedence constraint graph. The result of our topological sorting is a valid serial solution when precedence constraints are incorporated.

C. Neighborhood Functions

The algorithm iterates exploring test scheduling possibilities, TAM assignment, and wrapper configuration using three neighborhood solutions. The neighborhood functions are applied while ensuring precedence and concurrency constraints.

1) *Test Schedule Exploration* where the algorithm randomly selects a random core i from the current configuration and changes its starting time S_i to the *end time* F_j of a randomly chosen core j ($i \neq j$), $0 \leq i, j \leq N_c$. If core j is selected such that $j = N_c + 1$, then the start time of core i is set to be a 0.
2) *Test Schedule Swap* where the algorithm randomly selects two cores and swaps their test start time.
3) *TAM Exploration* where the algorithm finds all testing paths in the schedule such that $\sum C_j > LB_T$ where C_j is the test time for core j and LB_T is the lower bound. The algorithm allocates additional TAM bits to all cores until they reach the next Pareto-optimal point and selects the core that has the highest rate of test time improvement with respect to TAM bits.

The neighborhood functions are followed by two deterministic transformations that compact the schedule by removing idle slots.

D. Hierarchical Test Scheduling

The hierarchical core model is a recursive model where a wrapped parent core has an external TAM that connects externally to the parent core and an internal TAM that consists of the TAM architecture of the children cores. We start with

```
Annealing_TestScheduling()
{
   MaxTime = Total allowed time for the annealing process
   Assign one TAM bit to each core. Compute the Core test time, C_i.
   Design the Core Wrapper using the BFD algorithm.
   do {
        M = M_0
        do {
             if (iteration % 10 == 0)
                 NewS = TestScheduleSwap(CurrentS);
             else if (iteration % 11 == 0)
                 NewS = TAMExplore(CurrentS);
             else NewS = TestScheduleExplore(CurrentS);
             NewCost=Cost(NewS)
             Δ_Cost = NewCost − CurrentCost
             if (Δ_Cost < 0)
             {
              CurrentS=NewS
              CurrentCost=Cost(CurrentS);
              If (NewCost < BestCost) then
                 BestS=NewS
                 BestCost = Cost(BestS)
             }
             else if (Random < e^(−Δ_Cost/T) then      // T = temperature
              CurrentS=NewS
              CurrentCost = Cost(CurrentS);
             M = M − 1
        } while (M ≥ 1)          // M is time until next parameter update
        Time = Time + M_0;
        T = α *T;                          // α = Cooling rate
        M_0 = β * M_0;                     // β = Iteration multiplier
   } while (Time < MaxTime);
   CompactSchedule(BestS);
   Return(BestS);
}
```

Fig. 3. Annealing SOC test scheduling algorithm

```
Hierarchical_Test_Schedule()
{
   for all levels starting from level n − 1 to level 0
        find all mega cores
        for each mega core
             assign TAM bits for the children cores
             calculate the test time for each child core
             use simulated annealing to find the schedule for the children cores
             calculate overall test time
        if (level == 0)
             calculate the overall time of the SOC.
   for (i = 0; i < n; i++)
        Select a random core i from any level
        Randomly increase/decrease TAM bits in order to move to the
          next/previous Pareto-optimal point.
¿  If the transformation is accepted then
             simulated annealing();
        Update T_i and all parents cores that contain this core.
}
```

Fig. 4. Hierarchical test scheduling algorithm

the *mega-cores* that are at level n where each mega-core is considered as a separate SOC. The embedded child cores are initially assigned TAMs based on the algorithm shown in Figure 2 where we define a dominant core as a core such that when allocated the same TAM width as all its peers it returns a larger test time than its peers. For cores that are at level n and that have more than one peer, the algorithm allocates half the TAM bits of the parents to the child cores. However, the number of allocated TAM bits may change in the iterative part of the annealing process. During the TAM assignment process, the algorithm ensures that no mega-core is assigned more than the specified TAM bits and that no core is assigned more than the TAM bits of its parent. Once the TAM assignment is performed, the wrapper design is determined using the BFD algorithm.

Once the TAM has been assigned, the algorithm iterates over each mega-core using our simulated annealing in order to determine the *test schedule* as well as the wrapper design for the mega-core itself (Figure 4). On the other hand, test data serialization maybe necessary since a mega-core width maybe assigned less than the TAM width it requires. Thus, a mega-core i with top-level TAM width w_i maybe provided with w_i^* TAM bits such that $w_i^* \leq w_i$. The test time for mega-core i then reduces to $\mathcal{T}_i^* = \lceil \frac{w_i}{w_i^*} \rceil * \mathcal{T}_i$ where $\mathcal{T}_i$ is the total testing time for the embedded cores on the internal TAM partition for mega-core i. Finally, the testing time for mega-core i must include the test time for the top-level test time and this reduces to: $\mathcal{T}_i^{W_i^*} = \mathcal{T}_i^* + \mathcal{T}_i^{Mega}$ where $\mathcal{T}_i^{W_i^*}$ is the testing time for mega-core i with an external TAM of width w_i^* and $\mathcal{T}_i^{Mega}$ is the testing time at the system level TAM architecture. Once all cores at level i have been processed, the algorithm recurses back to level $i-1$ and considers the remaining mega-cores. The process repeats until all cores have been scheduled.

The last step of the algorithm is to explore further improvements by iterating over the schedule for n iterations ($n = 100$). During each iteration, the algorithm selects a random core from any level of the SOC hierarchy and applies one of two transformations based on a random probability. The first frees m TAM-bits by moving the core's test time to the previous Pareto-optimal point. The second reduces the core's test time by adding p TAM-bits and moving to the next Pareto-optimal point. If the move results with an improvement, then the SOC is test scheduled using the annealing algorithm. If the affected core is a child core then all its parents are updated accordingly.

IV. Results

We ran the test scheduling algorithm for flat as well as for hierarchical SOCs. The ITC'02 benchmark suite has only four benchmarks with a hierarchical structure which are shown in Table I and are compared to [3], [13], [14]. The TAM widths supplied to the mega-cores were fixed at 8 bits for SOCs *p22810* and *a586710*, and at 16 bits for SOCs *p34392* and *p93791*. As it is shown, our system clearly outperforms other systems in almost all attempted cases in a very short time. For flat SOCs, the algorithm assumes that all cores are at the same level with the ITC benchmark results for flat SOCs shown in Table II. The CPU time did not exceed 1.31 minutes. The simulated annealing cooling schedule was experimentally determined as follows: the temperature reduction multiplier, $\alpha = 0.99$ and T_{init} = 4000. The stopping criterion was $T \leq 0.001$ while the number of iterations, M, was set to 5 and the the iteration multiplier, β, to 1.05.

References

[1] K. Chakrabarty. Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming. *IEEE Trans. on CAD*, 19(10):1163–1174, 2000.

SOC	$\mathcal{W}$	[14]	[13]	[3]	Ours
a586710	16	5.27×10^7	–	4.44×10^7	**42117546**
	24	3.06×10^7	–	3.06×10^7	**22973206**
	32	2.19×10^7	–	2.28×10^7	**21058772**
	40	1.91×10^7	–	2.50×10^7	**17229905**
	48	1.53×10^7	–	2.14×10^7	**13401037**
	56	–		2.14×10^7	**13031723**
	64	1.41×10^7		2.14×10^7	**13031723**
p93791	16	**1865140**	–	–	1953223
	24	1486628	–	1650880	**1377332**
	32	1486628	1937130	1021320	**991915**
	40	**801271**	–	916852	845391
	48	801271	1272314	681816	**670660**
	56	–	–	632125	**629510**
	64	553775	947554	**521064**	545583
p22810	16	496804	–	505858	**496146**
	24	353619	–	412682	**334202**
	32	**280634**	466044	396473	**280634**
	40	**280634**	–	366260	**280634**
	48	**280634**	338374	366260	**280634**
	56	–	–	–	**280634**
	64	**280634**	285231	366260	**280634**
p34392	16	1467705	–	–	**1188916**
	24	1467705	–	1347023	**778273**
	32	776537	–	788873	**713071**
	40	776537	–	728426	**606261**
	48	**60261**	–	618597	**606261**
	56	–		618597	**606261**
	64	**60261**	–	618597	**606261**

TABLE I

HIERARCHICAL ITC'02 SOC RESULTS COMPARISONS

[2] K. Chakrabarty, V. Iyengar, and A. Chandra. *Test resource Partitioning for System-On-a-Chip*. Kluwer Academic Publishers, 2002.

[3] K. Chakrabarty, V. Iyengar, and M. Krasniewski. Test Planning for Modular Testing of Hierarchical SOCs. *IEEE Trans. CAD*, 24(3):435–448, 2005.

[4] S.K. Goel and E.J. Marinissen. SOC Test Scheduling Design for Efficient Utilization of Bandwidth. *ACM TODAES*, 8(4):399–429, 2003.

[5] Y. Huang, S.M. Reddy, W.-T. Cheng, P. Reuter, N. Mukherjee, C. Tsai, O. Samman, and Y. Zaidan. Optimal Core Wrapper Width Selection and SOC Test Scheduling on 3-D Bin Packing Algorithm. In *Proc. ITC*, pages 74–82, 2002.

[6] V. Iyengar, K. Chakrabarty, and E. Marinissen. Test Wrapper and Test Access Mechanism Co-Optimization for System-On-a-Chip. *JETTA*, 18:213–230, 2002.

[7] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization. In *Proc. VTS*, pages 253–258, 2002.

[8] E.J. Marinissen, S.K. Goel, and M. Lousberg. Wrapper Design for Embedded Core Test. In *Proc. ITC*, pages 911–920, 2000.

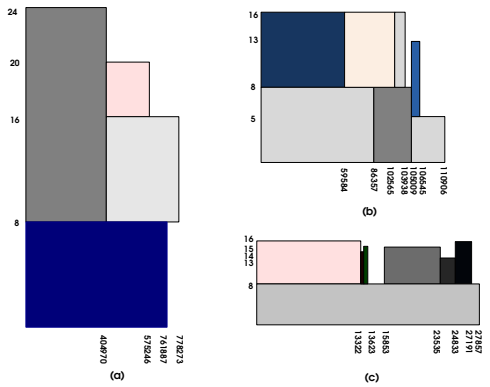

Fig. 5. a) Hierarchical test schedule for p34392; b) Mega-core 2 test schedule; c) Mega-core 1 test schedule

SOC	$\mathcal{W}$	[4]	[5]	[12]	[16]	[9]	Ours
d695	16	44307	42716	41905	41604	41847	42382
	24	28576	28639	28231	28064	29106	29103
	32	21518	21389	21467	21161	20512	21456
	40	17617	17366	17308	16993	18691	17480
	48	14608	15142	14643	14182	17257	14779
	56	12462	13208	12493	12085	–	12708
	64	11033	11279	11036	10723	13348	11069
g1023	16	34459	31444	32602	31139	-	31777
	24	22821	21409	22005	21024	-	20261
	32	16855	16489	17422	15890	-	16332
	40	14794	14794	14794	14794	-	14794
	48	14794	14794	14794	14794	-	14794
	56	14794	14794	14794	14794	-	14794
	64	14794	14794	14794	14794	-	14794
p22810	16	458068	446684	465162	438619	473418	464809
	24	299718	300723	317761	289287	352834	310456
	32	222471	223462	236796	218855	236186	233101
	40	190995	184951	193696	175946	195733	197566
	48	160221	167858	174491	147944	159994	166169
	56	145417	145087	155730	126947	–	145417
	64	133405	128512	145417	109591	128332	123543
p34392	16	1010821	1016640	995739	944768	-	940626
	24	680411	681745	690425	628602	-	637263
	32	551778	553713	544579	544579	-	544579
	40	544579	544579	544579	544579	-	544579
	48	544579	544579	544579	544579	-	544579
	56	544579	544579	544579	544579	-	544579
	64	544579	544579	544579	544579	-	544579
p93791	16	1791638	1791860	1767248	1757452	1827819	1777840
	24	1185434	1200157	1178776	1169945	1220469	1204168
	32	912233	900798	906153	878493	945425	943087
	40	718005	719880	737624	718005	787588	760868
	48	601450	607955	608285	594575	639217	638993
	56	528925	521168	539800	509041	–	557890
	64	455738	549233	485031	447974	457862	489591
u226	16	18663	13416	18663	13333	–	13333
	24	13331	10750	14745	8084	–	8084
	32	10665	6746	10665	6746	–	6746
	40	8084	5332	8084	5332	–	5332
	48	7999	5332	7999	5332	–	5332
	56	7999	4080	7999	4080	–	4080
	64	7999	4080	7999	4080	–	4080
f2126	16	372125	357109	357089	357088	–	357088
	24	335334	335334	335334	335334	–	335334
	32	335334	335334	335334	335334	–	335334
	40	335334	335334	335334	335334	–	335334
	48	335334	335334	335334	335334	–	335334
	56	335334	335334	335334	335334	–	335334
	64	335334	335334	335334	335334	–	335334
t512505	16	10530995	10531003	11210100	10530995	–	10530995
	24	10453470	10453470	10525823	10453470	–	10453470
	32	5268868	5268872	6370809	5268868	–	5268868
	40	5228420	5228420	5240493	5228420	–	5228420
	48	5228420	5228420	5239111	5228420	–	5228420
	56	5228420	5228420	5228474	5228420	–	5228420
	64	5228420	5228420	5228489	5228420	–	5228420
a586710	16	41523868	42198943	42067708	32626782	–	32417445
	24	28716501	27785885	27907180	23413604	–	22973206
	32	22475033	21735586	22704821	18838663	–	17195389
	40	19048835	19041307	19041307	14260216	–	14249791
	48	15315476	15071730	15212440	12811087	–	12811087
	56	13415476	14945057	13401034	12573448	–	1148660
	64	12700205	12754584	11567464	10659014	–	9572169
d281	16	8444	7948	8156	7946	–	7347
	24	6408	5486	5830	5485	–	4992
	32	5084	4070	4640	4070	–	3926
	40	3964	3926	3926	3926	–	3926
	48	3926	3926	3926	3926	–	3926
	56	3926	3926	3926	3926	–	3926
	64	3926	3926	3926	3926	–	3926

TABLE II

FLAT ITC'02 SOC RESULTS COMPARISONS

[9] J. Pouget, E. Larsson, and Z. Peng. Multiple-Constraint Driven System-on-Chip Time Optimization. *JETTA*, pages 599–611, 2005.

[10] R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. lemieux, P. Pande, C Grecu, and A. Ivanov. System-on-Chip: Reuse and Integration. *Proceedings of the IEEE*, 94(6):1050–1069, 2006.

[11] A. Sehgal, S.K. Goel, E.J. Marinissen, and K. Chakrabarty. IEEE P1500-Compliant Test Wrapper Design for Hierarchical Cores. In *Proc. ITC*, pages 1203–1212, 2004.

[12] C. Su and C. Wu. A Graph-Based Approach to Power-Constrained Test Scheduling. *JETTA*, 20:45–60, 2004.

[13] T.-P. Wang, C-Y. Tsai, M.-D. Shieh, and K.-J. Lee. Efficient Test Scheduling for Hierarchical Core Based Designs. In *Proc. VLSI-TSA-DAT*, pages 200–203, 2005.

[14] Q. Xu and N. Nicolici. Time/Area Tardeoffs in Testing Hierarchical SOCs with Hard Mega-Cores. In *Proc. ITC*, pages 1196–1202, 2004.

[15] Q. Xu and N. Nicolici. Resource-Constrained System-On-a-Chip Test: A Survey. *IEE Proceedings*, 152(1):67–81, 2005.

[16] W. Zou, S. R. Reddy, I. Pomeranz, and Y. Huang. SOC Test Scheduling Using Simulated Annealing. In *Proc. VTS*, pages 325–330, 2003.