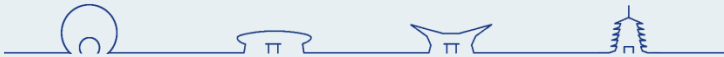
A Fast Test Compaction Method for Commercial DFT Flow Using Dedicated Pure-MaxSAT Solver

Zhiteng Chao[1,3,4], Xindi Zhang[2,3],
Junying Huang[1,3], Jing Ye*[1,3,4], Shaowei Cai[2,3], Huawei Li[1,3,4], Xiaowei Li*[1,3,4]
[1] State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences
[2] State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences
[3] School of Computer Science and Technology, University of Chinese Academy of Sciences
[4] CASTEST Co., Ltd.

Outline

- Backgrounds
  - Test compaction
  - Set covering problem
  - Pure-MaxSAT
- Proposed methods
  - A naive flow
  - Partial fault dictionary
  - Extended candidate pattern set
  - Dedicated Pure-MaxSAT solver
- Experimental results



Test compaction

- The increase in the scale of integrated circuits and test compression cause the expansion of the number of patterns:
  - Test volume increases
  - Test time increases
  - Test cost increases

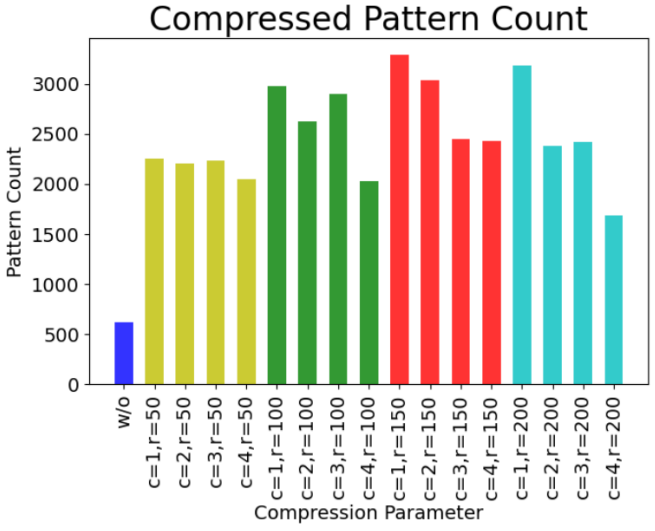


Fig. 1: Pattern count of b18 under different compression parameter settings.

Test compaction

- Test compaction technology is proposed to reduce the number of test patterns or the length of test sequence:
  - Static test compaction: does not change ATPG algorithm and work after ATPG ex ecution
    - Two by one algorithm [DAC’93: Kajihara S]
    - Essential fault reduction [ICCAD’98: Hamzaoglu I]
    - 1-D faults group [TVLSI’23: Eggersglüß S]
  - Dynamic test compaction: change ATPG algorithm and work during ATPG execution
    - Dynamic fault ordering [DAC’93: Kajihara S]
    - Guide line justification and fault propagation for test compaction [ITC’13: Ku mar A]
    - Unspecified bits filling compaction [TC’10: S. N. Neophytou]

Set covering problem

- Formally, we have $U = (X, Y, W)$, where X is the universal set, Y is the set of subsets of X, and W is the weight function for each subset of Y. The goal is to minimize the total weight of the chosen set S, while ensuring that all elements in X are covered by the chosen subsets within S, i.e., $\cup_{\forall y \in S} = X$.
- When all weights equal to 1, set covering problem could be transfor med into static test compaction.

Pattern		p_1	p_2	p_3	p_4
Fault	f_1	1	0	0	1
	f_2	0	0	1	0
	f_3	0	0	1	0
	f_4	0	1	0	1
	f_5	1	1	0	1

Fault dictionary
Original pattern set: {p1, p2, p3, p4}
Reduced pattern set: {p3, p4}

Pure-MaxSAT

- Input: WCNF with hard and soft clauses
  - Each clause C_i is a disjunction of literals, $C_i = l_{i1} \vee l_{i2} \vee \ldots \vee l_{ij}$
  - A literal is defined as a variable x_i or its negation $\neg x_i$
- Output: optimal complete assignment of a boolean variable set $\{x_1, x_2, \ldots, x_n\}$
  - maximize the number of fulfilled soft clauses
  - ensure the satisfaction of all hard clauses
- The Pure-MaxSAT problem is a special problem of Partial MaxSAT where all hard clauses are pure clauses with the same polarity (positive), and all soft clauses are pure clauses with the opposite polarity (negative).

Pure-MaxSAT

A specific example:

Pattern		p_1	p_2	p_3	p_4
Fault	f_1	1	0	0	1
	f_2	0	0	1	0
	f_3	0	0	1	0
	f_4	0	1	0	1
	f_5	1	1	0	1

Choose p_i or not

$\{x_1, x_2, \ldots, x_n\}$

WCNF generation

Soft clauses (patterns)

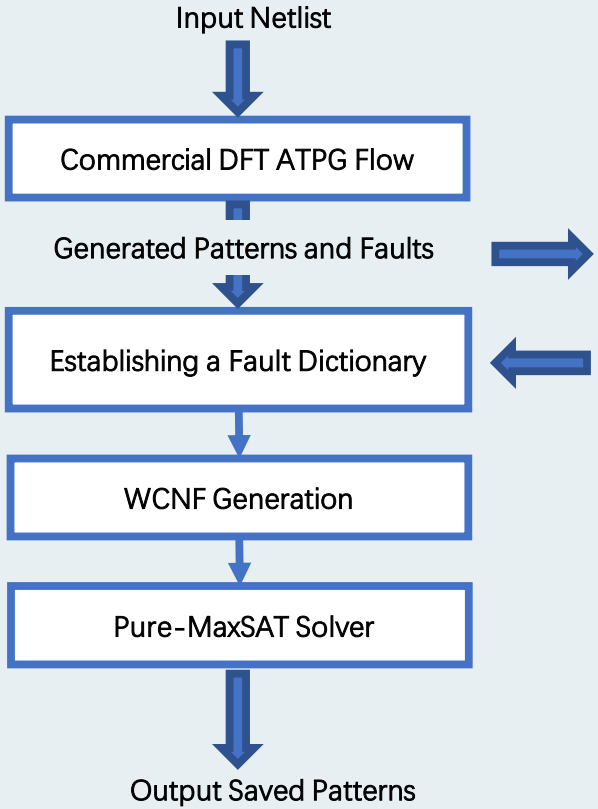
$C_1 = \neg x_1$
$C_2 = \neg x_2$
$C_3 = \neg x_3$
$C_4 = \neg x_4$

Hard clauses (faults)

$C_5 = x_1 \vee x_4$
$C_6 = x_3$
$C_7 = x_3$
$C_8 = x_2 \vee x_4$
$C_9 = x_1 \vee x_2 \vee x_4$

When$\{x_1, x_2, x_3, x_4\} = \{0, 0, 1, 1\}$
Maximal 2 soft clauses are satisfied
All hard clauses are satisfied

A naive flow



Pattern		p_1	p_2	p_3	p_4
Fault	f_1	1	0	0	1
	f_2	0	0	1	0
	f_3	0	0	1	0
	f_4	0	1	0	1
	f_5	1	1	0	1

- Questions:
  - Fault list size is too large (double of the circuit size)
  - Reduction effect is limited by pattern set quality
  - General Pure-MaxSAT Solver does not fit this kind of problem

Partial fault dictionary

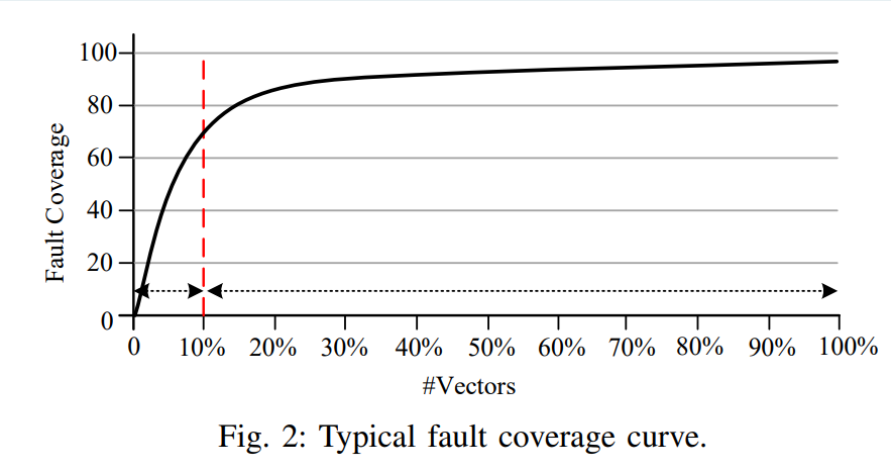
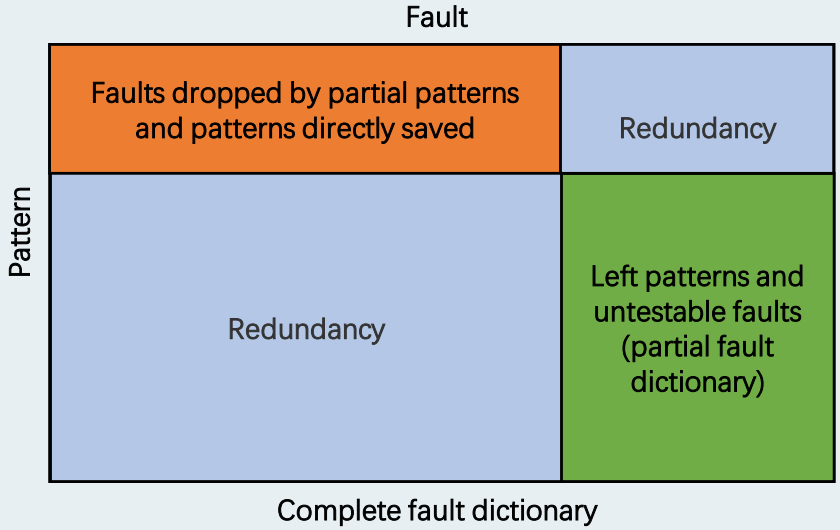


Fig. 2: Typical fault coverage curve.



- Benefits: reduce the scale of problem solving and time+space cost of program

Extended candidate pattern set

- “n-detect” ATPG refers to a fault detection metric that specifies the minimum number of test patterns required to detect all faults at least n times.
- “n-detect” ATPG is used to detect bridge faults widely, which could provide extended candidate pattern set for test compaction, improving final reduction effect.

Pattern		p_1	p_2	p_3
Fault	f_1	1	0	0
	f_2	0	0	1
	f_3	0	0	1
	f_4	0	1	0
	f_5	1	1	0

Fault dictionary of the original pattern set

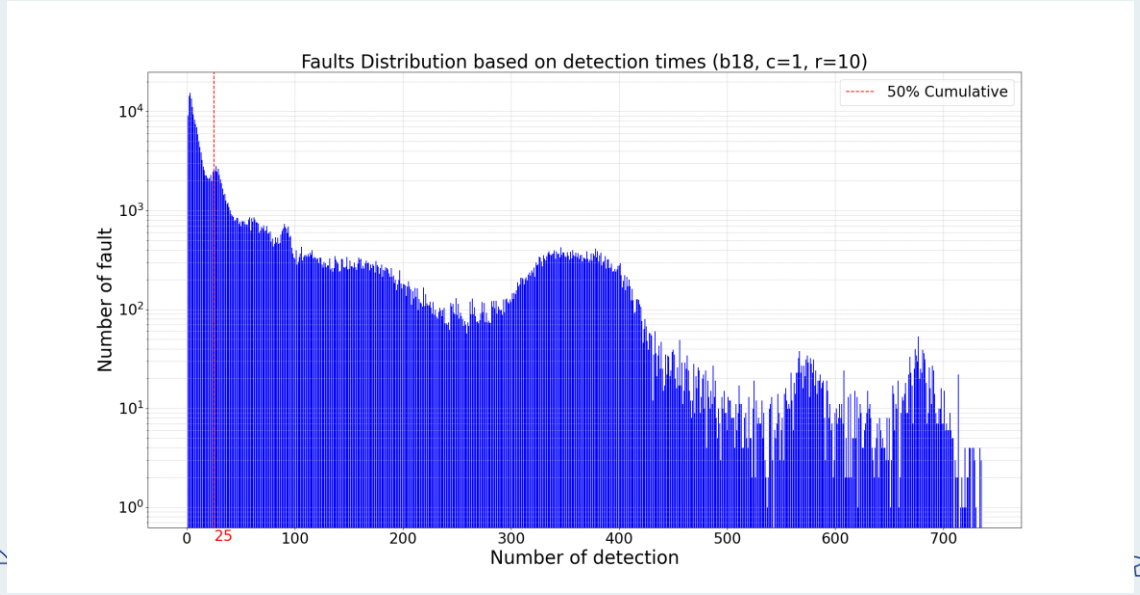
Pattern		p_1	p_2	p_3	p_4
Fault	f_1	1	0	0	1
	f_2	0	0	1	0
	f_3	0	0	1	0
	f_4	0	1	0	1
	f_5	1	1	0	1

Fault dictionary of the extended pattern set

Fig. 4: Explanation of the benefits of an extended pattern set.

Dedicated Pure-MaxSAT solver

- The baseline Pure-MaxSAT solver we use is proposed in [CP'20: S. Cai], which is a kind of the two-stage Linear Local Search (LLS) algorithm.
- A characteristic of the static test compaction benchmark compared to the general SCP benchmark is a huge fault detection number difference between easy-to-detect faults and hard-to-detect faults.



Dedicated Pure-MaxSAT solver

- Scoring function management is important for Pure MaxSAT solver to determine which variable to flip (from 1 to 0/ from 0 to 1) in the search phase.
- LLS will pick the variable with current highest $rscore$, original $rscore$ calculation is described as following:

$$hscore(\alpha, x) = \sum_{c \in H_f(\alpha)} hw(c) - \sum_{c \in H_f(\alpha')} hw(c) \quad (1)$$

$$sscore(\alpha, x) = cost_s(\alpha) - cost_s(\alpha') \quad (2)$$

$$rscore(\alpha, x) = \frac{hscore(\alpha, x)}{|sscore(\alpha, x)| + 1} \quad (3)$$

Dedicated Pure-MaxSAT solver

- We define FIF to measure the importance of a fault corresponding to a hard clause and adjust the scoring function of each variable corresponding to the pattern:

$$FIF(c) = Round(S \times (\frac{ls_{max}(c_m) - ls(c)}{ls_{max}(c_m)})^{\ln \frac{ls_{max}(c_m)}{ls_{avg}(c_a)}}) \quad (4)$$

- In (4), c denotes a hard clause, $ls(c)$ denotes the number of literals in hard clause c, $ls_{max}(c_m)$ denotes the maximum value, and $ls_{avg}(c_a)$ denotes the average value. S is a constant to scale and set to 2, $Round$ function is used to round a real number to an integer.
- We modify the $hscore$ to $hscore*$ as follows:

$$hscore^*(\alpha, x) = \sum_{c \in H_f(\alpha)} hw(c) \times FIF(c) - \sum_{c \in H_f(\alpha')} hw(c) \times FIF(c) \quad (5)$$

Experimental results

- Experimentation Setup:
  - Our dedicated Pure-MaxSAT solver is implemented in C++, which run on Dell servers equipped with Intel Xeon CPUs.
  - The performance and efficacy of our approach are evaluated using standard benchmarks from the ISCAS89, ITC99, and two open source RISC-V CPUs, openE906 (2.115×10^5primitives) and openC906 (4.816×10^5 primitives), synthesized using Synopsys Design Compiler.
- Evaluation criteria:
  - We initiate the test compaction process by generating a fault list and original test pattern set through a leading commercial DFT tool that uses EDT technology. We apply compression parameters that are selected at random and evaluate the effect of the compaction in ATE test cycles derived the same DFT tool.

Experimental results

- Effect of partial fault dictionary and extend pattern set:

TABLE I: Efficiency of Partial Fault Dictionary in "1-detect" mode ATPG algorithm

Benchmark	Compression Parameters	Complete Fault list Size	Reduction Fault list Size	Fault Dictionary Reduction Rate
s35932	c=2, r=10	41572	6287	86.39%
s35932	c=3, r=10	42472	6137	87.00%
s38417	c=1, r=50	43606	4838	90.01%
s38417	c=1, r=100	47258	5105	90.28%
s38417	c=1, r=200	54884	4870	92.01%
b17	c=1, r=50	127606	30479	78.50%
b17	c=2, r=100	139112	30111	80.52%
b17	c=2, r=200	153992	32591	80.95%
b18	c=3, r=100	349006	61205	84.21%
b18	c=3, r=200	371492	58427	85.85%
b18	c=4, r=150	371990	56713	86.28%
openE906	c=4, r=100	903334	120801	87.96%
openE906	c=4, r=150	916964	119722	88.25%
openE906	c=4, r=200	930474	122883	88.11%
openC906	c=4, r=100	1974388	198751	90.94%
openC906	c=4, r=150	1988046	210214	90.48%
openC906	c=4, r=200	2001496	224111	89.92%

TABLE II: Compaction Effect using Dedicated Pure-MaxSAT Solver in Different ATPG Modes

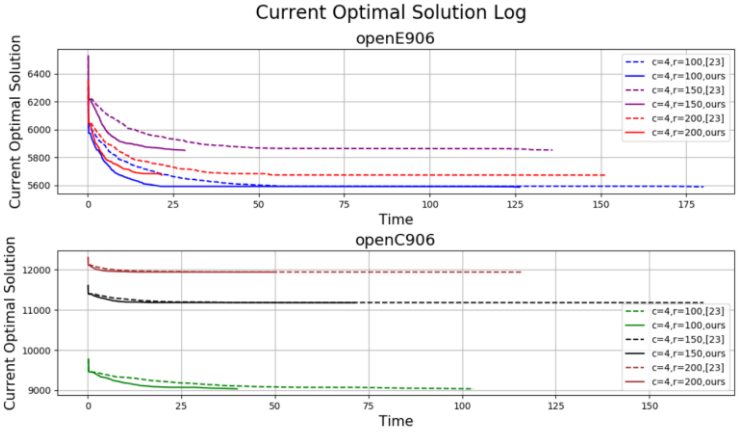
Benchmark	Compression Parameters	1D OP_n	1D CP_n	1D OC_n	1D CC_n	1D RR_c	2D CP_n	2D CC_n	2D RR_c	3D CP_n	3D CC_n	3D RR_c
s35932	c=2, r=10	47	41	6360	5748	9.62%	35	5136	19.25%	32	4830	24.06%
s35932	c=3, r=10	49	40	4644	3996	13.95%	34	3564	23.26%	32	3420	26.36%
s38417	c=1, r=50	611	524	39330	34197	13.05%	457	30244	23.10%	430	28651	27.15%
s38417	c=1, r=100	734	610	39610	33782	14.71%	536	30304	23.49%	524	29740	24.92%
s38417	c=1, r=200	747	597	52347	44397	15.19%	543	41535	20.65%	526	40634	22.38%
b17	c=1, r=50	1350	1135	75906	64296	15.30%	1078	61218	19.35%	1038	59058	22.20%
b17	c=2, r=100	1289	1082	52798	45139	14.51%	964	40773	22.78%	949	40218	23.83%
b17	c=2, r=200	1016	894	57636	52146	9.53%	855	50391	12.57%	811	48411	16.01%
b18	c=3, r=100	2896	2515	108936	95220	12.59%	2302	87552	19.63%	2278	86688	20.42%
b18	c=3, r=200	2423	2155	127923	115327	9.85%	1925	104517	18.30%	1865	101697	20.50%
b18	c=4, r=150	2425	2180	106676	96876	9.19%	1901	85716	19.65%	1845	83476	21.75%
openE906	c=4, r=100	9256	7588	547563	446190	18.51%	6898	406108	25.83%	6511	383662	29.93%
openE906	c=4, r=150	9700	7924	546932	444607	18.71%	7197	404622	26.02%	6820	383942	29.80%
openE906	c=4, r=200	9353	7649	551156	449336	18.47%	6955	409690	25.67%	6609	389854	29.27%
openC906	c=4, r=100	11178	10882	1174526	1142959	2.69%	10476	1100358	6.31%	10157	1067182	9.14%
openC906	c=4, r=150	13975	13380	1216220	1163894	4.30%	12973	1128892	7.18%	12579	1095008	10.00%
openC906	c=4, r=200	14765	14130	1198632	1146634	4.34%	13808	1120874	6.49%	13419	1089754	9.08%

Experimental results

- Effect of dedicated Pure-MaxSAT solver:

TABLE III: Performance Comparison for Relatively Large Benchmarks in "3-detect" Mode ATPG algorithm

Benchmark	Compression Parameters	OP_n	[12]		[13]		[20]		[18]		ours	
			CP_n	time(s)	CP_n	time(s)	CP_n	time(s)	CP_n	time(s)	CP_n	time(s)
openE906	c=4, r=100	22732	6529	434.18	6519	346.63	6533	198.47	6514	180.06	**6511**	**126.22**
openE906	c=4, r=150	23757	6835	489.38	6822	428.22	6846	237.05	6821	135.92	**6820**	**28.37**
openE906	c=4, r=200	22686	6616	449.03	6616	332.32	6615	44.41	**6607**	151.86	6609	**21.68**
openC906	c=4, r=100	22357	10162	384.22	10162	397.00	10171	112.90	10157	103.04	**10157**	**39.87**
openC906	c=4, r=150	23288	12588	334.15	12592	467.75	12593	77.38	12579	164.45	**12579**	**71.28**
openC906	c=4, r=200	23623	13422	442.57	13423	644.08	13430	**45.86**	13419	115.97	**13419**	49.85



Conclusion

We present a dedicated Pure-MaxSAT solver tailored for static test compaction, incorporating optimization techniques such as

- partial fault dictionary
- extended pattern set
- variable flipping score improvement

enabling pattern reduction within a finite time frame, thereby serving as a valuable complement to commercial DFT workflows.

THANKS

Q & A