Generování testu pro prostředky vestavěné diagnostiky

Robert Hülle
1. ročník prezenčního studia
Školitel: Petr Fišer, školitel specialista: Jan Schmidt

České vysoké učení technické v Praze, Fakulta informačních technologií
Thákurova 9, Praha, 16000
hullerob@fit.cvut.cz

***Abstrakt*—V tomto článku shrnuji své výsledky za 1. rok doktorského studia, porovnání poruchových modelů pro aplikačně specifické testování FPGA, zkoumání vlastností SAT instancí vzniklých v procesu ATPG. Dále nastiňuji další možný směr pokračování výzkumu, generování testu s nulovým aliasingem.**

***Klíčová slova*—test, testování, generování testu, poruchový model, FPGA, ATPG, SAT, kompakce odezvy, aliasing**

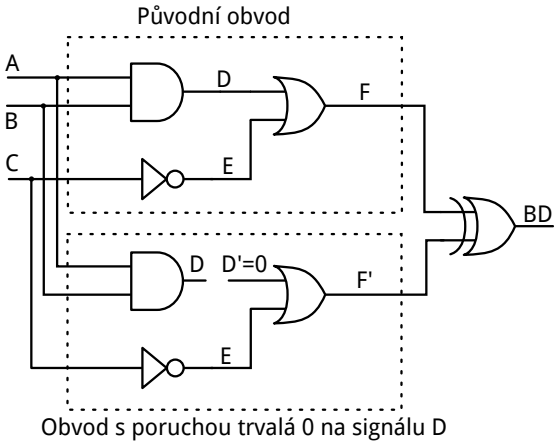
I. Úvod

S rostoucí hustotou integrace a složitosti číslicových obvodů roste také náročnost testování, ať už se jedná o generování testu, dobu aplikace testu či o plochu a spotřebu testovací logiky. Jedním ze způsobů, jak se vypořádat s rostoucí délkou testů a ceny ATE je využití prostředků vestavěné diagnostiky (built-in self-test, BIST).

Nedílnou součástí prostředků vestavěné diagnostiky je kompakce odezvy testovaného obvodu. Kompaktor odezvy je typicky tvořen sekvenčním obvodem, jehož vnitřní stav je ovlivňován výstupem testovaného obvodu. Příkladem takového obvodu je lineární zpětnovazební posuvný registr (LFSR) s paralelními vstupy (MISR).

Mimo klasického problému pokrytí poruch testovacím vektorem zde navíc vzniká problém aliasingu. Aliasing je jev, kdy porucha, která je pokrytá více testovacími vektory, vyvolá chybovou odezvu, která uvede kompaktor do stavu odpovídajícímu bezporuchovému obvodu, čímž klesá reálné pokrytí obvodu. Proti tomuto jevu lze bojovat různými prostředky, například zvětšením periody kompaktoru, či změnou typu kompaktoru [1], [2].

Cílem mé disertační práce je potlačit aliasing již ve fázi generování testovacích vektorů. Základní myšlenka je konstrukce dodatečných omezení výstupu obvodu. Výstup obvodu musí být omezen tak, aby nový testovací vektor generovaný pro nepokrytou poruchu nezpůsobil aliasing poruchy pokryté dřívějším testovacím vektorem. Zde se přímo nabízí generátor testu (ATPG) založený na řešení problému splnitelnosti booleovské formule (SAT), neboť ten umožňuje snadno vytvářet další omezení výstupu, na rozdíl od moderních strukturních algoritmů založených na původním D-algoritmu [3], mezi něž patří PODEM [4], FAN [5], SOCRATES [6] a další.



Obrázek 1. Obvod s modelovanou poruchou trvalá 0. Porucha je detekována, pokud výstup obvodu je 1.

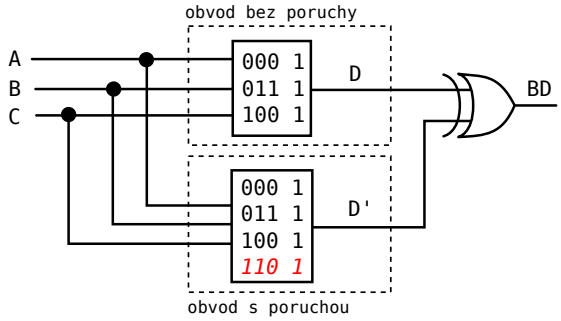
A. SAT ATPG

SAT ATPG překládá problém nalezení testovacího vektoru na řešení problému SAT. Tato ATPG využívají pokroků na poli řešičů SAT, které umožňují rychlé řešení, zejména SAT instancí vznikajících z logických obvodů. Ukazuje se, že SAT ATPG dokáží nalézt testovací vektor, nebo důkaz redundance poruchy rychleji než strukturní ATPG [7], [8].

Základní princip SAT ATPG spočívá v transformaci obvodu tak, aby jeho výstup byl jednotkový právě tehdy, když je na jeho vstup přiveden vektor, který detekuje testovanou poruchu. Toho je docíleno zdvojením obvodu a modelováním poruchy v jedné jeho polovině. Vstupy těchto dvou obvodů jsou společné, výstupy jsou spojeny hradly XOR [9].

Modelování poruchy je závislé na zvoleném poruchovém modelu. Poruchy typu trvalá 0/1 (SA) modelujeme rozpojením poruchového signálu a nastavením tohoto signálu na konstantní hodnotu, pro poruchu trvalá 0 na hodnotu 0, pro poruchu trvalá 1 na hodnotu 1. Příklad je uveden v obrázku 1.

Takovýto obvod je poté přeložen na instanci SAT v konjunktivní normální formě (CNF) jako konjunkce charakteristických funkcí všech hradel v obvodu.



Obrázek 2. Obvod s modelovanou poruchou bit-flip v LUTu.

Během přípravy SAT ATPG pro využití ve svém dalším výzkumu jsem změřil jeho vlastnosti a vlastnosti SAT instancí, které vyprodukoval.

II. SAT ATPG pro aplikačně specifické testování FPGA

Dostupná ATPG typicky podporují omezený poruchový model, nejčastěji model trvalé 0/1. Tento model je ovšem pro testování obvodu v FPGA nedostatečný [10], [11]. Jiné typy poruch se v takovýchto ATPG testují například transformací obvodu tak, aby tyto poruchy byly ekvivalentní (nějaké) poruše trvalá 0/1 v transformovaném obvodu.

Já jsem zvolil jinou cestu a implementoval jsem nový SAT-ATPG, který je schopen přímo pracovat s více poruchovými modely. Při výběru poruchových modelů jsem se zaměřil na aplikačně specifické testování FPGA, tedy testování logiky nahrané v FPGA čipu, bez nutnosti rekonfigurace. Pro tento scénář jsem zvolil modely „trvalá 0/1“ (stuck-at, SA) a „překlopení bitu“ (bit-flip, BF).

Porucha typu BF představuje změnu obsahu paměti vyhledávací tabulky (lookup table, LUT), kdy 1 bit v paměti LUTu je změněn. Tato porucha se projeví jako nesprávná hodnota na výstupu LUTu pro jeden jeho vstupní vektor.

Modelování BF poruchy při překladu obvodu do CNF provádím stejně jako v případě SA, mám tedy zdvojený obvod, s poruchovou a bezporuchovou částí. Na rozdíl od SA není v poruchové části žádná diskontinuita, pouze se v ní liší funkce uzlu (hradla či LUTu) reprezentujícího poruchový LUT. Stejně jako v případě SA je poté SAT řešičem nalezeno ohodnocení signálů, pro které se výstupy obvodů liší. Příklad je vyobrazen v obrázku 2.

Porovnal jsem vzájemný vztah poruch obou modelů, SA a BF, zejména jejich vzájemnou dominanci. Porucha SA je dominována poruchou BF vždy, pokud se nachází na vstupu či výstupu LUTu, nemusí však být dominována v jiných částech obvodu, například na vstupech či výstupech klasických hradel, například v obvodech pro rychlou propagaci přenosu (fast carry chain), které jsou součástí dnešních FPGA obvodů. Dominance v opačném směru, tedy poruchy BF dominované poruchou SA se v obvodu nevyskytují ve významném počtu, jediný případ, kdy k dominanci může dojít, je LUT implementující funkci s jedním mintermem [12].

Experimenty jsem provedl na 279 obvodech z benchmarků MCNC, LGSynth’91 [13], LGSynth’93, ISCAS’85 [14], ISCAS’89 [15] a IWLS 2005 [16]. Ze sekvenčních obvodů byly extrahovány kombinační části, poté byly obvody syntetizovány nástrojem Xilinx Vivado 2015.2 pro architekturu Artix-7. Tyto syntetizované obvody jsem převzal.

Změřil jsem podíl poruch BF pokrytých poruchami SA a podíl poruch SA pokrytých poruchami BF. Z neredundantních poruch bylo průměrně pokryto 99,6 % poruch SA a 69,5 % poruch BF. Ve většině obvodů bylo pokrytých 100 % poruch SA, s výjimkou obvodů, které obsahovaly primitivní hradla XOR a multiplexory, v některých obvodech byl také přítomný přímý spoj mezi vstupem a výstupem. Pokrytí poruch BF poruchami SA má mnohem větší rozptyl, pohybuje se od 17 % do 100 %

Překvapivý výsledek jsem získal z měření počtu redundantních poruch, kdy průměrný poměr redundantních poruch SA je dle očekávání nízký, 0,33 %. Naopak pro poruchy BF je poměr redundantních poruch průměrně 10,15 %. Celkový poměr všech redundantních poruch je pak 7,42 %.

Množství redundantních poruch v modelu BF je zapříčiněno horší kontrolovatelností a pozorovatelností, než je tomu v případu modelu SA.

Tyto výsledky jsem ve formě článku „SAT-ATPG for Application-Oriented FPGA testing“ poslal na konferenci Baltic Electronics Conference. Článek byl přijat k publikaci [12].

III. Vlastnosti SAT instancí

A. Výpočet charakteristické funkce

Ve výše zmíněném ATPG jsem implementoval dva různé způsoby generování CNF obvodu. Liší se generováním charakteristické funkce uzlu obvodu. Dále jsem měřil vliv minimalizace reprezentace uzlů v SOP na vlastnosti SAT instancí a její celkový vliv na generování testu.

1) Přímé generování charakteristické funkce: Základní myšlenka této metody je v tom, že obvodový uzel popíšeme jako logickou ekvivalenci výstupu a funkce vstupů. Tu poté algebraickými operacemi transformujeme do CNF.

V praxi stačí nalézt reprezentaci funkce uzlu a její komplement (on-set a off-set) ve formě SOP. Na tyto dvě reprezentace lze nazírat tak, že implikují výstup uzlu v hodnotě 1 (pro on-set) případně v hodnotě 0 (pro off-set). Tyto implikace odpovídají implikacím obsaženým ve výše zmíněné ekvivalenci výstupu a funkce vstupů. Použitím DeMorganových pravidel transformujeme implikaci na součin součtů (product of sums, POS), který je v součtu s výstupní proměnnou. Jednoduchou distribucí výstupní proměnné získáme polovinu charakteristické funkce (pro on-set a off-set) v CNF. Příklad takové transformace je uveden na Obrázku 3.

Výstupem tohoto generátoru je CNF s dlouhými klauzulemi, což může být nevýhoda.

2) Generování charakteristické funkce Tseitinovou transformací: Druhá metoda spočívá v Tseitinově transformaci [17] každého uzlu popsaného dvouúrovňovou logikou ve formátu PLA na dvě úrovně uzlů, v první úrovni jsou uzly AND, v druhé úrovni je uzel OR. Tato struktura vychází přímo z

$F(a,b,c) = \{0,2,3,6,7\}$

```
on-set
a b c    y
0 - 0    1   (a ∨ c ∨ y)
- 1 -    1   (¬b ∨ y)

off-set
a b c    y
1 0 -    0   (¬a ∨ b ∨ ¬y)
- 0 1    0   (b ∨ ¬c ∨ ¬y)
```

CNF

$$(a \vee c \vee y) \wedge (\neg b \vee y) \wedge (\neg a \vee b \vee \neg y) \wedge (b \vee \neg c \vee \neg y)$$

Obrázek 3. Příklad funkce zadané výčtem mintermů, její minimalizovaný on-set a off-set ve formě SOP a výsledná charakteristická funkce v CNF.

```
on-set
a b c    y
0 - 0    1
- 1 -    1
Tseitinova transformace
```

$y_1 = (\neg a \wedge \neg c)$
$y_2 = b$
$y = y_1 \vee y_2$

CNF

$$(a \vee c \vee y_1) \wedge (\neg a \vee \neg y_1) \wedge (\neg c \vee \neg y_1) \wedge$$
$$(b \vee \neg y_2) \wedge (\neg b \vee y_2) \wedge$$
$$(\neg y_1 \vee y) \wedge (\neg y_2 \vee y) \wedge (y_1 \vee y_2 \vee \neg y)$$

Obrázek 4. Příklad výpočtu charakteristické funkce v CNF pomocí Tseitinovy transformace.

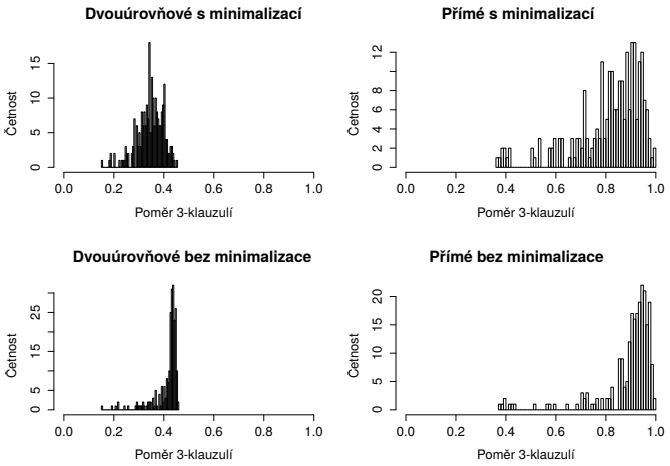
popisu uzlu jako součtu součinů. Charakteristické funkce tímto způsobem vzniklých virtuálních uzlů mohou být spočítány stejným způsobem výše uvedeným, ale protože se jedná pouze o dva druhy uzlů, můžeme si ušetřit práci použitím přepisovacích pravidel pro hradla AND a OR. Také není třeba počítat komplement (off-set) transformované funkce. Příklad takové transformace je uveden na obrázku 4.

Nevýhodou této verze je zvětšení počtu proměnných ve výsledné SAT instanci. Naopak výhodou je větší poměr dvouliterálových klauzulí.

B. Vlastnosti SAT instancí

3SAT je varianta problému splnitelnosti, kde každá klauzule má právě 3 literály. 3SAT patří mezi NP-úplné problémy a neposkytuje žádnou výpočetní výhodu proti obecnějšímu problému SAT. Vlastnosti 3SAT jsou však lépe pochopeny, konkrétně vztah mezi poměrem počtu klauzulí k počtu proměnných a mezi výpočetní složitostí náhodných 3SAT instancí je znám. Existuje fázový přechod kolem poměru 4,3 kde jsou instance nejtěžší [18], [19].

Podobně 2SAT je problém splnitelnosti, kde každá klauzule má právě 2 literály. Na rozdíl od 3SAT (a SAT) nepatří



Obrázek 5. Průměrná hodnota 2+p SAT u zpracovaných obvodů.

2SAT mezi NP-úplné problémy a jeho výpočetní složitost je polynomiální.

2+p SAT je varianta SAT, kde klauzule mají 2 nebo 3 literály. Pro náhodné instance bylo ukázáno, že pro $p > 0,4$ tedy pro poměr 3-literálových klauzulí, se vlastnosti náhodných instancí více podobají instancím 3SAT a pro $p < 0,4$ se podobají více instancím 2SAT [20].

Pro účely diskuse nad vlastnostmi instancí jsem instance vzniklé ATPG procesem transformoval na instance 2+p SAT tak, že na klauzule delší než 3 literály jsem použil klasickou redukci ze SAT na 3SAT. Tato transformace není třeba pro řešení instance, SAT řešič řeší netransformované instance.

C. Výsledky

V experimentech jsem použil 230 obvodů ze stejného benchmarku, zmíněného v předchozí sekci.

Pro každý obvod jsem vygeneroval SAT instance popisující jednotlivé poruchy, protože se vždy jedná o stejný obvod, lišící se pouze v místě poruchy, je rozptyl vlastností instancí malý, lze ho reprezentovat jednou hodnotou, průměrem. Na obrázku 5 jsem vyobrazil histogram poměru 2+p přes všechny obvody, pro obě metody generování charakteristické funkce, *s* i *bez* minimalizace reprezentace logických uzlů.

Je zřejmé, že použití metody využívající Tseitinovy transformace vede na instance s větším zastoupením 2-literálových klauzulí, kdežto metoda přímého generování vede na instance, ve kterých výrazně převažují 3-literálové klauzule. Minimalizace booleovských funkcí uzlů v obou případech mírně posunuje histogram k více 2-literálovým klauzulím. Změřený poměr naznačuje, že instance generované přímo by se měly chovat jako 3SAT, kdežto instance generované Tseitinovou transformací leží kolem fázového přechodu $p = 0,4$, chování takových instancí by tedy mělo být mezi 2SAT a 3SAT.

Stejným způsobem jsem zkoumal poměr počtu klauzulí k počtu proměnných. Při použití Tseitinovy transformace vyšel průměrný poměr 2,19 *s* a 2,16 *bez* minimalizace. Pro přímé generování jsem naměřil hodnoty 1,58 *s* a 1,43 *bez* minima-

lizace. V tomto případě neměla minimalizace velký vliv na poměr počtu klauzulí. Takovýto poměr klauzulí k proměnným v náhodném 3SAT vede na instance, které jsou téměř všechny splnitelné [18], poměr nesplnitelných instancí v měřených obvodech je však mnohem větší, což mě vede k domnění, že metriky vytvořené při zkoumání náhodných SAT instancí nemusí být nutně vhodné pro analýzu instancí vytvořených v ATPG procesu.

Změřil a porovnal jsem dobu generování a řešení SAT instancí. Jako řešič SAT instancí jsem použil Minisat, pro minimalizaci logických uzlů jsem použil Espresso.

Z dvou metod generování klauzulí nevychází ani jedna statisticky významně lepší, než druhá. Použití minimalizace ovšem vede na instance, které řešič vyřeší rychleji až o řád. Minimalizace má větší vliv na přímý generátor, nejrychleji řešené instance jsou tedy přímo generované a minimalizované.

Při srovnání doby celého generování testu, tedy včetně generování instancí, je situace jiná. Oba generátory bez minimalizace mají stejný výkon, s minimalizací roste doba generování SAT instancí víc, než uspoříme za běhu SAT řešiče.

Tyto výsledky jsem ve formě článku „On Properties of ATPG SAT Instances“ poslal na konferenci Euromicro Conference on Digital System Design. Článek získal jednu kladnou recenzi, dvě záporné recenze a nebyl přijat.

IV. Další směřování výzkumu

Pro snížení aliasingu v kompakci odezvy potřebuji poruchovou simulaci pro všechny pokryté poruchy a simulaci samotného kompaktoru odezvy. Ze znalosti stavu kompaktoru pro všechny poruchy a pro bezporuchový obvod spočítám, které odezvy obvodu by vedly na aliasing, a tyto odezvy zakóduji do SAT instance jako zakázané.

Tento přístup ovšem trpí jedním problémem: je potřeba znát stav kompaktoru po vygenerování následujícího testovacího vektoru již během jeho generování. Tento problém lze vyřešit extrakcí kombinační části kompaktoru, tedy simulace následujícího stavu je součástí výpočtu testovacího vektoru. Konceptuální obvod odražený v generované SAT instanci má tedy kromě výstupů testovaného obvodu (a jeho poruchového obrazu) také výstupy reprezentující budoucí stav kompaktoru. Podobně jako se musí lišit výstup bezporuchového obvodu a obvodu s poruchou, musí se také lišit výstup kompaktoru.

Takové řešení ovšem vede k dalším problémům, k nárůstu velikosti SAT instancí. Tento problém je potřeba analyzovat a zjistit, zda a nakolik překáží. Případně navrhnout způsob jeho zmírnění či odstranění, například výpočet stavu kompaktoru jen pro poruchy, kde hrozí aliasing a s tím související identifikace takových poruch.

Výsledkem by měla být metoda generování testu, která bude minimalizovat aliasing bez nutnosti zásahu do architektury kompaktoru. Dále by tato metoda měla jít použít i pro minimalizaci klopných obvodů v kompaktoru bez zvětšení míry aliasingu.

Poděkování

Autor děkuje za poskytnuté výpočetní a úložné zdroje programu Metacentra CESNET LM2015042 a CERIT-CS CZ.1.05/3.2.00/08.0144.

Tento výzkum byl částečně podporován z projektu ČVUT SGS16/121/OHK3/1T/18.

Tento výzkum byl částečně podporován z grantu české grantové agentury GA16-05179S.

Literatura

[1] P. D. Hortensius, R. D. McLeod, and H. C. Card, “Cellular automata-based signature analysis for built-in self-test,” *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1273–1283, Oct 1990.

[2] C. Stroud, *A Designer’s Guide to Built-in Self-Test*, ser. Frontiers in Electronic Testing. Springer, 2002.

[3] J. Roth, “Diagnosis of automata failures: A calculus and a method,” *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, July 1966.

[4] P. Goel, “An implicit enumeration algorithm to generate tests for combinational logic circuits,” *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215–222, March 1981.

[5] H. Fujiwara and T. Shimono, “On the acceleration of test generation algorithms,” *IEEE Transactions on Computers*, vol. C-32, no. 12, pp. 1137–1144, Dec 1983.

[6] M. Schulz, E. Trischler, and T. Sarfert, “SOCRATES: a highly efficient automatic test pattern generation system,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, Jan 1988.

[7] R. Drechsler, G. Fey, D. Tille, and S. Eggersglüß, *Test Pattern Generation using Boolean Proof Engines*, 1st ed., 12 2009.

[8] S. Eggersglüß and R. Drechsler, “Robust algorithms for high quality Test Pattern Generation using Boolean Satisfiability,” in *IEEE International Test Conference*, Nov. 2010, pp. 1 –10.

[9] T. Larrabee, “Test pattern generation using Boolean satisfiability,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[10] M. Rebaudengo, S. Reorda, Matteo, and M. Violante, “A new functional fault model for FPGA application-oriented testing,” in *17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2002, pp. 372–380.

[11] J. Borecký, M. Kohlík, P. Kubalík, and H. Kubátová, “Fault models usability study for on-line tested FPGA,” in *14th Euromicro Conference on Digital System Design (DSD)*, Aug 2011, pp. 287–290.

[12] R. Hülle, P. Fišer, J. Schnmidt, and J. Borecký, “SAT-ATPG for Application-Oriented FPGA testing,” in *The 15th Biennial Baltic Electronics Conference*, Oct 2016.

[13] S. Yang, “Logic synthesis and optimization benchmarks user guide: Version 3.0,” Jan. 1991.

[14] F. Brglez and H. Fujiwara, “A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran,” in *Proceedings of IEEE Int’l Symposium Circuits and Systems (ISCAS’85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[15] F. Brglez, D. Bryan, and K. Kozminski, “Combinational profiles of sequential benchmark circuits,” in *IEEE International Symposium on Circuits and Systems, 1989*, May 1989, pp. 1929–1934 vol.3.

[16] C. Albrecht, “IWLS 2005 benchmarks,” Tech. Rep., June 2005.

[17] G. Tseitin, “On the complexity of derivation in propositional calculus,” in *Automation of Reasoning*, ser. Symbolic Computation, J. Siekmann and G. Wrightson, Eds. Springer Berlin Heidelberg, 1983, pp. 466–483.

[18] B. Selman, “Stochastic search and phase transitions: AI meets physics,” in *International Joint Conference on Artificial Intelligence*, vol. 1. Morgan Kaufmann Publishers Inc., 1995, pp. 998–1002.

[19] J. Balcárek, P. Fišer, and J. Schmidt, “On properties of SAT instances produced by SAT-based ATPGs,” in *Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS)*, 2009, pp. 3–10.

[20] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, “2+p-SAT: relation of typical-case complexity to the nature of the phase transition,” *Random Structures & Algorithms*, vol. 15, no. 3-4, pp. 414–435, 1999.