The Efficient TAM Design for Core-Based SOCs Testing

Jiann-Chyi Rau, Po-Han Wu, Chih-Lung Chien, Chien-Hsu Wu
Department of Electrical Engineering, Tamkang University
151, Ying-Chuan Rd. Tamsui,
Taipei Country, Taiwan 25137, R.O.C.
{ jcrau, phwu, clchien, cswu}@ee.tku.edu.tw

Abstract: - This paper presents a framework and an efficient method to determine SOC test schedules. We increase the test TAM widths by the framework. Our method deals with the traditional scan chains and reconfigurable multiple scan chains. Experimental results for ITC'02 SOC TEST Benchmarks show that we obtain better test application time when compared to previously published algorithms. Test access mechanism (TAM) and test schedule for System-On-chip (SOC) are challenging problems. Test schedule must be effective to minimize testing time, under the constraint of test resources. This paper presents a core section method based on generalized 2-D rectangle packing. A core cuts into many pieces and utilizes the design of reconfigurable core wrappers, and is dynamic to change the width of the TAM executing the core test. Therefore, a core can utilize different TAM width to complete test.

Key-Words: - SOC Testing, TAM, Testing Scheduling

1 Introduction

The heading of each section should be printed in small, 14pt, left justified, bold, Times New Roman. You must use numbers 1, 2, 3, … for the sections' numbering and not Latin numbering (I, II, III, …)
Although the SOC design also called core-based-design is still a developing technology, but a lot of researches has proposed many idea to solve the problem in the future that will occur, especially how to test the SOC chip with efficiency method. Base on this reason, we have made a research to solve it. Now, I will first introduce the concept about SOC design below.

1.1 Test challenges in Core-based SOC designs

The more and more widening design productivity gap between VLSI system capabilities and design engineering capability, in a limited time to market scenario, has prompted many design houses to adopt a policy of design reuse at the core level [3]. The Semiconductor Industry Association's Technology Roadmap [1] predicts the percentage of reusable cores in SOC to be rising to 80% in 2006. However, with the increasing complexity and reduction of design cycle, the test application time of SOC is becoming a major bottleneck for time-to-market. It is more and more important for reusability of design to reduce the design time, but it is not enough when the verification and testing for reusable cores take up the most of design time.

One of the most effective and popular techniques is Boundary Scan Test, defined as IEEE standard 1149.1 ('JTAG'). The Boundary Scan Test standard defines the additional hardware for the IC, in order to solve the board interconnect test problem. Unfortunately, the boundary scan architecture allows only one pin for test data input and another one for data output, and hence can not efficiently support multiple scan chains of the core on SOC. So for testing a SOC, JTAG may not suitable.

There still exists an important challenge. For SOC integrators, the information inside the cores may be invisible. Most of cores are encrypted Intellectual Property blocks or hard cores that the users don't have enough information about the internal part of the cores. So it is difficult and inefficient for core user to develop the test set. Core vendors have responsibility to develop the core test. The most common method is Design-For-Testability (DFT). Core vendors would offer information to SOC integrators to design test processes.

The information is such as the I/O pins, internal scan chains, test patterns, power constraints, clock timing and etc. In order to increase the interoperability of the internal test with the core, the internal test to accompany its corresponding core need to be described in accepted standard format. Such a standard format is currently being developed by IEEE P1500 [4][5] and referred to as standardization of a core test description language.

1.2 TAM architecture

The basic TAM architecture is shown in Figure 1. It can be divided into three structural elements [6]:

Source and Sink: the source generates the test pattern for the embedded cores and sink compares the obtained responses with the excepted one to know if there exists difference. The two elements can generate two test methods on-chip and off-chip by its different position. With the method on-chip, like BIST or built-in ATE, all of that have the trouble with increasing area, but in opposite way, the requirement of TAM can be loosen. With the method off-chip, the total chip pins will decide the bandwidth under testing. If we increase the TAM width, it will raise the expensive cost on hardware.

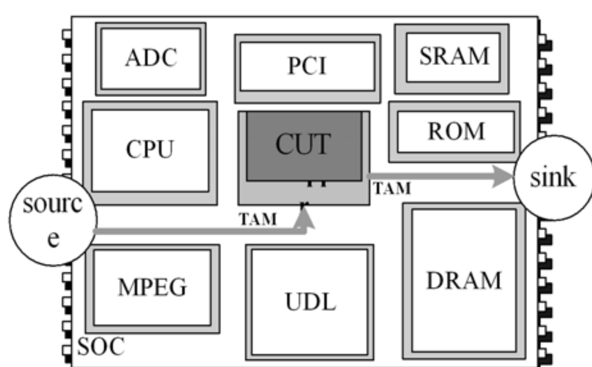


Fig. 1 The test access architecture overview

Test access mechanism (TAM): TAM is the mechanism of transporting the test data. TAM can transport the test set to the CUT and transport the response from CUT to the sink. With TAM design, there must be a trade-off between test bandwidth (capacity) and test cost. The larger TAM width can provides more bandwidth but it also increases the cost on routing. If the test source designs outside the SOC and the IC only with few pins, there is unmeaning to increase the TAM width.

Core test wrapper: wrapper is the interface between the embedded core and SOC, it connects between the core terminals and TAM and provides the switching mechanism to the function I/O and TAM.

1.2.1. Basic TAM architecture

There are three main kinds of test access architectures [7]: (a)Multiplexing Architecture; (b)Daisy-chain Architecture; (c)Distribution Architecture.

1.2.2. Test Bus architecture

Varma and Ahatia [9] proposed the Test Bus Architecture which combines the Multiplexing and Distribution Architectures. A single test bus actives the same as the operation of Multiplexing Architecture. Modules which connected to the same test bus can be only tested sequentially. The Test Bus Architecture allows that multiple test buses exist on one SOC and operate independently just like the Distribution Architecture. So modules connected to a same test bus will be in the common detrimental conditions to make the core-external testing difficult as in the Multiplexing Architecture.

1.2.3. TestRail architecture

The TestRail Architecture presented by Marinissen et al. [10] is a combination of the Daisychain and Distribution Architectures. A single TestRail is in essence the same as what is described by the Daisychain Architecture. Modules connected to the same TestRail can be tested simultaneously, as well as sequentially. The TestRail Architecture allows for multiple TestRails on same SOC, which operate independently, as in the Distribution Architecture. The advantage of the TestRail Architecture over the Test Bus Architecture is that it allows access to multiple or all wrappers simultaneously, which facilitates module-external testing.

TestRail Architecture supports multiple types of test schedules. It allows the modules on the common TAM operating in both serial testing mode and parallel testing mode. In parallel testing mode, the modules on TAM will be tested simultaneously.

1.3 Wrapper architecture

Good wrapper design will make the internal scan chains of core as balance as possible to reduce the core testing time. A standardized, but scalable test wrapper is an integral part of the IEEE P1500 working group proposal [11]. Apart from these mandatory modes, a core test wrapper might have several optional modes, e.g., a detach mode to disconnect the core from its system chip environment and the test access mechanism, or a bypass mode for the Universal BIST Scheduler [12] and the TestRail [13] test access mechanism.

Wrappers may need to perform test width adaptation when the TAM width is not equal to the number of core terminals. This will often be required in practice, since large cores typically have hundreds of core terminals, while the total TAM width is limited by the number of SOC pins.

1.4 Reconfigurable Core Wrappers Design

For easy testing, the providers usually use necessary Design-For-Testability (DFT) skills to the SOC chips. The integrators were allowed to insert a

wrapper cell to each input and output by these DFT skills. All the internal scan chains, input ports and output ports would be assigned into new scan chains. Test vectors will be recombined with the new scan chains. Reconfigurable Multiple Scan Chains is one kind of architecture to reassign the scan chains.

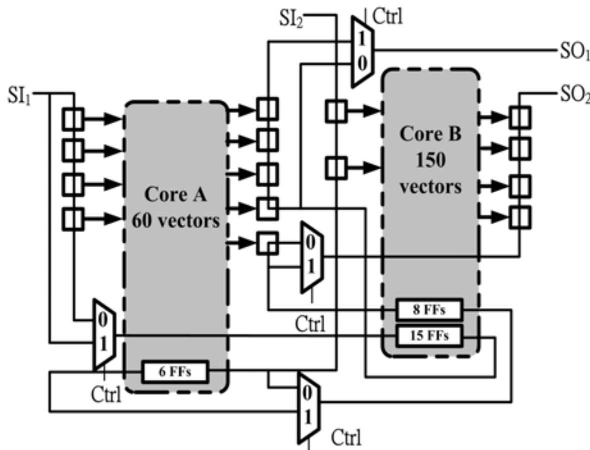


Fig. 2 An example of Reconfigurable Multiple Scan

2 Problem Formulation

2.1 Core test time

A scan test for a core consists of three phases: (1) scan in of the test patterns to the scan registers and ready for normal execution, (2) normal execution, and (3) capture and scan out of the responses by scan registers. We define for each core i the number of test pattern pi. Let si be the length of the longest wrapper scan-in chain to fill all flip flops for a core i, and so is the time of the longest wrapper scan-out to scan out all flip flops.

We assume that in each pattern exactly one time slot is used for the normal execution step; this means that the right input data has to be available at the core inputs at the moment of the normal execution step. This can be accomplished by adding scannable flip flops around the core [14]. The test time ti of core i becomes the sum of the scan-in time, the time for normal execution, and the scan-out time:

$$t_i = s_i \cdot p_i + p_i + s_o \cdot p_i \quad \text{(Eq. 1)}$$

In the scan test process it is common practice to use pipelining; when one pattern is scanned out, the next pattern is scanned in. This reduces the test time of a core to:

$$t_i = (1 + \max\{s_i, s_o\}) \cdot p_i + \min\{s_i, s_o\} \quad \text{(Eq. 2)}$$

When the term '+1' indicates that pipeling cannot be used for the scanning out the last pattern. The test time for a core decreases as both si and so are reduced. Therefore, the balance wrapper scan chains are important because the number of cycles to scan a test pattern to (from) a core is a function of the length of the longest wrapper scan-in (scan-out) chain.

2.2 The Popular Rectangle Packing Model

E.J. Marinissen presented a Rectangle Packing Model [15] as fixed width test buses. The total TAM width was partitioned among a number of fixed-width test buses and each core was assigned to one of these TAMs.

In Fig .5, each test of cores could be modeled as a rectangle by a fixed TAM width and the testing time. This is defined as a wrapper/TAM design problem in [16]. For different TAM widths, the same test could be modeled in different rectangles by width and the testing time. The rectangles in Fig. 3 are using the Design_wrapper algorithm [16] to model the Core 6 in SOC p93791. So based on the model, a test schedule problem would be treated as a 2D Bin-packing problem.

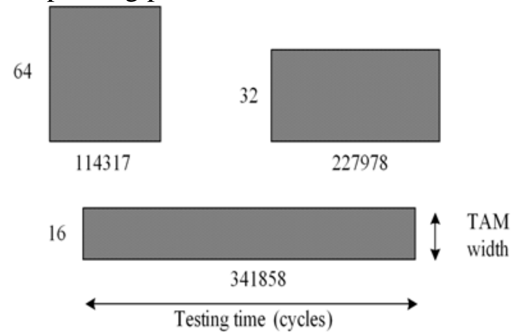


Fig. 3 Example rectangles for Core 6 in SOC p93791

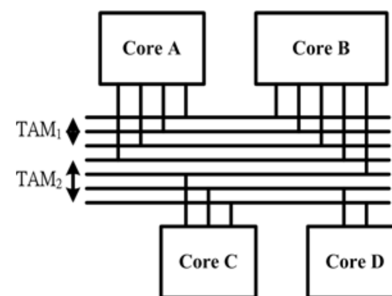


Fig. 4 TAM design using generalized rectangle packing

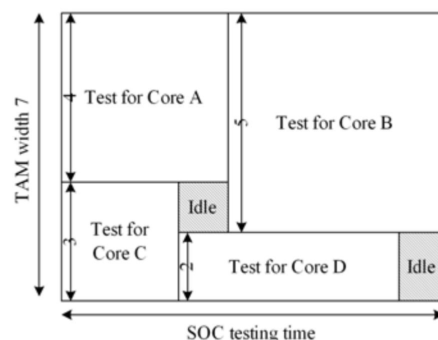


Fig. 5 The test schedule of fig.4

2.3 The General Test Schedule

Test schedule is the schedule for testing a SOC. Basically, a test schedule for a SOC should maintain all processes on testing. It includes the order of the cores for testing, the width for each core, and most important is that the test schedule would show the total test time. The test schedule is based on what TAM architecture the testing process used.

The total test time is one of the main factors to evaluate the testing cost. For designing a test schedule, there are three categories: (1) Serial Test Schedules, (2) Parallel Test Schedules, (3) Mixed test schedules.

2.4 Using dynamic TAM width to test a core

The past test scheduling that the partition of TAM width is fixed. The test scheduling will have a lot of idle time. An example is showed in Fig. 6 there are three idle time times existing in test scheduling. The more idle time will enable a result of the test scheduling worse.

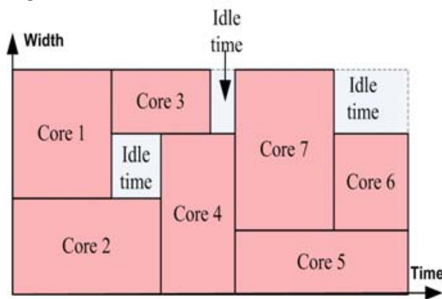


Fig. 6 SOC test scheduling

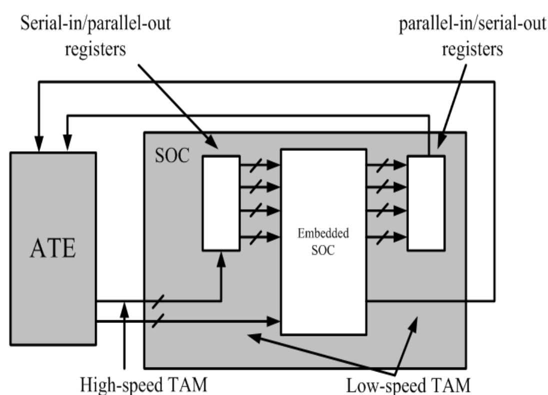


Fig. 7 Virtual TAM architecture

2.5 Virtual TAM

Sehgal [25] proposed an ideal to increase the test TAM width by using virtual TAM. We usually assume that the automatic test equipment (ATE) operates at the same frequency as the SOC core's internal scan chains frequency. In fact, the ATE operating frequency is higher than SOC core's internal scan chains frequency. Sehgal uses the characteristic of such frequency to increase the test width. Fig. 7 is the virtual TAM architecture. Quasem & Gupta (2004) proposed a framework of Reconfigurable Multiple Scan Chains. The frame can reduce the test application time, but it requires a lot of control signals. We proposed an algorithm to combine those methods and virtual enable signals. The attempt to minimize the test application time was a success.

We propose two methods for solving the problem of test scheduling. The first method is called stairway scheduling witch cut each core to become many piece among SOC. The second is using virtual TAM to control signal.

3 Proposed stairway scheduling

3.1 Concept of stairway scheduling

We propose a new test scheduling method. This method considers that testing of the core is cut many pieces and use different width of TAM to test. Then, the test scheduling of the core will become a form of stairway. We call stairway scheduling. Such, we can reduce the test application time of the core. In this paper we assume that all cores in the SOC can test at the same time, and we want to insert the core testing in the idle time. Then, the core must change the TAM width that to satisfy of TAM width of the current idle time. Such the core testing can use different TAM width to complete.

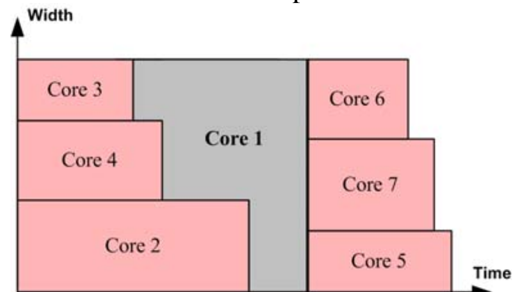


Fig. 8 SOC test of stairway scheduling

3.2 Proposed algorithm

We partition into two parts. One part describes basic core testing to complete the scheduling. This part not dynamically changes the TAM width to test the core. The design follow is shown in Fig. 9. Then, the width of TAM must have enough numbers to schedule the core at the current time. Until not exist any core enough to satisfy the TAM width at the current time. Then, we assign remnant TAM width to the largest core of scheduling. At this time, doesn't have any width of TAM enough to schedule any core. The total TAM width is zero at current time.

Another part describes dynamic testing of a core. That is called stairway scheduling. In this part, we can utilize the TAM width at the current time to test the core. Therefore, the core is completed testing through different testing resource (TAM). The scheduling is shown in Fig. 8 before.

In Fig. 10 we show the design follow of the stairway scheduling. The total TAM width is zero at current time. We move the current time to the next time and free the TAM width to test the core. This action is hold until to finish the testing of the core. Then, the TAM width equals the total width, we execute basic core testing (In Fig. 9)

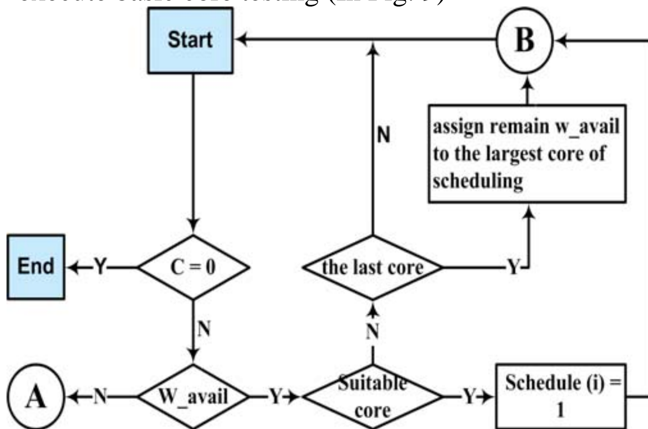


Fig. 9 The design follows of basic test scheduling (no stairway scheduling)

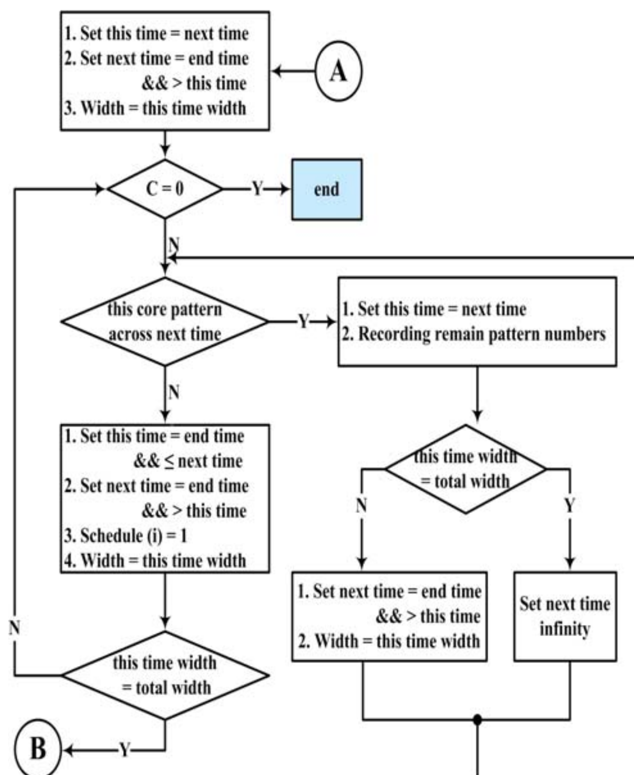


Fig. 10 The design follows of basic test scheduling (stairway scheduling)

Procedure *SOC_schedule_optimizer*

```
1 Compute collection of core rectangles using a design of wrapper;
2 Initialize(C, d, p);
3 Set w_avail = W_max;   this_w = W_max; this_time = 0; next_time = 0;
4 While C ≠ ∅ {
5     If w_avail > 0 AND this_w = W_max {
6       If Core i ∈ C can be found, such that
           Width (i) ≤ w_avail AND T_i(width (i)) is maximum; {
        Schedule (i) = 1 ;}
7       Else Core i ∈ C can be found AND C − {i} ≠ ∅ {
           In schedule structure, find a longest end_time (i); AND
           Assign remnant w_avail to this core ;}}
8     Else {
          Set this_time = next_time;
          In schedule structure, find the end_time (i) > this_time AND
             end_time (i) is minimum;
          Set next_time = end_time (i);
          Set w_avail = this_time width AND this_w = w_avail;
9         Calculate T_ic(width_c(i)) under current w_avail;
10        If T_ic(width_c(i)) > next_time {
11          While a Core pattern ≠ ∅ {
12             Calculate pattern numbers, such that
                 T_ip(width_ip(i)) will across next_time under current w_avail;
                 Records remain pattern numbers;
                 Set this_time = next_time;
                 Set w_avail = this_time width AND this_w = w_avail;
13               If this_time width = W_max {
                   Set next_tim = ∞;}
14               Else {
                   In schedule structure, find the end_time (i) > this_time
                      AND end_time (i) is minimum;
                   Set next_time = end_time (i) ;}}
                   Schedule (i) = 1 ;}
15        Else {
             Set next_time = this_time + T_ic(width_c(i));
             Set w_avail = this_time width AND this_w = w_avail;
          Schedule (i) = 1 ;}}}
16        Return Schedule;
```

Fig. 11 Proposed procedures for test scheduling

In Fig. 11, we detail the algorithm that we have developed to solve the problem of test scheduling. In our algorithm utilize the method [14] to find the initial rectangle of a core then to schedule the testing. We elaborate on each step of the algorithm in the following paragraphs.

3.3 Our algorithm explain

We use two data structures, which the one is core_initial and the core_sedule to store the TAM width, the pattern numbers, and testing time values, respectively. The structure for the core of SOC is presented in Fig. 12. This data structure update with the begin times and finish times for each core.

In Line 1, we use the method [14] to compute the collection of the initial rectangles. The Line 3 set initial values of w_avail, this_w, this_time, and next_time (In this paper, Wmax is chosen to be 64). In Line 5 is a general action to schedule the core. The w_avail is not equal the Wmax and the TAM width of the initial rectangle satisfy this w_avail such we schedule this core. In Line 7, we use the method [14] to assign the remained TAM width to the longest core at the current time (this_time). An example is shown in Fig. 13. Therefore, the TAM width (w_avail) equals the total (Wmax) width at the current time (this_time).

Structure *core_initail*
1. *num* (*i*) /* the number of a core */ 2. *end_time* (*i*) /* the initial time of a core */ 3. *width (i)* /* the initial width of a core */ 4. *remain_p* (*i*); /* the remain pattern numbers of a core */ 5. *Schedule* (*i*)/* boolean indicates test for Core *i* has completed */

Structure *core_sedule*
1. *num* (*i*) /* the number of a core */ 2. *this_time* (*i*) /* the scheduling begin time of a core */ 3. *end_time* (*i*) /* the scheduling finish time of a core */ 4. *Schedule* (*i*) /* boolean indicates test for Core *i* has completed */

Fig. 12 Data structures of the core

In Line8, we set the this_time to the next_time and release the TAM width at the current time (this_time). In Line 9, the testing time Tic (widthc (i)) of the core compute under current TAM width. In Line 10, Line 11, and Line 12, we will calculate whether the testing time Tip (widthip (i)) not across next_tme and compute the remnant pattern numbers. In Line 13, when the Width of the TAM equals the total TAM width (Wmax), we will set next_tme infinite. We schedule the core testing, when all patterns are finished to shift. In Line 15, the testing time Tip (widthip (i)) not across next_tme, we schedule this core testing.

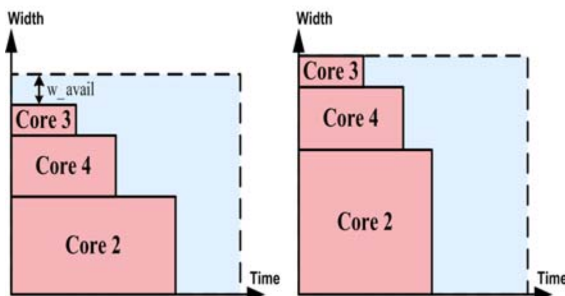


Fig. 13 Assign the remnant TAM width to the longest core

An ideal result of the stairway scheduling will not have occurred the idle time. But Tip (widthip (i)) is not completely to match the next time. A little idle time will exist in the scheduling of the core testing. An example is shown in Fig. 14 there are two idle times in the Core 1 testing. But idle times are small. An experimental result proves the times of idle will not affect the total time of testing.

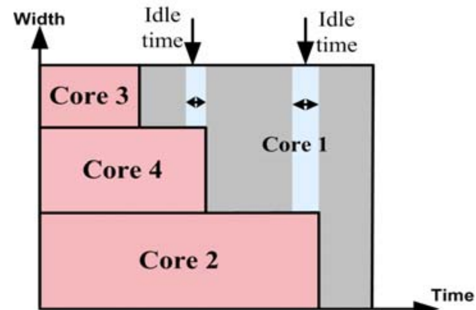


Fig. 14 the idle time of the stairway scheduling

4 Virtual Enable Signal

4.1 Concept of Virtual Enable

As mentioned in last section, the disadvantage of the virtual TAM method will limit the results. We consider that virtual TAM is a good method. Therefore the method which is the most beneficial is adopted. A method to enable signal and making the test TAM width increate was used. It shows the architecture of virtual enable signal in Fig. 15.

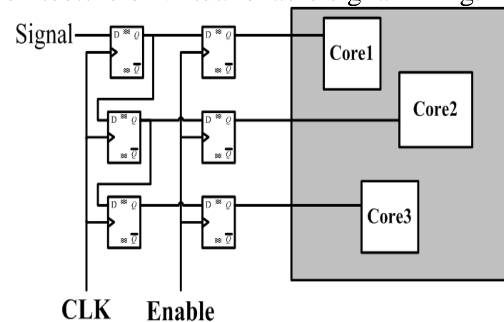


Fig. 15 Virtual control signals

In the figure, the Enable signal hold in low at start. Until the left Flip-flops holding the correct value, the Enable signal transfer from low to high. When Enable signal is high, the left Flip-flop's values shift to the right Flip-flops. Each module will need a control signal to change from normal mode to shift mode or form shift mode to normal node at mixed-architecture. Virtual TAM was used to reduce the pin's consumption and increase the test TAM width. The other benefit is that the bandwidth decreasing with control signal affects less to total test application time. The test time formula was modified in order to increase the amount of test TAM width.

4.2 Proposed algorithm of virtual control

The following section includes the algorithm that was used. The algorithm proposed refers to the TR architect algorithm [17]. We modify some steps of TR architect algorithm to match virtual enable signal and reconfigurable multiple scan chains. The algorithm is divided into 4 steps. They are STARTSOLUTION, BOTTMUP, TOPDOWN, and RESHUFFLE. In the algorithm, there are two differences with TR architect algorithm. Two ports are shown as following:

$$Rate = \begin{cases} \dfrac{ScanChains[L] - ScanChains[S]}{ScanChainNum} & ,\text{if } ScanChainNum \neq 0 \\ 0 & ,\text{others} \end{cases} \quad \text{(Eq. 3)}$$

The STARTSOLUTION step was modified. When the total test module numbers was grater than total test TAM widths, we use a priority to find the fit module. An equation was used to calculate the value, and show at Eq. 3. The ScanChainNum is means the number of the internal scan chains. The ScanChains[L] is means the largest number of Flip-flops of internal scan chains. The ScanChains[S] is means the smallest number of Flip-flops of internal scan chains. If the ScanChainNum equals to 0, the Rate is set to 0. The Rate with each module was calculated at the first of the STARTSOLUTION. When choosing the rate, the value which is closest was chosen. The closer to the two rates are the numbers of Flip-flops of their internal scan chains are more like, and let the Reconfigurable Multiple Scan Chain construct more balance.

```
01    assign the registers of the TAM;
02    compute the test time use lower bond equation;
03    save the test time in S-time;
04    sort the modules of the TAM with test patterns;
05    compute the test time with the RMSC;
06    save the test time in P-time;
07    if ( S-time < P-time){
08              return the S-time;
09    }else{
10              return the the P-time;
11    }
```

Fig. 16 Algorithm of Compute_Time

The second step was modifying the time function. The new algorithm is shown in the Fig. 16. The registers are assigned (the input/output ports, internal scan chains) to the test scan chains of the TAM at first (line 1), and the method will be introduced in as following. The time calculated is then compared by lower bond [27] with the time calculated by the Reconfigurable Multiple Scan Chains (RMSC) [28] (line 2 ~ line 6). The best of the two test times are selected (line 7 ~ line 11). There is the same number of control signals with the number of module of TAM. We will explain it in chapter 5.

In Fig. 17. In the step 1, the internal scan chains need to be balanced. (Firstly), the internal scan chains of module are sorted in descending order. If the sum the last two internal scan chains is smaller than the first internal scan chain, the two internal scan chains are added and sort the internal scan chains again (line 5 ~ line 6). In the step 2, the bidirs, input ports, output ports and internal scan chains is assigned to the TAM. The registers are assigned in this order of internal scan chains, bidirs, and input/output ports. In order to balance these scan chains, after the internal scan chains assigned, we assign these registers to the smallest value of the scan chain every time.

```
01      for each module {
02         /*step1: balance internal scan chain*/
03         sorting the internal scan chain of module
04            in decreasing order;
05         while((SC_{n-1}+SC_n) < SC_1) {
06                SC_{n-1}=SC_{n-1}+SC_n;
07                n=n-1;
08                sorting the internal scan chains
09                   again in decreasing order;
10         }
11         /*step2 : assign scan chains*/
12         assign these internal scan chains to TAM
13            in the order above;
14         assign the Bidirs to TAM;
15         if(the output ports > the input ports) {
16                assign the output ports to TAM;
17                assign the input ports to TAM;
18         }
19         else {
20                assign the input ports to TAM;
21                assign the output ports to TAM;
22         }
23      }
```

Fig. 17 Algorithm of registers assignment

5 Experimental Results

We used four SOC of the ITC'02 SOC test benchmarks [29]. The first letter d means that the SOC comes from Duke University and p means industrial Philips Company. SOC p22810 contains 6 memory cores and 22 scan-testable logic cores. SOC p34392 contains 15 memory cores and 4 scan-testable logic cores.

5.1 Stairway scheduling results

In the proposed stairway scheduling, we compare our results with three previously published approaches:

(1) Test Bus Architecture optimization method base on ILP and exhaustive enumeration in [31].

(2) Rectangle packing co-optimization in [14].
(3) Wrapper/TAM co-optimization in [30].

We use the TAM width 16, 24, 36, 48, 56, 64, to test every benchmark. Experimental results show our method get the better results than others method.

Table 1 Test application time for Stairway scheduling results

SOC	W	LB	ILP [21]	Rect. pack. [18]	Par_eval [20]	Our
d695	16	41231	42568	44545	42644	43023
	24	27487	28292	31569	30032	28878
	32	20615	21566	23306	22268	22234
	40	16492	17901	18837	18448	17848
	48	13743	16975	16984	15300	15864
	56	11780	13207	14974	12941	13722
	64	10307	12941	11033	12941	12279
p22810	16	412538	462210	489192	468011	499625
	24	275025	361571	330016	313607	373043
	32	206269	312569	312662	246332	242292
	40	165015	278359	278360	232049	212241
	48	137512	278359	268474	232049	178297
	56	117868	268472	266800	153990	138824
	64	103134	260638	260638	153990	129609
p34392	16	936881	998733	1053791	1033210	1051402
	24	624587	720858	759427	882182	715732
	32	544579	591027	544579	663193	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	1709095	1771720	1932331	1786200	1763635
	24	1138063	1187990	1310841	1209420	1268411
	32	853547	887751	988039	894342	871332
	40	682838	698583	794027	741965	696894
	48	569031	599383	669196	599373	603498
	56	487741	514688	568436	514688	514141
	64	426773	460328	517958	473997	464305

5.2 Virtual control results

To evaluate the proposed method, it was simulated in the ITC'02 SOC test benchmarks. In Table 2, we compare the test between proposed method (without virtual enable signal) and TR-Architect. W represents TAM widths and ΔT represents $(T_j - T_i)/T_i$. We can find that our method have better efficiency at large TAM widths. Besides these three test scheduling approaches in above, we compare the test time reduction algorithm for TestRail Architecture in [27].

The number after the SOC names represents the number of cores each SOC included, and t represents the TAM widths which increases after using virtual enable signal. We can see that as more cores embedded in SOC, the better performance for our method.

Table 2 Test application time for Virtual control results

SOC	W	TR-Achitect T_i	Proposed T_j	ΔT
d695	16	44307	41889	-5.45
	24	28576	28614	0.13
	32	21518	21518	0
	40	17617	17745	0.72
	48	14608	14450	-1.08
	56	12462	12426	-0.28
	64	11033	11029	-0.03
p22810	16	458068	458306	0.05
	24	299718	304064	1.45
	32	222471	228116	2.53
	40	190995	190995	0
	48	160221	160125	-0.05
	56	145417	135566	-6.77
	64	133405	121621	-8.83
p34392	16	1010821	1010811	-0.001
	24	680411	674170	-0.91
	32	551778	544579	-1.30
	40	544616	544579	-0.006
	48	544616	544579	-0.006
	56	544616	544579	-0.006
	64	544616	544579	-0.006
p93791	16	1791638	1778910	-0.71
	24	1185434	1193399	0.67
	32	912233	918197	0.65
	40	718005	732404	2.00
	48	601450	604857	0.56
	56	528925	520998	-1.49
	64	455738	452976	-0.60

Table 3 Simulation Results for benchmarks

SOC	W	ILP	GRP	Cluster	TR	Proposed	
						t	Time
d695 (10)	16	42568	44545	44330	44307	19	36330
	24	28292	31569	30021	28576	27	26150
	32	21566	23306	23488	21518	35	20430
	40	17901	18837	19034	17617	43	15980
	48	16975	16984	16194	14608	51	14610
	56	13207	14974	13479	12462	59	11740
	64	12941	11984	11033	11033	67	10630
p22810 (29)	16	462210	489192	-	458068	28	262052
	24	361571	330016	-	299718	36	205156
	32	312659	245718	259975	222471	44	174496
	40	278359	199558	206205	190995	52	145432
	48	278359	173705	173705	160221	60	138124
	56	268472	157159	146390	145417	68	126000
	64	260638	142342	133587	133405	76	112280
p34392 (20)	16	998733	1053491	-	1010821	24	674215
	24	720858	759427	876529	680411	32	544579
	32	591027	544579	585309	551778	40	544579
	40	544579	544579	544579	544616	48	544579
	48	544579	544579	544579	544616	56	544579
	56	544579	544579	544579	544616	64	544579
	64	544579	544579	544579	544616	72	544579
p93791 (33)	16	1771720	1932331	-	1791638	30	953536
	24	1187990	1310841	-	1185434	38	768736
	32	887751	988039	-	912233	46	633984
	40	698883	794027	816972	718005	54	542048
	48	599373	669196	677707	601450	62	466464
	56	514688	568436	542445	528925	70	418176
	64	460328	517958	467680	455738	78	371040

6 Conclusion

In this paper, we propose a new idea to test a system on chip (SOC). An idea of section a core to test, reform the SOC testing schedule. We called stairway scheduling. We cut each core to become many piece among SOC. Therefore, the core can split to complete a testing and follow the current time moving to utilize the test resource (TAM) under this current time.

We presented the test time by four SOC of ITC'02 SOC Test Benchmarks. By the comparison of [31], [14], and [30], the experimental results show our method get the better results than others method.

In this paper, we focus on the part of SOC test scheduling. Other part of hardware, we just propose a sketchy conception. As future work, we will research about the part of the hardware and try to fill the idle time to improve the total testing time. And in addition we may take more constraints, like power consumption, into consideration in our method.

Lastly, any testing schedule will require adding the hardware extra cost. Therefore, a good engineer must consider how to find a balance at test application time and hardware.

On the other side, we proposed a new method to use virtual TAM to do the control signal and enable signal. And we proposed an algorithm to solve core-based SOC schedule problem. The algorithm and the control enable signal is performance well for the ITC'02 benchmarks.

References:

[1] International SEMATECH. The International Technology Roadmap for Semiconductors (ITRS): 2001 Edition. http://public.itrs.net/Files/2001ITRS/Home.htm

[2] R.K. Gupta, Y. Zorian, "Introducing Core-Based System Design," In Proceedings IEEE Design & Test of Computers, Volume: 14, Issue: 4, pp. 15 - 25, Oct.-Dec. 1997.

[3] IEEE P1500 Web Site, http://grouper.ieee.org/groups/1500/.

[4] Y. Zorian, E.J. Marinissen, and S. Dey, "Test Requirements for Embedded Core- Based Systems and IEEE P1500," In Proceedings IEEE International Test Conference, pp. 191-199, Nov. 1997.

[5] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," In proceedings IEEE International Test Conference, pp. 130-134, Oct. 1998.

[6] J. Aerts and E.J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs," In Proceedings IEEE International Test Conference, pp. 448-457, Oct. 1998.

[7] V. Immaneni and S. Raman, "Direct Access Test Scheme-Design of Block and Core Cells for Embedded ASICS," In Proceedings IEEE International Test Conference, pp. 488-492, Sep. 1990.

[8] P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," In Proceedings IEEE International Test Conference, pp. 294-302, Washington, DC, Oct. 1998.

[9] E.J. 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 Proceedings IEEE International Test Conference, pp. 284-293, Oct. 1998.

[10] IEEE P1500 Web Site, http://grouper.ieee.org/groups/1500/.

[11] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," In Proceedings IEEE VLSI Test Symposium, pp. 6-11, April. 1993.

[12] H. Bleeker, P. van den Dijnden, and F. de Jong, "BoundaryScan Test-A Practical Approach. Kluwer Academic Publishers," Dordrecht, Netherlands, 1993.

[13] V. Iyengar, K. Chakrabarty and E.J. Marinissen, "On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization," In Proceedings IEEE VLSI Test Symposium, pp. 253-258., April. 2002.

[14] V. Iyengar, K. Chakrabarty and E.J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip," In Proceedings IEEE Internal Test Conference, pp.1023-1032, Oct. 2002.

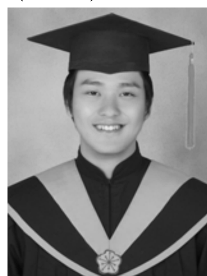
[15] V. Iyegnar, K. Chakrabarty and E. J. Marinissen, "Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip," In Proceedings IEEE Transaction on Computers, pp. 1619-1631, Dec. 2003.

[16] S. K. Goel and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SOCs," In Proceedings IEEE International Test Conference, pp. 529-538, Oct. 2002.

[17] Y. Huang and S.M. Reddy, W.-T. Cheng, P. Reuter, N. Mukherjee, Chien-Chung Tsai, O.Samman and Y. Zaidan, "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm", In Proceedings IEEE International Test Conference, pp. 74-82, Oct. 2002.

[18] E. Larrson and Z. Peng, "An Integrated framework for the Design and Optimization of SOC Test Solution", In Proceedings Date, Automation and Test in Europe Conference and Exhibition 2001, pp.138-144, March. 2001.
[19] A. Sehgal and K. Chakrabarty, "Efficient Modular Testing of SOCs Using Dual-Speed TAM Architectures", In Proceedings Date, Automation and Test in Europe Conference and Exhibition, pp. 422-427, Feb. 2004.
[20] Q. Xu and N. Nicolici, "Multi-frequency Test Access Mechanism design for Modular SOC Testing," In Proceedings 13th Asian Test Symposium, pp. 2-7, Nov. 2004.
[21] K. Chakrabarty, V. Iyengar and M.D. Krasniewski, "Test Planning for Modular Testing of Hierarchical SOCs," In Proceedings IEEE Transactions on Computer- Aided Design of integrated Circuits and Systems, pp. 435-448, March, 2005.
[22] K. Chakrabarty, V. Iyengar, M.D. Krasniewski and G.N. Kumar, "Design and Optimization of Multi-level TAM Architectures for Hierarchical SOCs," In Proceedings 21st VLSI Test Symposium, pp. 299-304, April. 2003.
[23] A. Sehgal, K. Chakrabarty and V. Iyengar, "SOC Test Planning Using Virtual Test Access Architectures," In Proceedings IEEE Transactions on Very Large Scale Integration Systems, pp. 1263-1276, Dec. 2004.
[24] A. Sehgal, K. Chakrabarty, V. Iyengar and M.D. Krasniewski, "Test Cost Reduction for SOCs Using Virtual TAMs and Lagrange Multipliers", In Proceedings Design Automation Conference, pp. 738-743, June. 2003.
[25] Agilent Technologies. Winning in the SOC market, available online at: http://cp.literature.agilient.com/litweb/
[26] K. Chakrabarty, "Optimal test access architectures for system-on-a-chip", ACM Trans Design Automation of Electronic Systems, 2001, pp. 26-49.
[27] Md. S. Quasem, S. Gupta, "Designing reconfigurable multiple scan chains for systems-on-chip", VLSI Test Symposium, Proceedings. 22nd IEEE, 2004, pp. 365-371.
[28] E. J. Marinissen, V. Iyegnar, and K. Chakrabarty, "ITC2002 SOC benchmarking initiative". http://www.extra.research.philips.com/itc02socbenchm
[29] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Efficient Wrapper/TAM Co-Optimization for Large SOCs". In Proc. Design, Automation, and Test in Europe (DATE), pages 491-498, March 2002.
[30] K. Chakrabarty, "Design of System-on-a-chip Test Access Architecture Using Integer Linear Programming". In Proceedings IEEE VLSI Test Symposium.

Jiann-Chyi Rau was born in Miaoli, Taiwan, Republic of China. He received the B.S. degree in computer and information science in 1993 from Tunghai University, the M.E. and Ph. D. degrees in computer science and information engineering in 1995 and 2000 from National Chung-Cheng University. In 2001, he joined the Department of Electrical Engineering at Tamkang University, Taipei, Taiwan, where he is being an assistant professor. In 2003, he received the Young Professor Chair from MXIC. His research interests include VLSI design and testing, logic synthesis and optimization, computer architecture, and computer-aided design (CAD) and electronic design automation (EDA).



Po-Han Wu received B.S. and M.E. degree in electrical engineering from Tamkang University, Taiwan, in 2004 and 2006. He is currently a Ph.D. program in electrical engineering at Tamkang University.

Chih-lung Chien received B.S. and M.E. degree in electrical engineering from Tamkang University, Taiwan, in 2003 and 2005.

Chien-Hsu Wu received B.S. and M.E. degree in electrical engineering from Tamkang University, Taiwan, in 2005 and 2007.