Vol.2012-SLDM-155 No.5
Vol.2012-EMB-24 No.5
2012/3/2

論理関数の充足不可能性に注目した論理回路デバッグ手法の検討

李　在城[†1]　松本剛史[†2]　藤田昌宏[†2,†3]

回路中の誤り箇所を特定するデバッグは時間のかかる作業である。従来手法では反例に基づいてデバッグを始めるが、本稿では反例が得られない状況におけるデバッグ方法を考える。論理関数の充足可能性 (SAT) 問題を用いた検証では、充足可能な場合に設計が正しくなるように定式化される場合、反例を特定することができない。この場合には、論理式中で矛盾を起こしている部分集合 (UNSAT コア) がデバッグのヒントになると考えられる。そこで本稿では、複数の UNSAT コアを用いることで、誤り候補を絞り込む手法を考える。しかし、UNSAT コアを多数求めようとすると、現在のアルゴリズムでは計算量が爆発的に増加し、このままでは使えないという問題がある。本稿では、簡単な回路を対象に上記の方法を説明し、問題点を克服するための考察を行う。

A consideration of logical circuit debugging method using unsatisfiability

Jaesung Lee,[†1] Takeshi Matsumoto[†2]
and Masahiro Fujita[†2,†3]

Detecting errorneous behaviors in logic circuits and debugging are time-consuming tasks in circuit designs. While there exist many debugging methodologies starting with counter-examples, we consider to debug circuits when the circuits have no or few counter-examples. When a verification problem is formulated as Boolean satisfiability, a variable assignment generated by SAT solvers is a counter-example, when the formula is set to be *false* if the circuit behavior is correct. On the other hand, we can formulate a verification problem so that the Boolean formula is *true* if the circuit behavior is correct. In the case, we can find hints for debugging in UNSAT cores which are subsets of caluses including a logical conflict inside, since UNSAT cores correspond to a part of the circuit which has a logical conflict to the verified property. In this work, we consider to locate bugs using an intersection of multiple UNSAT cores for a property. However, the computation to get UNSAT cores exponentially increases with the number of UNSAT cores. In this paper, we discuss circuit debugging with UNSAT cores and more efficient ways to get UNSAT cores.

1. はじめに

VLSI(Very Large Scale Integration) 回路の大規模化・複雑化に伴い、設計期間の長期化が問題となっており、その効率化が必要となっている。中でも、設計が期待通りの動作をしているかどうかを確認する検証、そして、設計誤りがある場合のデバッグに、設計期間の大半が費やされている。デバッグにおける誤り箇所の特定は、現在でも設計者の手によって行われており、自動化による効率化が強く望まれている。本研究では、論理回路を対象にバグ位置特定の自動化手法を検討する。

検証によって、回路に誤りがあることが分かった場合、回路中の誤り箇所を特定する必要がある。この誤り箇所の特定を支援する手法として、回路中の信号値を他の値と置換することによって出力値を正すことができる個所を探索する手法[1)] と誤った出力値を伝搬させている経路を特定する手法[2)] が存在し、それに基づいた商用ツールも開発されている[3)]。前者は、図 1 に示すようなマルチプレクサ付きの回路に対して回路制約の式を生成する。このとき、入力値を反例と同じ入力値、出力値を期待される正しい値で制約して、充足可能性問題 (Satisfiability problem, SAT 問題) を解き、得られた変数割当てから誤り箇所の候補を得ることができる。この手法では、複数の反例を用いることによって、誤り箇所候補の精度を高めることができるが、与えられる反例によっては、多数箇所の候補が得られ、十分に絞り込むことができない可能性もある。一方、後者は、誤った出力値から各論理ゲートの制御値 (control value) をたどることによって、どの経路によって、誤った値が伝搬しているかを特定することができる。ただし、ゲート値が非制御値の場合には、複数の経路を別々にたどる必要があるため経路数が非常に多数になる可能性があること、さらに、得られた各経路が必ず誤り箇所を含むわけではないことに注意が必要である (全ての経路の和集合には

†1 東京大学大学院 工学系研究科 電気系工学専攻
Department of Electrical Engineering and Information Systems, School of Engineering, The University of Tokyo
†2 東京大学 大規模集積システム設計教育研究センター
VLSI Design and Education Center, The University of Tokyo
†3 科学技術振興機構 戦略的創造研究推進事業 CREST
CREST, Japan Science and Technology Agency

Vol.2012-SLDM-155 No.5
Vol.2012-EMB-24 No.5
2012/3/2

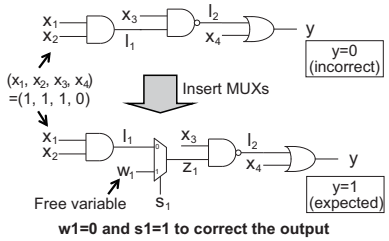

図 1 SAT に基づくデバッグ
Fig. 1 SAT-based debugging.

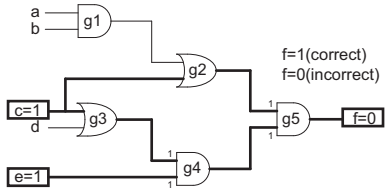

図 2 例
Fig. 2 An example.

必ず含まれる)。

本研究では「回路の出力値が誤っている場合には偽」となるような定式化によって SAT 問題を利用した検証を前提とした誤り位置特定を考える。この場合、1 つの充足可能な変数割当ては、回路のある正しい動作に相当する。つまり、どのような入力パタンを与えても、回路が誤った出力値を出す場合に充足不可能となる。回路に誤りがあり、式が充足不可能となる場合、SAT ソルバにより UNSAT コアが生成される。UNSAT コアには必ず 1 つ以上の誤り箇所が含まれており、本研究では、これを誤り箇所特定に利用することを考える。UNSAT コアを利用する手法は、前述の既存手法に比べて、誤り箇所候補が反例に依存しないこと、また、必ず誤り箇所を含んだ範囲を比較的狭い範囲に特定できることが期待できる。

図 2 では、$c = 1, e = 1$ の制約下で正しい出力 $f = 0$ であることを検証する例題を示している。しかし、回路に誤りがあるため、他の a, b, d のどのような入力パタンに対しても $f = 0$ とすることができない。この例で得られる UNSAT コアは $g2, g3, g4, g5$ を含む。一方、パストレースによって誤りを含むパスを求める場合、$a = 1, b = 1, d = 1$ の場合には全てのゲートが誤り候補となるが、$a = 0, b = 0, d = 0$ の場合には $g1$ は含まれない。このように、パストレースによる手法では、与えられる反例によって、制御値をたどった結果が異なることがある。このような反例によって、誤り箇所の候補が変化することは、信号値の置換によって出力値を正すことができる箇所を探す手法[1] でも同様である。これらに対して、本研究では、反例に依存せず、誤り箇所候補を求めることができる。

本稿の構成は次の通りである。第 2 節で、本研究で用いる SAT 問題と UNSAT コアを説明し、第 3 節で UNSAT コアをデバッグ支援に用いる関連研究を紹介する。第 4 節では、UNSAT コアを用いて求めることができる誤り箇所候補の範囲を実験結果を用いて示し、考察を行う。回路デバッグに UNSAT コアを用いる際には、1 つの SAT 問題から、多くの UNSAT コアを得た方が誤り箇所候補の範囲を狭めることができると期待される。そこで、第 5 節では、効率的に複数の UNSAT コアを求める手法の検討を行う。第 6 節でまとめと今後の課題を述べる。

2. 充足可能性問題と UNSAT コア

ブール式の充足可能性問題 (Boolean Satisfiability Problem, Boolean SAT 問題または単に SAT 問題) とは、与えられたブール式に対して、式中の変数値を 0 か 1 に定めることによって、式全体の値を 1 にすることができるかどうかを判定する問題である。多くの SAT ソルバーは乗法標準形 (Conjunctive Normal Form, CNF) で書かれたブール式を処理できる。CNF 式は 1 つ以上の項 (clause) の論理積から成り、1 つの項は 1 つ以上のリテラル (literal) の論理和である。リテラルは、ある変数または変数の否定である。式 1 に CNF 式の例を示す。

$$f = (x1 \vee \neg x3) \wedge (x2 \vee \neg x3) \wedge (\neg x1 \vee \neg x2 \vee x3) \quad (1)$$

上に示した例では、$(x1, x2, x3) = (0, 0, 0)$ の場合、f の値を 1 にすることができる。このように与えられた式を 1 とすることができる変数値の割当てが存在する場合、その式は充足可能 (Satisfiable, SAT) であるという。一方、式を 1 とする変数値の割当てが存在しない場合には、その式は充足不可能 (Unsatisfiable, UNSAT) である。

ここで、ある CNF 式が UNSAT であるとは、どのような変数値の割当てに対しても 1 とならないことを意味する。このような CNF 式から取り出された項の部分集合が UNSAT であるとき、その集合を UNSAT コア (Unsatisfiable core, UNSAT core) と呼ぶ。また、UNSAT コアのうち、どの 1 つの項を取り除いても、UNSAT コアの残りの項の集合が SAT になるものを最小 UNSAT コア (Minimal Unsatisfiable Subset, MUS) と呼ぶ。以降では、特に断りがない限り、UNSAT コアは最小 UNSAT コアを指すものとして議論を進める。

$$f = (x1 \vee \neg x3) \wedge (x2 \vee \neg x3) \wedge (\neg x1 \vee \neg x2 \vee x3) \wedge (\neg x1) \wedge (\neg x2) \wedge (x3) \quad (2)$$

式 2 に UNSAT な CNF 式を示した。式中の 3 つの項 $(x1 \vee \neg x3) \wedge (\neg x1) \wedge (x3)$ はどのような x1, x3 の値の組合せに対しても 1 とはできない。さらに、3 つのうちどの項を除いても、残された 2 項は SAT となる。従って、この項の集合は最小 UNSAT コアである。さらに $(\neg x2) \wedge (x2 \vee \neg x3) \wedge (x3)$ も最小 UNSAT コアである。このように、1 つのブール式が UNSAT の場合、最小 UNSAT コアは複数存在することもある。

次に、回路要素の CNF 式表現を説明する。各論理ゲートの入力値と出力値の間で成り立

Vol.2012-SLDM-155 No.5
Vol.2012-EMB-24 No.5
2012/3/2

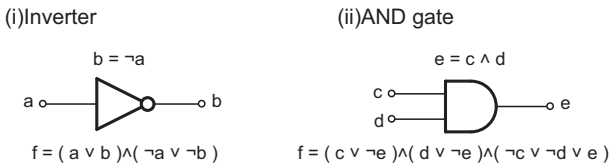

図 3 回路要素の SAT 式
Fig. 3 SAT instance of circuit elements.

つ等式を変換することにより、論理ゲートの入出力変数の間で成り立つ関係を CNF 式として表現することができる。図 3 に、いくつかの論理ゲートの CNF 表現を示した。入力値と出力値の関係がゲートの論理の通りであるとき、各 CNF 式 (図中の f) は常に 1 となる。論理回路中の各信号値は、CNF 式上で 1 つの変数として表現される。この回路から CNF 式への変換は、ゲートの論理と入力信号線、出力信号線が与えられれば可能であり、ゲート数に比例する計算量で変換できる。

3. UNSAT コアをデバッグに用いる研究

本節では、UNSAT コアを用いたデバッグ手法の研究を 2 つ紹介する。

文献[4] では、図 1 に示した方法によって誤り箇所を求める際に、より大規模な回路を扱うために、候補となり得る信号線の集合を小さくするような手法を提案している。回路中に含まれる全ゲートに対する回路制約から CNF 式を生成した場合、SAT ソルバの実行時間が長くなってしまう。そこで、回路中のある部分 (モジュール) における入出力値を、与えられた反例によって求めた値によって制約し、この部分 (モジュール) 内の論理を表す回路制約を省略することで効率化を実現している (抽象化)。この時、省略された部分内に誤りがある場合、CNF 式が UNSAT となる。その場合、UNSAT コアに含まれる部分 (モジュール) の回路制約を抽象化された式に追加することにより、その部分 (モジュール) 内の誤り箇所候補を求めることができるようになる (抽象化の修正)。抽象化の修正方法として、UNSAT コアに含まれるモジュールを全クロックサイクルにおいて復元する方法と、当該モジュールの特定クロックサイクルのみを復元する方法の 2 つが提案されている。

ある回路で反例が得られた場合、反例と同じ入力値、正しい出力値、そして、回路制約から成る CNF 式は必ず UNSAT となる。文献[5] では、そのようにして得られる UNSAT コアの中に含まれるゲートのみを対象にして、図 1 に示す手法を適用することによる効率化を提案している。つまり、図 1 におけるマルチプレクサを用いた信号値の置換を全てのゲートに適用するのではなく、UNSAT コア内のゲートのみに適用することにより、SAT ソルバの実行時間を短縮している。文献[5] では、複数の UNSAT コアの共通部分を取ることにより、複数の誤り箇所を求める手法も提案されている。

以上の 2 つの既存研究では、どちらも与えられた反例に対して誤り箇所候補を求めているのに対し、本研究では、反例が与えられない状況でデバッグを行うことを目指している。既存研究における UNSAT コアの利用は、文献[1] で提案されている、反例の誤った出力値を正しい値にするために回路中のどの信号値を変更すればよいかを求める手法の効率化が目的である。一方、本研究では、反例を用いず、UNSAT コアのみから誤り箇所候補を求めるため、与えられた反例に依存しない結果を得ることが期待できる。

4. 実験:UNSAT コアを用いた誤り位置の特定

本節では、いくつかの誤りを含む例題回路に対して UNSAT コアを求め、それを用いて誤り位置をどの程度の範囲で特定できるかを実験結果とともに示す。

4.1 実 験 環 境

実験では、Verilog HDL で記述された論理回路から CNF 式を生成し、その式を SAT ソルバを用いて解くことにより UNSAT コアを生成した。対象回路が順序回路の場合には、指定回数だけ時間軸方向に展開した組合せ回路を変換の対象としている。検証対象のプロパティについては、論理回路中の信号線を参照してプロパティの成否を判定するモニタ回路を作る。このモニタ回路の出力 p は、回路がプロパティを満たしている場合に 1 になるように設計する。モニタ回路は Verilog HDL で検証対象の論理回路と同様に記述されており、回路とともに CNF 式に変換している。このようにして得られた CNF 式に対して、$(p = 1)$ という制約を与えて SAT ソルバで充足可能性を解く。このとき、どのような入力パタンに対しても、プロパティが成立しない場合、$(p = 1)$ を満たす変数割当てが存在しないため、CNF 式は UNSAT となり、UNSAT コアを得ることができる。

今回の実験では、複数の UNSAT コアによって、どれだけ誤り位置を狭い範囲に特定できるかを評価するため、複数の UNSAT コア (可能であれば、全ての UNSAT コア) を求める必要がある。そこで、文献[8] で提案されている、全ての UNSAT コアを求める手法を適用している。この手法では、まず、Minimal Correction Set (MCS) と呼ばれる項の部分集合を求め、そこから最小 UNSAT コアを導出する。MCS は、UNSAT な CNF 式の項の部分集合のうち、元の CNF 式から取り除くことにより残された CNF 式が SAT になり、かつ、MCS に含まれるどの項をその SAT な CNF 式に追加しても CNF 式が UNSAT にな

Vol.2012-SLDM-155 No.5
Vol.2012-EMB-24 No.5
2012/3/2

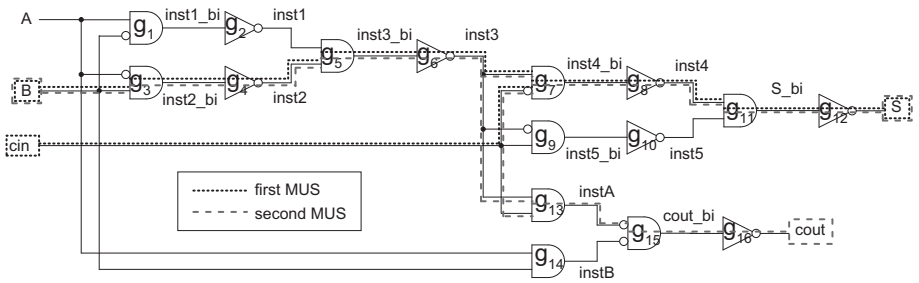

図 4 FA_bug1 において得られた UNSAT コア
Fig. 4 UNSAT cores in FA_bug1.

るような項の集合である。MUS と MCS は互いに Irreducible Hitting Set の関係にあり、図 5 に示す方法で変換することができる。詳細は文献[8] を参照されたい。実験では、SAT ソルバとして PicoSAT[6][7] を用い、PicoSAT 上に、全ての MCS を求める機能と MCS から MUS を求める機能を追加して全ての UNSAT コアを求めている。

実験に用いた例題回路は以下の通りである。

- **全加算器 (FA)** AND ゲートと INV ゲートで構成された 1 ビット全加算器。誤りとして 3 箇所の変更 (bug1, bug2, bug3) を用意し、単一の誤りを持つ 3 つの回路と複数の誤りを持つ 4 つの回路を用いた。
- **2 ビットカウンタ (C2)** 2 個の全加算器を接続した順序回路。最下位ビットのキャリー入力は 1 に固定してある。各全加算器の片方の入力は現在のカウンタ値に接続し、他方を 0 に固定している。出力値が次時刻のカウンタ値となっている。誤りとして異なる 1 箇所を変更した 2 つの回路 (bug1, bug2) を用いた。
- **4 ビットカウンタ (C4)** 2 ビットカウンタと同様に、4 個の全加算器を接続した順序回路。2 ビット目から 3 ビット目へ伝搬するキャリー信号が反転する誤りを挿入している。
- **実設計例題 (Industry1, Industry2)** 組合せ回路部分のゲート数がそれぞれ 20523, 4082、状態変数がそれぞれ 1417, 282 の順序回路。回路中に誤りがあるため、到達すべきある状態に到達しない回路となっている。実験では、この順序回路をそれぞれ 50 回ずつ展開した回路を用いている。

全ての実験は、CPU Xeon 2.6GHz, メモリ 4GB を有する計算機上で行った。

4.2 実験結果

表 1 実験結果 (1)
Table 1 Experimental results(1).

Circuit	Time	# of MCS	# of MUS	# of gates	# of gates in every MUS	Ratio
FA_bug1	3 ms	14	2	16	8	50%
FA_bug2	4 ms	16	2	16	10	63%
FA_bug3	6 ms	27	6	16	2	13%
FA_bug1_2	8 ms	37	15	16	6	38%
FA_bug1_3	6 ms	74	8	16	0	0%
FA_bug2_3	7 ms	78	6	16	0	0%
FA_bug1_2_3	3 ms	20	6	16	2	13%
C2_bug1	9 ms	61	2	33	23	70%
C2_bug2	9 ms	34	4	33	13	39%
C4_unroll1	>24 h	-	-	67	-	-

表 2 実験結果 (2)
Table 2 Experimental results(2).

Circuit	Time(ms)	# of gates	# of gates in MUS	Ratio
Industry1	2121	1026200	9466	0.9%
Industry2	411	204200	917	0.4%

表 3 実験結果 (3)
Table 3 Experimental results(3).

Circuit	# of Bug	# of candidates w/o UNSAT cores	# of candidates with UNSAT cores	Ratio
FA_bug1_2	2	240	180	75%
FA_bug1_3	2	240	156	63%
FA_bug2_3	2	240	182	76%
FA_bug1_2_3	3	3360	2041	61%

実験結果を表 1 に示す。UNSAT コアが得られた回路において、全ての UNSAT コアに誤り箇所が含まれていることを確認した。誤りが 1 箇所の場合には、UNSAT コアの共通部分に誤りがあるため、全体のゲート数に比べて、UNSAT コアに含まれるゲート数がどの程度狭い範囲になっているかが重要となる。誤りを 1 つ含む全加算器では平均 42%、2 ビットカウンタでは平均 55%のゲート数が UNSAT コアの共通部分に含まれていた。

図 4 に、FA_bug1 において得られた全ての UNSAT コアを示す。この例では、全部で 2

Vol.2012-SLDM-155 No.5
Vol.2012-EMB-24 No.5
2012/3/2

つの UNSAT コアが得られた。誤りはゲート $g4$ であり、UNSAT コアの共通部分に含まれていることが分かる。また、UNSAT コアが、出力からつながった一連の部分回路となっていることが分かる。これは、全ての実験結果で得られた UNSAT コアでも同様である。誤った出力からつながった一連の部分回路と離れた部分が UNSAT コアとなることはない。そのような誤った出力を含む部分回路から離れている部分を UNSAT コアから取り除いた項の集合も UNSAT コアとなる。しかし、本研究では、最小 UNSAT コアを用いているため、そのような離れた回路部分は UNSAT コアに含まれない。

表 1 からも分かるように、4 ビットカウンタの例では全ての UNSAT コアを 24 時間以内に生成することができなかった。この結果から、文献[8] で提案されている手法で全ての UNSAT コアを生成する場合、小規模な回路であっても、非常に長い時間がかかる場合があることが分かる。これは、UNSAT コアを回路デバッグに利用する場合に大きな問題となる。これについては、次節で、より効率的な手法を検討する。

回路に m 個の誤りがある場合、UNSAT コアを用いない手法では、ゲート数 n の回路における誤り箇所の組合せ総数は ${}_nC_m$ となる。UNSAT コアを用いる場合、UNSAT コアの共通部分とそれ以外の部分を区別して考えることにより、あり得る組合せの数を減らすことができる。表 3 では、UNSAT コアを利用することにより、誤り箇所の組合せの総数が減少していることが確認できる。

表 2 には、Industry1 と Industry2 において得られる UNSAT コアに含まれるゲート数を示す。これらの回路では、回路規模が大きいため、全ての UNSAT コアを求めることはできなかった。しかし、得られたそれぞれ 1 つの UNSAT コアに含まれているゲート数は、それぞれ全体の 0.9%と 0.4%であり、全体の回路規模に対して、かなり狭い範囲に誤り箇所を特定できていると言える。このような規模の例題で、複数の UNSAT コアを用いて、さらに狭い範囲に誤りを特定することができれば、回路デバッグに非常に有効であると考えられる。

5. 効率的な複数 UNSAT コア生成手法の検討

前節の実験結果からも分かるように、ある程度規模の大きな回路では、最小 UNSAT コアを全て求めることは現実的に不可能である。一方で、誤り箇所をより狭い範囲で特定するためには、より多くの UNSAT コアを用いる必要がある。そこで、本節では、効率的に複数の UNSAT コアを生成する手法を検討する。

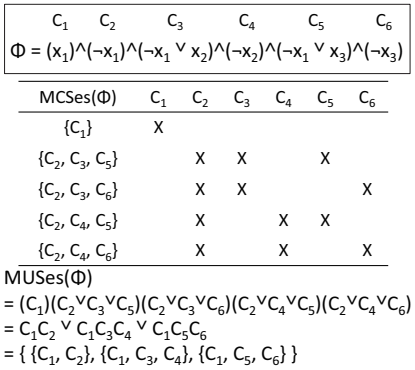

図 5 MCS から MUS を求める方法[8)]
Fig. 5 A method to calculate MUSes from MCSes.[8)]

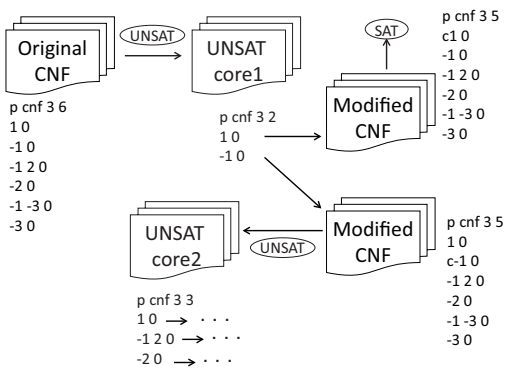

図 6 1 つの UNSAT コアから他の UNSAT コアを求める手法
Fig. 6 A method of getting another UNSAT core from existing one.

5.1 一部の MCS からの MUS 生成

前節の実験では、文献[8)] で提案されている MCS から最小 UNSAT コアを求める手法を適用して、全ての UNSAT コアを求めようとした (図 5)。しかし、実験結果でも見られたように、回路の論理や規模によっては、MCS の数は非常に多くなる場合がある。そこで、ここでは、全てではなく、一部の MCS から UNSAT コアを得ることを考える。例えば、図 5 の例では、全部で 5 つの MCS のうち、初めの 3 つが与えられた場合、そこまでの計算で得られる不完全な UNSAT コアは、$(C_1, C_2), (C_1, C_3), (C_1, C_5, C_6)$ の 3 つである。このうち、(C_1, C_3) は、全ての MCS が与えられた場合には、最終的に (C_1, C_3, C_4) となる。このように、一部の MCS に対して計算された不完全な UNSAT コアにいくつかの項を追加することによって、完全な UNSAT コアとすることができる場合がある。

実験結果でも見られたように、誤った回路から得られる UNSAT コアは、誤った出力につながった部分回路となっている。そのため、一部の MCS から計算された不完全な UNSAT コアが表す回路を拡張して、誤った出力につながる部分回路を作ることにより、完全な UNSAT を求める手法が考えられる。この場合、

- 得られた UNSAT コアの候補が UNSAT であるかどうかを SAT ソルバを使って確認する必要があること
- 得られた UNSAT コアが最小 UNSAT コアでない可能性があること

Vol.2012-SLDM-155 No.5
Vol.2012-EMB-24 No.5
2012/3/2

- 誤った出力につながる部分回路の作り方が複数通りある場合、複数の候補を試す必要があること

といった問題点を解決する必要がある。

5.2 1 つの UNSAT コアから他の UNSAT コアを求める手法

SAT ソルバは、全ての UNSAT コアの計算と比較して、ある 1 つの UNSAT コアを求めることは非常に短時間で行うことができる。例えば、表 2 では、数十万ゲートの回路において、UNSAT コアを 1 つ求めることに成功している。これを利用して、1 つの UNSAT コアから、他の UNSAT コアを求めることができれば、比較的小さな計算量で複数の完全な UNSAT コアを求められると期待できる。

1 つの UNSAT コアから、他の異なる UNSAT コアを求める方法の概要を図 6 に示す。この方法では、まず、得られた UNSAT コアに含まれる項を 1 つずつ除いた CNF 式を作る。これにより、それぞれの CNF 式が UNSAT である場合でも、既に得られた UNSAT コアと同じ UNSAT コアが含まれることはない。このようにして作られた CNF 式において、UNSAT コアが得られれば、複数の UNSAT コアを比較的短い計算時間で得ることができる。ただし、UNSAT コアに含まれる項を除いた CNF 式は SAT になる場合もある。その場合は、その CNF 式から UNSAT コアを得ることはできない。図 6 では、1 つ目の UNSAT コアとして、$(v_1), (\neg v_1)$ が得られる (ここでは、CNF 式で n で表される変数を v_n とする)。そこで、これと同じ UNSAT コアが再び得られるのを防ぐため、それぞれの項を除いた CNF 式を作る。項 (v_1) を除いた CNF 式は SAT となるため、UNSAT コアを得ることはできない。一方、$(\neg v_1)$ を除いた CNF 式は再び UNSAT となるため、元の UNSAT コアとは異なる UNSAT コア $(v_1), (\neg v_1 v_2), (\neg v_2)$ を得ることができる。これを繰り返すことにより、複数の UNSAT コアを次々に求めることが可能であると考えられる。

この方法によって、全ての UNSAT コアを求めようとすると、項の除き方の組合せの数が非常に大きくなり、計算時間も長くなることが予想される。しかし、この方法では、全ての UNSAT コアが求まらない場合であっても、その途中結果として、完全な UNSAT コアを複数求めることができるため、規模の大きな回路において、有効な UNSAT コアの求め方となる可能性がある。

6. ま と め

本稿では、複数の UNSAT コアを用いた効率的な回路デバッグの検討を行った。この際、誤りが 1 つの場合は UNSAT コアの共通部分を、複数の場合は各 UNSAT コアへの出現関係から複数ゲートの組を候補と考えることで、回路全体を誤り候補と考える手法より対象のサイズを小さくできることを確認した。しかし、今回用いた手法では、計算時間が非常に長くなるため、全ての UNSAT コアを求めることは現実的ではない。そこで、複数の UNSAT コアを効率的に得る手法として、(1)MCS から最小 UNSAT コアを求める方法、(2) 得られた 1 つの UNSAT コアから他の UNSAT コアを得る方法を検討した。今後は、これらの考えを基にして、論理回路デバッグにおいて、効率的に複数の UNSAT コアを用いて誤り箇所を特定する手法を研究していく予定である。

参 考 文 献

1) A. Smith, A. Veneris, M.F. Ali, and A. Viglas, "Fault Diagnosis and Logic Debugging Using Boolean Satisfiability," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.24, No.10, pp.1606–1621, 2005.
2) S. Venkataraman and W.K. Fuchs, "A deductive technique for diagnosis of bridging faults," *Proc. of International Conference on Computer-Aided Design*, pp.9–13, 1997.
3) Vennsa Technologies: OnPoint, available from 〈 http://www.vennsa.com/ 〉 (accessed 2012-02-02).
4) B. Keng and A. Veneris, "Managing complexity in design debugging with sequential abstraction and refinement," *Proc. of 16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pp.479–484, 2011.
5) Andre Suelflow, Goerschwin Fey, Roderick Bloem, and Rolf Drechsler, "Using unsatisfiable cores to debug multiple design errors," *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pp.77–82, 2008.
6) Armin Biere: PicoSAT, available from 〈 http://fmv.jku.at/picosat/ 〉 (accessed 2012-02-02).
7) A. Biere, "PicoSAT Essentials," *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, vol.4, pp.75–97, 2008.
8) M.H. Liffiton, K.A. Sakallah, "Algorithms for computing minimal unsatisfiable subsets of constraints," *Journal of Automated Reasoning*, Vol.40, No.1, pp.1–33, 2008.
9) Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, *HANDBOOK of satisfiability*, IOS Press, 2009.