doi:10.11887/j.cn.201806014 http://journal.nudt.edu.cn

加强约束的布尔可满足硬件求解器*

马柯帆,肖立权,张建民,黎铁军,周善祥
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:利用现场可编程门阵列固有的并行性和灵活性,提出在硬件可编程平台上基于随机局部搜索算法的布尔可满足性求解器,用于求解大规模的布尔可满足性问题。相对其他求解器,该求解器的预处理技术能极大提高求解效率;其变元加强策略避免了同一变元被反复连续翻转,降低了搜索陷入局部最优的可能。评估结果表明,求解器最多能处理32 000个变元/128 000个子句的实例。相比当前同类型的求解器,其求解效率明显提高。

关键词:现场可编程门阵列;布尔可满足性;加强约束;不完全算法
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2018)06-105-07

Hardware Boolean satisfiability solver with enhanced constraint

MA Kefan, XIAO Liquan, ZHANG Jianmin, LI Tiejun, ZHOU Shanxiang
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: Taking advantage of parallelism and flexibility of field-programmable gate array, a novel Boolean satisfiability solver based on an improved local search algorithm on the reconfigurable hardware platform was proposed to solve large-scale Boolean satisfiability problems. In comparison with the past solver, the preprocessing technology can strongly improve the efficiency of solver; the strategy of strengthening the variable selection can avoid the same variable flipped continuously and repeatedly. It can reduce the possibility of search falling into local optimum. The experimental results indicate that the solver can solve problems of up to 32k variables/128k clauses, and has better performance than previous works.

Key words: field-programmable gate array; Boolean satisfiability; enhanced constraint; incomplete algorithm

布尔可满足性(Boolean SATisfiability,SAT)问题作为第一个被证明的非确定性多项式完全(Non-deterministic Polynomial Complete,NPC)问题,是计算机理论与应用的核心问题,被广泛应用于电子设计自动化、芯片形式化验证、密码学、芯片映射、人工智能等领域。最坏的情况下,SAT求解时间随着实例的规模呈现指数增长的趋势。因此,探索不同的方法,提高SAT求解效率具有非常重要的现实意义。现场可编程门阵列(Field-Programmable Gate Array,FPGA)由于其设计灵活性和并行性等特点,逐渐成为研究可满足性问题的热点。文献[1]对基于FPGA的SAT求解算法的研究现状做了详细的介绍。然而,仍存在两方面的因素制约着硬件SAT求解器性能的提升[2]。一方面是算法,对实际应用中的许多问题来说,简单的算法并不能有效解决大的现实问题,而使用复杂的算法必然会增加复杂度,并且在硬件中实现复杂的控制逻辑和存取序列并不容易;另一方面是存储,一般来说,工业实例中的SAT问题规模很大,问题变元数目可达数百万,甚至具有千万量级的约束子句,算法求解过程中数据存储方式势必影响求解器性能。

针对以上问题,本文设计了基于传统局部搜索的加强约束随机行走(Enhanced Constraint WalkSAT, ECWSAT)算法,用以解决随机生成的大规模SAT问题。算法对搜索变元随机的初始指派值做了一定的约束,以增加可满足子句数。

1 可满足性问题

合取范式(Conjunctive Normal Form, CNF)的构造规则是:文字是变元及其否定形式,而若干文字的析取构成子句,若干子句的合取组成公式。

* **收稿日期**:2017-09-18
基金项目:国家自然科学基金资助项目(61103083, 61133007, 61572509);国家重点基础研究发展计划资助项目(2016YFB0200203)
作者简介:马柯帆(1985—),男,湖南永州人,博士研究生,E-mail:makefan14@nudt.edu.cn;
肖立权(通信作者),男,教授,博士,博士生导师,E-mail:xiaolq2015@163.com

定义1 对给定的CNF表达式 F 以及 F 中变元的集合 V,若存在一组 V 的逻辑赋值使 F 的值为1,则说明 F 满足;反之,F 不满足。

例如,合取范式 $F=(\bar{a}\vee b\vee\bar{c})\wedge(b\vee c)$,当 $a=0,b=0,c=1$ 时,F 的值为1。则说明此公式是可满足的。

定义2 若CNF公式表示的SAT问题中任意子句最多包含 k 个文字,则称之为 k - SAT问题。

已经证明,通过引入新的变元 k - SAT均能在多项式时间内转化为3 - SAT问题[1],因此,对探索一般SAT问题求解算法来说,研究3 - SAT求解具有重要的意义。

SAT问题的求解算法通常可分为完全算法和不完全算法两大类。完全算法总是可以找到使公式满足的解或者推断出问题不可满足。其优点是能准确判定SAT问题是否满足,当实例无解时可以给出完整的证明;但缺点是算法解空间的复杂度会随着问题规模的不断增大呈指数增长,因此,仅适合求解小规模的SAT问题,且计算效率不高。经典的完全算法大多基于Davis和Putnam的DPLL算法[3]。

不完全算法主要包括局部搜索算法[4]和遗传算法[5]。算法试图将不可满足的子句数量最少化,通过不同的启发式引导算法在解空间内进行随机搜索,最终逐步逼近问题的解。不完全算法不能保证在规定的时间内判定问题的满足性。换句话说,当算法找到解时说明可满足;反之,不能确定此问题不可满足。相对于完全算法,不完全算法单位时间内的迭代次数更多,因此,求解速度更快。对于某些类型的SAT问题例如3 - SAT,特别是规模较大的3 - SAT问题,要比很多完全算法更为有效。

2 ECWSAT算法

针对如何解决大规模的SAT问题,在经典不完全算法的基础上进行了大量改进,通过计算变元初始指派为正的概率 P_i,对各变元的初始指派做了适当的约束,以增加可满足子句数。为了避免搜索陷入局部最优,算法引入一个噪声扰动机制,当搜索进入局部最优时,允许算法按照一定策略选择和当前候选解相同质量或者次质量的解,使算法进入不同的优化方向,从而跳出局部最优。

ECWSAT主算法伪代码见算法1,软件预处理按照一定的策略计算各变元初始指派为正的概率。主程序中,根据计算的概率给变元赋初值,此

算法1 算法主程序

Alg. 1 Main procedure of our algorithm

输入:CNF公式F, Maxtries, Maxflips
输出:公式可满足的赋值(若有)

```
ECWSAT (F, Maxtries, Maxflips)
{
/*预处理*/
    计算各变元的初始指派为正的概率 P_i;
/*主程序*/
  for (tries = 1 to Maxtries) {
    V = 各变元以概率 P_i 为正,产生一组初始随机
        指派;
    for (flips = 1 to Maxflips) {
      if V 满足 F,then return V;
      C = 随机选择一个不可满足子句;
      p = 以一定的启发式 Heuristic(c)选择一布尔
          变元;
      V = V with p flipped;
      噪声扰动机制();
    }
  }
  Heuristic(c){
    L = ∅;
    for (c 中每一个文字 l_i){
      计算 break-value 并排序;
      if(break-value = 0)
      L = L∪l_i
      }
    if (L 非空且 L 中变元最近未被翻转){
        随机选择 L 中的任一变元 p;
      }
      else{
        if(p 具有最小 break-value 且最近未被翻转){
            选择变元 p;
        }
        else if(p 具有次小 break-value 且最近未被翻
    转){
            选择变元 p;
        }
        else{
            选择最早被翻转的变元;
        }
      }
}
噪声扰动机制(){
    if (搜索结果在最近 N_c × θ + 3 步并未改善){
      if (p 具有次小 break-value 且最近未被翻转){
          选择变元 p;
        }
      else{
          选择具有第3小 break-value 的变元;
        }
      }
  }
  return "没有发现可满足的指派";
}
```

时若 CNF 公式满足,则求解完成,并返回变元当前赋值。否则,随机选择一个不可满足的子句,以启发式 Heuristic(c)选择某一布尔变元,并翻转变元赋值(由 1 变为 0 或者 0 变为 1),然后判定 CNF 公式是否满足,若满足,则求解完成,返回变元当前赋值;若不满足,重复上一步骤直至找到问题的解或者在规定的时间内找不到解。其中,*Maxtries* 和 *Maxflips* 用于控制算法的最长执行时间,分别代表未找到解时算法重新开始搜索的次数和单次搜索中允许变元翻转的次数。

定义 3　对 CNF 公式子句中的变元 p,当其中一个变元翻转时,原本在前一次赋值下满足的子句可能变为不可满足,这些由真变假的子句数称为该变元 p 的 break-value 值。

算法引入一种新的启发式策略,用于选择下一个将被改变赋值的变元。该策略通过记录最近翻转过的变元信息,避免了同一个变元被连续翻转。执行过程中,对不满足子句的 3 个文字,分别计算各自的 break-value 并排序,选择具有下列特征的变元。若存在一组最近均未被翻转的变元 V,且其 break-value 都为 0,则从 V 中随机选择一个变元。如果存在某个变元 p,具有最小 break-value,且最近未被翻转,则选择此变元。选择具有次小 break-value 值并且最近未被翻转的变元。若 3 个变元最近均被翻转,则选择最早被翻转的变元。

若求解过程持续选择最优变元进行翻转,搜索有可能陷入局部最优。逃离局部最优最普遍的方法是局部搜索策略在每次搜索的过程中并不一定都保持最优移动,而是以一定的概率做非最优移动。这种方式称之为扰动移动,其概率又称为扰动参数或噪声参数[6]。为了简化搜索过程,当搜索结果在最近 $N_c \times \theta + 3$ 步没有改善,且具有次小 break-value 值的变元最近未被翻转时,选择此变元,否则选择具有第 3 小 break-value 值的变元。其中,N_c 为给定实例中子句的数量,θ 为常量,取 1/6[7]。

3　ECWSAT 求解器的实现

ECWSAT 求解器主要包含软件预处理和硬件求解两方面。软件预处理由主机在求解开始前完成,主要负责变元的初始指派以及从 DIMACS 数据格式中提取地址和子句信息,FPGA 实现算法中的主程序部分。

3.1　软件预处理

大量研究表明,k-SAT 公式的可满足性与其子句变元比密切相关[8-10]。假设一个 k-SAT 公式包含 m 个变元以及 n 个子句,则该实例的子句变元比定义为 $r = n/m$。对于随机的 3-SAT 问题,当 r 比某个常量 r_0 小时,该公式极有可能是可满足的,也就是说问题的可满足性接近于百分之百。当 r 高于常量 r_0 时,该问题不可满足的可能性接近于 100%,常量 r_0 被称为可满足阈值或临界值。Achlioptas 等证明了当 $r < 3.145$ 时,随机 3-SAT 公式满足的可能性接近 1[8];Dubois 等证明当 $r > 4.267$ 时,随机 3-SAT 公式不可满足的可能性接近 100%[9]。这种现象被称为 SAT 问题的相变。

文献[6,10]等通过理论和实践对临界值的大小进行了详细研究。不同文献的讨论结果有所差异,但一致的观点是,当变元赋初值后产生的不可满足子句数远远小于变元产生的子句样本空间时,公式满足的可能性较大。对典型的 3-SAT 问题,由于每一个子句最多包含 3 个文字,每个文字是变元或其否定形式,因此 m 个变元可产生的子句样本为 $2^3C_m^3$,临界值 r_0 约为 4.3,临界点 r_0 处子句的数量 $n \approx 4.3m$。当 $n \ll 2^3C_m^3$ 时,各文字在子句中的分布非常不均,有些文字甚至自始至终都不在子句中出现。研究表明,这些未出现的文字为问题解的可能性要小于出现过的文字[11]。在局部搜索算法中,若能对随机产生的初始变元进行改进,尽量使初始指派的可满足子句更多,则可大大提高问题的收敛速度,减少搜索陷入局部最优的现象的发生。

针对上述思想,提出变元初始指派取正的概率 P_i 的计算公式。

$$P_i = \begin{cases} m_i \times \Delta/(m_i + n_i) + r_1, & m_i > n_i \\ m_i \times \Delta/(m_i + n_i) + r_2, & m_i < n_i \\ 0.5, & m_i = n_i \end{cases} \quad (1)$$

其中:m_i 为实例中变元 i 取值为 1 的子句的个数;变元 i 取值为 0 的子句的个数表示为 n_i;Δ 为 0.5 ~ 1之间的常数,此时取 0.9;当 $m_i > n_i$ 时,r_1 就[$(1-\Delta)/2, 1-\Delta$)之间的随机数,当 $m_i < n_i$ 时,r_2 是(0,$(1-\Delta)/2$]之间的随机数。通过对 r 取值的约束,使得当变元 i 为 1 的子句的个数大于为 0 的子句的个数时,其初始指派取正(为 1)的概率大于 0.5;反之,概率小于 0.5。整个系统的预处理代码和数据提取工作使用 C 语言在 Linux Ubuntu - 14.04 环境下实现。通过适当约束各变元的初始赋值,算法在搜索的过程中,保证了可满足的子句数尽可能地增加,从而很大程度

上减少了变元翻转的次数,提高了算法的求解效率。

3.2 硬件体系结构

ECWSAT 的硬件体系结构如图 1 所示,主要包括地址和子句映射模块、子句评估模块、子句暂存和计数器模块、变元加强模块、并串转换模块、先进先出(First Inpat First Output, FIFO)队列树、随机地址产生模块和不可满足子句存储器模块。

地址和子句映射表主要完成软件预处理后的地址和子句信息存储,算法执行过程中,此模块根据变元查找对应的子句信息并输出。定义 N_{c_i} 为公式中包含文字 i 的子句数量,对于有 N_c 个子句 N_v 个变元的 3-SAT 公式来说,N_{c_i} 的平均值可表示为 $N_{c_i}=\frac{3\times N_c}{2\times N_v}$。可知,$N_{c_i}$ 的最大值并不取决于公式规模的大小,而由 N_c 和 N_v 的比率决定。对大多数公式来说,N_{c_i} 的值小于 20,并且电子设计自动化领域中的 SAT 求解器的信号扇出数一般也小于 20[12]。为增大求解实例的规模,设计的 N_{c_i} 为 30,表中每个子句包含 2 个文字,第 3 个文字即为寻址地址,因此整个子句表的大小为深 32 768 bit、宽 32 × 30 bit。地址映射表宽度为 45 bit,包含地址部分和掩码部分,地址部分各比特位分别对应 N_{c_i} 个子句,某位有效则选中相应的子句编号(纵坐标),掩码部分用于选择子句表的横坐标,地址映射表本身地址分别对应 N_v 个变元以及取反,用这种方法可以迅速找出包含文字 i 的子句。求解开始前,地址和子句信息被分别下载到地址映射表和子句表的 ROM 中。

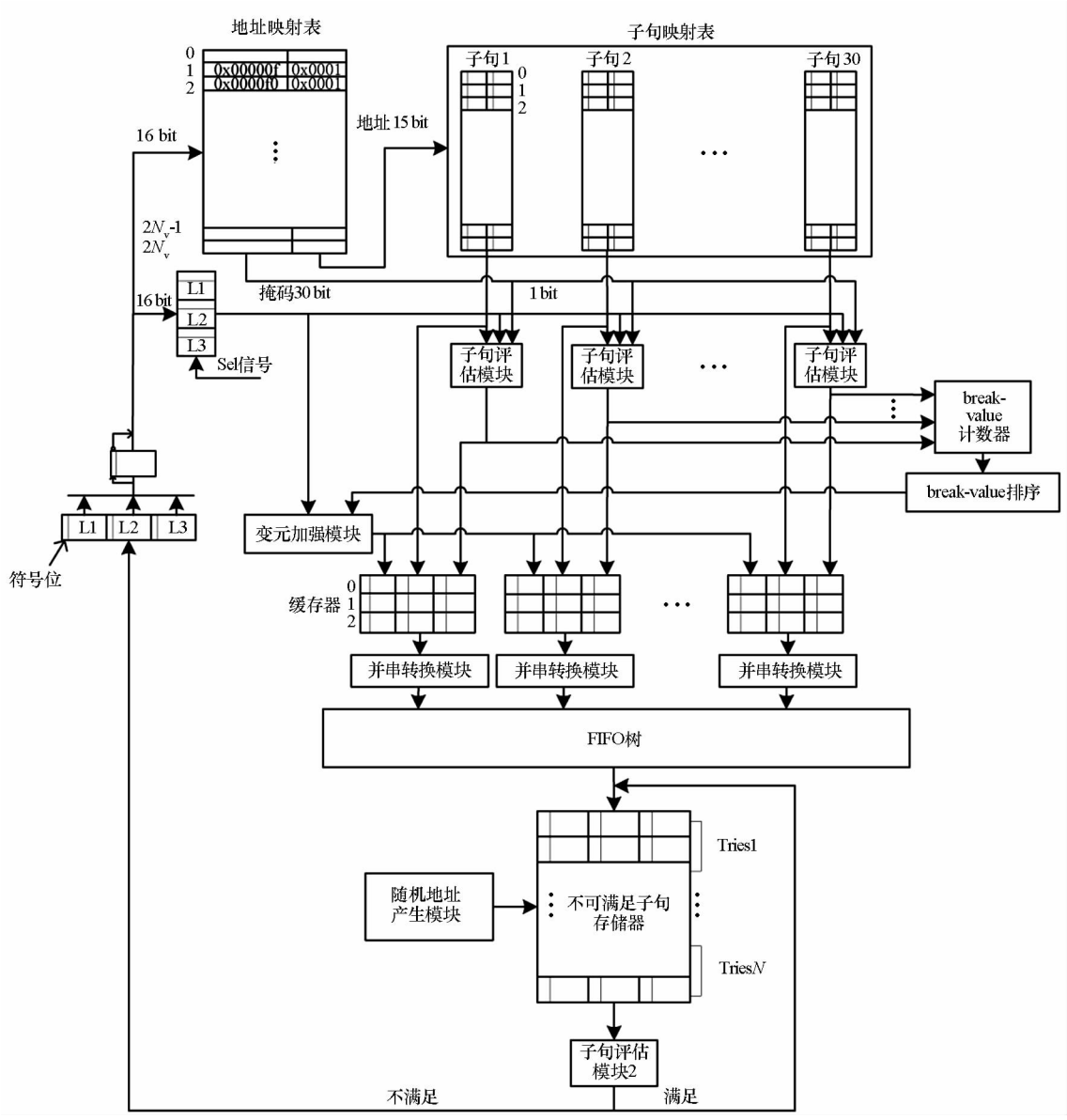

图 1 硬件框图

Fig. 1 Hardware overview

图 2 为子句评估模块 2 的内部结构,由 3 个片上存储器和一些简单的逻辑组成,*Maxtries* 组变元初值依次下载到 FPGA 的存储器中,以实现 *Maxtries* 次搜索。该模块实现的功能为:选择信号决定搜索时实际的变元取值,当地址使能信号为高时,根据子句映射模块输出的子句信息,在存储器中查找相应的变元值并进行运算,判断此子句是否满足;更新待翻转变元的值,即根据实际翻转变元的地址将所有子句评估模块内对应的地址的值翻转。子句评估模块跟模块 2 具有类似结构,只是少 1 个存储器,第 3 个文字即为临时翻转的变元,值恒为 0。

子句评估模块的输出结果通过计数器对各变元的 break-value 值进行计数和排序,3 个变元评估结果为不满足的子句分别暂存于缓存器中,根据最终翻转的变元将对应缓存器中的子句信息输出到 FIFO 树模块并清除缓存器的内容。变元加强模块主要为一个先入先出队列,搜索时每进行一次变元翻转需要及时更新队列内容,并在求解过程中对将要翻转的变元进行约束。变元加强模块决定最终翻转的变元。

ECWSAT 求解过程如下:

步骤 1:对给定的 DIMACS 数据格式的 CNF 公式,主机提取相关的数据信息。

1)计算各变元的初始指派为正的概率 P_i,并根据概率产生一组初始指派。*Maxtries* 组变元的初始指派被依次存储在片上存储器中。

2)提取地址和子句信息。将地址映射表的 ROM 地址作为变元信息,提取包含对应变元的子句信息。

3)对变元赋不同的初始值,查找不可满足子句。*Maxtries* 组不可满足子句被依次存储在片上存储器中。

步骤 2:求解开始时,地址产生模块随机产生地址,从存储器中输出一个不可满足子句,由于变元的翻转,此子句或许会变成满足,运用子句评估模块 2 对其进行评估,当满足时则重新选择子句。

步骤 3:对输出的不可满足子句的 3 个文字 L_1、L_2、L_3,分别从 RAM 中读出对应变元的值并进行暂时翻转,此时,所有包含 L_1 文字的子句将变为真,但是包含¬ L_1 的子句将可能由真变为假。求解器通过地址映射表查找包含¬ L_1 的子句,评估模块选择出不满足的子句,break-value 计数器计算各变元的 break-value 值并进行排序,按上文所述策略翻转变元,并将翻转的值更新到评估模块的 RAM 中。

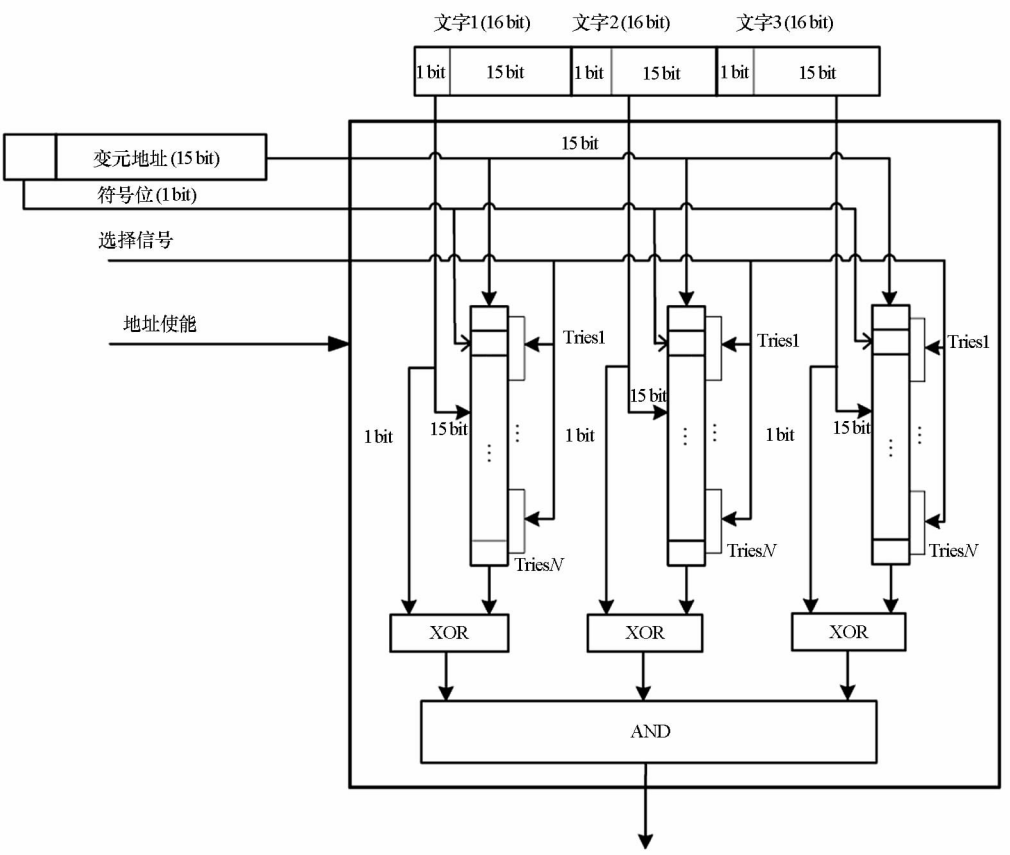

图 2　子句评估模块 2

Fig. 2　Clause evaluator 2

步骤 4:由变元翻转产生的不可满足子句将经过多级 FIFO 进行汇聚,新产生的不可满足子句将被存储在最后一级 FIFO 中,供后续处理。数据写入 FIFO 的顺序是随机的。此时,求解器完成一次变元翻转,整个 SAT 求解过程即为多次变元翻转的过程。

步骤 5:求解过程中同一组初始指派变元翻转的次数由 *Maxflips* 决定,尝试次数由 *Maxtries* 组变元初值和对应的不可满足子句决定,若一组子句求解完成但未找到可满足的解,寻址地址将增加指定偏移量并继续进行下次求解。当执行 *Maxtries* 次仍无解或者找到可满足的解时,算法停止。

4 性能评估

大多数研究是基于实验统计与经验分析的[13]。同样地,本文求解器的性能与改进效果也通过实验进行统计比较与数据分析。

性能评估结果基于 Xilinx Virtex - 6 FPGA (xc6vhx565t)开发板,能处理多达 32 000 个变元/128 000个子句的实例,占用4% 的 slice 和 81% 的片上 RAM/FIFO,并通过求解不同规模的随机 SAT 实例对其性能进行测试。表 1 和表 2 将本文求解器分别与目前最新的软件求解器(WalkSAT Version 51[14])和另一款硬件求解器[15]的性能做了比较。随机选择三个不同规模的测试用例,前两个来源于 SATLIB Benchmark Problems[16],后一个来源于 SAT Competition[17]。WALKSAT Version 51 算法在 Intel(R) Core(TM) i5 32-bit 2.3 GHz CPU 4.0 GB RAM Linux Ubuntu - 14.04 环境下编译,*Maxflips* 取值 30 000 000。依据 SAT 竞赛中的测试标准,每个测试实例测 100 次,取 100 次测试中成功测试的平均运行时间和变元翻转次数作为记录结果。表中,N_c、N_v 和 m 分别表示子句数量、变元数量和子句长度,*#flip* 为找到解时需要翻转的总步数,$\#fps_{avg}$表示平均翻转速率,t 为求解器找到解所需的执行时间。

由表 1 可知,ECWSAT 求解器执行时间均比软件求解器少,系统加速比都有不同程度的提高(1.1 ~ 2.0 倍)。这主要是因为加强了对变元初始指派的约束,并且通过约束搜索过程中的翻转变元,使得算法不是每次都以一种贪心搜索方式来调整变元的赋值,而是根据变元的权值确定需要调整的变元以及调整的方向,一定程度上优化了算法,提高了求解效率。对于表 2 中 sat04 - 910 实例,文献[15]使用了多线程策略。本文求解器为单线程,因此在求解很大规模的 SAT 实例时变元翻转次数相对较多,但是对于其他两个实例,本文求解器均表现出了良好的性能,并且由于未使用片外求解器,搜索过程中不会带来额外的时钟开销。实验表明,本文求解器相对于当前最新的软件求解器和文献[15]硬件求解器来说,求解效率都有明显的提高,若能进行多线程并行处理,求解效率将会有进一步提升。

表 1 WalkSAT 与 ECWSAT 性能比较

Tab. 1 Performance comparison between WalkSAT and ECWSAT

实例	合取范式			WalkSAT[14]			ECWSAT			加速比
	N_v	N_c	m	*#flip*	$\#fps_{avg}$/(Mflip/s)	t/s	*#flip*	$\#fps_{avg}$/(Mflip/s)	t/s	
uf225 - 028	225	960	3	10 907	2.709	0.004	8271	3.94	0.002	1.9
f2000	2000	8500	3	446 781	3.282	0.136	1 130 182	9.36	0.121	1.1
sat04 - 910	22 000	58 740	3	8 136 943	3.320	2.451	6 491 388	5.25	1.237	2.0

表 2 Hardware 与 ECWSAT 性能比较

Tab. 2 Performance comparison between Hardware and ECWSAT

实例	合取范式			Hardware[15]			ECWSAT			加速比
	N_v	N_c	m	*#flip*	$\#fps_{avg}$/(Mflip/s)	t/s	*#flip*	$\#fps_{avg}$/(Mflip/s)	t/s	
uf225 - 028	225	960	3	11 455	2.08	0.006	8 271	3.94	0.002	2.6
f2000	2000	8500	3	3 420 044	6.91	0.495	1 130 182	9.36	0.121	4.1
sat04 - 910	22 000	58 740	3	2 308 837	15.81	0.584	6 491 388	5.25	1.237	0.5

5　结论

本文提出基于 FPGA 的 SAT 求解器,可用于解决大规模的 SAT 问题,算法对各变元的初始指派做了适当的约束,以减少不可满足子句数。求解过程中采用的变元加强策略,避免了同一变元被反复连续翻转。当搜索陷入局部最优时,允许选择和当前候选变元相同质量或者次质量的变元,使算法进入不同的区域,从而跳出局部最优。

评估结果基于 Xilinx Virtex－6 FPGA 开发平台,由于所有的数据存储均使用片上 RAM,使用更大容量的 FPGA 或片外 RAM 将进一步增强求解器求解能力。另外,当不可满足存储器输出的子句满足时,替代子句的选择策略以及求解过程中多线程策略也将影响求解效率。对翻转变元的选取,使用更优化的启发式[18],充分发挥硬件的并行处理能力是未来研究的重点。

参考文献(References)

[1] Skliarova I, de Brito Ferrari A. Reconfigurable hardware SAT solvers: a survey of systems [J]. IEEE Transactions on Computers, 2004, 53(11): 1449－1461.

[2] 马柯帆, 肖立权, 张建民, 等. 基于硬件可编程逻辑的 SAT 求解算法研究与进展[J]. 计算机工程与科学, 2016, 38(4): 634－639.
MA Kefan, XIAO Liquan, ZHANG Jianmin, et al. State of the art and future research of a SAT problem solver on FPGA [J]. Computer Engineering and Science, 2016, 38(4): 634－639. (in Chinese)

[3] Davis M, Logemann G, Loveland D. A machine program for theorem proving [J]. Communications of the ACM, 1962, 5(7): 394－397.

[4] 张建民, 沈胜宇, 李思昆. 基于悖论证明与局部搜索的不可满足子式求解算法[J]. 计算机学报, 2014, 37(11): 2262－2267.
ZHANG Jianmin, SHEN Shengyu, LI Sikun. An unsatisfiable subformula extraction algorithm base on refutation proof and local search [J]. Chinese Journal of Computer, 2014, 37(11): 2262－2267. (in Chinese)

[5] 陈阳志. 基于遗传算法的 SAT 问题求解研究[D]. 哈尔滨: 哈尔滨工程大学, 2014.
CHEN Yangzhi. Research on genetic algorithm-based SAT problem solving [D]. Harbin: Harbin Engineering University, 2014. (in Chinese)

[6] 许可, 李未. SAT 问题的相变现象[J]. 中国科学: E 辑, 1999, 42(4): 354－360.
XU Ke, LI Wei. The SAT phase transition [J]. Science in China: Series E, 1999, 42(4): 354－360. (in Chinese)

[7] Hoos H H. An adaptive noise mechanism for walkSAT[C]// Proceedings of Eighteenth National Conference on Artificial Intelligence, 2002: 655－660.

[8] Achlioptas D. Lower bounds for random 3-SAT via differential equations [J]. Theoretical Computer Science, 2001, 265(1/2): 159－185.

[9] Dubois O, Boufkhad Y, Mandler J. Typical random 3-SAT formulae and the satisfiability threshold[C]//Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, 2000: 126－127.

[10] Crawford J M, Auton L D. Experimental results on the crossover point in satisfiability [J]. Artificial Intelligence, 1996, 81(1/2): 31－57.

[11] 杨晋吉, 苏开乐. SAT 问题中局部搜索算法的改进 [J]. 计算机研究与发展, 2005(1): 60－65.
YANG Jinji, SU Kaile. Improvement of local research in SAT problem [J]. Journal of Computer Research and Development, 2005(1): 60－65. (in Chinese)

[12] Kanazawa K, Maruyama T. Solving SAT-encoded formal verification problems on SoC based on a WSAT algorithm with a new heuristic for hardware acceleration[C]//Proceedings of the IEEE 7th International Symposium on Embedded Multicore/Manycore System-on-Chip, 2013: 101－106.

[13] Kilani Y, Bsoul M, Alsarhan A, et al. A survey of the satisfiability-problems solving algorithms [J]. International Journal of Advanced Intelligence Paradigms, 2013, 5(3): 233－256.

[14] SATLIB Solvers [OL]. [2017－08－05]. http://www.satlive.org/solvers/.

[15] Kanazawa K, Maruyama T. An approach for solving large SAT problems on FPGA[J]. Acm Transactions on Reconfigurable Technology & Systems, 2010, 4(1): 10.

[16] SATLIB Benchmark Problems [OL]. [2017－08－05]. http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html.

[17] SAT Competition [Z/OL]. [2017－08－05]. http://www.satcompetition.org/.

[18] LiC M, Li Y. Satisfying versus falsifying in local search for satisfiability[C]//Proceedings of International Conference on Theory and Applications of Satisfiability Testing, 2012: 477－478.