面向功能向量的并行逻辑模拟

赖李洋[1)], 杨玉鑫[2)], 李华伟[2)], 林晓泽[1)]

[1)] (汕头大学电子信息工程系 汕头 515063)
[2)] (中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)
(lylai@stu.edu.cn)

摘 要: 为了提高数字系统中功能向量的模拟效率并获得良好的性能扩展性, 提出一种并行逻辑模拟算法. 首先针对异质计算中图形显卡和多核微处理器的架构特性, 结合同层级逻辑门仿真的任务并行性和竞争包容的事件管理机制; 然后采用基于位操作的门运算、基于时帧的模拟以及基于 VCD 的功能向量进行逻辑仿真. 在超百万门的开源电路 openMSP 和 ISCAS89 上的实验结果表明, 对比单线程算法, 功能向量模拟可以获得 8.9~86.8 的性能加速比, 且加速比与电路的规模成正比.

关键词: 并行逻辑模拟; 功能向量仿真; 异质并行计算
中图法分类号: TP391.41 **DOI:** 10.3724/SP.J.1089.2023.19443

Parallel Logic Simulation for Functional Test

Lai Liyang[1)], Yang Yuxin[2)], Li Huawei[2)], and Lin Xiaoze[1)]

[1)] (*Department of Electronics and Information Engineering, Shantou University, Shantou* 515063)
[2)] (*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

Abstract: To improve the efficiency of functional test simulation in digital systems with good performance scalability, an algorithm of parallel logic simulation is proposed. By tailoring to the architectural feature of graphics cards and multi-core microprocessors in heterogeneous computing, the algorithm combines task parallelism in gate evaluation at the same level and a race-tolerant mechanism for event management. It makes use of bitwise logical operation-based gate evaluation, frame-based simulation, and VCD-based functional test stimulus. Experiments are conducted on open-source circuits of openMSP and ISCAS89 with over one million gates. Compared to the single-threaded algorithm, a performance speedup of 8.9 to 86.8 can be achieved; moreover, the speedup is proportional to the size of the circuit.

Key words: parallel logic simulation; functional test simulation; heterogeneous parallel computing

作为芯片设计自动化中的关键技术之一, 逻辑模拟[1]在芯片设计流程的各个环节都有广泛的应用, 如前端设计、逻辑验证、物理验证和自动测试向量生成(automatic test pattern generation, ATPG)等. 随着晶圆制程的进步和芯片设计方法学的发展, 高端芯片集成的晶体管数目越来越多, 系统模块构成也越来越复杂, 导致逻辑模拟所需的时间呈现爆炸式成长, 尤其是在设计验证阶段. 在电子

收稿日期: 2021-11-01; 修回日期: 2022-11-24. **基金项目**: 国家自然科学基金(62090024); 广东省自然科学基金(2022A1515011084); 广东省扬帆计划紧缺拔尖人才项目(140-14600602); 计算机体系结构国家重点实验室开放课题(CARCH201912, 140-15220011). **赖李洋**(1975—), 男, 博士, 副教授, 硕士生导师, CCF 会员, 主要研究方向为集成电路可测试性设计、电子设计自动化、容错计算、芯片设计; **杨玉鑫**(1995—), 男, 硕士, 主要研究方向为硬件安全; **李华伟**(1974—), 女, 博士, 研究员, 博士生导师, CCF 会员, 主要研究方向为数字电路设计自动化、测试验证、智能计算、近似计算; **林晓泽**(1996—), 男, 硕士研究生, 主要研究方向为电子设计自动化.

设计自动化(electronic design automation, EDA)领域，通过算法创新不断提高逻辑模拟效率以满足芯片设计的迫切需求，一直是学术界和产业界孜孜不倦的追求和所面临的持续挑战.

并行逻辑模拟[1-2]的发展历史已有数十年，但应用在实际的工业级电路上仍有一定的难度. 商用工具中成熟的并行逻辑模拟是基于电路分区[2]的多核多线程模拟，而合理、高效的电路分区需要架构设计工程师的深度介入，实现成本很高.

逻辑模拟分为顺序性模拟和事件驱动模拟[1]，已有学者提出其相应的并行模拟策略. 当然，模拟的电路对象可以经过一定的优化，如代码编译. 在顺序性模拟中，所有的逻辑门都根据其拓扑排序依次仿真一遍，在拓扑排序中处于同一层级的逻辑门，它们之间由于没有数据依赖关系，可以并行仿真. 在逻辑模拟中，大多数情况下只有极小比例的逻辑门会有仿真值的改变，当一个逻辑门的输入值不变时，其输出是保持稳定的. 因此，在顺序性模拟中，大部分的逻辑门仿真是不必要的，称之为冗余计算. 为了解决顺序性模拟中的冗余计算问题，人们提出事件驱动模拟，即只仿真输入值有改变的逻辑门. 事件驱动模拟的主流实现形式是同步事件驱动模拟，同步指的是逻辑门的仿真顺序需要与全局的仿真时间轮同步. 随着时间轮中的时间印章向前推进，各个时间印章点上的逻辑门被依次仿真，如果是零延时模拟，则用逻辑门的层级替代时间印章. 与顺序性模拟类似，在事件驱动模拟中，同一时间印章或层级上的逻辑门可以进行并行仿真，而不同时间印章或层级上的逻辑门则需要根据其在时间轮上的先后顺序进行仿真. 事件驱动模拟并行算法的难点在于其全局的事件队列，所有并行的线程需要在运行过程中读取和更新这个队列，这就导致了并行计算中众所周知的数据竞争问题. 为了避免数据竞争，需要采用特别的原子操作或数据锁等机制，以保证单一时间只有一个线程可以访问共享数据，但这些操作会极大地影响并行算法的性能.

由于计算机架构的限制，早期的并行逻辑模拟基本上采用多处理器多节点分布式计算[1-2]，但实际效果对比优化后的单线程逻辑模拟并没有太大的提升. 随着异质并行计算架构[3-6]近 10 余年的飞速发展，单个芯片上可以集成成百上千个运算核，如高性能图形处理器(graphics processing unit, GPU)和各类定制加速芯片已被广泛应用于机器学习、科学计算等领域，这也为并行逻辑模拟带来了新的发展机遇. Chatterjee 等[7-8]将电路网表分割成多个逻辑门簇，且允许逻辑门簇之间的逻辑门有一定的重叠，在逻辑门簇分区的基础上，分别实现了基于 GPU 的顺序性模拟并行算法[7]和粗粒度的事件驱动模拟并行算法[8]，但冗余计算和数据竞争的问题并没有得到解决；Sen 等[9]的并行逻辑模拟与文献[7]类似，但电路网表使用了一种不同的等价转化；Zhu 等[10]实现了一个基于 GPU 的异步并行逻辑模拟器，由于异步逻辑模拟的内存分页和数据同步开销过大，该模拟器并不适用于大规模的工业电路. 以上工作的一个共同点是它们的实验对象都是小于 100 万门的逻辑电路，不足以说明所提算法对大规模工业电路的适用性.

为了解决逻辑模拟并行算法中的冗余计算和数据竞争问题，文献[11]提出一种混合式并行逻辑模拟算法，其有机地结合并行顺序模拟和并行事件驱动模拟各自的优势，同时避免了各自的缺点，是已知文献中首次在 GPU 上实现千万门级工业电路的逻辑模拟工作，且获得了优异的性能加速. 但该算法以芯片结构测试向量为对象，不同结构测试向量的仿真相互独立，其逻辑模拟大量使用位并行性以实现单线程模拟中的多向量并行仿真，这一点与功能向量仿真是不一样的.

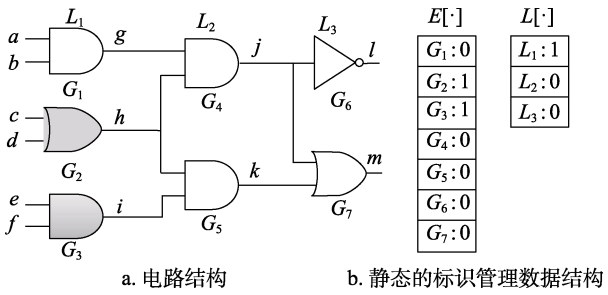
本文在文献[11]的基础上，根据功能向量的特点对混合式并行逻辑模拟算法进行改进和优化. 首先实现了同时适用于多核处理器(central processing unit, CPU)和 GPU 的面向功能向量仿真的并行逻辑模拟，证明该算法的架构普适性；其次通过对逻辑仿真值的特殊编码，设计一种新颖的、基于逻辑位操作的逻辑门仿真方法，获得了与快速查表法类似的门仿真性能；最后通过实验验证了并行逻辑模拟更适用于大规模的数字电路(超百万门甚至千万门)，且电路规模越大性能加速越好.

1 混合式并行逻辑模拟算法和架构适配

1.1 混合式并行逻辑模拟算法

混合式并行逻辑模拟算法[11]突破了已知并行顺序模拟和并行事件驱动模拟算法的局限性，具有较好的可并行性和扩展性，非常适用于拥有大量运算单元的并行计算架构.

图 1a 用于解释混合式并行逻辑模拟算法的基本思想. 这是一个简单的逻辑电路，共有 G_1~G_7 这 7 个逻辑门，a~f 为输入，l~m 为输出；电路通过拓扑排序分 3 个层级 L_1~L_3.



a. 电路结构　　b. 静态的标识管理数据结构

图 1　混合型事件驱动模拟的简单实例

混合式并行逻辑模拟算法的关键在于引入静态的标识管理数据结构，即 2 个静态一维数组 $E[\cdot]$ 和 $L[\cdot]$，如图 1b 所示；其中，数组 $E[\cdot]$ 的大小同电路的逻辑门数相同，$L[\cdot]$ 的大小则与电路的层级数相同. 如果电路层级 l 的标识 $L[l]$ 是 1，则表明这个层级上有逻辑门需要被计算；如果标识 $L[l]=0$，则该层级的所有逻辑门都可以略过. 一个层级上具体哪些逻辑门需要被计算，由 $E[\cdot]$ 来决定. 如果逻辑门 g 的标识 $E[g]$ 是 1，则表明这个逻辑门需要被模拟；如果 $E[g]$ 是 0，则不需要. 混合型模拟与顺序性模拟同样是逐级进行的. 在图 1b 中，假定层级 L_1 上有 $L[L_1]=1$ $E[G_1]=0$ $E[G_2]=1$ $E[G_3]=1$. 由于当前 $L[L_1]=1$，3 个并行线程 T_1，T_2 和 T_3 会被分别用于处理逻辑门 G_1，G_2 和 G_3，并随后将 $L[L_1]$ 重置为 0. 线程 T_1 检查了 $E[G_1]$ 后会即刻退出；线程 T_2 和 T_3 则会对 G_2 和 G_3 进行重新计算，并将 $E[G_2]$ 和 $E[G_3]$ 重置为 0. 假定 G_2 和 G_3 有了新的逻辑值，那么 T_2 需要进一步设置 $E[G_4]=1$ $E[G_5]=1$ 和 $L[L_2]=1$；而 T_3 则需要操作 $E[G_5]=1$ 和 $L[L_2]=1$. 其中，T_2 和 T_3 出现了数据竞争：两者需要同时对 $E[G_5]$ 和 $L[L_2]$ 进行写 1 操作. 但这是“安全”的数据竞争，因为不论哪个线程先完成操作，最终的结果都是正确的，即竞争鲁棒性. 竞争鲁棒性的出现，使混合型事件驱动模拟算法的并发线程可以独立地运行，消除了事件驱动模拟中的数据竞争瓶颈；而静态标识管理数据结构的应用，解决了并行顺序性模拟的冗余计算问题.

总体而言，混合型并行逻辑模拟通过巧妙的数据结构和算法设计，有机结合并行顺序性模拟和并行事件驱动模拟的各自优势，同时避免了两者的缺点，使可扩展的基于任务并行性的事件驱动模拟成为可能.

1.2　算法复杂度

逻辑模拟属于线性复杂度的算法. 假定数字电路的逻辑门数为 n，需要模拟的时钟周期数为 c，则其算法复杂度为 $O(c\times n)$. 对于大规模数字芯片来说，n 可以是上千万甚至上亿；c 至少是数百万，甚至上亿. 因此，逻辑模拟的主要问题在于超大规模的数据集，即便是线性复杂度的算法，其运行时间也是极其可观的. 逻辑模拟算法的性能优化方向是降低需要计算的逻辑门数目，以及提升算法的可并行性和性能扩展性.

1.3　算法的架构适配性

虽然文献[11]的工作是在 GPU 上实现的，但混合式并行逻辑模拟算法本身并没有对硬件的并行架构有特别的限制. 因此，理论上该算法可以运行在 GPU、多核 CPU 和各类定制加速芯片等不同并行架构上. 当然，由于架构的差异性，其运行效率有所不同，本文对所采用的多核 CPU 和 GPU 架构做一些定性的分析，表 1 为这 2 种并行架构的基本特征对比.

表 1　多核 CPU 和 GPU 的基本特征对比

并行架构	多核 CPU	GPU
运算核心数	几个至几十个	数百至上万
单核性能	强	较弱
内存大小	几 GB 至几 TB	几 GB 至 48 GB
共享内存	是	是
高速缓存的支持	有	非常有限
虚拟内存的支持	有	无
代码路径一致性要求	无	有
同步机制(效率)	共享内存(高)	共享内存(高)

多核 CPU 的核心数通常为几个至几十个，要远小于 GPU，如 2022 年发布的英伟达高端 GPU 有超过 1 万的核心数，但 CPU 的单核性能要远强于 GPU. 多核 CPU 往往比 GPU 有更大的内存，如专业工作站可以有几 TB 的内存，而最高端的 GPU 的内存大小也只有 48 GB；而且多核 CPU 系统有完善的虚拟内存支持，能够支持超大数据集的应用. 因此，在对 GPU 的应用中，为了最大限度地发挥其海量运算核心的优势，需要将其数据集控制在 GPU 显存可以容纳的范围内. 必须特别指出的是，这 2 种架构对不同代码路径的支持的差异性. 多核 CPU 中的每个 CPU 核都有其独立的指令控制逻辑，并行线程可独立运行在不同的指令路径上；而 GPU 则是多个运算核共享一个指令控制逻辑，如英伟达的 GPU 是 32 个核共享一个指令控制逻辑，这就要求并行线程尽可能地在相同的代码路径上运行，否则不同路径的代码其实是串行执行的，将极大地影响并行算法的加速性能.

对于混合式并行逻辑模拟算法，随着电路规模的增大，每个电路层级上的逻辑门数也变得更多，极端的情况下，一个层级可以有超百万个逻辑门，而其中真正需要仿真的逻辑门只占很小比例. 面对这种情况，GPU 上大量的运算核心将更有利于算法中对 $E[\cdot]$ 数组的快速访问和逻辑门的仿真. 因此，对于大规模的数字芯片，混合式并行逻辑模拟在拥有海量运算核心的并行架构上更具有优势. 这个分析同样适用于各类集成了大量运行核的定制加速芯片，并且如果加速芯片的运算核是独立的 CPU 核，如 ARM 核，那么其算法性能会有进一步的提升.

2 面向功能向量的算法优化

2.1 逻辑门运算优化

在结构测试向量(也称扫描向量)的逻辑模拟中，不同测试向量是相互独立的，可以充分利用单线程中的位并行性. 每个逻辑门的仿真值 $\{0,1,\mathrm{X},\mathrm{Z}\}$ 使用 2 个 64 位长整数，表示 64 个测试向量的仿真值，因此，一次逻辑门的运算实际上是模拟了 64 个独立测试向量的仿真结果.

功能向量本质上是单一的时序向量，在时间轴的不同点上有不同的向量输入值，不同时间点的输入值不能进行并行仿真. 因为电路内部节点的逻辑值依赖前面输入的仿真结果，所以结构测试向量模拟中的位并行方法不再适用于本文.

在功能向量仿真中，每次逻辑门的运算只针对一个向量，常用的方法是查表法，即将不同类型逻辑门的真值表存起来，每次计算的时候通过查表的方式得到输出值. 本文采用一种新的基于逻辑运算的门运算方法，不再依赖真值表，但所获得的性能和查表法的相差无几. 其基本思路是首先对 4 值逻辑 $\{0,1,\mathrm{X},\mathrm{Z}\}$ 采用一种特殊的 8 位无符号字符编码，如表 2 所示；然后根据逻辑门类型，采用简单的逻辑运算计算出逻辑门的输出值.

表 3 为 5 种常见逻辑门的运算实例. 其中，符号 ~，& 和 | 分别是 C 语言中的位非、位与和位或逻辑运算；A，A_1，A_2 代表逻辑门的输入值；函数 swap_hi4_lo4(·) 表示将 8 位字符变量的高 4 位和低 4 位进行交换. 注意，逻辑值中的 Z 值只会在三态门和总线出现，其他的逻辑门仿真实际上只需要考虑 $\{0,1,\mathrm{X}\}$ 这 3 个值. 以逻辑门 NAND 为例，为了计算其输出值，只需将其所有输入值作 & 操作，并将所得值的高 4 位和低 4 位交换即可.

表 2 4 值逻辑的无符号字符编码

逻辑值	编码
0	0x00
1	0xFF
X	0x0F
Z	0xF0

表 3 5 种常见逻辑门的运算实例

逻辑门	运算方法
NOT(A)	swap_hi4_lo4$(\sim A)$
AND(A_1,A_2)	$A_1 \,\&\, A_2$
NAND(A_1,A_2)	swap_hi4_lo4$(A_1 \,\&\, A_2)$
OR(A_1,A_2)	$A_1 \mid A_2$
NOR(A_1,A_2)	swap_hi4_lo4$(A_1 \mid A_2)$

2.2 基于时帧的逻辑模拟

在很多工作中对时序电路的逻辑模拟往往只考虑基于周期的模拟，即组合逻辑在每个时钟周期只仿真一次，触发器在当前周期捕获的新值会用在下一个周期的仿真中. 但这种方法只适用于理想化的时序电路模型，即只有单一类型的触发器，且所有触发器的有效时沿均是一致的，都是上升沿触发或是下降沿触发. 而实际的数字芯片要比理想模型复杂得多，除了上升沿有效和下降沿有效的触发器会混合使用之外，还有不少电平敏感的锁存器会被用在时钟控制逻辑中.

如图 2 所示，电路中有 2 个 D 触发器 DFF_1 和 DFF_2. 其中，DFF_1 为上升沿触发，DFF_2 为下降沿触发. 在时钟的上升沿，节点 a 的值会传送到 b；在时钟的下降沿，b 的值会传送到 c. 即在一个时钟周期内 a 的值会传送到 c，按照理想时序电路模型的仿真无法得到该结果. 为了保证逻辑模拟的准确性，本文使用基于时帧的仿真，每次时钟信号的变化都会产生一个新的事件，触发电路进行一

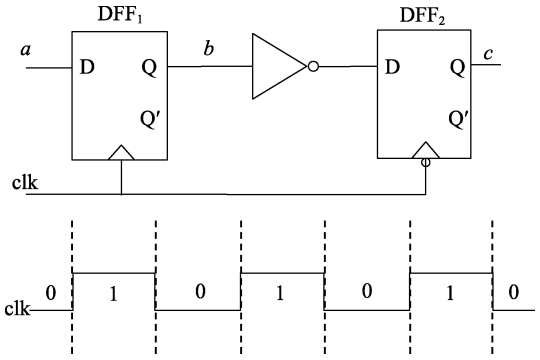


图 2 混合边沿触发的时序电路实例

轮新的事件驱动模拟, 则不同边沿的触发器和锁存器都可以被正确地仿真, 因此能够处理真实的工业电路.

2.3　功能向量加载

在功能向量模拟中, 实现测试环境的搭建和加载是非常重要的. 本文采用被测电路的顶层模块边界信号的 VCD(value change dump)文件作为逻辑模拟器的输入. VCD 文件是数字电路模拟中用于记录指定电路节点信号值变化的文件, 商用的逻辑模拟工具都支持在逻辑模拟过程中输出 VCD 文件. 因此, 使用 VCD 文件可以保证功能向量输入的正确性.

由于功能向量可以持续数万甚至上百万的时钟周期, 因此 VCD 文件也会因此变得非常大. 当逻辑模拟在 GPU 上运行时, 功能向量数据需要通过 PCIe 总线从 CPU 内存上传输到 GPU 显存上, 如图 3 所示.

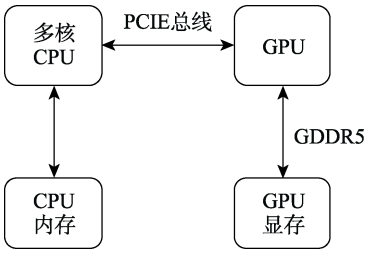


图 3　多核 CPU 和 GPU 的数据传输

由于 GPU 内存有限, 无法一次放下整个 VCD 文件的数据内容, 因此, VCD 的数据内容需要按时间顺序进行切割. 一次只向 GPU 传送一部分功能向量内容, 当这部分功能向量模拟完成后, 再传送下一部分的向量. 与文献[11]不同, 本文并没有把功能向量的传输和 GPU 的 Kernel 运算进行重叠操作, 因为数据传输时间并不是功能向量模拟的性能瓶颈, 其只占整体运行时间的很小一部分. 这样做的另外一个好处是, 相同的算法代码路径(数据传输除外)可以同时支持在 GPU 上和在多核 CPU 的并行逻辑模拟.

3　实验及结果分析

本文所有实验都是在一台 Ubuntu 14.04 Linux 服务器上运行的: CPU 为 Intel Xeon E5-2630 v4 2.20 GHz, 10 核 20 线程; GPU 为 NVIDIA GeForce GTX 1080Ti, 3584 CUDA 核, 11 GB 显存; 系统支持 OpenCL 1.2, 且安装有 Synopsys 公司的商用 VCS 逻辑模拟器.

本文基于多核 CPU 和 GPU 的功能向量并行逻辑模拟算法采用 OpenCL[6]实现. 为了验证并行模拟算法的性能加速, 实现了 CPU 单线程的事件驱动逻辑模拟算法. OpenCL 是一个开源的异构并行计算平台, 其最大优势是可以支持不同厂商的多种类型的并行计算器件, 包括 Intel 和 AMD 的多核 CPU, NVidia, AMD, Intel 的显卡, 以及 Xilinx 的 FPGA 等. 因此, 本文采用 OpenCL 可以最大限度地实现不同异构器件之间的代码共享, 大幅提高算法实现效率.

3.1　实验电路

实验中选用 ISCAS89[12]电路 s38417 和 s38584, 以及开源的 IP 核 openMSP430(简称 openMSP). 所有电路都使用 Synopsys 的 Design Compiler 进行逻辑综合, 采用 SMIC18 标准单元库. 考虑到现实中难以获得开源的超百万门级的工业电路, 为了验证并行逻辑模拟算法对不同规模电路的加速作用, 选中的实验电路被多次例化后进行规模扩展, 从而获得超百万门的电路规模. 这些扩展是宽度上的扩展, 电路的深度并没有改变. 虽然这些电路并不理想, 但其对于评估模拟算法在不同电路规模下的性能加速趋势仍具有指标意义, 这已在文献[11]中得到了验证, 因此, 本文也采取同样的方法.

表 4~表 6 为电路 s38417, s38584 和 openMSP 在多次例化后电路网表的标准单元数和逻辑门数. 其中, 电路名称后面的“x+数字”(如 x10, x25 等)表示电路例化的次数, 次数越多, 则扩充后电路的规模越大.

表 4　s38417 多次例化的电路规模统计

电路	标准单元数	逻辑门数
s38417	23815	25085
s38417x10	238150	250575
s38417x25	595375	626450
s38417x50	1190750	1252871
s38417x75	1786125	1879263
s38417x100	2381500	2505683
s38417x200	4763000	5011307

表 5　s38584 多次例化的电路规模统计

电路	标准单元数	逻辑门数
s38584	20679	22610
s38584x10	206790	225734
s38584x25	516975	564350
s38584x50	1033950	1128660
s38584x75	1550925	1692930
s38584x100	2067900	2257239

表 6 openMSP 多次例化的电路规模统计

电路	标准单元数	逻辑门数
openMSP	4730	11128
openMSPx10	47160	110432
openMSPx25	117900	275987
openMSPx50	235800	551819
openMSPx75	353700	827609
openMSPx100	471600	1103446
openMSPx200	943200	2206761

在芯片设计流程中，构成电路网表中的标准单元和在基准电路中常见的基本逻辑门是有区别的. 典型的逻辑门(如 NAND, AND, NOR, OR, NOT 等)都有其对应的标准单元实现，但并不是所有标准单元都可以用单个的逻辑门来表示其逻辑功能. 例如，在单元库中常见的 AOI 标准单元即可以用 3 级的逻辑门结构来表示其功能，第 1 级是多个与门 AND，与门的输出送到第 2 级的或门 OR，或门的输出再送到第 3 级的反向器 NOT；也可以用或非门 NOR 合并第 2, 3 级逻辑门. 除了 AOI 外，单元库中还有一些更复杂的标准单元，如扫描单元和全加器单元等，需要使用更多的逻辑门来构建其等效的功能实现.

3.2 性能加速比分析

表 7 至表 9 为电路 s38417, s38584 和 openMSP 在不同例化次数下逻辑模拟的运行时间，例化次数越多，电路规模越大. 其中, s38417 和 s38584 使用的是 1 024 个时钟周期的随机功能向量输入；电路 openMSP 则使用的是 IP 核验证时的真实功能向量测试序列，逻辑模拟的运行时间分为在主机 CPU 上的单线程运行时间(简称单线程)、在多核 CPU 上的多线程并行算法时间(简称多核 CPU)以及在 GPU 上的多线程并行算法时间(简称 GPU).

表 7 s38417 逻辑模拟运行时间

例化次数	逻辑门数	运行时间/s		
		单线程	多核 CPU	GPU
x1	25085	1.49	5.58	3.55
x10	250575	15.07	8.24	4.27
x25	626450	44.37	13.43	4.07
x50	1252871	99.23	20.41	3.98
x75	1879263	153.24	29.16	4.13
x100	2505683	210.90	44.63	4.71
x200	5011307	515.42	70.01	5.93

表 8 s38584 逻辑模拟运行时间

例化次数	逻辑门数	运行时间/s		
		单线程	多核 CPU	GPU
x1	22610	2.19	6.14	4.00
x10	225734	22.23	11.38	4.46
x25	564350	87.06	19.37	4.88
x50	1128660	198.46	33.95	5.14
x75	1692930	253.95	46.98	5.50
x100	2257239	338.33	57.54	5.68

表 9 openMSP7 逻辑模拟运行时间

例化次数	逻辑门数	运行时间/s		
		单线程	多核 CPU	GPU
x1	11128	0.36	6.92	4.53
x10	110432	3.86	7.73	4.47
x25	275987	9.87	7.80	5.14
x50	551819	20.29	9.96	5.01
x75	827609	35.00	13.70	5.31
x100	1103446	45.10	16.75	5.07
x200	2206761	92.82	20.12	5.14

从表 7~表 9 可以看出，随着电路规模的增长，CPU 上单线程的逻辑模拟时间接近于线性增长；多核 CPU 上的并行模拟时间也有所增长，但速度要远低于 CPU 单线程的速度；而 GPU 上并行模拟时间则变化不大，只有非常小幅度的增加，说明 GPU 的并行运算资源还没有被充分利用，还可以模拟更大规模的电路. 与单线程的逻辑模拟相比，不论是在多核 CPU 上还是在 GPU 上，并行算法的性能提升都是显著的，但两者之间还是有区别的. 为了更好地展示这 2 种架构下的性能加速差异，图 4~图 6 分别显示了在不同例化次数下，电路 s38417, s38584 和 openMSP 的电路规模与性能加速比的关系. 若单线程运行时间为 T_s，并行算法运行时间为 T_m，则性能加速比 S 定义为 $S = T_s/T_m$.

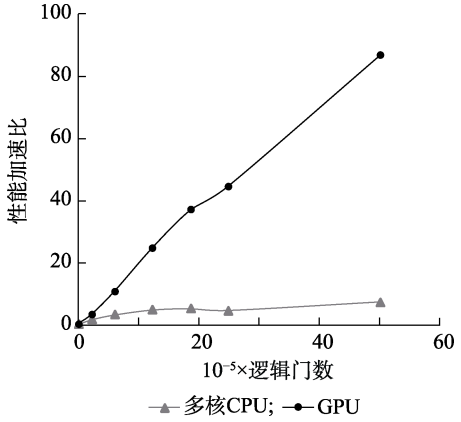


图 4 s38417 不同例化次数下的性能加速比

从图 4 至图 6 可以看出，多核 CPU 上的并行算法性能加速比随着电路规模的增加而趋于饱和，这是由于多核 CPU 的物理线程数是非常有限的(只

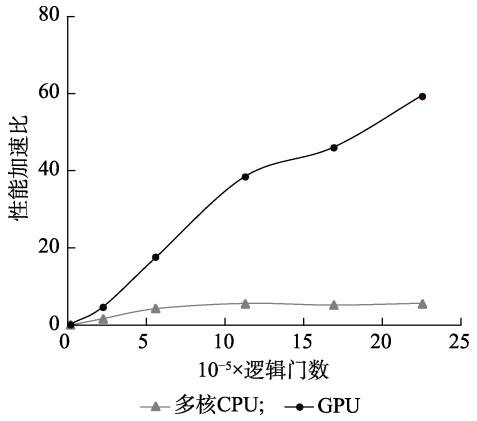


图 5　s38584 不同例化次数下的性能加速比

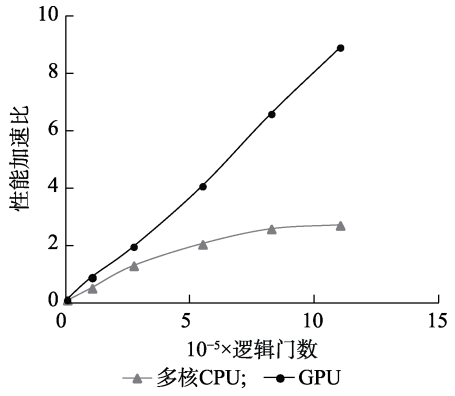


图 6　openMSP 不同例化次数下的性能加速比

有 20)，当电路足够大时其利用率已经非常高了，因此加速比也很难提高了．然而，实验配置中的 GPU 有 3584 个 CUDA 核，可以并行执行 3584 个物理线程，即便是实验中的百万门级电路，也没有将所有的 CUDA 核充分利用. OpenCL 并行计算教程[6]中指出，为了最大化运算资源利用率，要求程序向调度管理器申请的并行线程数要远大于物理的运算核心数．显然，多核 CPU 上的并行算法已满足这个要求，但 GPU 还没有达到．可以预见，随着电路规模增大到千万门级甚至是上亿门级，当 GPU 的运算资源利用率足够高时，其性能加速比也会趋于饱和.

需要指出的是，并行逻辑模拟在电路 openMSP 的性能加速比要比基准电路 s38417 和 s38584 低得多．在 GPU 上使用大约 100 万门的电路规模下，s38417x50(125.3 万门)的性能加速比是 24.9；s38584x50(112.9 万门)的加速比为 38.6；openMSPx100(110.3 万门)的加速比为 8.9．导致这种差异有 2 个原因：(1) openMSP 的功能向量激活的电路信号翻转率要低于 s38417 和 s38584，因此其在同层级需要并行执行的事件就会变少，从而降低了 GPU 的运算资源利用率；(2) openMSP 在电路分层后的层数较多，大约是 s38417 和 s38584 的 2 倍，而且不少层级上逻辑门数量较少．由于混合式并行逻辑模拟算法的加速来源是在同一层级上的逻辑门可做并行计算，而不同层级之间的仿真仍是顺序进行的，因此每个层级都要发起一次并行任务的调度申请，且这个调度申请是需要付出性能代价的．电路层数增多，算法中发起的并行任务调度申请也随之增加，这从另一个方面影响了算法的性能加速比.

一个非常有意义的实验是对比商用逻辑模拟工具 VCS 的运行时间．已知的功能向量并行模拟[7-10]大都语焉不详，只是简单地列出了一个加速比数字，或是说，自己的单线程性能与商用工具类似，并未给出商用工具的详细信息、具体运行参数以及硬件平台，而这些参数实际上对逻辑模拟的性能影响非常大，甚至反转数据分析结论.

针对 openMSP 电路，表 10 所示为在同一台服务器上使用商用工具 VCS 缺省选项的运行时间，作为比较，表中同时加入了单线程逻辑模拟算法和 GPU 上并行逻辑模拟算法的运行时间．从表 10 可以看出，商用的逻辑模拟工具 VCS 是非常高效的，其性能平均是本文中单线程算法的 9~10 倍．需要指出的是，文献[11]用同样的方法比较了商用工具和单线程算法的性能差异，结果表明商用工具的性能大约是单线程算法的 2 倍．但不同的是，文献[11]的研究对象是面向结构测试向量的并行逻辑模拟，且对标的商用工具是西门子公司的 ATPG 工具 Tessent FastScan．商用工具产生这么大的差异的关键在于网表优化．ATPG 是用于产生数字电路的结构测试向量和进行故障模拟的，需要完整地保留电路网表的结构信息和所有内部节点的仿真值，这与本文的逻辑模拟器类似．通用的逻辑模拟器是为了验证电路的输入和输出是否符合预期，而电路内部结构和仿真过程中的内部节点值并不是其重点关注的问题，所以商用逻辑模拟器会对电路网表进行类似于代码编译的深度优化．也就是说，商用 VCS 和本文的逻辑模拟算法实际上仿真的内部网表并不一致．只有当 2 种工具的逻辑模拟算法使用同样的优化网表时，对比才真正有意义．即便如此，当电路足够大，即 openMSP 例化 200 倍、电路规模约为 200 万门时，GPU 上的并行逻辑模拟性能是可以反超 VCS 的．因此，在 GPU 上进行功能向量的并行模拟还是有其优势的.

表 10 openMSP 的商用工具运行时间对比

例化次数	逻辑门数	运行时间/s		
		单线程	GPU	商用 VCS
x1	11128	0.36	4.53	**0.25**
x10	110432	3.86	4.47	**0.48**
x25	275987	9.87	5.14	**0.88**
x50	551819	20.29	5.01	**1.90**
x75	827609	35.00	5.31	**3.10**
x100	1103446	45.10	5.07	**4.90**
x200	2206761	92.82	**5.14**	11.3

注. 加粗表示最优值.

表 11 所示为文献[8-10]和本文在 GPU 上进行功能向量并行逻辑模拟的数据对比, 其中列出了各个文献实验中的最大电路规模、对比单线程逻辑模拟的性能加速比, 以及对比商用工具的性能加速比. 本文引用的数据是电路在最大例化次数下的实验结果. 不同文献使用的电路、功能向量和 GPU 型号不尽相同, 而电路结构和功能向量特性对并行模拟算法的影响非常大, 并且商用工具在不同网表优化选项下也有很大的性能差异, 因此表 11 的数据只能作为参考. 从表 11 可以看出, 本文在对比单线程的性能加速方面已不输于文献[8-10]的工作. 另外, 并行逻辑模拟的真正驱动力来源于超大规模的数字芯片模拟, 而这正是本文算法所擅长的, 同时也是相比于文献[8-10]的一个重要优势.

表 11 4 种算法 GPU 功能向量并行逻辑模拟对比

算法	最大电路规模	加速比	
		单线程	商用
文献[8]	1.200M	未报告	平均 13
文献[9]	0.218M	1.7~5.1	未报告
文献[10]	0.117M	平均 47.4	未报告
本文	5.000M	8.9~86.8	2.2(openMSP)

4 结 语

本文在前期面向结构测试向量的 GPU 逻辑模拟工作基础上, 通过算法的改进和优化实现了面向功能向量的并行逻辑模拟. 实验数据和分析结果表明, 本文算法同样能够用于功能向量的仿真, 可运行在包括多核 CPU 和 GPU 在内的多种并行计算架构中; 同时, 该算法在 GPU 架构上有着更为优秀的电路规模扩展性, 非常适用于集成有海量运算核心的 GPU 和各类加速芯片.

参考文献(References):

[1] Bailey M L, Briner JR J V, Chamberlain R D. Parallel logic simulation of VLSI systems[J]. ACM Computing Surveys, 1994, 26(3): 255-294

[2] Chang K H, Browy C. Parallel logic simulation: myth or reality?[J]. Computers, 2012, 45(4): 67-73

[3] Lee V W, Kim C, Chhugani J, *et al*. Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU[C] //Proceedings of the 37th Annual International Symposium on Computer Architecture. New York: Association for Computing Machinery, 2010: 451-460

[4] Du P, Weber R, Luszczek P, *et al*. From CUDA to OpenCL: towards a performance-portable solution for multi-platform GPU programming[J]. Parallel Computing, 2012, 38(8): 391-407

[5] Cheng J, Grossman M, McKercher T, *et al*. Professional CUDA C programming[M]. New York: John Wiley & Sons, 2014

[6] Munshi A, Gaster B R, Mattson T G, *et al*. OpenCL programming guide[M]. New Jersey: Addison-Wesley Professional, 2011

[7] Chatterjee D, DeOrio A, Bertacco V. GCS: high-performance gate-level simulation with GPGPUs[C] //Proceedings of Design, Automation & Test in Europe Conference & Exhibition. Los Alamitos: IEEE Computer Society Press, 2009: 1332-1337

[8] Chatterjee D, DeOrio A, Bertacco V. Event-driven gate-level simulation with GP-GPUs[C] //Proceedings of the 46th ACM/IEEE Design Automation Conference. New York: Association for Computing Machinery, 2009: 557-562

[9] Sen A, Aksanli B, Bozkurt M, *et al*. Parallel cycle based logic simulation using graphics processing units[C] //Proceedings of 9th International Symposium on Parallel and Distributed Computing. Los Alamitos: IEEE Computer Society Press, 2010: 71-78

[10] Zhu Y H, Wang B, Deng Y D. Massively parallel logic simulation with GPUs[J]. ACM Transactions on Design Automation of Electronic Systems, 2011, 16(3): 1-20

[11] Lai L Y, Zhang Q T, Tsai H, *et al*. GPU-based hybrid parallel logic simulation for scan patterns[C] //Proceedings of the IEEE International Test Conference in Asia. Los Alamitos: IEEE Computer Society Press, 2020: 118-123

[12] Brglez F, Bryan D, Kozminski K. Combinational profiles of sequential benchmark circuits[C] //Proceedings of the IEEE International Symposium on Circuits and Systems. Los Alamitos: IEEE Computer Society Press, 1989: 1929-1934