基于FPGA的图着色问题求解

张益豪[①] 张子超[①] 刘小青[①②] 冷 煌[①②] 王之元[*③] 许 进[①②]

[①](北京大学信息科学技术学院 北京 100871)

[②](北京大学高可信软件技术教育部重点实验室 北京 100871)

[③](国防科技创新研究院人工智能研究中心 北京 100073)

摘 要：图着色问题是在满足相邻顶点不能分配相同颜色且颜色数最少的约束条件下，将图的顶点划分为不相交的集合，且每个集合中的顶点分配相同的颜色。由于图着色问题属于NP-完全问题，求解图着色问题的算法复杂度会随顶点个数的增加呈指数级增长。当顶点个数非常大时，通用处理器求解图着色问题的性能将会显著下降。因此，该文基于现场可编程逻辑门阵列(FPGA)实现求解图着色算法的专用硬件加速器。首先依据FPGA模块化的设计思路提出并实现了基于回溯法的图着色问题求解的硬件架构；其次分析了FPGA内部消耗资源与图着色顶点数之间的关系；最后利用通用异步收发传输器协议实现了通用处理器与FPGA的通信。实验结果表明，相比于在通用处理器上利用软件实现图着色算法，基于FPGA所实现的图着色算法运行时间减少了一个数量级。除此之外，FPGA内部消耗资源数与顶点个数呈线性关系，且每次迭代时FPGA运算所消耗的时间与顶点个数无关。

关键词：图着色问题；回溯法；FPGA

中图分类号：O157.6 **文献标识码**：A **文章编号**：1009-5896(2022)09-3328-07

DOI: 10.11999/JEIT210646

Solution of Graph Coloring Problem Based on FPGA

ZHANG Yihao[①] ZHANG Zichao[①] LIU Xiaoqing[①②] LENG Huang[①②]
WANG Zhiyuan[③] XU Jin[①②]

[①](*School of Electronic Engineering and Computer Science*, *Peking University*, *Beijing* 100871, *China*)

[②](*Key Laboratory of High Confidence Software Technologies*, *Peking University*, *Beijing* 100871, *China*)

[③](*Artificial Intelligence Research Center*, *Defense Innovation Institute*, *Beijing* 100073, *China*)

Abstract: Graph coloring problem divides the vertices of the graph into disjoint sets and the vertices in each set are assigned by the same color under the constraints that adjacent vertices can not be assigned the same color and the number of colors is the smallest. Since graph coloring problem belongs to the class of NP-complete problems, the complexity for solving the graph coloring problem increases exponentially with the number of vertices. The performance of solving graph coloring problem by general-purpose processors decreases significantly when the number of vertices is large enough. This paper implements a dedicated hardware accelerator for solving graph coloring problem based on Field Programmable Gate Array (FPGA). Firstly, by utilizing the rule of FPGA modular design, the hardware architecture for solving graph coloring problem based on the backtracking is proposed and implemented. Secondly, the relationship between the resource consumption of FPGA and the number of vertices is analyzed. Finally, the general-purpose processor and FPGA can communicate through universal asynchronous transmitter-receiver protocol. The experimental results show that the running time of graph coloring algorithm based on FPGA is about an order of magnitude smaller than that of graph coloring algorithm based on software on general-purpose processors. Besides, the resource consumption of FPGA is linear with the number of vertices and the time consumption at each iteration is independent of the number of vertices.

Key words: Graph coloring problem; Backtracking; FPGA

收稿日期：2021-06-25；改回日期：2022-02-28；网络出版：2022-02-19

*通信作者： 王之元 alice_zyw@foxmail.com

基金项目：国家重点研发计划(2019YFA0706401)，国家自然科学基金(61632002, 61872166, 61902005, 62002002)

Foundation Items: The National Key Research and Development Program (2019YFA0706401), The National Natural Science Foundation of China (61632002, 61872166, 61902005, 62002002)

1　引言

图着色问题是计算机科学中的一项重要研究内容，不论在理论还是应用方面均具有重要的研究价值。例如，图着色问题可应用于求解第4代移动通信系统中设备到设备通信的资源分配问题[1]，将大量目标程序分配给少量中央处理器(Central Processing Unit, CPU)中寄存器的寄存器分配问题[2]，在满足一定约束条件下的时间表安排问题等[3]。尽管图着色问题在实际生活中应用广泛，但该问题属于NP-完全问题，因此如何寻找精确算法来求解NP-完全问题一直是近年来研究的热点[4]。然而，NP-完全问题的一个重要特性是如果采用穷举法求解，随着问题规模的增加，时间复杂度将会以指数级增长[5]。因此，一些算法被提出以降低求解图着色问题的时间复杂度，常见的算法包括动态规划、分支定界、整数线性规划以及回溯法。

动态规划算法通过计算并结合子问题的解来降低直接求解原问题的复杂度，但需要消耗一定的空间来存储子问题的解。Lawler[6]首次提出了基于动态规划求解图着色问题的算法，该算法的时间复杂度和空间复杂度分别为$O\left(2.4423^n nm\right)$和$O\left(2^n\right)$，其中$n$和$m$分别是图的顶点数与边数。随后Eppstein[7]和Byskov[8]对Lawler所提出的方法进行了修正，将求解图着色问题的时间复杂度分别降低为$O\left(2.4150^n\right)$和$O\left(2.4023^n\right)$。分支定界法通过枚举所有可能的解来搜索解空间，同时利用剪枝减小搜索的空间从而降低时间复杂度[9]。此外，结合分支定界法与列生成算法，可以将图着色问题建模成为整数线性规划问题并求解[10]。

动态规划、分支定界与整数线性规划均可求得图着色问题的一组解，但有时求得图着色问题的所有可行解是有意义的。例如，当一个图$G=(V,E)$存在一条割边$e=<v_s,v_t>$时，其中$V=\{v_1,v_2,\cdots,v_n\}$是由顶点v_i构成的集合且$E=\{e_1,e_2,\cdots,e_m\}$是由边e_i构成的集合，则$G-e$的两个连通分量可以作为两个图分别进行图着色。此时，由于$G-e$的两个连通分量仅通过割边$e=<v_s,v_t>$连接，因此只要v_s与v_t的颜色不同，就能够得到原图G的一组有效着色方案。若$G-e$的两个连通分量着色完成后v_s与v_t的颜色相同，则可通过置换函数改变任一连通分量的着色方案，使得v_s与v_t的颜色不同。回溯法[11]一般用来求解图着色问题的所有解，其基本思想是以深度优先搜索整个解空间，同时通过剪枝来减小所需搜索解空间的大小。

通常求解图着色问题的算法是在通用处理器上利用软件所实现的。然而，随着顶点规模的增加，通过软件实现图着色算法会造成通用处理器性能的下降以及能耗的提升。尽管已经存在许多软件解决方案来提高求解图相关问题的性能与能量效率，但当前通用处理器的硬件结构仍然是影响性能和能量效率的一个重要因素[12]。现场可编程逻辑门阵列(Field Programmable Gate Array, FPGA)是由可配置逻辑块(Configurable Logic Blocks, CLBs)，输入输出(Input/Output, I/O)模块与可编程的互连资源所构成的可编程逻辑电路[13]。FPGA具有能耗低，开发周期短，易于对设计中出现的错误及时修正，以及适用于实时性系统和分布式系统等特点[14]，因此与图相关的问题能够基于FPGA设计专用硬件电路来求解[15]。

2　图着色问题的回溯算法

回溯法求解图着色问题的思路是对图的所有可能着色方案进行搜索，并判断着色方案是否满足相邻顶点所分配的颜色不同。以未着色的图作为根节点，依据图中顶点与顶点之间的邻接关系和深度优先搜索策略，从根节点出发遍历图中的每个顶点，从而对解空间进行遍历。在每一次访问图中节点的过程中，需要判断当前节点是否存在满足图着色问题约束的着色方案。若存在满足约束的颜色，则对该节点进行着色并从该节点继续遍历未访问的节点；若不存在满足约束的颜色，则逐层向该节点的祖先节点回溯，从而完成剪枝，避免对以该节点为根节点的子树进行搜索。

接下来本文以对图1中的平面图进行四着色为例说明回溯法的流程。在图1中，数字表示顶点标号。利用回溯法，可以得到图1的24种着色方案，如图2所示。详细来说，图2中树的根节点(树的第1层)表示所有的顶点均未着色，树的第i层表示已有$(i-1)$个顶点着色，且连接第i层与第$(i+1)$层边的颜色表示顶点i被分配的颜色。假设先用红色对顶点1进行着色，当对顶点2进行着色时，由于顶点1与顶点2连接，若为顶点2分配红色，则相邻顶点具有相同的颜色，此时不满足图着色问题的约束。因此只能用绿色、紫色、咖啡色之一对顶点2进行着色。若存在所有着色方案均无效的顶点，则返回该顶点的父节点，改变父节点的着色方案。

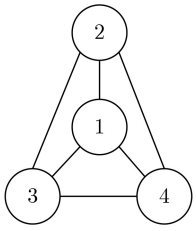

图 1 四顶点平面图

由图2可以看出，图1的四顶点平面图共有24种着色方案。然而，给定任意一种着色方案，剩余的着色方案可以根据置换函数得到。图1的色组划分[16]是唯一的，则其独立着色方案集(其中任意一个方案都不能通过其他着色方案经置换函数得到)仅有1个元素。因此，如果利用回溯法仅求取独立着色方案集，可以减小搜索解空间的大小。除此之外，在FPGA中，存储容量主要由随机存取存储器(Random Access Memory, RAM)的大小决定。FPGA内部的RAM由块RAM(Block RAM)和分布式RAM(Distributed RAM)组成[17]。给定FPGA型号，Block RAM的大小是固定的。Distributed RAM由查找表(Look Up Table, LUT)构成，会占用FPGA内部大量的逻辑资源[18]。因此相比于通用处理器，FPGA内部的存储资源受限[19]。随着图的顶点数逐渐增大，求取所有的四着色方案不仅会增加时间复杂度，也有可能造成需存储解的个数以指数级增长。因此，为了减小在FPGA内部存储所求解的数量，需要求取图的独立着色方案集。

令$K_{n'}$表示n'个顶点的完全图，由于任意两个顶点之间都存在一条边，则要使$K_{n'}$满足图着色问题的约束，至少需要n'种颜色。因此一个S-可着色的图G只能含有$n' \le S$的完全子图$K_{n'}$，其中$n' \le S$。若图G的最大完全子图顶点数为T，则将图G记为G_T。首先利用T种颜色对任一最大完全子图中的T个顶点着色并固定这T个顶点的着色方案。接下来利用回溯法对G_T中剩余顶点进行着色，则能够有效避免图着色问题中解的同构性。综上所述，依据图G的最大完全子图的顶点数，可将图G分为不同的类型。例如，4-可着色图可以分为G_2, G_3, G_4。算法的流程图如图3所示。

3 基于FPGA的图着色实现方案

FPGA通常自顶向下地设计层次化及正则化的

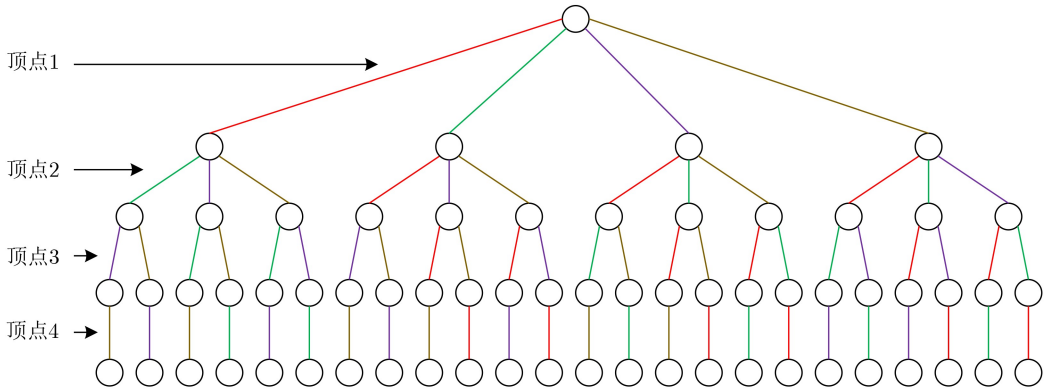

图 2 四顶点平面图的着色方案

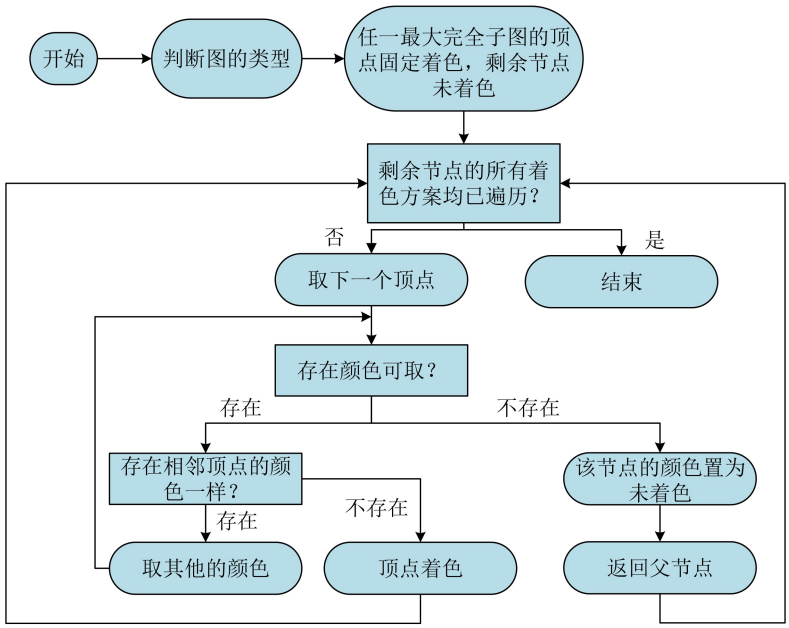

图 3 回溯法算法流程图

模块，其中正则化设计指的是最大化利用已设计的模块以提高模块的复用率。每个模块具有不同的功能，通过上层模块对下层模块的调用来实现最终的功能函数。合理地设计模块划分可以将复杂的问题转化为较小的问题，从而简化设计的复杂度[20]。本文将基于回溯法的图着色问题求解分为3个模块，分别是输入模块、核心计算模块与输出模块，如图4所示。3个模块共享时钟源信号clk与低有效复位信号rst_n。接下来以图的四着色问题为例说明该流程图。

在输入系统中，输入信号data_in由两部分组成：图G的邻接矩阵与图类型G_2, G_3, G_4的表示信号initial_state，其中initial_state是两比特的信号，01, 10与11分别表示G_2, G_3与G_4。图类型表示信号initial_state代表图G的最大完全子图，而求解图G的最大完全子图也属于NP完全问题。为了简化求解的复杂度，本文在通用处理器上采用启发式算法求取图G的最大完全子图。值得注意的是，即使求取图G的完全子图并不是最大的，只会造成FPGA内部所需存储解的个数增加，而不会影响图着色问题的求解。图G的邻接矩阵通过串行的方式传输到输入系统中，输入系统利用一块RAM实现串并转换：RAM中每一个地址ram_addr_rd对应一个顶点，而ram_addr_rd对应的值为与该顶点相邻的信息。即输入系统通过从核心计算系统获得ram_addr_rd来确定当前正在对哪个顶点进行着色，而输入系统根据ram_addr_rd读取RAM中代表该顶点邻域信息的data_internal信号并传送给核心计算系统。输入信号的有效使能data_en用来表明输入数据中哪些信号是有效的。在系统初始化时，核心计算系统处于空闲状态，而当输入模块接收整个邻接矩阵与图类型信号后，用来表示信号已接收完成的一个脉冲信号initial_en被传送给核心计算系统，表明核心计算系统可以开始进行图着色的相关运算。

在核心计算系统中，当接收到initial_en信号

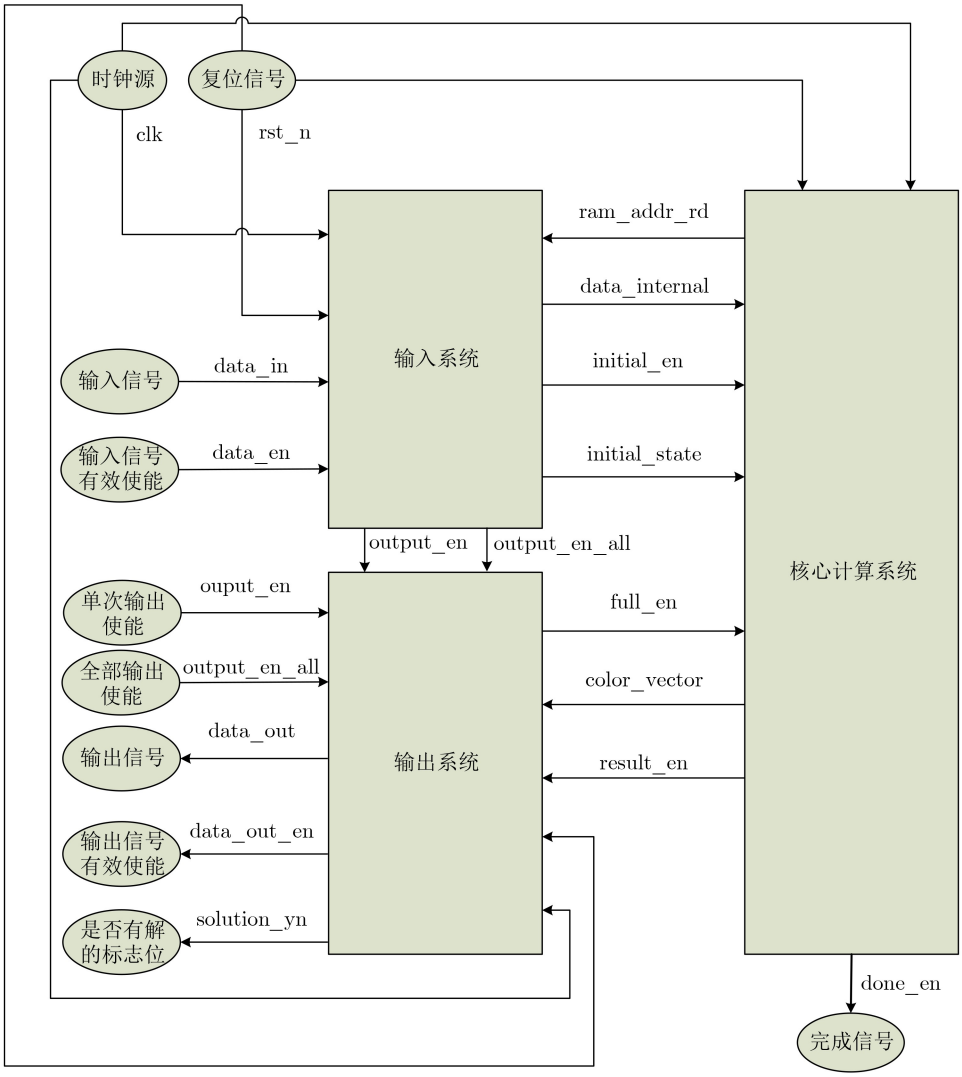

图 4 基于FPGA的图着色实现方案

后，根据图的类型信息initial_state对前T个顶点进行固定的着色初始化。通过ram_addr_rd信号来控制从输入系统中读取哪个顶点的邻域信息。核心系统中增加一个信号v_color_yn表示对应的顶点是否被着色。若未着色则从第1种颜色开始尝试，若已着色则根据当前着色计算下一个可能的着色方案color_next。结合data_internal，v_color_yn和color_next来判断当前顶点的邻接顶点是否存在相同的着色。若不存在则将color_next赋值给包含所有顶点着色方案的信号color_vector，若存在则依据color_next的值决定是否需要回溯。回溯的操作可以通过改变ram_addr_rd的值来回溯到当前节点的父节点。若核心计算系统通过运算得到一种有效的四着色方案时，则将着色方案信号color_vector和表示着色方案有效的使能信号result_en传送到输出系统进行保存。当所有的着色方案完成遍历后，核心运算系统会将done_en信号置为有效位，表明所有运算已结束。若运算已结束并且result_en始终无效，则说明当前图不存在有效的着色方案。由于图着色方案可能存在大量的解，而解的数量无法预先得知。同时，FPGA内部用来存储解的RAM是预先固定好的，且FPGA内部能够用来存储的RAM 容量受限。因此当输出系统无法保存新的着色方案时，需要置full_en有效，使核心系统停止当前的运算，并保存当前运算的信息。待输出系统中的RAM有新的存储空间接收着色方案时，置full_en无效，此时核心计算系统继续进行之前的运算。注意到回溯法中每一次迭代的主要步骤是验证当前节点的着色与其邻接节点的着色是否相同，该验证可以通过或运算实现。即只要存在一个邻接顶点的着色与当前节点的着色相同，则当前节点无法着色，需要改变着色方案或进行回溯。

在输出系统中，result_en信号用来判断当前着色方案color_vector是否有效。若有效则将color_vector存储在RAM中，若当前RAM中存储的着色方案个数已达到最大值，则置full_en有效，使得核心计算系统停止当前运算。根据输入使能信号output_en与output_en_all来决定是否将RAM中的着色方案输出，其中output_en 1次只输出1个有效着色方案，而output_en_all一次性将所有着色方案全部输出。RAM中存储的着色方案通过并串转换生成data_out信号输出，使能信号data_out_en用来表明当前输出是否有效。注意到图着色问题可能无解，因此本文利用信号solution_yn表示该图是否存在有效的着色方案。

4 硬件实现

本文在型号为EP4CE6E22C8的FPGA上，利用Verilog语言实现了基于回溯法的图的四着色问题求解。在Altera平台的Quartus Prime 18.0的开发环境中经过综合后，得到图的顶点个数与FPGA内部资源消耗的关系，如表1所示。在表1中，本文采用逻辑元件(Logic Element, LE)和寄存器的个数来表示FPGA内部的资源。LE是构成FPGA中CLB模块的重要单元，通常用来实现逻辑电路[21]。不同的FPGA型号会有不同数量的LE，例如EP4CE6E22C8仅有6000个LE，而Stratix GX10M拥有10200000个LE[22]。由表1可以看出FPGA内部的资源LE与寄存器均基本与图的顶点个数成正比，造成这种现象的主要原因是表示顶点是否被着色的信号v_color_yn与顶点着色方案信号color_vector所需的比特数与顶点数成正比，且验证当前节点着色方案与邻接节点着色是否相同是通过或运算实现的，而或运算的操作数是color_vector。表1的资源消耗主要用于实现图4的输入系统、核心计算系统与输出系统。但在输入输出系统中并不包含用于和外部通信的传输协议模块。

接下来以12顶点的图四着色为例，分析基于FPGA的图着色算法性能。进行静态时序分析后，得到当FPGA内核电压为1.2 V，温度为85°C时，系统允许工作的最大时钟频率为139.65 MHz，而当FPGA内核电压为1.2 V，温度为0°C时，系统允许工作的最大时钟频率为147.17 MHz。由图3的回溯算法流程图可知，算法的运行复杂度由改变顶点着色方案的次数和每次迭代时判断能否着色所消耗的时间共同决定。由于在FPGA平台上实现图着色算法不能减小改变顶点着色方案的次数，因此仅需考虑回溯法中每次迭代所消耗的时间。图5可以看出每次color_next的改变最多仅需10个时钟周期，注意到每次顶点着色方案的改变所需时间周期并不

表 1 FPGA资源消耗

	图的顶点个数						
	12	16	32	64	128	256	512
LE	499	547	717	1045	1700	2997	5580
寄存器	262	323	439	667	1119	2019	3815

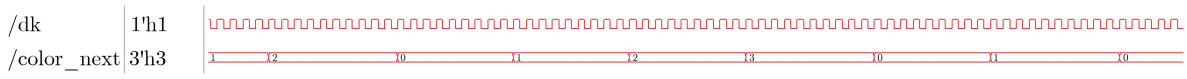

图 5 顶点着色方案验证的功能仿真

$$\boldsymbol{A}=\begin{bmatrix}0&1&1&1&1&0&1&0&0&0&0&0\\1&0&1&0&1&1&0&0&0&0&0&0\\1&1&0&0&0&1&1&0&0&0&0&0\\1&0&0&0&1&0&1&1&1&0&1&0\\1&1&0&1&0&1&0&0&1&0&0&0\\0&1&1&0&1&0&1&1&1&1&0&1\\1&0&1&1&0&1&0&0&0&0&1&1\\0&0&0&1&0&1&0&0&1&1&1&0\\0&0&0&1&1&1&0&1&0&0&0&0\\0&0&0&0&0&1&0&1&0&0&1&1\\0&0&0&1&0&0&1&1&0&1&0&1\\0&0&0&0&0&1&1&0&0&1&1&0\end{bmatrix}$$

图 6 12顶点图的邻接矩阵

固定，这是因为除了判断当前着色方案是否有效外，还需根据是否回溯改变ram_addr_rd以选取当前着色的顶点。因此，若FPGA内部使用100 MHz的时钟，则每次迭代所消耗的时间为10/100=0.1 μs。同时，在主频为3.0 GHz的通用处理器中，利用Matlab对1000组12顶点的图进行四着色，平均每次迭代所消耗的平均时间为5.397 μs。由此可见，相比于在通用处理器中采用软件实现图着色算法，基于FPGA实现的图着色算法执行速度更快。若采用性能更高的FPGA，可以进一步提高系统允许工作的最大时钟频率，从而减小基于回溯法求解图着色问题的时间。

随着顶点个数的增加，在每次迭代中判断当前着色方案是否有效所需的运算量增加。具体来说，利用3个周期的或运算来判断当前顶点的着色方案是否与邻接节点着色方案相同。本文基于空间换时间的思想，通过增加单个周期内或运算的个数，保证每次迭代时FPGA内部运算所需时钟周期个数与顶点的个数无关。因此随着顶点个数的增加，每次color_next的改变仍然仅需10个周期，且每次迭代时FPGA执行运算所消耗的时间与顶点个数无关。而随着顶点个数的增加，每次迭代时软件实现方案会消耗更多的时间。例如当图的顶点个数为512时，利用Matlab实现每次迭代所消耗的时间为15.611 μs。

为了验证所实现硬件的正确性，本文基于通用异步收发传输器协议实现了FPGA与通用处理器之间的通信。对具有如图6所示邻接矩阵$\boldsymbol{A}$的12顶点图G进行着色。根据该图的类型，通用处理器需将146 bit信息传输到FPGA中，其中前144 bit为邻接矩阵$\boldsymbol{A}$的1维表示，最后2 bit信息是表示图类型的initial_state信号。基于16进制将该146 bit表示为961186010D2B29AA59029BB2E582DD24622D01，并将该信息通过串口传输到FPGA内部进行运算。通过将FPGA运算的结果与通用处理器中图着色的运算结果对比，验证了FPGA内部输出的解是具有邻接矩阵$\boldsymbol{A}$的图G的所有独立着色方案集。

5　结束语

图着色问题一直是近年来研究的热点问题，本文基于FPGA利用回溯法实现了图着色问题的求解。首先利用最大完全子图固定了部分顶点的着色，使得图着色问题的解是独立着色方案集。随后在Altera公司的EP4CE6E22C8芯片上设计并实现了基于回溯法的图着色问题求解。最后的实验结果表明，基于FPGA的图着色算法在每次迭代时，判断能否着色所消耗的时间小于在通用处理器中利用软件实现迭代所需的时间，并且FPGA内部所消耗的资源与图的顶点个数成正比。

参 考 文 献

[1] GUO Jianding. Theoretical research on graph coloring: Application to resource allocation in device-to-device 4G radio system (LTE)[D]. [Ph. D. dissertation], Université Bourgogne Franche-Comté, 2018.

[2] ELUMALAI A. Graph theory applications in computer science and engineering[J]. *Malaya Journal of Matematik*, 2020, S(2): 4025–4027. doi: 10.26637/MJM0S20/1043.

[3] PODDAR N and MONDAL B. An instruction on course timetable scheduling applying graph coloring approach[J]. *International Journal of Recent Scientific Research*, 2018, 9(2): 23939–23945. doi: 10.24327/ijrsr.2018.0902.1567.

[4] DE LIMA A M and CARMO R. Exact algorithms for the graph coloring problem[J]. *Revista de Informática Teórica e Aplicada*, 2018, 25(4): 57–73. doi: 10.22456/2175-2745.80721.

[5] VENKATASUBRAMANIAN M. Failure evasion: Statistically solving the NP complete problem of testing difficult-to-detect faults[D]. [Ph. D. dissertation], Auburn University, 2016.

[6] LAWLER E L. A note on the complexity of the chromatic number problem[J]. *Information Processing Letter*, 1976, 5(3): 66–67. doi: 10.1016/0020-0190(76)90065-X.

[7] EPPSTEIN D. Small maximal independent sets and faster exact graph coloring[J]. *Journal of Graph Algorithms and Applications*, 2003, 7(2): 131–140. doi: 10.7155/jgaa.00064.

[8] BYSKOV J M. Chromatic number in time $O(2.4023^n)$: Using maximal independent sets[R]. BRICS Report Series RS-02-45, 2002.

[9] BRÉLAZ D. New methods to color the vertices of a graph[J]. *Communications of the ACM*, 1979, 22(4): 251–256. doi: 10.1145/359094.359101.

[10] MEHROTRA A and TRICK M A. A column generation approach for graph coloring[J]. *Informs Journal on Computing*, 1996, 8(4): 344–354. doi: 10.1287/ijoc.8.4.344.

[11] MARAPPAN R and SETHUMADHAVAN G. Complexity analysis and stochastic convergence of some well-known evolutionary operators for solving graph coloring problem[J]. *Mathematics*, 2020, 8(3): 303. doi: 10.3390/math8030303.

[12] GUI Chuangyi, ZHENG Long, HE Bingsheng, *et al*. A survey on graph processing accelerators: Challenges and opportunities[J]. *Journal of Computer Science and Technology*, 2019, 34(2): 339–371. doi: 10.1007/s11390-019-1914-z.

[13] BROWN S. FPGA architectural research: A survey[J]. *IEEE Design & Test of Computers*, 1996, 13(4): 9–15. doi: 10.1109/54.544531.

[14] CHINNERY D and KEUTZER K. Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design[M]. Boston, USA: Springer, 2002: 1–414. doi: 10.1007/b105287.

[15] BESTA M, STANOJEVIC D, DE FINE LICHT J, *et al*. Graph processing on FPGAs: Taxonomy, survey, challenges[EB/OL]. https://arxiv.org/pdf/1903.06697.pdf, 2021.

[16] 许进. 极大平面图理论(上册: 结构-构造-着色)[M]. 北京: 科学出版社, 2019.
XU Jin. Theory of Maximal Planar Graph (Volume One: Structure-Construction-Colorings)[M]. Beijing: Science Press, 2019.

[17] FAZAKAS A, NEAG M, and FESTILA L. Block RAM versus distributed RAM implementation of SVM Classifier on FPGA[C]. Proceedings of 2006 International Conference on Applied Electronics, Pilsen, Czech Republic, 2006: 43–46. doi: 10.1109/AE.2006.4382960.

[18] JUNGK B and STÖTTINGER M. Serialized lightweight SHA-3 FPGA implementations[J]. *Microprocessors and Microsystems*, 2019, 71: 102857. doi: 10.1016/j.micpro.2019.102857.

[19] CLIFFORD W. Introduction to FPGA acceleration[EB/OL]. https://www.stemmer-imaging.com/media/uploads/websites/documents-/tech-tips/en_GB-TechTip-_Intoduction-to-FPGA-acceleration-20110211.pdf, 2021.

[20] TRIMBERGER S, ROWSON J, LANG C, *et al*. A structured design methodology and associated software tools[J]. *IEEE Transactions on Circuits and Systems*, 1981, 28(7): 618–634. doi: 10.1109/TCS.1981.1085035.

[21] LEWIS D, AHMED E, BAECKLER G, *et al*. The stratix II logic and routing architecture[C]. Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2005: 14–20. doi: 10.1145/1046192.1046195.

[22] Intel stratix 10 GX/SX product table[EB/OL]. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/pt/stratix-10-product-table.pdf, 2021.

张益豪：男，博士，研究方向为信号处理、非凸优化、组合优化.

张子超：男，博士生，研究方向为社交网络、图神经网络与生物信息网络等.

刘小青：女，特聘副研究员，研究方向为图论与组合优化、新型计算.

冷 煌：男，特聘副研究员，研究方向为新型计算、生物计算、组合优化.

王之元：女，副研究员，研究方向为智能模型、智能算法.

许 进：男，教授，研究方向为图论与组合优化、生物计算机、社交网络与信息安全等.

责任编辑：陈 倩