

Measure true ATPG performance improvements

By Rohit Kapur
Scientist

Chris Allsup
Marketing Manager
Test Automation Products

Synopsys Inc.

IC tests aim to isolate good devices from those that are defective to reduce the shipped defective parts per million to as low as 100. These low defect levels are achieved by generating high-coverage patterns that target faults, which are abstractions of defects. By making design modifications to assist ATPG algorithms, it is possible to achieve high fault coverage with a reasonable number of test patterns. Like other automation applications, ATPG results can be analyzed in benchmarks and tool improvements can be monitored over successive product generations to achieve more efficient IC test automation.

Intermediate runs

While it is a batch-run process expected to be executed at the end of the design flow, ATPG is run many times in trial modes throughout the design implementation phase. These intermediate

runs are done to determine as early as possible if the test coverage is achievable, and to generate sample patterns for validation. The goal is to overcome any deficiencies in the fault models so that the most compact and highest-coverage patterns are produced at the end of the design cycle.

But these iterations cannot be performed efficiently with long ATPG runs. Consequently, much attention is focused on ATPG runtime performance characterization and improvements. It turns out that ATPG runtime performance—reflected by the CPU time required to generate results—is closely coupled with fault efficiency and pattern count. Fault efficiency represents the percentage of faults that the ATPG has resolved successfully. Pattern count represents the number of tests generated to achieve the fault coverage. While shorter runtimes are always desirable—along with higher fault coverage and lower pattern counts—all these factors cause the algorithms to evaluate a full complement of trade-offs. For example, in practice, lower CPU times and pattern counts are achievable at the expense of fault coverage. Likewise, lower pattern

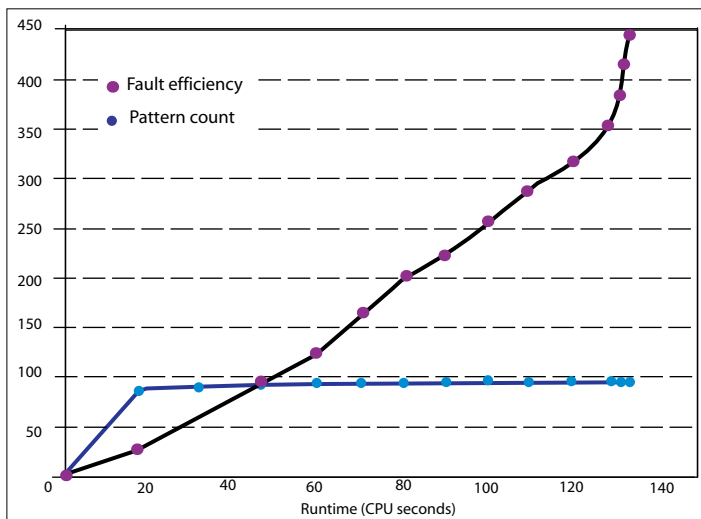


Figure 1: The two curves of fault efficiency and pattern count are quite linear over about 90 percent of the runtime range.

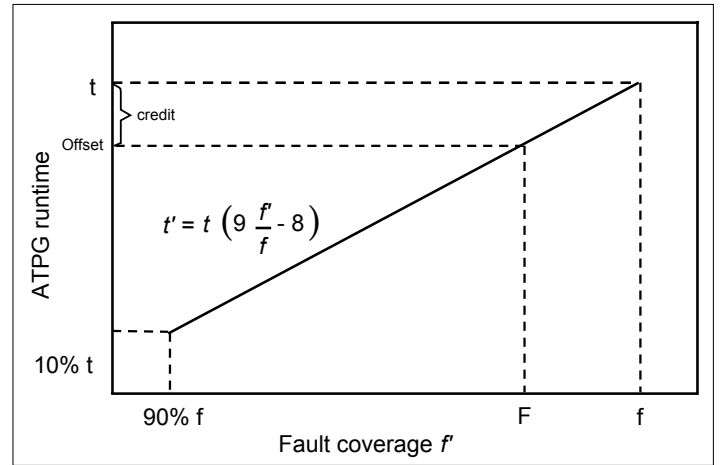


Figure 2: Based on the linear relationship between fault coverage and runtime, difference in fault coverage can be converted into a time 'credit'.

counts are achievable through complex compaction schemes, at the expense of longer runtimes.

Hence, it is difficult to achieve realistic and accurate ATPG performance analysis when running disparate tools to determine runtime, fault efficiency and pattern count. Comparison is possible when two of the three results have similar value, but this seldom occurs.

ATPG numbers

A direct method to measure ATPG performance is to compare runtime numbers vs. fault coverage when the two runtime numbers are within a certain percentage of each other. This approach works fine if the pattern count number is also within a reasonable percentage of the other two. However, this is usually not the case. For most comparative ATPG executions, this assumption cannot be made, hence the need to equilibrate all the ATPG performance metrics for the different executions.

To accomplish this equilibration, compute a metric that equalizes the fault efficiency and pattern count differences in the numbers. **Figure 1** shows the relationship between ATPG runtime, fault efficiency and pattern count for a sample run.

The two curves are quite linear over about 90 percent of the runtime range. After an initial ramp-up in fault coverage, the coverage is a straight line, where achieving the last 10 percent of coverage consumes 90 percent of the total runtime. Similarly, the number of patterns generated grows linearly over about 90 percent of the runtime range, with some anomalies around the ends of the curve. While linearity applies across designs, the slopes of the curves vary from one design to the next.

Improvement factor

We can use this linearity to our advantage. Assume that two versions of the same tool are being executed. The older version provides results of runtime t , pattern count p and fault coverage f . The new version of the same tool with the same settings ends with a runtime T , pattern count P and fault coverage F . Assuming that the two execution runs have similar pattern count and fault coverage, the ATPG performance improvement can be described using a ratio called the Improvement Factor (IF). If the new tool version is better, then IF is simply the ratio of the old runtime to the new runtime: $IF = t/T$ if $p = P, f = F$.

However, if there is a difference in fault coverage, the time to achieve that fault coverage must be accounted for appropriately. Based on the linear relationship between fault coverage and runtime, as well as the observation that 90 percent of the runtime is spent gaining the last 10 percent of the fault coverage, difference in fault coverage can be converted into a time "credit." **Figure 2** shows the ATPG run timeline and its equation in the range of linearity—i.e. between 90 percent and 100 percent of f , and between 10 percent and 90 percent of t .

We can use this equation to determine the runtime offset, which occurs at the ATPG runtime corresponding to F :

$$\text{Offset} = t \left(\frac{9 \frac{F}{f} - 8}{f} \right) = t - \frac{(f - F) \times 0.9t}{0.1f}$$

The IF now reflects a time credit that properly accounts for the fault coverage difference:

$$IF = \frac{t - \frac{(f - F) \times 0.9t}{0.1f}}{T} \quad \text{if } p = P$$

A difference in the pattern count can similarly be translated into time, $(p - P) \times t/p$. However, while equalizing the faults, the time taken to detect those extra faults was already accounted for. By taking credit for the difference in patterns, the patterns generated to detect the difference in faults accounted for is double-counted. The patterns used to detect the extra faults is thus

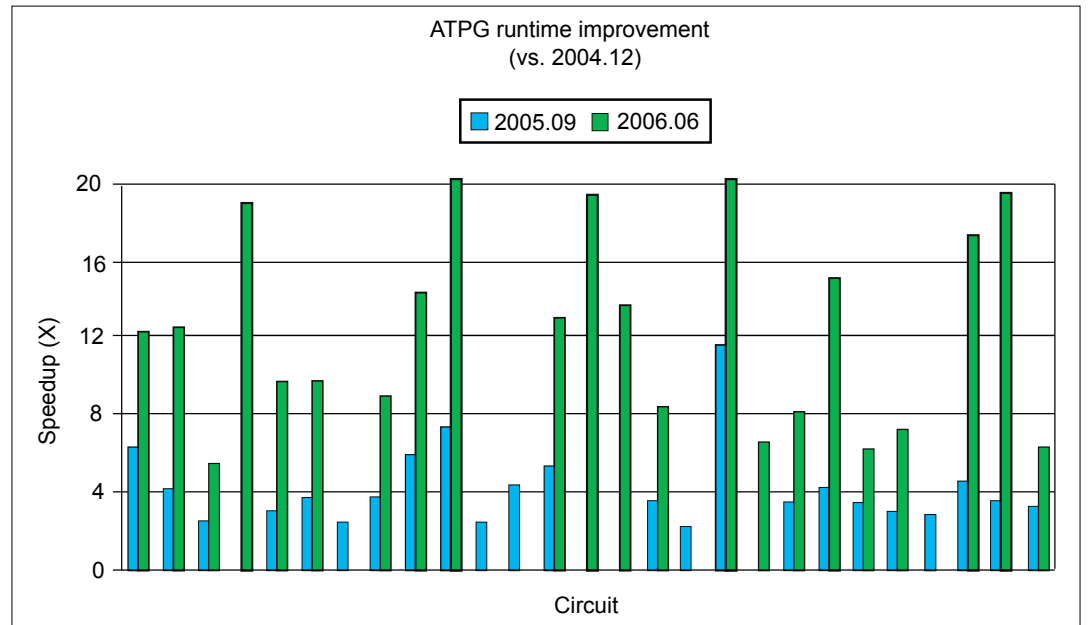


Figure 3: Using IF, the 2004.12 and 2006.06 versions of TetraMAX have achieved, on average, a runtime of more than 12x speed-up for both stuck-at and transition-delay pattern generation.

computed as $(f - F) \times 0.9p/0.1f$ and removed from the pattern accounting. Hence, the improvement factor is given by:

$$IF = \frac{t - \frac{(f - F) \times 0.9t}{0.1f} + (p - P - (f - F) \times (0.9p / 0.1f)) \times \frac{t}{p}}{T}$$

Equilibration results

To observe the calibration of this equation, run the same version of the ATPG tool under two different vector compaction conditions:

1. With compaction turned on, $p = 422$ patterns were generated to achieve $f = 97.11$ percent fault coverage in $t = 147$ CPU seconds.
2. With compaction turned to a lower setting, the same tool produced $P = 444$ patterns to achieve $F = 96.96$ percent

fault coverage in $T = 134$ CPU seconds.

Thus, the execution runtime is $147/134 = 1.1$ times faster

for the second condition. But using the IF equation, the equivalent improvement factor is computed to be 1, which is certainly the case because the ATPG program was identical, and only the parameters had changed.

The fourth IF equation can then be used to compare the performance improvements between two versions of the same ATPG tool: Version 1 is the

older version, and version 2 is the newer version. **Table 1** shows the IF metrics for the data collected.

The ATPG CPU runtime improved for both designs (1.62x and 1.07x for designs A and B, respectively) using the two different tool versions. But, as demonstrated by the IF metrics, this simple runtime comparison underestimates the true performance improvement. When accounting for the changes in both fault coverage and pattern count, the IF metric indicates a larger performance increase: 1.83x for design A and 2.05x for design B.

Fault efficiency, runtime and pattern count are fundamental ATPG performance characteristics that must be properly considered in the evaluation of performance across tools, or across versions of the same tool. The IF formulation correlates fault efficiency and pattern count differences among different executions so that consistent and reliable comparisons can be made to assess ATPG runtime improvements.

Design	Version 1			Version 2			IF
	t	f	P	T	F	P	
A	48166	91.84	7147	29684	93.15	8085	1.83x
B	619591	90.60	1897	581265	94.59	1641	2.05x

Table 1: When accounting for the changes in both fault coverage and pattern count, IF metrics indicate a significantly larger performance increase.