An Efficient Scheduling Algorithm Based On Multi-frequency TAM for SOC Testing

Jiann-Chyi Rau, Po-Han Wu, Jia-Shing Ma
Department of Electrical Engineering
Tamkang University
151, Ying-Chuan Rd. Tamsui,
Taipei County, Taiwan 25137, R.O.C.
{jcrau, phwu, jsma}@ee.tku.edu.tw

Abstract: - In recent years the advance of CMOS technology has led to a great development, especially on the complexity of the system-on-chip (SOC). As the development of circuit with different technology, the embedded cores embedded into system-on-chips (SOCs) usually have multi-frequency to drive it. In this paper, we present a heuristic approach of TAM optimization according to the reality and reduce the test application time. The proposed method is applicable to the design model with hierarchy SOCs. We pay the price in hardware overhead in order to decrease test application time.

Key-Words: - SOC, Testing, TAM

1 Introduction

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 challenge in 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.

1.2 Core test access

A typical core is often deeply embedded in a system chip so that direct physical access to its peripheries is not available by default. Hence an electronic access mechanism is needed to help core users to test the core. This access mechanism requires additional logic, such as a wrapper around the core and wiring, such as a test access mechanism to connect core peripheries to the test sources.

Test-access-mechanism (TAM) is the mechanism that transports the test vectors from the source to CUT and transports the response from the CUT to the sink. Here, the source can be Automatic-Test-Equipment (ATE) or Built-in-Self-Test (BIST) and the sink can be ATE or Multiple-input-signature-register (MISR).

1.3 TAM architecture

The basic TAM architecture is shown in Figure 1. It can be divided into three structural elements [6]: (1) Source and Sink, (2) Test access mechanism (TAM), (3) Core test wrapper.

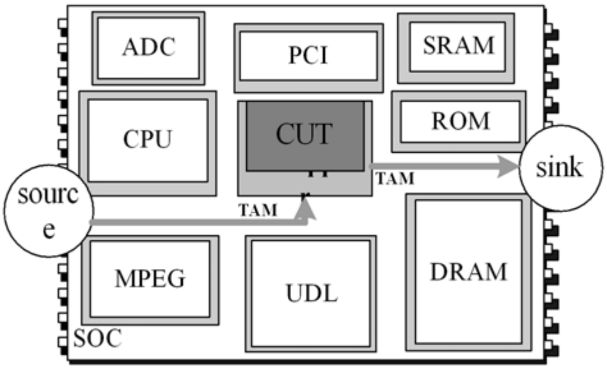


Fig. 1 The test access architecture overview

1.3.1 Basic TAM architecture

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

1.3.2 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.

The modules connected to a common test bus are tested in an arbitrary but sequential order.(note: the order of test for each core may be different in practice). We call this schedule as “serial testing mode”, because the modules are tested in order on each TAM. Test bus architecture only support serial testing mode. We also show the parallel testing mode on the TestRail Architecture in sequel of this session.

1.3.3 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.

1.4 Wrapper architecture

In this section, we will introduce the test wrapper design and explain the relationship between test wrapper design and core test time. 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.

2 Problem Formulation

Because each core in SOC has a lot of information about itself, so they sometimes access the information in order to reduce the data. With the reducing data, we can simplify the problem as easily.

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:

$$t_i = s_i \cdot p_i + p_i + s_o \cdot p_i$$

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:

$$t_i = (1 + \max\{s_i, s_o\}) \cdot p_i + \min\{s_i, s_o\}$$

When the term '+1' indicates that pipelining cannot be used for the scanning out the last pattern.

2.2 The Popular Rectangle Packing Model

E.J. Marinissen presented a Rectangle Packing Model in [15]. In [15], they modeled TAMs 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 Figure 2, 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 Figure 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.

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

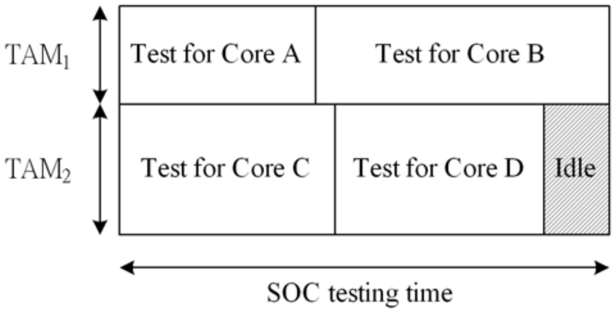


Fig. 2 The test schedule of Fig. 12

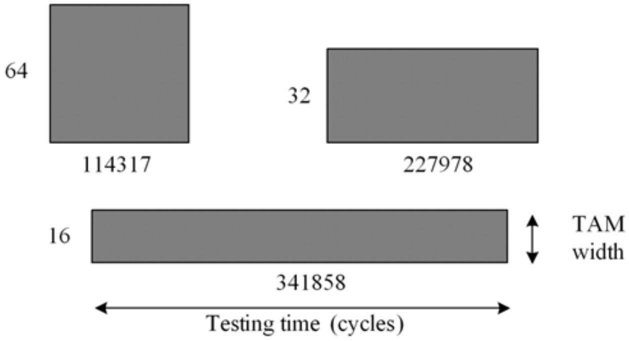


Fig. 3 Example rectangles for Core 6 in SOC p93791

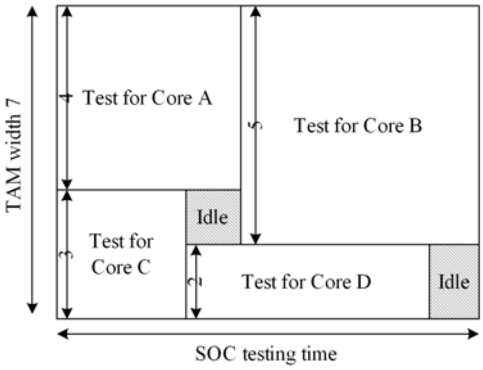


Fig. 4 The test schedule of Fig. 15

2.4 Pareto-optimal points

In general, if we assign the more TAM widths used for a given core test, we get the less time to test the core. So we can achieve the distinct reduction in testing time by adding the TAM width for the core test. But not all values of TAM width between 1 and W, where W is the total SOC TAM width, have time reduction for the core test in practice. It was shown in [17] that for a given core, the testing time varies with TAM width as a "staircase" function.

3 Problem Solution

In this section, we focus on the issue that models the SOC cores reasonable. In order to obtain the prefer model, we take the follow two things into consider. First, how can we get a more efficiency rectangle of each core? Second, the evaluation of the lower bound of each core. These methods can help us to make sure the final result.

3.1 Previous discussion

In order to simplify the test scheduling problem, modular testing has proposed to reduce the complexity of the test access and application [6]. Each core is isolated by its surrounding logic. Such logic include test wrapper and test access mechanism (TAM) which is designed to transfer data from I/O pins independent. This promotes the reuse of pre-computed the cores individually and independent partition for SOC.

Although the problem of the test scheduling to be NP-hard [16], many efficient heuristic technique has be proposed to solve the TAM optimization [15, 16, 18, 19]. However these methods are assumed all of the SOCs cores are flatten and the ATE provides the stimuli to the SOC at the same frequency. In fact, with this optimization technique, we can't obtain the most availability in real world. On the other hand, someone put forward the new method to solve the frequency, hierarchy and power consumption [19, 21, 22, 23, 24]. If we want to test a core at high-speed, we must to be transported the test data at high data rate. This work can be done by ATEs include the Agilent 93000 series tester [27]. By this way, many researches had proposed many methods in order to speed up the test finish time. In [21, 22], although they proposed the multi-frequency to use the ATE resource adequately, but they still not include the hierarchy condition; in [23, 24], the hierarchy has considered, but its structure does not has very great elasticity; in [19], the heuristic method has take care of the power consumption, but it's a pity that it doesn't take hierarchy into account. First, we define the optimization problem as follow and then we propose a complete a complete test solution, it will describe in section 4.

3.2 Problem statement of bin packing

We assume some parameters for SOCs; the total SOC TAM width W, the total available high-speed TAM width V (V < W), the ratio of high-frequency to low-frequency f_ratio, the Σ which indicates the maximum power dissipation of the SOC. And we determine each core's wrapper design, TAM width and its test schedule according to the core's frequency, such that (i) the total TAM width does not exceed the maximum contain W and V, (ii) minimize the SOC testing time, (iii) total power consumption can not over Σ at any piece of time interval.

All of the embedded cores have its own information include the primary inputs, primary outputs, Bi-direction I/O ports, internal scan chain lengths, test patterns and peak power consumption. Then according to [15], we can compute its rectangle.

All the above describe is similar to the method of dual-speed rectangle. However, dual-speed rectangle is 2-D format [21]; the information of rectangle contains width and test time. But here we use the concept of 3-D bin packing [19]. Besides the two constraints that mention in [21], we add the each core's peak power information into consider, and change the rectangles into 3-D format, e.g. figure 6.

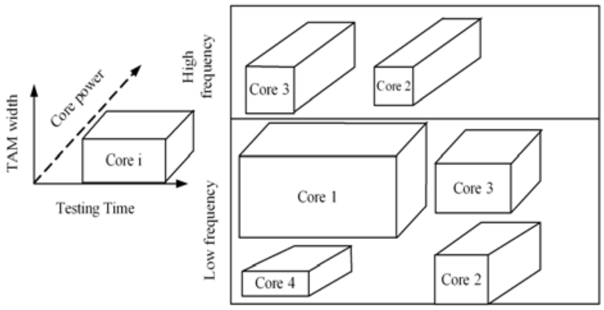


Fig. 6 3-D rectangle model

As regards the same core, if the core works at the same frequency with different width, we assume its power consumption Pi is the same. But if the core can operate at high frequency, its test time can decreases vary much with the same width which compares to the core works at low frequency, but the power consumption compares to the low frequency will increase a lot. The relation can be express as equation (1).

The parameter Pireal means the core i real power consumption when test scheduling. The f_ratio indicates the ratio which high-frequency compared to low frequency, can be expressed by equation (2). The most important is that at any time interval, the total peak power can not over the maximum power dissipation Σ. The Σ indicates the maximum power consumption when SOC under testing, and the parameter can be user defined.

$$P^{i}_{real} = P_i \times f\_ratio \quad \text{(Eq. 1)}$$

$$f\_ratio = \frac{high\_frequency}{low\_frequency} \quad \text{(Eq. 2)}$$

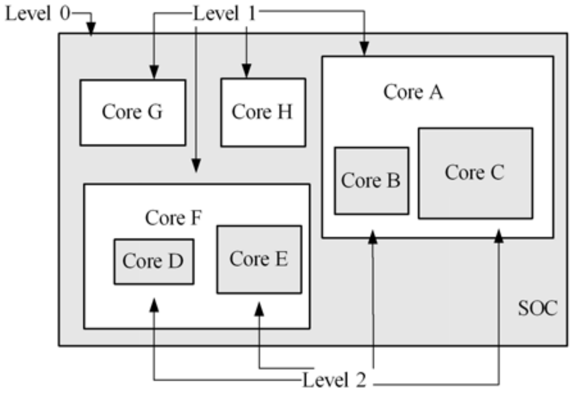


Fig. 7 Illustration of hierarchy circuit

Beside this we also include the hierarchy in considering. From figure 7 we can see each core's relation in the SOC. Because core B and C are embedded in core A, when core A works, core B and core C can not work at the same time. Come to say to the other hand, core G and core B does not conflict with each other. Sometimes we also consider the use of TAM width of a core according to the relation of circuit level. For example, if core A uses n TAM widths at most, the core which contained in it also has n TAM width at most to use at single-speed TAM wire. But if core A can be runs at high-speed and core B and C can just runs at low-speed, the width that B and C can use can be increasing to n * f_ratio.

This concept is called Virtual TAM [25, 26]. Like figure 8. WH indicates the TAM line which can operate at high speed, WL indicates the TAM which can work just at low speed. By the way, W, the total available TAM width, is the summation of the two types, can be expressed as equation (3).

$$W = W_L + W_H \times f\_ratio \quad \text{(Eq. 3)}$$

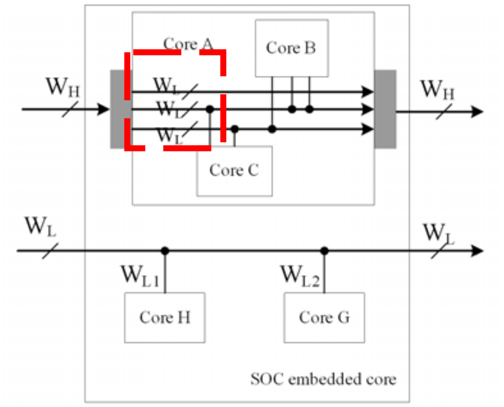


Fig. 8 Example of virtual TAM

And the gray region is the virtual mechanism. This idea not only increases the TAM width for embedded core to use when test scheduling, but also reduce a lot of test application time. According to these that we mentioned, we target at the TAM-optimization for SOC containing power consumption with hierarchy for reducing test application time in real-world.

3.3 Evaluation of the lower bound

Because our approach is extend from dual-speed [21], so we continue to use the method to evaluate the testing time for our architecture. We briefly describe the concept. A test time of lower bound for a core can be computed as equation (4):

$$T_i(w) = \lceil (\max(ts_i, tr_i) \cdot p_i + \min(ts_i, tr_i)) / w \rceil + p_i \quad \text{(Eq. 4)}$$

,where tsi is the scan-in length that scanned into the core i; tri is the scan-out length that scanned out of the core i; pi is the total patterns that applied to the core i. For single-speed architecture, the evaluation of the rectangle relates to its width (W) and testing time (T). The area of the rectangle is given by T×W.

With different wrapper design, each core has its own set of different areas. The area of core i that we can represented as Ri(w) = Ti(w)×w. As we know, the actual total area can not less than the minimum total area, so we simply express the test time as this: $T \geq \sum_{i=1}^{N} R_i^{\min} / W$ (Eq. 5)

In dual-speed TAM optimization, this idea of rectangle packing extends into two parts. Let xi1=1 (xi2=1) if core i assigned to high (low) bin. Significantly, xi1+ xi2=1, 1 ≤ i ≤ N. The object is to minimize C:

$$C \geq T_h \quad \text{(i)} \qquad T_h \geq (\sum_{i=1}^{N} R_{ih}^{\min} \cdot x_{i1}) / V \quad \text{(ii)}$$

$$C \geq T_l \quad \text{(iii)} \qquad T_l \geq (\sum_{i=1}^{N} R_{il}^{\min} \cdot x_{i2}) / (W - V) \quad \text{(iv)}$$

$$x_{i1} + x_{i2} = 1 , \; 1 \leq i \leq N \quad \text{(v)}$$

In section 5, we will present the lower bound result and compared with others.

4 Heuristic optimization algorithm

In this section, we will explain our heuristic method to solve the problem that we described before. We extended the approach from dual-speed TAM optimization [21] to solve the rectangle packing over two bins. According to the concept of rectangle packing [15], we can understand the reliable of test schedule depends on certain preferred width to each core. But it does still determine by the enough available TAM width. After that, when the core finished, its TAM width can be freed to reassign to another cores as available TAM wires. This character makes the following unscheduled core can obtain the best preferred width for testing, as long as there are more TAM lines available.

4.1 Initialization

According to each core's circuit information, we can compute all of core rectangles at the beginning. The data of rectangle that contains widthl(i) (width-h(i)),the preferred width for core i at low-frequency (high-frequency); width(i), the TAM lines assign to core i exactly; begin(i), begin time of core i; end(i), end time of core i; pow(i), the real power dissipation of core i; freq(i), boolean value, indicates whether the core can run at high-speed or not; schedule(i), boolean value, indicates the core whether be scheduled or not; complete(i), boolean show whether the core be finished the test or not.

As the knowledge we have, each core can run at low-speed, so we first compute all of the low-frequency rectangles. Then refer to the parameter freq(i), if the core can operate at high-speed we must compute its high-frequency rectangle for scheduling. Last, because the rectangle is constructed by T×W, according to the complexity of T and W, as the knowledge we have, the more test time or the more TAM widths we have to pay, the more difficulties we must to solve, under this CUT (circuit-under-test). So we sort the rectangles by decreasing sequence according to each core's low-frequency data, in order to make the most complicated circuit cab be solved as soon as possible.

4.2 Part-scheduling

Here we pick the rectangle for scheduling into the appropriate bin under the constraints that we define. The approach is almost the same as rectangle packing [15], and here we just present the differences with its. Similarly, we still consider the parameter freq(i). If freq(i)=1, we just put the core i into the high-speed bin; oppositely, if freq(i)=0, we let the core i schedule into the low-speed bin. Here, we should make sure the available TAM lines prepare for specified core is enough to use.

After checking freq(i), we take the hierarchy level into account. Here, we use a parameter level(i) to record the relation of level between each hierarchy cores. Before this, we first take the top level (level 0) module into account lastly. And we will trace the level to the top-module if needed. As example like figure 6, level (C) = A, level (B) = A, if core B picks out and prepares for scheduling, we must check the core A has been scheduled or not. Like figure 9, we simply use 2-D format to explain. Figure 9 violates the rule what we proposed, but figure 9 is allowable.

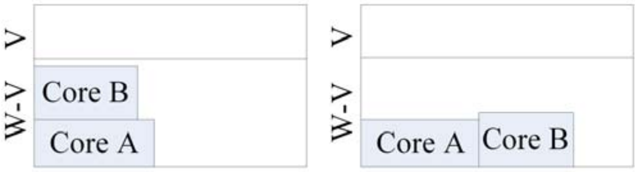


Fig. 9 Core's test scheduling

Even thought this method will increase the SOC test finish time, but its behavior be more closed to the real-world circuit. Finally, we should check the power consumption is satisfied with our restriction or not. For all of the scheduled core i, we check the total power dissipation at this_time. The pointer this_time indicates the earliest time at which TAM lines are available between two bins and next_time indicates the minimum width to which either of the two bins is filled.

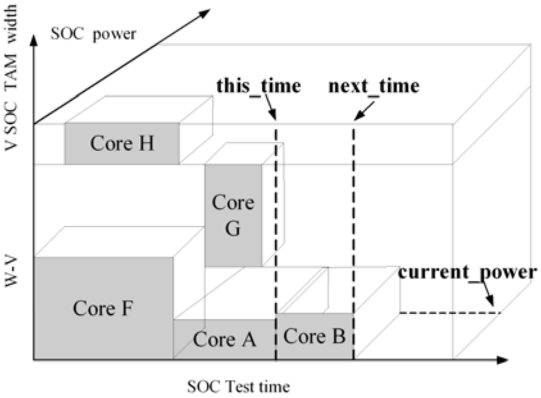


Fig. 10 3-D bin packing with hierarchy

4.3 Optimized-scheduling for each level

Because of the available TAM lines is limited, sometimes TAM lines may be not enough for core to be assigned. If now we want to schedule a core, we must wait for more TAM lines freed. This might be wasted a lot of idle spaces in the schedule. In order to make use of the remainder TAM lines more efficiently, we adopt two methods to solve this problem.

4.3.1 Minimize the idle space

First, refer to the TAM_optimizer procedure presented in rectangle packing [21], we extend the idea. If there exist idle TAM width w_idle (w_avail≠0), for all of the scheduled core under the condition of begin(i)= this_time, find out the minimum end time end(i), then assign w_idle to the remainder unscheduled core j that has the maximum end time end(j) under width w_idle, such that end(j)<end(i), i≠j. By this way, we can obtain more benefit by assigning relatively less TAM lines to unscheduled core. For example of figure 11, we have the idle space and we want to fill it.

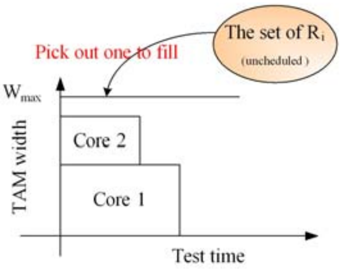
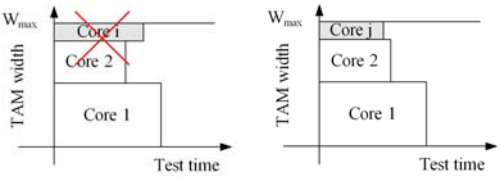


Fig. 11 Fill the idle time



(a) (b)

Fig. 12 (a) Fill the idle time (b) Fill the idle time

4.3.2 Redistribution of TAM lines

In addition, we use another method to reduce test application time further. Sometimes we may not find the core assign w_idle suitably. It means there still have available TAM widths that leave unused.

In the past, TAM_optimization [21] assign all of the w_idle to the core that has the maximum reducing time. This is not efficiency to the test schedule. Here, for all of the cores' start time which the same as the beginning of the idle space, we find the core which has maximum end time to assign one TAM line once until w_idle=0.

For example of figure 13, we assume there is no core match our condition to fill the idle space. So we distribute the available width (assume the available TAM width is 2) to the core that scheduled.

First, we pick the scheduled core which has the largest end time (here we pick core 1) and assign one TAM line to it (now, the remainder available width is 1), like figure 14(a). After that, we re-compute the test application time and compared to core 2, to find out which one has the largest end time (it is still core 1). Finally we assign the last one TAM line to core 1 and obtain its test application time, figure 14(b). By this way, the available TAM lines can be balanced distributed and the test time can be reduced averagely.

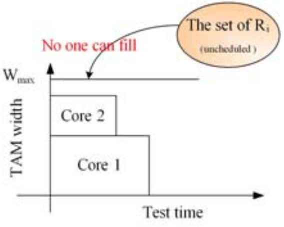
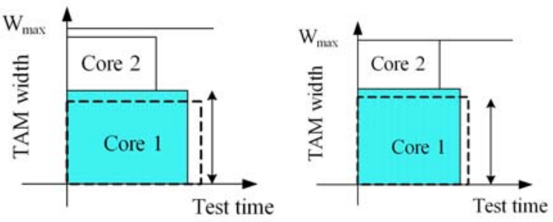


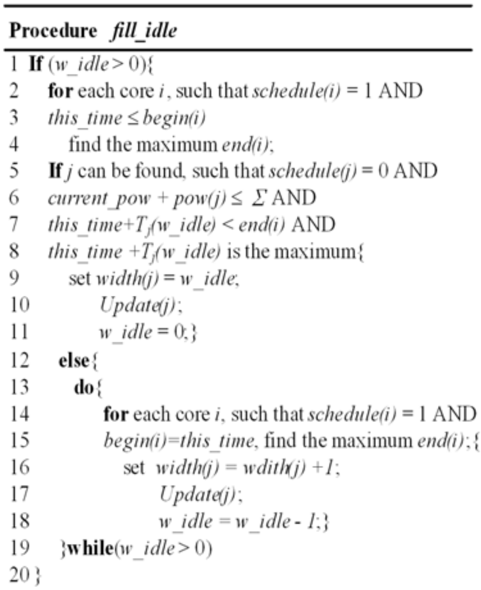
Fig. 13 Minimize the idle space further



(a) (b)

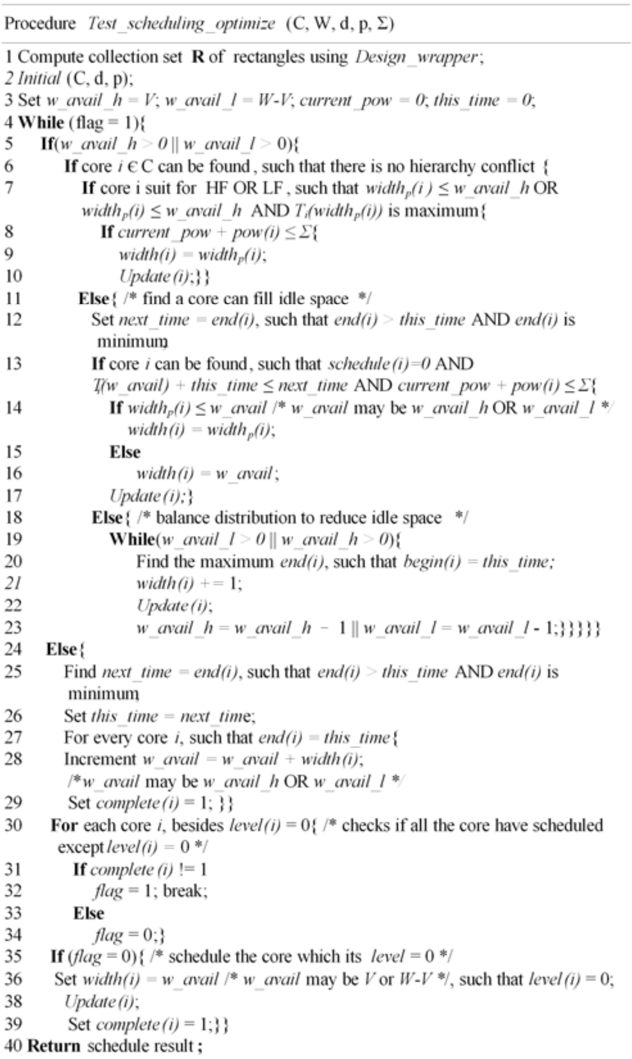
Fig. 14 (a) Minimize the idle space further (b) Minimize the idle space further

According to this approach, we average the TAM widths as possible, in order to avoid wasting TAM widths. In figure 15, we just show the pseudo code of our algorithm that different from TAM_optimizer.



```
Procedure fill_idle
If (w_idle > 0){
   for each core i, such that schedule(i) = 1 AND
   this_time ≤ begin(i)
      find the maximum end(i);
   If j can be found, such that schedule(j) = 0 AND
   current_pow + pow(j) ≤ Σ AND
   this_time+$T_j$(w_idle) < end(i) AND
   this_time +$T_j$(w_idle) is the maximum{
      set width(j) = w_idle;
         Update(j);
         w_idle = 0;}
    else{
      do{
         for each core i, such that schedule(i) = 1 AND
         begin(i)=this_time, find the maximum end(i);{
            set  width(j) = wdith(j) +1;
               Update(j);
               w_idle = w_idle - 1;}
    }while(w_idle > 0)
}
```

Fig. 15 Pseudo code for fill_idle



```
Procedure Test_scheduling_optimize (C, W, d, p, Σ)
Compute collection set R of rectangles using Design_wrapper;
Initial (C, d, p);
Set w_avail_h = V; w_avail_l = W-V; current_pow = 0; this_time = 0;
While (flag = 1){
    If(w_avail_h > 0 || w_avail_l > 0){
        If core i ∈ C can be found , such that there is no hierarchy conflict {
            If core i suit for HF OR LF , such that width_p(i ) ≤ w_avail_h OR
            width_p(i) ≤ w_avail_h AND T_i(width_p(i)) is maximum{
                If current_pow + pow(i) ≤ Σ{
                    width(i) = width_p(i);
                    Update (i);}}
            Else{ /* find a core can fill idle space */
              Set next_time = end(i), such that end(i) > this_time AND end(i) is
               minimum
              If core i can be found, such that schedule(i)=0 AND
              T_i(w_avail) + this_time ≤ next_time AND current_pow + pow(i) ≤ Σ{
                If width_p(i) ≤ w_avail /* w_avail may be w_avail_h OR w_avail_l */
                  width(i) = width_p(i);
                Else
                    width(i) = w_avail;
                Update(i);}
              Else{ /* balance distribution to reduce idle space */
                While(w_avail_l > 0 || w_avail_h > 0){
                    Find the maximum end(i), such that begin(i) = this_time;
                    width(i) += 1;
                    Update(i);
                    w_avail_h = w_avail_h - 1 || w_avail_l = w_avail_l - 1;}}}}}
    Else{
        Find next_time = end(i), such that end(i) > this_time AND end(i) is
         minimum
        Set this_time = next_time;
        For every core i, such that end(i) = this_time{
        Increment w_avail = w_avail + width(i);
         /*w_avail may be w_avail_h OR w_avail_l */
         Set complete(i) = 1; }}
    For each core i, besides level(i) = 0{ /* checks if all the core have scheduled
      except level(i) = 0 */
        If complete (i) != 1
            flag = 1; break;
        Else
            flag = 0;}
    If (flag = 0){ /* schedule the core which its level = 0 */
     Set width(i) = w_avail /* w_avail may be V or W-V */, such that level(i) = 0;
      Update(i);
      Set complete(i) = 1;}}
Return schedule result ;
```

Fig. 16 The pseudo code of the algorithm

4.4 Finalize overall SOC scheduling

Repeat above the mention steps until all of the cores finished test scheduling which do not contain level 0 module. We assign the maximum TAM width w_max to level 0 module and schedule it to the bin which according to its specific frequency. Figure 17(b) expresses the hierarchical test schedule result on the benchmarck core p34392.

After these complex steps, we can obtain the entire SOC test application time finally. The complete schedule chart will be like figure 10.

Among the pseudo code we presented, w_idle(i) and T(i) can be subdivided into two sets, low-speed w_idle_l(i) and high_speed w_idle_h(I); T_h(i) and T_l(i), according to the parameter freq(i). It means we must check high-speed and low-speed bin at the same time as considering the procedure fill_idle. No matter how these value change, the two data, begin(i) and end(i) will truly record the order between each core. Figure 16 shows the pseudo code of this algorithm.

5 Experimental Results

In this section, we performed some simulation experimental result to compare our technique with [19, 24]. With these benchmarks, p93791, p34392 and p22810 have the character of multi-level in their circuit; only h953 has the information about power consumption.

5.1 Comparison with previous work

Table 1 shows the detail power information which obtained from ITC’02 benchmarcks h953.

Table 2 shows the result for a given number of TAM widths with power restriction under single speed for h953. Compared with the previous work, because of h953 without hierarchy, therefore the optimization result is according to the number of the TAM widths and the bottleneck core.

Table 1 Power information about h953

h953 module	Power
1	5,658,600
2	5,753,800
3	59,000
4	33,200
5	178,800
6	20,425,228
7	1,420,875
8	3,025,200

Table 2 Test application time for benchmark h953

TAM width SoC power		16	24	32	40	48	56	64
$6*10^9$	Method in [19]	122,636	122,636	12,636	122,636	122,636	12,636	12,636
	Our method	12,636	12,636	12,636	12,636	12,636	12,636	12,636
$7*10^9$	Method in [19]	119,357	119,357	119,357	119,357	119,357	119,357	119,357
	Our method	119,357	119,357	119,357	119,357	119,357	119,357	119,357

In Table 3, we compared the optimization result used in hierarchy SOC benchmark with [24]. With our method the TAM width can be changeable if needed, but in [24], the TAM width is fixed after the first core which scheduled in test each TAM. ∆T is calculated as {(TM - Th) / TM}* 100 and ∆ ε = {(Th - Tf) / Tf}* 100. With our method, we can reduce test time about 1.26% to 59.34% compared with the result in [24]. Relative to the case of flatten Tf, the increasing of test time Th is between 0% and

8.33%. With this comparison, we can know our method can be more flexible and more efficient under the hierarchy SOCs.

Table 3 Test application time for (a) p22810, (b) p34392, (c) p93791, compared the method in [11].

W	Method in [24] T_M	Our method (hierarchy) T_h	Our method (flatten) T_f	Δ T (%)	Δ ε (%)
16	505,858	499,491	492,657	1.26	1.36
24	412,682	347,305	346,503	15.84	0.23
32	396,473	286,786	285,205	27.67	0.55
40	366,260	225,253	224,653	38.5	0.26
48	366,260	198,881	198,380	45.7	0.25
56	366,260	176,291	163,923	51.87	7.01
64	366,260	148,938	148,538	59.34	0.26

(a)

W	Method in [24] T_M	Our method (hierarchy) T_h	Our method (flatten) T_f	Δ T (%)	Δ ε (%)
24	1,347,023	838,871	838,643	37.72	0.02
32	788,873	594,097	544,579	24.7	8.33
40	728,426	581,731	544,579	18.6	6.38
48	618,597	581,729	544,579	5.6	6.38
56	618,597	581,702	544,579	5.96	6.38
64	618,597	569,336	544,579	7.96	4.34

(b)

W	Method in [24] T_M	Our method (hierarchy) T_h	Our method (flatten) T_f	Δ T (%)	Δ ε (%)
16	2,044,124	1,861,913	1,861,913	8.91	0
24	1,351,710	1,390,322	1,390,322	-2.85	0
32	1,087,300	1,117,314	1,117,314	-2.76	0
48	839,796	811,018	811,018	3.43	0
56	839,796	728,992	721,921	13.19	0.97
64	839,796	639,898	639,898	23.8	0

(c)

Table 4 shows the result under dual-speed constraint. We assume the relation between high-speed and low-speed as fh = 2*fl. Therefore, according to Table 4, if there exit more cores within the SOCs can run at high-speed and the available TAM width is enough to use, the test time can reduce a lot.

Table 4 Test application time with dual-speed for p34392

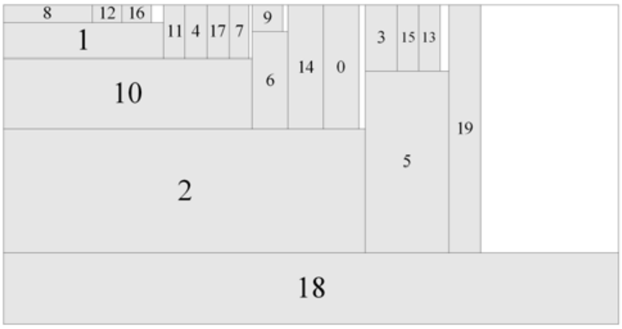
Width \ V/W	0	0.5
16	1,126,062	869,908
24	838,871	683,252
32	594,097	537,935
40	581,731	419,580
48	581,729	419,549
56	581,702	321,835
64	569,336	309,470

Additionally, we present the final result of bin-packing of p34392. By Table 1, we can prove our simulation result is accurate under the condition that we presented. Here we show the Table 1 by simple 2-D format under the relation of hierarchy and flatten. We also show the scheduling result with dual-speed by Table 2.

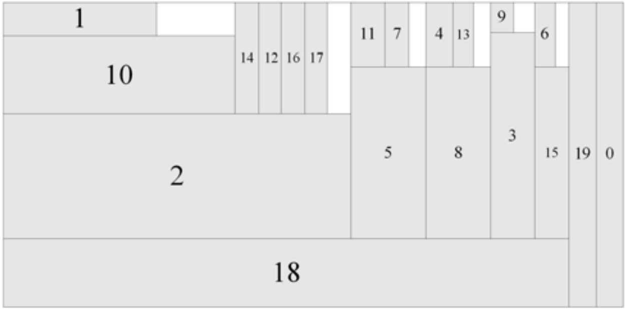
6 Conclusion

We have shown the method of TAM optimization to solve the real-world problem. As the result we presented, we can reduce a lot of test time by using virtual width as needed. Although we pay the price of area additionally to achieve this effect, but in recent years, the cost of area is not the major problem with SOCs.

With the advance of the CMOS technology, the core operating frequency goes higher and higher and the power consumption is less than before. In general speaking, the complexity of the SOCs grows up with the integration between each core.

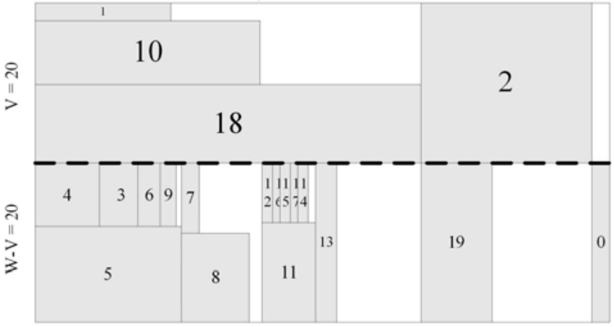


(a) Flatten: test time = 544579 time unit



(b) Hierarchy: test time = 581731 time unit

Fig. 17 Final bin-packing for p34392 (width = 40, single speed)



Hierarchy test time = 419580 time unit

Fig. 18 Final bin-packing for p34392 (width = 40, dual-speed)

If we can make a good manner before test scheduling, then we can easily to obtain a well plan on SOCs integrating. Therefore, the test application time can be reduced further when testing. All of the important we have to do is to make a beneficial testing tactic under these rigid restrictions.

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] M. Keating and P. Bricaud, "Reuse Methodology Manual for System-on-Chip Designs," Kluwer Academic Publishers, 1998.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Agilent Technologies. Winning in the SOC market, available online at: http://cp.literature.agilient.com/litweb/