Abstract—a new method is presented for TAM/wrapper co-optimization problem. The new method is based on Hybrid Genetic Algorithm and 2D rectangle packing problem: core test is represented by rectangles, and hybrid genetic algorithm that provides highly optimal solution for 2D packing problem is introduced for TAM allocation and test scheduling. The flexible-width TAM architecture is combined with the HR algorithm to calculate the value of fitness function in HGA. This HGA based method was implemented in C and applied to ITC’02 SOC Test Benchmark. Experimental results show that lower testing time was obtained by this new method compared to other methods [1, 4, 5].

I. Introduction

The development of SOC technology reduces the cost of electronic products. However, the highly integrated functional modules make SOC testing more complex at the same time. A lot of IC companies can not afford the expensive cost of testing. Therefore, academia and industry have been trying to find an effective test program to reduce test cost and shorten the testing time. Basing on the IEEE P1500 test Strategy, SOC test optimization problem can be divided into Wrapper/TAM design and test scheduling optimization, these problems have been researched by scholars and certain solutions have been proposed. Most prior researches have studied wrapper/TAM design and test scheduling optimization as independent problems [4, 6]. In this studies, modus operandi is to divide a test bus of certain width into several groups, the wrapper of IP core was connected to a certain group of bus, each IP core use a fixed TAM width throughout the testing process. However, there is close relationship between Wrapper/TAM design and test scheduling optimization, solving these two problems in conjunction will certainly shorten the testing time.

V. Iyenga introduced a co-optimization in 2002[1]. In his paper, a flexible-width TAM architecture is proposed to minimize the waste of test resources and ILP algorithm is used to solve the test scheduling problem. Studies have proved that ILP algorithm is not very effective for solving test scheduling problem, so the method in [1] didn’t obtain desired result. Addressing all issues above, flexible-width TAM architecture is combined with Hybrid genetic algorithm to complete the co-optimization in this paper.

In the new method, BFD algorithm is used to
completed the design of wrapper to minimize the test time of each IP core, and Hybrid genetic algorithm is used to solve the problem of TAM design and test scheduling. The method which combine HR algorithm with design of flexible_width TAMs is used to Calculate the fitness of chromosome.

II. SOC test optimization model

The co-optimization problem in this paper is as follows:

Given the total TAM width Wmax for the SOC, the wrapper and TAM assignation should be determined for each core, and a test schedule should be design for the SOC, such that:
1. The total number of TAM wires utilized at any moment does not exceed Wmax.
2. The overall SOC test completion time is minimized.

In this paper, co-optimization is transformed into rectangle packing problem. The rectangle packing problem is described as follows:

Given a collection of rectangles, and a bin of fixed width and unbounded height, pack the rectangles into the bin, such that no two rectangles overlap, and the height of the bin is minimized. In the co-optimization model, the test of each core will be represented by a rectangle. The width of the rectangle corresponds to the TAM width assigned to the core, and the height of the rectangle corresponds to the testing time of the core. When core test is represented with rectangle, two parameters must be initialized: the width and height of each rectangle, namely the width of bus assigned to each IP core and the test time of the core for this width. The method described in [1] is used to initialize these two parameters. The pseudocode for the initialization is presented in Figure 1 and the definition of highest Pareto-optimal width can be seen in [1].

The parameters in the pseudocode are as follows:
◆ wpi—preferred TAM width for core i;
◆ Tpi—testing time of core i for the width of wpi;
◆ Ti(w)—testing time of core i for the width of w;
◆ wi—the width of rectangle for core i;
◆ hi—height of rectangle for core i;
◆ Wmax—the TAM total width;
◆ pc—control parameter, control the value of Ti and thus control the value of wi;
◆ dc—control parameter, control the equivalence probability between wpi and highest Pareto-optimal width;
In this paper, BFD algorithm is used to design a set of wrappers for each core. A preferred TAM width is assigned to each core in line 1-9, and then BFD algorithm is used for calculating the test time of each core for the width of wpi. Width and height of rectangle i are made respectively equal to the preferred TAM width and test time of the core i. A set of rectangles for core test can now be constructed and the test scheduling is transmuted into a process of packing, the goal of optimization is to pack all rectangles into a bin of fixed width (Wmax) with using a smallest height of the bin, namely using the shortest testing time.

1. compute collection A of cores of SOC
2. Set g= the quantity of cores belong to C
3. FOR i = 0 to g
4. Set T(Wmax)=BFD(i,Wmax), T(1)=BFD(i,1)
5. calculate T=T(Wmax)+
6. Set wp=wh, such that T(w)-T is minimum
7. Calculate highest Pareto-optimal width wh;
8. IF wp<wh then
9. Set wp=wh, Tp=BFD(i,wp),
10. END
11. END
12. FOR i = 0 to g
13. Set w_i=wp, h_i=tp_i
14. END

Figure 1. Initialization for rectangle

III. Co-Optimization based on HGA

In this paper, hybrid genetic algorithm [7] is used to solve the problem of test scheduling. First, the search ability of genetic algorithm is used to find the order in which IP cores are connected to the bus, namely the order in which rectangles are filled into the box. Then fitness value of chromosome is calculated by the method described in section 3.1. After several iterative operations, the best result of HGA is chosen as the solution for test scheduling problem. Construction of hybrid genetic algorithm is divided into two parts: design of fitness function and design of genetic operators.

A. fitness function

In this paper, the value of fitness function of chromosome is the height of the box that has been filled, namely the testing time of SOC. The flexible-width TAM architecture is combined with HR algorithm to calculate the value of fitness function.

The rule of HR algorithm is as follows [6]:

(1). Insert rectangle to fill idle[1]. Width and height of rectangles can be adjusted because flexible-width TAM architecture is adopted. At this time a rectangle is selected to be inserted into the gap after being assigned a new width, then the waste of space can be reduced. In practice, the width of the gap corresponds to the width of the idle bus, and the height of the gap corresponds to the idle time of the bus. The preferred TAM width that selected core obtained before is replaced by the width of idle bus, then this IP core can be connected to the idle bus. After being connected to the bus, the testing time of the selected core will be calculated by BFD algorithm.

Let Q be the set of IP cores meeting the condition difj=0, the selected core should meet the following condition: difselected<dif (selected ∈ Q,j ∈ Q,selected ≠ j)

Where:
idle_t---idle time of bus;
new_ti--- the testing time of core i after core i being connected to bus; 
dif i --- dif i= idle_t - new_ti;

(2). Widen rectangle to till idle[1]. If the suitable rectangle can not be found in previous step, a filled rectangle will be selected to obtain an extra width of idle_w, so that its height is decreased (figure 6). In practice, an IP core that have been connected to bus is selected to obtain an extra width of idle_w, then test time of this IP core will be shortened. After obtaining an extra width, the testing time of the selected core will be calculated by BFD algorithm.

Let P be the set of IP cores meeting the condition that begini=idle_begin, the selected core should meet the following condition:

\[ D_n > D_i (sel \in P, j \in P, selected \neq j) \]

Where:

- idle_begin---time when bus enters idle status;
- begin_i --- time when the test of core i begin;
- orig_ti --- testing time of core i before it obtaining the extra width;
- new_ti --- testing time of core i after it obtaining the extra width;
- Di --- Di=orig_ti - new_ti ;

B. realization of HGA

Hybrid genetic algorithm in this paper is based on real coding. The value of ith gene of the chromosome represents the test sequence of core i. Hybrid genetic algorithm is completed by using some genetic operators described in [7]. The fitness value is calculated by the method described in 3.1. Detailed process of HGA is as follows:

Step1: Population initialization. A sequence coded chromosome is used as the first chromosome. Two random breakpoints are selected in this chromosome. The genes on both sides of the breakpoints are exchanged. This method is used to get other chromosomes.

Step2: Calculate the fitness value of individuals of population.

Step3: Crossover. Two individuals are randomly selected from the old generation. Two new chromosomes are generated from the couple of selected chromosomes by using the method of single-point crossover. This operation was executed POPSIZE / 2 times to get the new generation.

Step4: Mutation. For each chromosome, two random breakpoints are selected and the genes between these two breakpoints are swap. If the fitness value of the new chromosome is lower than the old chromosome, the new chromosome is accepted as the offspring; otherwise the old chromosome is accepted as the offspring.

Step5: Optimal preservation strategy. Chromosome with the highest fitness value in new generation is replaced by the chromosome with the lowest fitness value in old generation.

Step6: If the number of iterations reaches MAXG, return to step2; otherwise, the shortest time is return and the corresponding chromosome is selected as the solution of co-optimization.

MAXGENS is the largest number of iterations, POPSIZE is the population size, PXOVER is the crossover probability. In this paper, MAXG=100~400, POPSIZE=50, PXOVER=0.8.

IV. Experimental results

Our algorithm is implemented in C, and experiments is conducted on Intel Pentium IV, 2.5G Hz processor with 256 MB memory.

In this section, experimental results are presented. These results are for three benchmark SOCs: d695, p93791, p22810. d695 is an academic benchmark which contains 3 memory cores and 8 scan-testable logic cores. p93791 is the largest example SOC which contains 18 memory cores and 14 scan-testable logic cores. SOC p22810 contains 6 memory cores and 22 scan-testable logic cores.

All possible integer values of the parameters pc and dc in the range 0<pc<10, 0<dc<4 are considered and the best results are shown in tables.

<table>
<thead>
<tr>
<th>Table 1. Experimental results for d695</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wmax Tnew</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>64</td>
</tr>
</tbody>
</table>
A new co-optimization method for SOC test is proposed in this paper. In the new method Wrapper/TAM optimization and test scheduling problem are transformed into rectangle packing problem. And then Hybrid genetic algorithm is used to search the best option. At last our co-optimization technique is used to an academic benchmark SOC and two industrial SOCs. Compared with other method, lower test time is obtained by our method in most instances. A careful analysis reveals the reasons for the poor results and to solve the problem related to these reasons is the direction for our further work.

Table 2. Experimental results for p93791

<table>
<thead>
<tr>
<th>( n_{\text{max}} )</th>
<th>( T_{\text{new}} )</th>
<th>( T[1] )</th>
<th>( \triangle T )</th>
<th>GA</th>
<th>( \triangle T )</th>
<th>QEA</th>
<th>( \triangle T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1842659</td>
<td>1851135</td>
<td>-0.46%</td>
<td>1751423</td>
<td>+5.21%</td>
<td>1743162</td>
<td>+5.71%</td>
</tr>
<tr>
<td>24</td>
<td>1244061</td>
<td>1248795</td>
<td>-0.38%</td>
<td>1180948</td>
<td>+5.34%</td>
<td>1173202</td>
<td>+6.04%</td>
</tr>
<tr>
<td>32</td>
<td>948601</td>
<td>975016</td>
<td>-2.71%</td>
<td>935313</td>
<td>+6.19%</td>
<td>877066</td>
<td>+8.16%</td>
</tr>
<tr>
<td>40</td>
<td>754843</td>
<td>794020</td>
<td>-4.93%</td>
<td>723400</td>
<td>+4.35%</td>
<td>705524</td>
<td>+6.99%</td>
</tr>
<tr>
<td>48</td>
<td>593328</td>
<td>627934</td>
<td>-5.51%</td>
<td>593841</td>
<td>+0.09%</td>
<td>590525</td>
<td>+0.48%</td>
</tr>
<tr>
<td>56</td>
<td>534997</td>
<td>568436</td>
<td>-5.88%</td>
<td>511001</td>
<td>+4.70%</td>
<td>508459</td>
<td>+5.22%</td>
</tr>
<tr>
<td>64</td>
<td>501661</td>
<td>511286</td>
<td>-1.88%</td>
<td>485148</td>
<td>+9.50%</td>
<td>441410</td>
<td>+12.96%</td>
</tr>
</tbody>
</table>

Table 3. Experimental results for p22810

<table>
<thead>
<tr>
<th>( n_{\text{max}} )</th>
<th>( T_{\text{new}} )</th>
<th>( T[1] )</th>
<th>( \triangle T )</th>
<th>GA</th>
<th>( \triangle T )</th>
<th>QEA</th>
<th>( \triangle T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>384082</td>
<td>452639</td>
<td>-15.14%</td>
<td>461548</td>
<td>-16.75%</td>
<td>438619</td>
<td>-12.43%</td>
</tr>
<tr>
<td>24</td>
<td>263303</td>
<td>307780</td>
<td>-14.48%</td>
<td>311258</td>
<td>-27.13%</td>
<td>298914</td>
<td>-11.91%</td>
</tr>
<tr>
<td>32</td>
<td>201320</td>
<td>246150</td>
<td>-18.21%</td>
<td>213891</td>
<td>-35.88%</td>
<td>230813</td>
<td>-12.78%</td>
</tr>
<tr>
<td>40</td>
<td>170162</td>
<td>197293</td>
<td>-13.73%</td>
<td>287260</td>
<td>-40.76%</td>
<td>198184</td>
<td>-14.14%</td>
</tr>
<tr>
<td>48</td>
<td>151128</td>
<td>167256</td>
<td>-9.64%</td>
<td>288512</td>
<td>-43.72%</td>
<td>174496</td>
<td>-13.90%</td>
</tr>
<tr>
<td>56</td>
<td>143325</td>
<td>154117</td>
<td>-1.43%</td>
<td>271429</td>
<td>-47.20%</td>
<td>151886</td>
<td>-5.64%</td>
</tr>
<tr>
<td>64</td>
<td>132170</td>
<td>136941</td>
<td>-3.48%</td>
<td>260645</td>
<td>-49.29%</td>
<td>137532</td>
<td>-3.90%</td>
</tr>
</tbody>
</table>

In the table, the percentage difference between testing time obtained by our method and other methods is calculated by the formula \( \triangle T = \frac{(T_a - T_{\text{new}})}{T_a} \), \( T_a \) represents the SOC testing time with using other method and the \( T_{\text{new}} \) represents the SOC testing time obtained by our method. It can be seen from the tables that compared with the method which use the flexible-width TAM architecture[1], the new method proposed in this paper obtain lower testing time in almost all instance. This shows Advantage of hybrid genetic algorithm compared with ILP algorithm. Compared with the GA algorithm and the QEA algorithm, our method obtain lower testing time in most instances for d695 and p22810. However, for p93791, our method obtain comparable or higher testing time. A careful Analysis reveals reasons for this poor result:

1. It is difficult to optimize the system testing time with using only a single value of pc and dc for all cores.
2. The scale of the problem is so large that the algorithms need long computation time, so the iteration number is set to 100, this results that genetic manipulation has been terminated before the best solution obtained.

To solve the problem related to these reasons is the direction for further work.

V. Conclusion

References