SOC Test Time Minimization Under Multiple Constraints

Julien Pouget, Erik Larsson, and Zebo Peng
Embedded Systems Laboratory,
Computer Science Department
Linköpings Universitet, Sweden

Abstract[1]

In this paper, we propose a SOC (system-on-chip) test scheduling technique that minimizes the test application time while considering test power limitations and test conflicts. The test power consumption is important to consider since exceeding the system's power limit might damage the system. Our technique takes also into account test conflicts that are due to cross-core testing (testing of interconnections), unit testing with multiple test sets, hierarchical SOCs where cores are embedded in cores, and the sharing of test access mechanism (TAM). Our technique handles these conflicts as well as precedence constraints, which is the order in which the tests has to be applied. We have implemented our algorithm and performed experiments, which shows the efficiency of our approach.

1. Introduction

Test time minimization is a major problem when developing a system-on-chip (SOC) test solution. Long testing times are due to excessive test data volumes, a direct consequence of high design complexity and the use of core-based SOC design methodology, where pre-defined logic blocks, cores, are integrated with UDL (user-defined logic) to form a system.

An efficient test schedule can reduce the testing time by allowing tests to be executed concurrently. However, executing tests concurrently increases the activity in the system, which leads to higher power consumption. It is important that the test power constraints are not violated since it might damage the system. Furthermore, test conflicts such as cross-core testing (interconnection testing), unit testing with multiple test sets, hierarchical SOCs where cores are embedded in cores, and the sharing of test access mechanism (TAM) wires, must be considered during the test scheduling process.

Several approaches have been proposed for SOC test scheduling [1,2,3,4,5,6,7,9,10,11,12,14]. The basic problem is to minimize the test time for a design where the test sets are stored in an Automatic Test Equipment (ATE) and the main limitation is the number of available pins in the system. Goel and Marinissen, for instance, proposed, for systems where each core has a dedicated wrapper, a technique that schedules the test data transportation on the TAM wires in such a way that the total test application time is minimized. Huang *et al.* proposed also a method to address the test power consumption [3], where the test time for a system with wrapped cores is minimized while test power limitations are considered and tests are assigned to TAM wires. Recently, Iyengar *et al.* proposed a scheduling technique minimizing the testing time while taking hierarchical constraints into account [8]. We have in our previous work considered design hierarchy constraints, power limitations, precedence constraints, multiple test set and interconnection test [14]. However, the wrapper design and the test scheduling were considered as two sequential steps, which has the consequence that even if locally optimal wrapper configurations are selected a global system optimum is usually not achieved.

In this paper we address the SOC test scheduling problem by proposing a test scheduling technique that minimizes the test application time while considering test power consumption and test conflicts. In our approach we take power constraints and hierarchical constraints into account. We allow the cores to be tested by multiple test sets. It means that a core can for instance be tested using one test set generated by an LFSR and one test set stored in the ATE. In our approach we also consider cross-core (interconnection) testing, which is the testing of logic and interconnections placed between wrapped cores. We also take precedence constraints into account which is important when a particular order has to be enforced between some of the tests. The main advantage of our proposed approach, compared to our previous, is that we integrate the wrapper design algorithm with the test scheduling algorithm, which makes it possible to explore the design space in a more efficient way since we do not fix the wrapper configuration prior to the scheduling.

The rest of the paper is organized as follows. In Section 2, we give the background to the problem and formulate precisely the problem. Our combined wrapper design and test scheduling approach is then described in Section 3. The experimental results are reported in Section 4 and the paper is concluded with conclusions in Section 5.

2. Background and Problem Formulation

In this section we give the background and our problem formulation. Let us consider a core-based system as given in Figure 1. Such a system is said to be testable if every

1. The research is partially supported by the Swedish National Program on Socware.

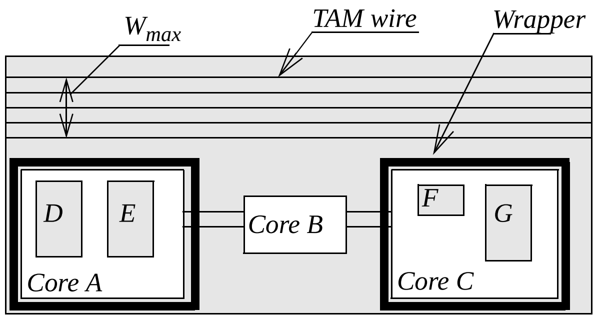

Figure 1. A core-based design with a TAM, cores in wrappers (Core A and core C) and hierarchy (core D and E are embedded in core A and core F and G are embedded in core C).

testable unit in the system is equipped with a test method and corresponding test sets. A testable unit can be a core, UDL, or interconnections. It is also assumed that a set of pins can be used for the TAM (the total number of wires in the TAM is denoted by W_{max} in Figure 1) and in order to connect the cores to the TAM some cores are equipped with wrappers.

The problem we focus on is basically how to assign a start time, an end time and if needed a set of TAM wires for each test in such a way that the total test time is minimized. The assignment should consider the conflicts discussed below.

A wrapper is the interface between a core and the TAM and it can normally be in one of the following modes at a time: *normal operation mode*, *internal test mode*, *external test mode*, or *bypass mode*. Some cores are equipped with wrappers while others are not. In order to access test data on the TAM, a wrapper must be used. If a testable unit does not have its own wrapper, some other wrapper must be used. For example, in order to test core B in Figure 1 with an ATE stored test, the wrapper at Core A can be used to feed test stimuli to core B and Core C can be used to receive test responses from core B. Note, that since a wrapper can be only in one mode at a time, testing of Core B cannot be performed concurrently with the testing of core A and core C, since there is a wrapper conflict. We call this type of testing cross-core testing.

Another conflict illustrated in Figure 1 is the design hierarchy conflict. The two cores named F and G are embedded within core C. Such embedding of cores leads to test conflicts since concurrent testing of core F and/or core G with core C is not possible.

Each testable unit can be tested by one or more test sets. If more than one test set exists for a testable unit, there is a test conflict since only one test set can be applied at a time to a testable unit. The test time at a testable unit can often be modified. An example of such is the scan-tested core given in Figure 3 where the scan-chains and the wrapper cells are configured into two wrapper-chains. A higher number of wrapper-chains reduces the testing time at the expense of

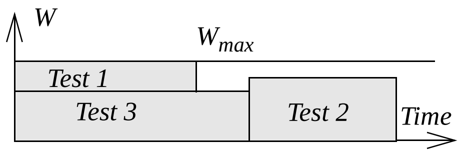

Figure 2. TAM wire-constrained test scheduling.

more TAM wires and vice versa. Iyengar *et al.* showed that the problem is NP-hard [6]. A wrapper design algorithm computes the test time at a given number of wrapper chains. A Pareto-optimal point is a configuration where there exists no lower testing time for a lower number of wrapper chains.

We assume that a test set for a testable unit is either stored in an ATE or generated at a dedicated BIST engine placed at the testable unit. It means that if a testable unit is tested by only a BIST test set there is no need to make use of TAM wires. On the other hand, for a test stored at the ATE, TAM wires are required for the transportation of test stimuli from the ATE via the TAM to the testable unit and TAM wires are required for the transportation of test response from the testable unit to the ATE. At any time, only one testable unit can use a wire. There is a sharing conflict, which is illustrated in Figure 2. Figure 2 illustrates the assignment of TAM wires to three tests over time. Each test is assigned to TAM wires for a certain period of time. In terms of TAM costs, at the moment and for small systems, the area overhead induced by the bus based TAM we propose might be negligeable compared to the cores total area. With the apparition of more complex systems and larger SoCs (up to 100 cores), the SoC concept should move to a Network On Chip one (NoC) and the wiring additionnal area should be considered. In our approach, we only provide as a TAM information the number of wires, considered as an upper bound to schedule the tests considering all the constraints. We do not provide more details on the wiring area of the TAM because we do not propose a physical implementation of the test architecture. Therefore, we do not consider the routing cost.

The execution of a test results in switching activity, which consumes power. Figure 4 shows the execution of a test and its power consumption. The power varies over time. However, to simplify the analysis, we will assume a fixed power value attached to each test. The total power consumed by a system under test at a certain point is the

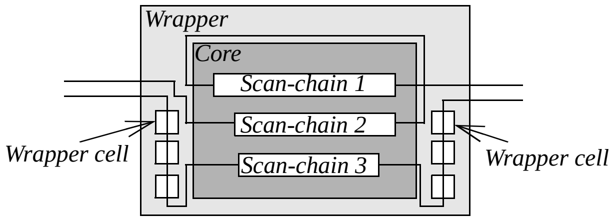

Figure 3. A wrapped scan tested core where the scan-chains and wrapper cells are configured into two wrapper chains.

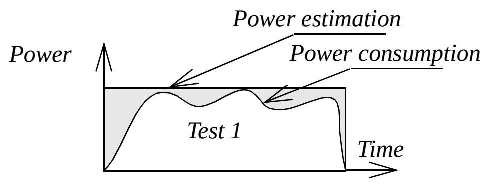

Figure 4. Modelling of test power consumption.

summation of the test power of the tests that are executed at the point. At no time it is allowed to consume more power than the power budget.

In some cases, the order in which the tests are executed is important. It imposes precedence constraints which means that some tests must be executed prior to others.

3. Proposed Test Scheduling Technique

In this section we describe our technique to integrate wrapper design (scan-chain chaining) and test scheduling.

The wrapper chain design algorithm configures the scanned elements (scan-chains, input wrapper cells, output wrapper cells and bidirectional wrapper cells) into a given number of wrapper chains and computes the testing time for the wrapper configuration. We compute the Pareto optimal points for each core. In the scheduling step, we use a heuristic aiming to minimize the total test time taking into account the constraints and we make use of the Pareto optimal points provided by the wrapper designs heuristic.

Our previously proposed technique [14] considered design hierarchy constraints, power limitations, precedence constraints, multiple test sets, and cross-core testing. However, the technique consists of two consecutive steps. First it selects a wrapper configuration for each core and then the tests, configured according to the selected configuration, are scheduled. The main disadvantage is that it does not allow an efficient exploration of the possible solutions since a locally optimal wrapper configuration does not guarantee a global optimal solution. In our current work method, we integrate the wrapper design and the test scheduling. The advantage is that a wide range of wrapper designs can be explored.

The proposed wrapper design heuristic is illustrated in Figure 5. We use an internal chaining function aiming at balancing the scan chains in order to reduce the longest wrapper chain. The longest wrapper chain is the one that limits the solution (the testing time) as shown in [14]. The generated designs are memorized so that all the possible architectures for each core can be checked during the TAM building and the test scheduling steps.

The scheduling heuristic is outlined in Figure 6. First the tests are sorted in decreasing test time order. For each test, one Pareto optimal point is selected considering the maximal width use (*i.e.* the couple T_i, W_i with W_i being the closest to W_{max} (W_{max} is given)).

At step two, the *VirtualTime* test time is estimated in order to obtain a lower bound for the system test time. This bound is used in the scheduling heuristic during the selection of configurations for each core. The advantage is that points with a testing time higher than *VirtualTime* will not be selected since they will increase the total test time.

The *VirtualTime* is calculated using the formula:

$$VirtualTime = \frac{\sum_i W_i \times T_i}{W_{limit}}$$

where W_{max} is the number of available pins for test access (the TAM bandwidth), W_i is the lowest number of TAM

```
Wmax=number of TAM connections
NbLines=(int)(Wmax/2)
#SC=number of Scan Chains

Process 'Internal chaining'
Sort the internal scan chain in decreasing length order
Select the (NbLines) longest scan chains as the (NbLines) lines
While (#SC>NbLines)
    Chain the shortest line with the shortest scan chain
    Update #SC
    Update length of the longest scan chain
    Sort scan chains in decreasing length order
End process
Add functional I/Os balancing the scan chains
End
```

Figure 5. Our wrapper chain design heuristic (scan-chain chaining).

```
1. L1=list of sorted tests in decreasing test time order
2. VirtualTime= $\sum W_i \times T_i / W_{limit}$
3. For tolerance=0 to tolerance=80
4. While all tests are not scheduled
5.     While L1 not empty
6.         For each test T in L1
7.             For each time point t defining the beginning of a
               test session
8.                 Select the best Pareto optimal point such that a)
                   it respects the tolerance; b) the width constraint
                   is satisfied, c) the test time does not exceed Vir-
                   tualTime, and d) precedence, power, incompati-
                   bilities constraints are respected.
9.                     If (the current total test time will not change
                       when T is scheduled to start at t)
10.                        Schedule T at t with the selected Pareto
                           point; remove T from L1.
11.                    Else
12.                        If T is the first test of L1
13.                            Schedule T at t with the selected Pareto
                               point; remove T from L1.
14.                        Else
15.                            Put the test T in L2; remove T from L1.
16.    L1<=L2
17. End
```

Figure 6. Our Test Scheduling Heuristic

wires assigned to a core, and T_i is the testing time assuming W_i TAM wires.

The formula gives the lower bound for the system test time without considering any constraints. In this ideal case, the schedule does not contain any idle times (*i.e.* there is no time loss), and it is therefore the best it can ever achieve. In practice, it is usually impossible to find a test schedule with the *VirtualTime* test time because the tests incompatibilities due to design hierarchy and test resource sharing constraints influence the schedule. Through the scheduling process, we slowly increase the *VirtualTime*. We then use a *tolerance percentage* in order to select the Pareto optimal points in the scheduling. The Pareto optimal points correspond to optimal wrapper designs for a given width constraint and are pre-calculated by our wrapper design heuristic. The tolerance is a percentage of cost loss compared to the best cost computed by the wrapper design algorithm (the cost is defined for each wrapper design by the product W_i*T_i). For instance, if the tolerance is 10%, the dynamic choice of the Pareto optimal point will be done in each session trying to schedule the test checking every wrapper configuration with a cost loss between 0% and 10%.

The main idea in the heuristic is to schedule the tests as soon as possible using the Pareto optimal points defined in the wrapper design heuristic. For each test, the heuristic tries to place each test in a session starting from time t=0, and also trying all the Pareto optimal points (*i.e.* changing the values of W_i and T_i) of the considered test with a cost loss lower or equal to the tolerance to fit in the constraints. The heuristic defines one schedule and one TAM configuration for each tolerance (*i.e.* 80 schedules and TAM configurations from 0% to 80%) and memorizes the solution with the smallest test time fitting into the limits imposed by the constraints.

For all the tests that are first sorted into a list L1, if one test can not be scheduled, it is placed in a auxiliary list L2 to be scheduled later. When L1 is empty, *i.e.* all tests are scheduled or placed in L2, then L2 becomes L1 and the process is re iterated until all tests are scheduled.

4. Experimental Results

We have implemented our test scheduling technique and performed experiments using the ITC’02 benchmarks. Note that none of the previous approaches consider more test conflicts than TAM wire sharing but Iyengar *et al*. [8] who consider design hierarchy constraints in the benchmarks. In all other approaches the design is assumed to be flat.

We are, as discussed above, considering the test conflicts and we are also considering cases when one core is tested by several tests. These realistic assumptions, obviously, make the problem more complicated.

In the first experiment we compare our technique with the approach presented by Huang *et al*. [3] using the d695 circuit considering the same power values depicted in Table 1. The results are given in Table 2 for different TAM

Test	d695	p22810	p93791
1	660	173	7014
2	602	173	74
3	823	1238	69
4	275	80	225
5	690	64	248
6	354	112	6150
7	530	2489	41
8	753	144	41
9	641	148	77
10	1144	52	395
11	-	2505	862
12	-	289	4634
13	-	739	9741
14	-	848	9741
15	-	487	78
16	-	115	201
17	-	580	6674
18	-	237	113
19	-	442	5252
20	-	441	7670
21	-	167	113
22	-	318	76
23	-	1309	7844
24	-	260	21
25	-	363	45
26	-	311	76
27	-	2512	3135
28	-	2921	159
29	-	413	6756
30	-	508	77
31	-	-	218
32	-	-	396

Table 1. Power consumption values for the tests in design d695, p22810, and p93791.

the power constraint is relaxed the better are our results compared to the ones presented in [3]. Otherwise, the results by the two approaches are similar even if we in our approach consider the test conflicts.

In our second experiment, we compared our approach to previous proposed techniques using the d695, p22810 and p93791 without considering any power limitation. The results are for a range of TAM bandwidths given in Table 3. We list first the lower bound from Goel and Marinissen [10] and the *VirtualTime* extracted from our formula above. Then we compare the test times for each TAM width for [3,5,6,14]. Note that in p22810 and p93791 there are design hierarchy constraint that we are considering.

In our last experiment, we applied our algorithm assuming different power constraint values. We made use of two designs with a high number of tests; p22810 containing 30 tests and p93791 containing 32 tests. As power values are not given in these benchmarks, we added values as depicted in Table 1. The power limitations for p93791 are in the range from 30000 down to 10000 and for p22810 the range is from 10000 down to 3000 units. The results are presented in Table 4 and Table 5.

The computation time of our algorithm including the wrapper design and test scheduling is only a few seconds using an AMD 1800 machine (1.53 GHz, 512 MB RAM).

5. Conclusions

In this paper we have proposed a test scheduling technique that takes test power consumption and test conflicts into account when minimizing the test application time. It is important to consider test power consumption since exceeding it might damage the system. The test conflicts we consider are important since they appear in SOC designs. For instance, cross-core testing (interconnection testing), unit testing with multiple test sets, hierarchical SOCs where cores are embedded in cores, and the sharing of test access mechanism (TAM) wires. Another important conflict that we consider is precedence constraints, which is the order in which the tests are to be applied.

We have implemented our technique and performed several experiments where we compare our technique with previous proposed approaches. The experiments show that our technique has a low computational cost and the results are comparable with other techniques which do not consider all the constraints and limitations that we are handling.

References

[1] E. Cota, L. Carro, M. Lubaszewski, and A. Orailoglu, “Test Planning and Design Space Exploration in a Core-based Environment”, *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*, pp. 478-485, Paris, France, March 2002.

[2] H-S Hsu, J-R Huang, K-L Cheng, C-W Wang, C-T Huang, and C-W Wu, “Test Scheduling and Test Access Architecture Optimization for System-on-Chip”, *Proceedings of IEEE Asian Test Symposium (ATS)*, pp. 411-416, Tamuning, Guam, USA, November 2002.

[3] Y. Huang, S.M. Reddy, W-T Cheng, P. Reuter, N. Mukherjee, C-C Tsai, O. Samman, Y. Zaidan, “Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm”, *Proceedings IEEE of International Test Conference (ITC)*, pp. 74-82, Baltimore, MD, USA, October 2002.

[4] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, “Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip”, *Journal of Electronic Testing; Theory and Applications (JETTA)*, pp. 213-230, April 2002.

[5] V. Iyengar K. Chakrabarty, and E. J. Marinissen, “Efficient Wrapper/TAM Co-Optimization for Large SOCs”, *Proceedings of Design and Test in Europe (DATE)*, pp. 491-498, Paris, France, March 2002.

[6] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, “On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization”, *Proceedings of IEEE VLSI Test Symposium (VTS)*, pp. 253-258, Monterey, California, USA, April 2002.

[7] V. Iyengar, S. K. Goel, E. J. Marinissen and K. Chakrabarty, “Test Resource Optimization for Multi-Site Testing of SOCs under ATE Memory Depth Constraints”, *Proceedings of IEEE International Test Conference*, pp. 1159-1168, Baltimore, MD, USA, October 2002.

[8] V. Iyengar, K. Chakrabarty, M. D. Krasniewski, and G. N. Kuma, "Design and Optimization of Multi-level TAM Architectures for Hierarchical SOCs", *Proceedings of IEEE VLSI Test Symposium (VTS)*, pp. 299-304, 2003.

[9] S. K. Goel and E. J. Marinissen, “Cluster-Based Test Architecture Design for System-On-Chip, *Proceedings of IEEE VLSI Test Symposium (VTS)*, pp. 259-264, Monterey, California, USA, April 2002.

[10] S. K. Goel and E. J. Mariniseen, “Effective and efficient test architecture design for SOCs”, *Proceedings of IEEE International Test Conference*, pp. 529-538, Baltimore, MD, USA, October 2002.

[11] S. Koranne, “On Test Scheduling for Core-based. SOCs”, *Proceedings of International Conference on VLSI Design*, pp 505-510, Bangalore, India, January 2002.

[12] S. Koranne and V. Iyengar, “On the use of k - tuples for SoC test schedule representation”, *Proceedings of International Test Conference (ITC)*, pp. 539- 548, Baltimore, MD, USA, October 2002.

[13] E. J.Marinissen, R. Kapur, and Y. Zorian, “On Using IEEE P1500 SECT for Test Plug-n-play”, *Proceedings of IEEE International Test Conference (ITC)*, pp. 770-777, Atlantic City, NJ, USA, October 2000.

[14] J. Pouget, E. Larsson, Z. Peng, M.-L. Flottes, B. Rouzeyre, “An Efficient Approach to SoC Wrapper Design, TAM configuration, and Test Scheduling”, *Proceedings of IEEE European Test Workshop (ETW)*, Maastricht, The Nederlands, May 2003.

Design: d695	**TAM width=32**		**TAM width=48**		**TAM width=64**		**TAM width=80**		**TAM width=96**		**TAM width=112**		**TAM width=128**	
Approach:	**[3]**	**Our**	**[3]**	**Our**	**[3]**	**Our**	**[3]**	**Our**	**[3]**	**Our**	**[3]**	**Our**	**[3]**	**Our**
P_{max}=1500	45560	43541	31028	32663	27573	26973	20914	24369	20914	23425	16841	19402	16841	19402
P_{max}=1800	44341	42450	29919	32054	24454	23864	20467	18774	18077	18774	14974	18774	14899	16804
P_{max}=2000	43221	42450	29419	29106	24171	21942	19206	18691	17825	17467	14128	14563	14128	14469
P_{max}=2500	43221	41847	29023	29106	23721	21931	19206	18691	15847	17257	14128	13963	12993	13394

Table 2. Power constrained test time on design d695 - Comparison between Huang et al. [3] and our approach.

		Test Time								
Design	TAM Width	LB[10]	Virtual Time	Multiplexed [14]	Static [14]	[3]	[6]	[5]	[8]	Our
d695	128/64	10247	9584	36158	13348	11279	11604	12941	-	13348
	96/48	13659	12780	36232	19932	15142	15698	15300	-	17257
	80/40	16388	15335	36232	19932	17366	18459	18448	-	18691
	64/32	20482	19169	45798	32857	21389	23021	22268	-	20512
	48/24	27305	25559	45972	33031	28639	30317	30032	-	29106
	32/16	40951	38339	78077	65136	42716	43723	42644	-	41847
p22810	128/64	104868	105493	503088	142360	128512	136941	153990	-	128332
	96/48	139823	140578	503534	215339	167858	167256	232049	-	159994
	80/40	167787	168790	503635	223463	184951	197293	232049	-	195733
	64/32	209734	210988	531631	294046	223462	246150	246332	-	236186
	48/24	279644	281317	619537	418226	300723	307780	313607	-	352834
	32/16	419466	421976	664665	574120	446684	452639	468011	-	473418
p93791	128/64	436673	413565	639827	618150	459233	511286	473997	481896	457862
	96/48	582227	551420	672119	650402	607955	627934	599373	635710	639217
	80/40	698670	661704	1174475	1155800	719880	794020	741965	758156	787588
	64/32	873334	827131	1240170	1221495	900798	975016	894342	863765	945425
	48/24	1164442	1102841	1377123	1358448	1200157	1248795	1209420	1293990	1220469
	32/16	1746657	1654261	2432511	2432511	1791860	1851135	1786200	1927010	1827819

Table 3. Experimental results. Comparison between Multiplexed approach [14], Pouget et al. [14], Huang et al. [3], Iyengar et al. [6], Iyengar et al. [5], Iyengar et al. [8], and our approach.

p22810	Virtual Time	No P_{max}	P_{max} = 10000	P_{max} = 8000	P_{max} = 6000	P_{max} = 5000	P_{max} = 4000	P_{max} = 3000
TAM Width		Test Time	Test Time	Test Time	Test Time	Test Time	Test Time	Test Time
128	103 344	128 332	128 332	142 056	157 568	246 110	268 856	293 021
112	118 108	138 410	138 542	147 535	159 686	257 600	268 272	293 528
96	137 792	159 994	159 994	159 994	174 928	266 166	285 814	311 632
80	165 351	195 733	195 733	195 733	209 559	264 038	285 307	356 215
64	206 688	236 186	236 186	236 186	250 487	321 930	324 478	309 255
48	275 584	352 834	352 834	352 834	346 461	382 507	389 243	392 525
32	413 376	473 418	473 418	473 418	475 951	472026	480 223	482 963
24	551 168	635 583	635 583	635 583	638 116	638 316	653 699	680 622
20	661 402	819 465	819 465	819 465	819 530	845 469	845 469	845 469
16	826 753	892 713	892 713	892 713	893 050	891 457	891 457	948 481
12	1 102 337	1 206 986	1 206 986	1 206 986	1 206 986	1 206 986	1 206 986	1 206 986

Table 4. Power-constrained scheduling on p22810.

p93791	Virtual Time	No P_{max}	P_{max} = 30000	P_{max} = 25000	P_{max} = 20000	P_{max} = 15000	P_{max} = 10000
TAM Width		Test Time	Test Time	Test Time	Test Time	Test Time	Test Time
128	424 847	457 862	457 862	493 599	472 653	486 469	568 734
112	485 539	515 020	515 020	549 669	549 669	598 487	629 051
96	566 462	639 217	639 217	639 217	658 132	631 214	691 866
80	679 755	787 588	787 588	821 475	821 575	848 050	1 091 210
64	849 694	945 425	945 425	965 383	957 921	1 014 616	1 117 385
48	1 132 924	1 220 469	1 220 469	1 220 469	1 220 469	1 220 469	1 220 469
32	1 699 387	1 827 819	1 827 819	1 827 819	1 827 819	1 827 819	1 827 819
24	2 265 850	2 399 834	2 399 834	2 399 834	2 399 834	2 399 834	2 399 834
20	2 719 020	2 951 651	2 951 651	2 951 651	2 951 651	2 951 651	2 951 651
16	3 398 775	3 574 150	3 574 150	3 574 150	3 574 150	3 574 150	3 574 150
12	4 531 700	4 728 023	4 728 023	4 728 023	4 728 023	4 728 023	4 728 023

Table 5. Power-constrained scheduling on p93791.