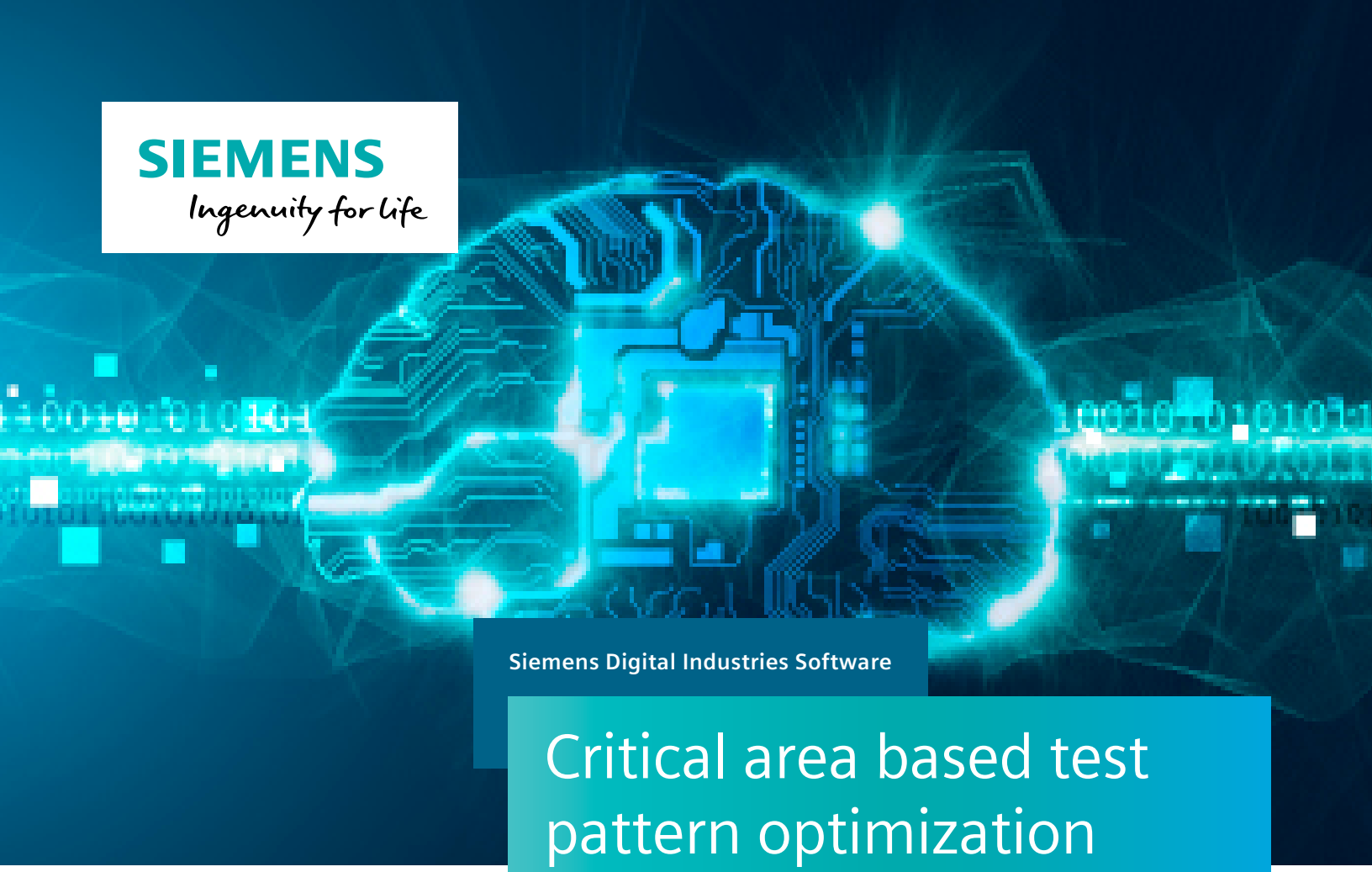


SIEMENS
Ingenuity for life

Siemens Digital Industries Software

Critical area based test pattern optimization

Improve test quality and lower test pattern count

Executive summary

Among the challenges for DFT engineers is how to set a target metric for ATPG and how to choose the best set of patterns. Traditional coverage targets based on the number of faults detected doesn't consider the likelihood of one fault occurring compared to another. Tessent developed total critical area ATPG technology that enables the sorting and ordering of patterns based on their likelihood to detect defects. Users can create a pattern set for multiple fault models in one run, reducing the overall pattern size significantly.

Ron Press and Andreas Glowatz, Siemens EDA

Introduction

Among the persistent challenges for DFT engineers include:

- How to set a target metric for automatic test pattern generation (ATPG)
- How to choose the best set of patterns targeting different faults to get the best results

DFT teams can spend several years establishing ATPG targets—which can be a coverage goal, pattern size, or some other metric—just for stuck-at and transition fault models. Targets need to be adjusted for a new technology node when new important fault models are introduced. How do companies decide what targets to set for them? If you want to apply a sample of patterns aimed at a new fault model, then which are the best patterns to try? Using newer models makes setting fault model target metrics even more complicated. These are realistic and common issues faced today. ATPG tool users need an easy method to determine the best mix of patterns targeting specific faults and which samples of patterns to experiment with.

New methods of scan testing have been introduced to the industry in recent years with unique, production defect detections. Fortunately, there is an approach to measuring pattern value that provides a consistent, "apples to apples" assessment of patterns detecting defects based on the likelihood the physical defects occurring. The approach is to use the total critical area (TCA) associated with faults detected by a pattern.

TCA provides a common metric to assess a pattern's impact on DPM, which can be used to sort or order patterns to achieve the lowest DPM for a given number of patterns. As a result, you can mix in patterns targeting new fault models for a more effective pattern set, even with the same number of patterns as your original pattern set. You can select or sort the most effective patterns from your entire pattern set based on their ability to detect physical defects. Pattern selection and ordering based on TCA provides a faster coverage ramp, based on the likelihood to detect physical defects compared to traditional approaches, as shown in figure 1.

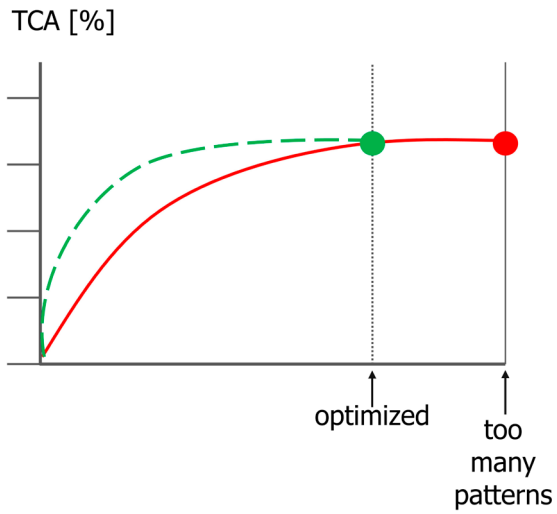


Figure 1: TCA-weighted pattern ordering (green, dashed line) speeds up coverage ramp compared to traditional approaches (red).

ATPG fault models

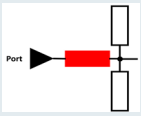
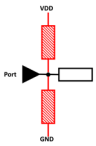
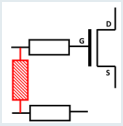
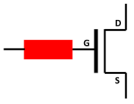
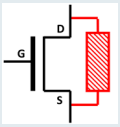
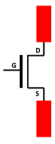
Scan ATPG has advanced over the years with higher quality patterns, embedded compression to greatly reduce that pattern size, and new fault models to achieve low DPM.

High-quality test pattern sets target new fault models and can dramatically improve DPM by providing unique production defect detection (W. Howell, 2018). This section describes some of these newer defect-oriented fault models, including cell-internal/cell-aware, interconnect, and inter-cell bridge. Cell-internal/cell-aware test has wide industry usage and a proven record of improves detection of unique defects.

The other fault models are either growing in usage or are new to the industry. Because of their effectiveness in very-low DPM applications, these fault models are referred to as automotive-grade ATPG.

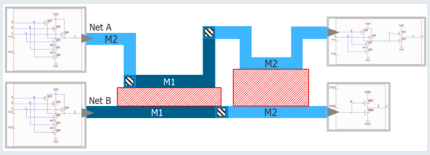
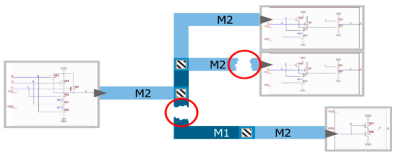
Cell-internal (cell-aware test)

This fault model targets physical defects that can occur within the technology standard cells (Hapke F., 2014). The cell library physical design is characterized to produce a user-defined fault model (UDFM) used by ATPG. The table below shows the defects that are modeled with cell-internal faults. In addition to the detection condition data, the cell-internal defects also contain information about whether a defect behaves like a small delay and if it should be considered for timing-aware pattern generation.

Port open defect	A disconnected port, such as D1, to analyze the effect of cell-external disconnects to cell ports	
Port bridge defect	A bridge between a port and VSS, VDD, or any other port of the cell	
Bridge defect	Any cell-internal bridge defect, such as bridges between adjacent objects in the same layer or different layers. In the case of non-layout-aware runs, bridges are calculated based on parasitic capacitors in the SPICE netlist	
Open defect	Any cell-internal open defect, such as an open in poly, metal, diffusion, or vias. In the extracted SPICE netlist, the tool matches these defects to existing parasitic resistors by increasing their resistance values	
Ton defect	Any cell-internal transistor defect that switches a transistor partially on with a specific resistive value	
Toff defect	Any cell-internal defect that switches a transistor partially off with a specific resistive value	

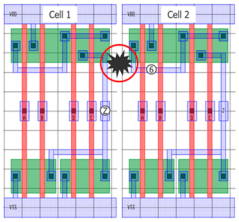
Interconnect

Interconnect fault models are created using the design layout to determine where physical defects can occur.

Interconnect bridge	Any bridge defect between adjacent objects of two nets in the same layer or different layers. The figure shows two possible interconnect bridges between Net A and Net B, highlighted as red shaded rectangles.	
Interconnect open	Any open defect of a net, such as an open in metal1 or metal2. The figure shows two possible open defects between standard cells.	

Inter-cell bridge

This fault model is sometimes referred to as a “cell neighborhood” fault.

Inter-cell bridge	Cell neighborhood defects are chip-dependent bridge defects located at the interface between one instance of a standard cell and a neighboring cell. Such defects can occur between cell-internal nets (layers) not directly accessible from the cell interconnect nets.	

NOTE: The Tessent Shell User’s Manual includes an automotive reference flow and corresponding testcase that describes the steps to generate any of these automotive-grade ATPG patterns. The testcase is included in the Tessent software based on an open-source design.

The challenge of inconsistent ATPG targets

The test coverage targets for static and dynamic patterns targeting stuck-at and transition fault models, respectively, are usually based on the percent of faults detected. These targets vary by company and it often takes years of production fail data to decide on appropriate goals. When an additional fault model is determined to be necessary, the target could be completely different than test coverage based on the full fault list.

For example, consider targeting a test coverage for all potential bridge faults—this could be a huge list. You might achieve 99% detection of all bridge faults but miss hundreds of the most likely bridges. To reduce DPM, it is more effective to choose the subset of bridges that is most likely to occur.

Cell-aware test patterns add significant unique detections beyond stuck-at and transition patterns. The value of cell-aware test is clear but the target is often difficult to decide on. A cell-aware test coverage report is based on cell-internal defects. Even though it has a higher defect detection rate seen in silicon compared to stuck-at and transition, the coverage metric is typically lower than stuck or transition. Thus, there is a problem of how to establish a meaningful target for cell-aware coverage.

The challenge of selecting patterns for production test is complicated by the inconsistent targets for various fault models. Patterns targeting stuck-at and transition faults traditionally target the number of faults detected. Bridge faults could be weighted based on parasitic capacitance, to create a smaller list than simply all potential bridges, but it would not be consistent with targets for other fault models.

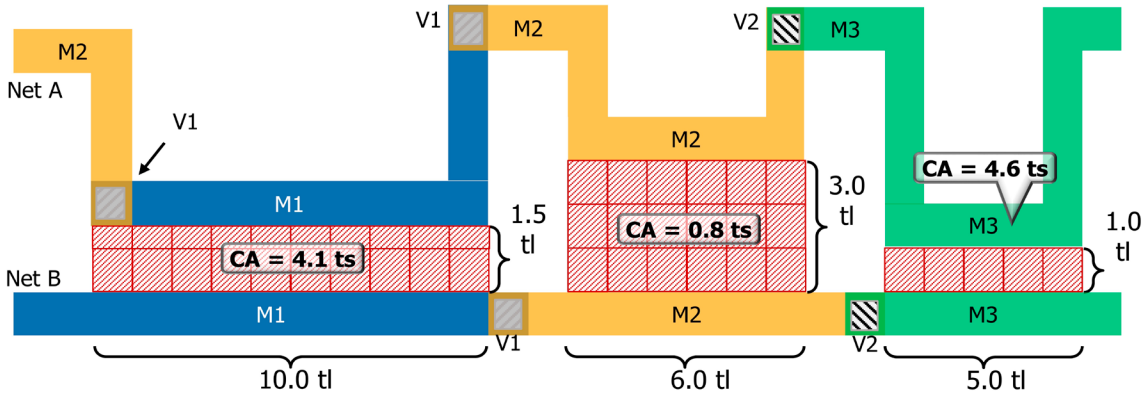
We propose a new common metric for grading a pattern's ability to detect defects for any fault model by using total critical area (TCA).

Using critical area as a common metric

Critical area (CA) refers to the area in a design layout that determines the likelihood that a specific physical defect can cause a failure in the design (Simon Favre, 2019). It is the area in which the center of a spot defect must fall to cause incorrect behavior. Critical area is normalized to the technology length, which refers to the width of metal1 for a particular technology. We use "technology square" (ts) units when calculating critical area.

CA provides a more realistic metric to assess a pattern's ability to detect a physical defect and defines a fault-specific weight. TCA is the sum of all individual critical areas of a short between two connectors, or an open in a connection, weighted by the probability of occurrence of that spot size. Instead of just counting faults, TCA coverage takes the likelihood of defects into account and provides a consistent metric for all fault models.

Figure 2 shows how the critical area is calculated for a bridge defect. The critical area for each segment where the two nets are in proximity is calculated and summed for a TCA. Note that the CA increases as the nets get closer and as the parallel run length increases. TCA can give the user a quick understanding of how a pattern set based on a new fault model might compare to the current pattern sets.



$$\int_{s_{min}}^{s_{max}} \frac{3 * dist_{min}^{2} * (s + length) * (s - dist)}{s^3} ds$$

- s= spot size in technology length [tl]
- s_{min} = minimum spot size in [tl], that can create a bridge
- s_{max} = maximum spot size in [tl] to be considered
- $dist_{min}$ = technology dependent minimum distance of objects in [tl]
- length = length of the bridging area in [tl]
- dist = distance of the bridging objects in [tl]

Figure 2: Total critical area (TCA) calculation for a bridge between two nets.

Including physical data in standard files for ATPG

The TCA values are calculated using physical layout information and stored in the UDFM file. A layer-based TCA is provided for each UDFM defect type so we can calculate a potential 100% TCA for the cell-internal (cell-aware) faults and a potential for 100% TCA detection of faults caused by interconnect bridges. Tessent retrieves the layout information from the Layout Database (LDB) generated and used by the Tessent Diagnosis tool. The LDB is created using the design LEF/DEF and used for bridge, open, and inter-cell TCA calculations.

A separate UDFM file is produced for each defect type (cell-internal, bridge, open, cell-neighborhood), using the flow shown in figure 3. Tessent defect site extraction calculates the TCA values for interconnect bridges, opens, and vias. Tessent CellModelGen calculates the TCA values for all cell-internal and cell-neighborhood defects.

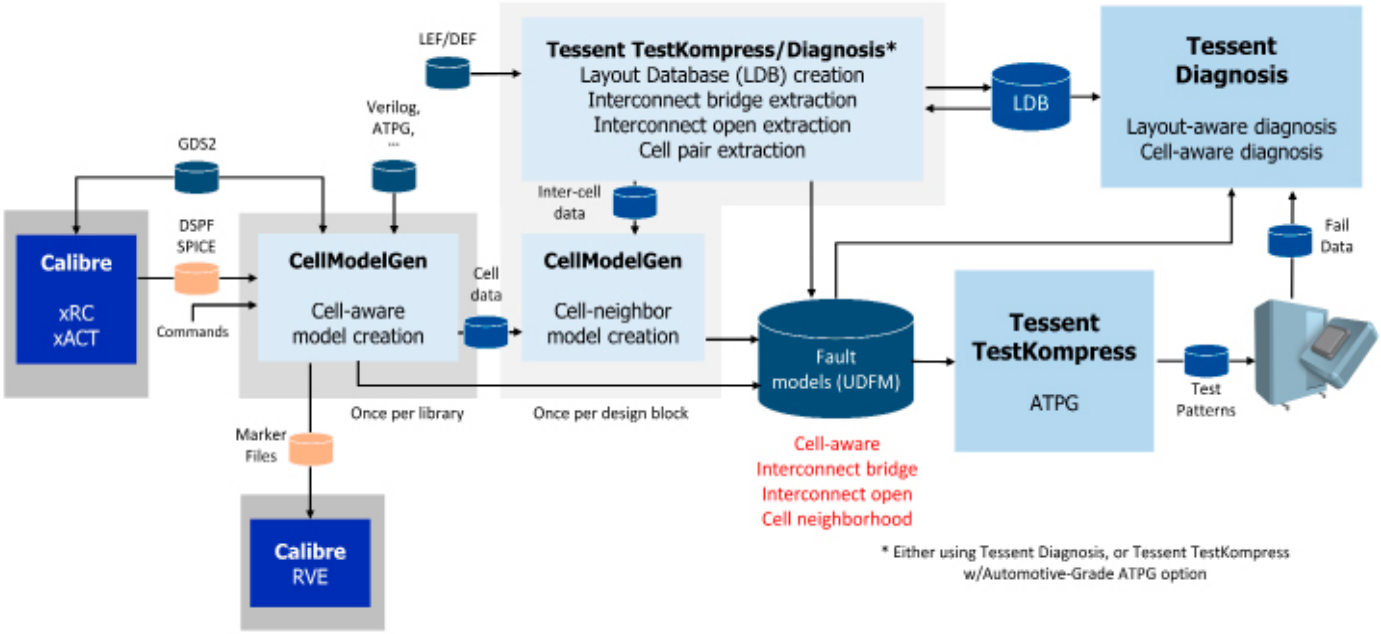


Figure 3: The flow for creating defect-oriented fault models and patterns using physical data for critical area calculations.

Traditional stuck-at and transition faults in the Cell-Aware model

Stuck-at and transition faults have been the baseline fault models for many companies. The stuck-at and transition fault models are represented as “port faults” in the cell-aware fault model. Just like stuck-at and transition fault models, every library cell port will have two faults associated with it. As a result, all stuck-at faults are included as a subset of the cell-aware static fault model. Similarly, all transition faults are included as port faults in the dynamic cell-aware fault model.

There isn’t a critical area associated with port faults. However, any traditional stuck-at or transition fault pattern can be graded for its critical area-weighted impact on DPM using the cell-internal/cell-aware UDFM critical area simulations. Thus, traditional stuck-at and transition faults are a subset of cell-aware tests, and existing stuck-at and transition patterns can be graded for their critical area-weighted coverage.

Pattern types targeting small delay

Timing-aware ATPG uses SDF design information to target and detects small delay defects along long paths during pattern generation. While timing-aware weighting does not correlate directly with TCA weighting, timing-aware patterns are still an important pattern type, especially when combined with cell-aware test, as illustrated in figure 4. This type of pattern set can target any faults based on the most critical paths, and grade faults based on the percent of the longest path tested, on a fault by fault basis. So it has its own built-in pattern sorting. It is useful to try a sample of timing-aware ATPG and determine the mix useful for your production environment.

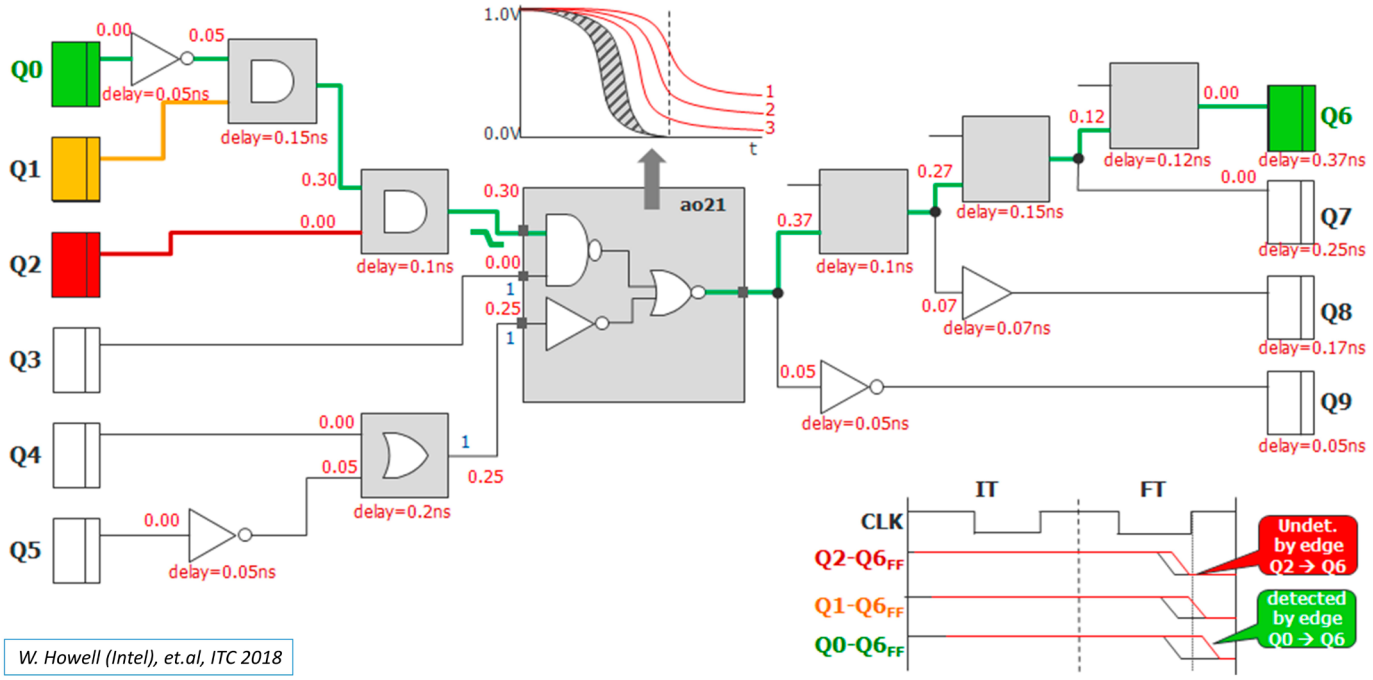


Figure 4: Timing-aware ATPG combined with cell-aware test uses SDF to test for small delays.

Selecting the patterns most likely to detect physical defects

DFT engineers need a way to compare patterns for multiple fault models and select the most effective of each using a consistent and realistic metric. The `order_patterns` command in Tessent can read in multiple patterns created for various fault models and sort them from the highest TCA to lowest. The only restriction is that fault models for static and dynamic patterns must be sorted separately.

Figure 5 shows how to load various fault models to optimize your pattern set. It can be used to simulate and calculate the TCA for an existing pattern set or to create a new pattern set from scratch. Reports can be generated to show the TCA included during ATPG, summary coverage, fault list, and a layer-based TCA summary. Appendix A shows examples of these reports.

Grading and sorting existing patterns based on critical area

Once you create UDFM files with TCA data for the various fault models, you can grade and sort any existing pattern set based on TCA. This is true even for patterns created without any TCA considerations, such as stuck-at patterns. The steps to sort existing patterns based on TCA are:

1. Load the UDFM files for the various fault models that you want to use.
2. Load the existing pattern sets.
3. Simulate the patterns using TCA from UDFMs.
4. Sort the patterns.

A small pattern set might result in a similar sorting based on fault count and based on TCA because a small set can have a big variation in the number of faults detected between different patterns. So, if pattern 1 detects 2000 faults and pattern 2 detects 1000 faults, then the TCA is likely higher for pattern 1 since there are so many more faults. However, if there is a large pattern set then we are more likely to see patterns with similar numbers of faults detected and a bigger difference when sorting by TCA.

TCA sorting is also useful for selecting a sample of additional new patterns to try in production. When considering a new pattern for a new fault model, you can find the likely most effective patterns using TCA sorting. Then you can choose just a small sample of the highest TCA patterns to apply. ATPG commands used to sort existing pattern sets based on TCA are shown in figure 6.

```
read_fault_sites cells.udfm
read_fault_sites bridge.udfm
read_patterns stuck_at.patdb
read_patterns static_bridge.patdb -append
set_critical_area_options -reporting ON
add_faults -all
simulate_patterns -source external -store_patterns all
order_patterns -critical_area
```

Figure 6: ATPG commands to sort based on TCA

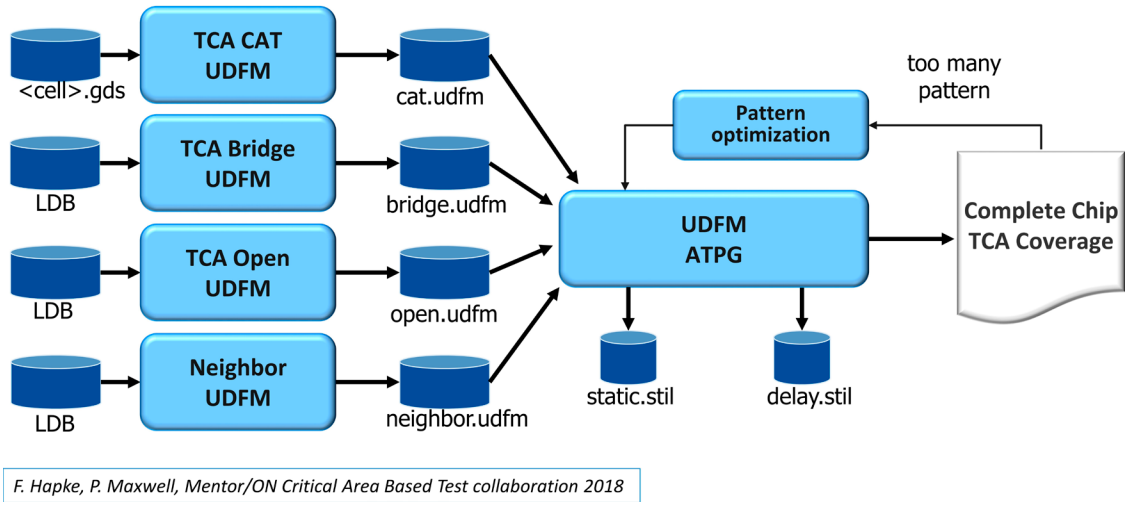


Figure 5: The separate UDFM files can be read in to order and optimize the combined patterns based on TCA.

Creating an optimized pattern set targeting multiple fault types in parallel

To reduce DPM, it is more efficient to run ATPG with multiple fault models than to sequentially target patterns, as is the common practice. Results show 10-40% pattern size reduction when targeting the UDFM TCA fault models together. You just needs to read in the UDFM fault models before ATPG.

If you cannot use the full set of patterns when optimizing an existing pattern set based on TCA or creating a new pattern set, then you need to decide on the relative value of static vs. dynamic TCA. You do not need to work out the value of bridge fault detection compared to stuck-at or other fault models.

Critical area-weighted sorting can order them for you. Similarly, timing-aware ATPG, especially combined with cell-aware test, is shown to have important unique detections and might be a consideration for production test, but timing-aware weighting doesn't correlate directly with TCA weighting.

The Tessent `order_pattern` command can have a significant impact on sorting the patterns by likelihood to detect physical defects. Figure 7 shows ATPG results for three UDFM fault models in one ATPG run. The initial ATPG run is shown in the dark curve. After one `order_pattern` sorting of that pattern we see a much-improved curve in yellow.

You can optimize the curve even more by using the `order_pattern` command to perform multiple consecutive ordering analyses. The green curve shows "`order_pattern 3`". Three passes is usually good enough to get near-optimal results.

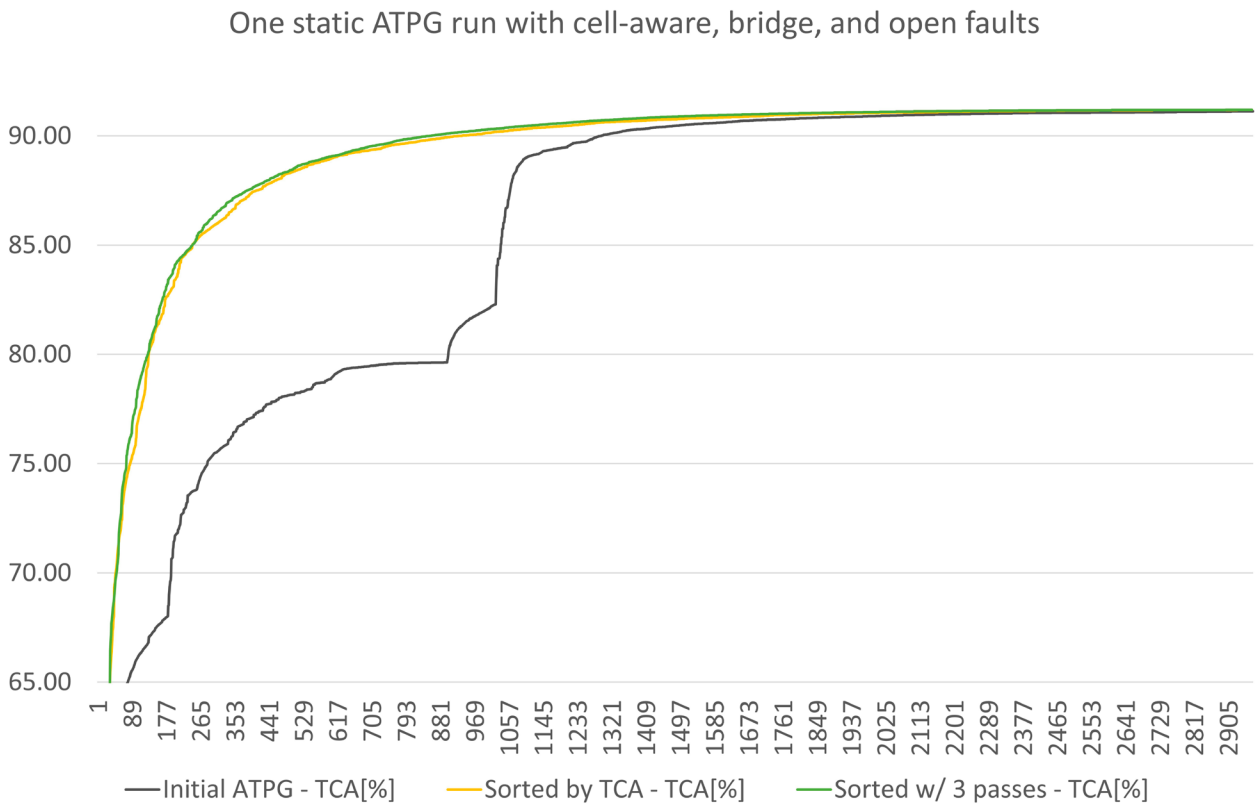


Figure 7: Curve showing the benefits of sorting patterns.

Conclusion

Critical area-weighted pattern selection is a new method of measuring the quality of and optimizing test patterns for various fault models. This TCA coverage is a significantly better measure for the quality of the applied test patterns than counting just the number of faults or defects. TCA-weighted ATPG represents a major step forward in ATPG for high-quality test and provides a consistent metric to select or optimize patterns for the best DPM in the shortest test time.

The suggested process for generating high-quality test and selecting or sorting patterns based on critical area is included in the Tessent Shell User's Manual as a description and corresponding open-source testcase in the Tessent software.

References

1. Gahdhi, R. (2019). Beyond traditional ATPG: A test methodology targeting in-cell defects. available on Arm community site - https://community.arm.com/developer.
2. Hapke, F. (2014). Cell-Aware Test. Trans on CAD, vol 33, no 9, 1396-1409
3. Hapke, F., Howell, W., Maxwell, P., Brazil, E., Venkataraman, S., Rudrajit, D., Rajski, J. (2020). Defect-Oriented Test: Effectiveness in High Volume Manufacturing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
4. Peter Maxwell, F. H. (2017). Bridge Over Troubled Waters: Critical Area Based Pattern Generation. European Test Symposium (ETS). IEEE.
5. Simon Favre. (2019). Getting started with critical area analysis. Siemens White Paper.
6. W. Howell, F. H. (2018). DPPM Reduction Methods and New Defect Oriented Test Methods Applied to Advanced FinFET Technologies. International Test Conference. IEEE.

Appendix A: critical area reporting

There are multiple reports available for viewing total critical area. The main command to activate the reporting is "`set_critical_area_options -reporting ON`". The calculations are layout-based so the TCA for stuck-at patterns or any other fault model can be normalized to a total of 100% for that fault model.

As shown below, if critical area reporting is on then you get a column during ATPG with the normalized TCA% similar to the test coverage % reported in column 2.

```
//  Simulation performed for #gates = 395716  #faults = 223481
//  system mode = analysis    pattern source = internal patterns
//  ------------------------------------------------------------------
//  #patterns  test      #faults  #faults    # eff.    # test    TCA test       process      RE/AU/AAB/EAB
//  simulated  coverage  in list  detected   patterns  patterns  coverage       CPU time
//  deterministic ATPG invoked with comb/seq abort limit = 200/60
//   ---       ------     ---      ---       ---       ---       ------      34.01 sec     53/8783/142/0
//   64        73.65%    55984    167420       64        64      74.16%     187.17 sec
//   ---       ------     ---      ---       ---       ---       ------     214.25 sec     228/8851/291/0
//   128       82.36%    35174     20548       64       128      83.50%     235.34 sec
//   ---       ------     ---      ---       ---       ---       ------     275.16 sec     708/9107/521/0
//   192       86.37%    25088      9292       64       192      87.18%     281.68 sec
//   ---       ------     ---      ---       ---       ---       ------     321.95 sec     1484/9482/825/0
//   256       88.57%    19167      4652       64       256      89.09%     326.51 sec
//   ---       ------     ---      ---       ---       ---       ------     356.21 sec     2226/9884/1081/0
//   320       89.97%    15152      2782       64       320      90.31%     361.23 sec
//   ---       ------     ---      ---       ---       ---       ------     395.24 sec     3066/10234/1300/0
//   384       90.95%    12278      1588       64       384      91.03%     398.27 sec
//   ---       ------     ---      ---       ---       ---       ------     428.03 sec     3981/10588/1563/0
//   448       91.79%     9642      1246       64       448      91.78%     432.07 sec
//   ---       ------     ---      ---       ---       ---       ------     464.63 sec     5031/11034/1879/0
//   512       92.57%     7034       969       64       512      92.42%     468.52 sec
//   ---       ------     ---      ---       ---       ---       ------     512.11 sec     6176/11492/2197/0
//   576       93.29%     4602       694       64       576      92.88%     515.24 sec
//   ---       ------     ---      ---       ---       ---       ------     522.09 sec     6221/11504/2213/0
//   640       93.46%     4141       402       63       639      93.02%     525.31 sec
//   ---       ------     ---      ---       ---       ---       ------     527.31 sec     6221/11504/2213/0
//   704       93.54%     3953       188       60       699      93.11%     529.74 sec
//   ---       ------     ---      ---       ---       ---       ------     531.58 sec     6221/11504/2213/0
//   768       93.61%     3745       208       60       759      93.17%     534.17 sec
//   ---       ------     ---      ---       ---       ---       ------     536.14 sec     6221/11504/2213/0
//   832       93.67%     3592       153       59       818      93.24%     538.44 sec
//   ---       ------     ---      ---       ---       ---       ------     538.65 sec     6221/11504/2213/0
//   895       93.72%     3422       170       57       875      93.31%     540.97 sec
//  73 faults were identified as detected by implication.
//   ---       ------     ---      ---       ---       ---       ------     552.70 sec     6221/11504/1666/0
//   953       93.83%     3158       222       34       909      93.39%     554.23 sec
//   ---       ------     ---      ---       ---       ---       ------     554.66 sec     6221/11504/1666/0
//   988       93.84%     3134        24        7       916      93.40%     556.55 sec
//   ---       ------     ---      ---       ---       ---       ------     559.59 sec     6221/11504/1664/0
//   992       93.85%     3125         9        3       919      93.40%     559.93 sec
//   ---       ------     ---      ---       ---       ---       ------     562.78 sec     6221/11504/1664/0
//   993       93.85%     3123         2        1       920      93.40%     563.04 sec
//   ---       ------     ---      ---       ---       ---       ------     565.28 sec     6221/11504/1664/0
//  EDT with 1-hot scan masking.
//   ---       ------     ---      ---       ---       ---       ------     567.36 sec     6221/11504/1664/0
```

Each fault has an associated TCA value which is included in the fault reporting.

```
  command: report_faults
Class    UdfmType                Fault          TCA
------   --------------------    ------------   ------
DS       interconnect_bridges    B10_A0_dom_B   359.12
DS       interconnect_bridges    B10_A1_dom_B   359.12
AU       interconnect_bridges    B10_B0_dom_A   359.12
AU       interconnect_bridges    B10_B1_dom_A   359.12
DS       interconnect_bridges    B11_A0_dom_B   323.11
AU       interconnect_bridges    B11_A1_dom_B   323.11
DS       interconnect_bridges    B11_B0_dom_A   323.11
AU       interconnect_bridges    B11_B1_dom_A   323.11
AU       interconnect_bridges    B12_A0_dom_B   320.59
RE       interconnect_bridges    B12_A1_dom_B   320.59
DI       interconnect_bridges    B13            318.75
AU       interconnect_bridges    B14_B0_dom_A   316.00
AU       interconnect_bridges    B14_B1_dom_A   316.00
AU       interconnect_bridges    B15_A0_dom_B   313.49
AU       interconnect_bridges    B15_A1_dom_B   313.49
DS       interconnect_bridges    B16_A0_dom_B   307.08
AU       interconnect_bridges    B16_A1_dom_B   307.08
DS       interconnect_bridges    B16_B0_dom_A   307.08
UO,UNS   interconnect_bridges    B16_B1_dom_A   307.08
DS       interconnect_bridges    B17_A0_dom_B   306.87
AU       interconnect_bridges    B17_A1_dom_B   306.87
DS       interconnect_bridges    B17_B0_dom_A   306.87
AU       interconnect_bridges    B17_B1_dom_A   306.87
DI       interconnect_bridges    B18            302.77
DS       interconnect_bridges    B19_A0_dom_B   296.03
AU       interconnect_bridges    B19_A1_dom_B   296.03
DS       interconnect_bridges    B19_B0_dom_A   296.03
DS       interconnect_bridges    B19_B1_dom_A   296.03
AU       interconnect_bridges    B1_B0_dom_A    522.83
AU       interconnect_bridges    B1_B1_dom_A    522.83
```

Another useful report is produced with report_statistics to show the TCA% similar to test coverage and the TCA value similar to the fault counts.

```
                          Statistics Report
                            UDFM Faults
-------------------------------------------------------------------------------
Fault Classes                  #faults          #faults          TCA
                               (total)          (total relevant) (total)
----------------------------  ---------------  ---------------  ---------------
  FU (full)                      241131           240347           1627952
  ------------------------    ---------------  ---------------  ---------------
  UC (uncontrolled)              381 ( 0.16%)     same  ( 0.16%)    1786 ( 0.11%)
  UO (unobserved)               1500 ( 0.62%)     same  ( 0.62%)    7829 ( 0.48%)
  DS (det_simulation)         210569 (87.33%)     same  (87.61%) 1391149 (85.45%)
  DI (det_implication)          7658 ( 3.18%)     same  ( 3.19%)   88641 ( 5.44%)
  PU (posdet_untestable)         729 ( 0.30%)     same  ( 0.30%)    7140 ( 0.44%)
  PT (posdet_testable)          1241 ( 0.51%)     same  ( 0.52%)    8053 ( 0.49%)
  UU (unused)                    925 ( 0.38%)     same  ( 0.38%)   10692 ( 0.66%)
  TI (tied)                      390 ( 0.16%)     same  ( 0.16%)    2688 ( 0.17%)
  BL (blocked)                    12 ( 0.00%)     same  ( 0.00%)      52 ( 0.00%)
  RE (redundant)                6607 ( 2.74%)     same  ( 2.75%)   23859 ( 1.47%)
  AU (atpg_untestable)         11119 ( 4.61%)    10335  ( 4.30%)   86063 ( 5.29%)
-------------------------------------------------------------------------------
Fault Sub-classes
  ------------------------
  AU (atpg_untestable)
    UDN    (undriven)              23 ( 0.01%)     same  ( 0.01%)     159 ( 0.01%)
    BB     (black_boxes)         1132 ( 0.47%)     same  ( 0.47%)   14312 ( 0.88%)
    EDT    (edt_blocks)           784 ( 0.33%)       deleted         4334 ( 0.27%)
    PC*    (pin_constraints)        8 ( 0.00%)     same  ( 0.00%)     104 ( 0.01%)
    TC*    (tied_cells)          3866 ( 1.60%)     same  ( 1.61%)   25289 ( 1.55%)
    SEQ    (sequential_depth)     977 ( 0.41%)     same  ( 0.41%)   12586 ( 0.77%)
    Unclassified                 4329 ( 1.80%)     same  ( 1.80%)   29277 ( 1.80%)
  UC+UO
    AAB    (atpg_abort)          1051 ( 0.44%)     same  ( 0.44%)    5134 ( 0.32%)
    Unclassified                  830 ( 0.34%)     same  ( 0.35%)    4481 ( 0.28%)
  *Use "report_statistics -detailed_analysis" for details.
-------------------------------------------------------------------------------
Coverage
  ------------------------
  test_coverage                  94.00%           94.32%           93.51%
  fault_coverage                 90.91%           91.21%           91.37%
  atpg_effectiveness             98.96%           98.96%           99.16%
-------------------------------------------------------------------------------
#test_patterns                                                        920
  #basic_patterns                                                     174
  #clock_sequential_patterns                                          406
  #multiple_load_patterns                                             340
#simulated_patterns                                                   993
CPU_time (secs)                                                    1120.9
-------------------------------------------------------------------------------
```

The layer-based fault model TCA value and the normalized TCA% for that fault model can be reported with the report_udfm_statistics command. This gives you a view of the relative importance of each fault model related to TCA-weighted likelihood of being impacted by defects.

UDFM Fault Statistics

Type	Block	Name	Instances	Port-Insts	Faults	TCA[ts]	DS[%]	DI[%]	PT[%]	TI[%]	BL[%]	RE[%]	AU[%]	UC[%]	UO[%]	UU[%]	PU[%]	FC[%]	TC[%]	TCA[%]
auto-inserted-faults	Instance	*	1	1	70	0.00	21.43	2.86	0.00	0.00	0.00	0.00	72.86	0.00	0.00	2.86	0.00	24.29	25.00	0.00
inter_cell_bridges	Instance	*	1	0	263158	251824.39	91.59	0.01	0.07	0.09	0.00	5.95	1.81	0.02	0.28	0.13	0.05	91.66	97.69	97.75
interconnect_bridges	Instance	*	1	0	9799895	42306688.00	82.03	0.97	0.87	0.23	0.00	5.19	8.00	0.35	1.12	0.51	0.74	83.80	89.08	85.22
interconnect_opens	Instance	*	1	1	1499930	33329146.00	74.67	22.13	0.05	0.13	0.02	0.64	1.86	0.00	0.18	0.32	0.02	96.83	97.91	96.32
intra_cell_defects	Module	*	197986	8975	12701231	80161624.00	93.17	2.68	0.05	0.08	0.00	1.10	2.31	0.01	0.27	0.31	0.02	95.88	97.34	97.39
undefined-cells	Cell	*	1151	1147	15338	0.00	49.84	31.83	0.11	9.65	0.14	0.16	6.08	0.00	0.03	2.01	0.16	81.80	92.91	0.00
		Total	199141	10124	24279622	156049120.00	87.48	3.18	0.38	0.15	0.00	2.77	4.58	0.15	0.61	0.39	0.31	91.01	94.13	93.86

Siemens Digital Industries Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Our solutions help companies of all sizes create and leverage digital twins that provide organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

siemens.com/eda

© 2019 Siemens. A list of relevant Siemens trademarks can be found here. Other trademarks belong to their respective owners.

82978 V2 5/2020 BM