面向集成电路逻辑仿真的高效数字波形压缩方法

谢雨洋[1,2)], 李凌劼[1,2)], 喻文健[1,2)*]

[1)] (清华大学计算机科学与技术系 北京 100084)
[2)] (北京信息科学与技术国家研究中心 北京 100084)
(yu-wj@tsinghua.edu.cn)

摘 要: 电路仿真在集成电路设计中占有十分重要的地位. 超大规模集成电路仿真输出的信号波形会占用大量存储空间, 输出波形的压缩对提升电路仿真效率至关重要. 逻辑仿真的输出波形主要包含信号跳变时刻信号值, 还包含全部信号名、信号类型、信号位宽等辅助信息. 首先提出对辅助信息的压缩处理方法, 然后针对信号值数据的特点改进现有工作中的信号名压缩方案, 并据此提出一种更高效的数字波形压缩存储格式. 所提出的格式有利于变长编码压缩的同时, 可以使用通用压缩算法进行二次压缩, 进一步提升了压缩率. 最后通过引入并行策略, 压缩和解压缩过程能够以 3 阶段流水线模式运行. 在缩短压缩和解压缩时间的同时, 能更好地与逻辑仿真器相结合. 实验结果表明, 所提方法的压缩率最大达到 720 倍, 相较于现有方法, 在更短的压缩、解压缩耗时下文件压缩率最大提升近 23 倍.

关键词: 电路仿真; 数字波形; 波形压缩; 并行计算
中图法分类号: TP391.41 **DOI:** 10.3724/SP.J.1089.2021.18799

Efficient Digital Waveform Compression Method for Logic Simulation of Integrated Circuits

Xie Yuyang[1,2)], Li Lingjie[1,2)], and Yu Wenjian[1,2)*]

[1)] (*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)
[2)] (*Beijing National Research Center for Information Science and Technology, Beijing* 100084)

Abstract: Circuit simulation becomes more and more important in integrated circuit design. For VLSI circuits, the simulation usually outputs signal waveforms occupying massive storage space. The compression of these signal waveforms becomes crucial to the efficiency of circuit simulation. Logic simulation mainly outputs the signal values at the time of signal transition and some auxiliary information such as signal name, signal type, signal width. A compression method for auxiliary information is proposed. Then, the signal name compression scheme in existing work is improved according to the characteristics of signal value data, and a more efficient digital waveform compression storage format is proposed. The proposed format is more adaptive to the variable-length coding for compression. At the same time, general compression algorithms can be used for secondary compression, thereby further improve the compression rate. Finally, through parallel computing, the compression and decompression procedure can run in a three-stage pipeline mode. The proposed method largely reduces the compression and decompression time, and can be better integrated within the logic simulator. The experimental results show that with the proposed method the compression rate can be as large as 720. Compared with the existing methods, the compression rate is increased by nearly 23 times with shorter compression and decompression time.

收稿日期: 2020-11-23; 修回日期: 2021-05-28. **基金项目**: 国家重点研发计划(2019YFB2205002); 国家自然科学基金(61872206).
谢雨洋(1996—), 男, 博士研究生, 主要研究方向为数值计算、大数据挖掘; **李凌劼**(1995—), 男, 博士研究生, 主要研究方向为模拟波形压缩、张量计算; **喻文健**(1977—), 男, 博士, 副教授, 博士生导师, CCF 高级会员, 论文通讯作者, 主要研究方向为参数提取、数值算法与应用、高性能计算.

Key words: circuit simulation; digital waveform; waveform compression; pipeline technology

集成电路逻辑仿真的对象是由门和功能块等元件组成的逻辑电路, 逻辑仿真是对硬件描述语言所定义的数字电路行为的预测与检验, 其基本原理是使用计算机软件模拟一个激励信号, 然后观察所设计电路的响应行为并判断是否满足预期的功能. 逻辑仿真采用的是事件驱动的仿真方法, 仿真结果波形仅当事件发生的某些时刻才产生变化. 逻辑仿真已经成为数字集成电路设计中最重要的步骤之一, 其得到的数字波形为后续的设计验证与优化提供重要的参考. 随着电路规模不断变大, 逻辑仿真生成的数字波形数量不断增多, 这导致需要大量的存储空间存储波形. 对于集成电路逻辑仿真过程, 仿真时间节点的数量从成百上千到上亿不等, 而一个时间点上可能包含了成千上万的数字信号值变化. 针对集成电路逻辑仿真生成的大规模数字波形文件, 需要设计一个高效且无精度损失的数字波形压缩存储格式和相应算法以减少波形数据的存储开销.

集成电路设计流程相关的数据压缩问题还有测试数据压缩和模拟波形压缩. 测试是将预先设置的一系列测试输入信号, 施加到电路器件的输入上, 在电路输出端得到输出信号, 将输出和已设置好的预期输出对比验证电路是否存在故障. 因此, 预先准备一套正确而充分的测试数据, 对于成功测试是至关重要的. 考虑对测试数据的压缩能极大减少存储开销, 文献[1]提出了一种动态更新参考切片的 IP(intellectual property)核测试数据压缩方法; 文献[2]提出了一种广义折叠技术的集成电路测试数据压缩方案. 模拟信号的瞬态仿真需计算出随时间变化的电路响应, 得到电压和电流等电气量的连续变化数值. 因此, 针对瞬态仿真得到的模拟波形的压缩也是至关重要的. 文献[3]提出了一种使用递归最小二乘格子与算术编码结合的无损自适应预测方法, 但是该方法的压缩率不高; 文献[4-5]通过执行主成分分析来压缩波形信号值, 但是这意味着存在大量的计算和内存的开销; 文献[6]提出了一种具有波形预测和量化的流式波形数据压缩方法; 文献[7]使用递归多项式表示压缩技术库. 值得注意的是, 虽然在测试数据压缩和模拟波形压缩问题上有一些相关工作, 但与数字波形压缩相关的工作非常少.

文献[8]提出了一些适用于数字波形的压缩技术, 但是文献中没有较好的数据结构和对应的实现方法. 针对文献[8]的不足, 本文设计出更好的数据结构以及使用二次压缩方法, 从而达到更高的压缩率. 另外, 文献[8]中没有能够将数字波形压缩和解压缩算法与逻辑仿真器相结合的具体方案, 总的来看, 文献[8]方法缺乏实用性. 另外, 还有一些商业工具支持压缩转储波形格式, 如快速信号数据库格式(fast signal database, FSDB)①, 但是未见介绍该工具压缩算法的文献. 其他一些常见的无损压缩算法, 如霍夫曼编码、Deflate[9]和 LZW 编码(Lempel-Ziv-Welch Encoding)压缩算法[10], 也可用于减少波形数据的存储, 但是这些算法没有考虑数字波形数据的数据特性以及前后波形信号变化的相关性, 因此不能提供显著的压缩率.

考虑实际情况中的逻辑仿真输出波形包含信号跳变时刻信号值, 还包含全部信号名、信号类型、信号位宽等辅助信息. 本文首先对辅助信息压缩存储, 提出了对逻辑仿真输出的辅助信息的压缩方案, 使压缩算法能完整地结合到逻辑仿真器当中. 然后根据输出的信号值特点, 针对已有工作中信号名压缩方案和存储格式进行改进; 在信号名压缩部分设计限制大小的对照表, 从而使程序实际运行内存变得可控. 将压缩过程得到的仿真时间点、访问索引值、信号值编码数据分别存储为数据流, 再拼接数据流得到高效的数字波形压缩存储格式. 该格式有利于变长压缩和二次压缩, 进一步提升了压缩率. 基于该格式, 本文进一步提出了对应原始波形文件的压缩文件存储格式. 综合以上技术, 最后提出高效的数字波形信号压缩和解压缩算法, 并以流水线技术改进, 加速了算法的运行, 并使之能更好地与逻辑仿真器结合. 实验显示, 与现有方法相比, 在更短的压缩、解压缩耗时下, 本文方法文件压缩率最大提升近 23 倍.

1　相关工作

1.1　数字波形与 VCD 格式

数字波形信号值包括 4 种状态, 分别是高电平 1、低电平 0、高阻态 *z* 和未定态 *x*. 图 1 所示为一段逻辑仿真输出的 3 个数字波形, 其对应于 3 个节

① https://www.synopsys.com/verification/virtual-prototyping/saber/cosmos-scope.html

点上的信号, 它们的关系为 $y=x_1 \& x_2$. 逻辑仿真的时间点是以半个时钟周期为单位的, 因此可以将仿真时间表示为 $t=\{0,1,2,3,4,5\}$, 信号 x_1 的变化表示为 $x_1=\{1,1,1,1,0,0\}$, 信号 x_2 的变化表示为 $x_2=\{0,1,0,0,1,1\}$, y 的变化表示为 $y=\{0,1,0,0,0,0\}$. 其中, 1 表示高电平, 0 表示低电平. 可以看出, 每个信号仅在某几个仿真时间点会发生变化, 据此可以得到信号值变化数据为

$$\begin{aligned}&\#0,\ 1x_1,\ 0x_2,\ 0y,\\&\#1,\ 1x_2,\ 1y,\\&\#2,\ 0x_2,\ 0y,\\&\#4,\ 0x_1,\ 1x_2,\end{aligned}$$

这样, 只需要记录有信号值发生变化的仿真时间点 $t=\{0,1,2,4\}$以及对应发生变化后的信号值即可, 这样记录信号值变化的数据显然比记录完整的波形数据要节省空间, 这也是工业中许多逻辑仿真器采用的数字波形数据的输出方式.

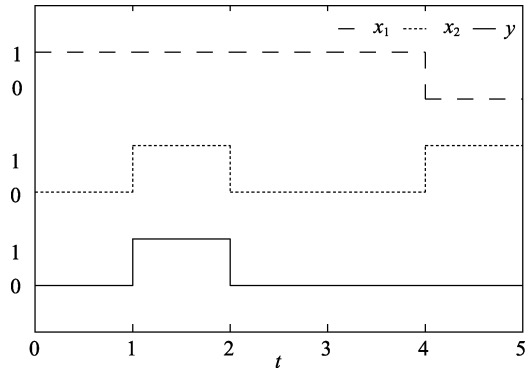

图 1 逻辑仿真结果波形图

基于 ASCII 字符的 VCD 文件格式就是一种在工业中广泛应用的数字波形表示格式[11], 该数字波形表示格式在完全满足以上信号值变化规则之外还保留了信号名、信号定义、信号位宽等组成的辅助信息. 辅助信息不仅对后续波形压缩和解压缩算法有影响, 而且对仿真器也是至关重要的. 在 VCD 文件中, 辅助信息所占比重往往很小, 而信号值变化的信息占比重很大. 针对图 1 的样例和对应的信号值变化数据, 可以给出相应的 VCD 文件的内容, 如图 2 所示. 图 2 中"$enddefinitions $end"之前都是 VCD 文件的辅助信息, 而"#0"开始的后续数据都是信号值变化信息. VCD 文件中的信号类型可以划分为 SCALAR 类型和 VECTOR 类型, 两者的区别就是前者信号值的位宽始终为 1, 而后者信号值的位宽始终大于 1. 因此, 本文完全可以将 VCD 文件当作初始的波形文件, 后续的研究工作可以考虑对该文件进行压缩.

```
$date
    November 22, 2020 11::11::23
$end
$version
    sim6415 _123 01920klaskdsa
$end
$timescale
    10 fs
$end
$scope module rc_pif $end
$var reg 1 x1    TxDeemph $end
$var reg 1 y1    user_rx_info $end
$var reg 1 y     ltssm_tx $end
$upscope $end
$enddefinitions $end
#0
1x1
0x2
0y
#1
1x2
1y
#2
0x2
0y
#4
0x1
1x2
```

图 2 样例对应的 VCD 文件内容

为了方便后续的说明, 先给出信号值变化数据的表示方法. 首先约定完整信号值变化部分的数据(vector of transition block, VTRB)表示为 $\mathrm{VTRB}=\{\mathrm{TRB}_0,\mathrm{TRB}_1,\cdots,\mathrm{TRB}_{N-1}\}$, 其中 N 表示 VTRB 一共有 N 块信号值变化块(transition block, TRB), $\mathrm{TRB}_i=\{t,\mathrm{VTR}_i\}$表示一个仿真时间点 t 以及对应的一系列信号值变化信息(vector of transition, VTR). 再约定 $\mathrm{VTR}=\{\mathrm{TR}_0,\mathrm{TR}_1,\cdots,\mathrm{TR}_n\}$, 其中一个信号的变化(transition, TR)表示为 $\mathrm{TR}=\{v,\mathrm{ID}\}$, v 表示变化后的信号值, ID 表示 ASCII 字符组成的信号名.

图 2 中的信号值变化信息用上述提出的方法表示, 可知 N=4, $\mathrm{VTRB}=\{\mathrm{TRB}_0,\mathrm{TRB}_1,\mathrm{TRB}_2,\mathrm{TRB}_3\}$. 其中, $\mathrm{TRB}_0=\{0,\mathrm{VTR}_0\}$, 而 $\mathrm{VTR}_0=\{\mathrm{TR}_0,\mathrm{TR}_1,\mathrm{TR}_2\}$, $\mathrm{TR}_0=\{1,x_1\}$, $\mathrm{TR}_1=\{0,x_2\}$, $\mathrm{TR}_2=\{0,y\}$, 同样可以类似地得到 $\mathrm{TRB}_0\sim\mathrm{TRB}_3$ 的组成.

1.2 现有压缩方法

文献[8]提出了一些针对 VCD 格式的信号值变化的压缩技巧, 但没有针对辅助信息的压缩处理. 由于仿真器是分段地输出信号值变化信息 TRB={t, VTR}, 文献[8]主要是对 VTR 压缩处理. 因为 VTR 里的 TR={v, ID}主要由信号名 ID 和信号值 v 组成, 文献[8]对信号名和信号值分别进行压缩,

从而得到编码压缩后的信号值变化信息(compressed TRB_i, $CTRB_i$). 已知 VTRB 一共有 N 块 TRB, 容易得到如图 3 所示全部信号值变化信息压缩存储格式. 对该存储格式可以分批进行二次压缩(gzip[12]/bzip2[13])并写入文件.

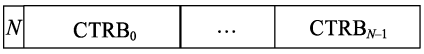

图 3　全部信号值变化信息的压缩存储格式

下面介绍文献[8]中对信号值 v 的编码压缩方案. 因为数字信号值 v 只由 0, 1, x 和 z 组成, 故很容易想到将信号值中的 0 编码成二进制的 00, 1 编码成 01, x 编码成 10, z 编码成 11. 由于数字信号值中最频繁出现的是 0 和 1, 因此信号值编码后的二进制表示中的奇数位往往都是 0. 针对该特点, 可以从信号值编码后的二进制数据中提出 4 个字节, 放置奇数位在前 2 个字节, 放置偶数位在后 2 个字节, 这样数据换位后可以更契合变长编码[14]的适用情况, 再进行变长编码可以较好地压缩, 以上就是文献[8]对信号值的基础编码方案. 变长编码方法的基本思想是将每个字节的第 1 位设置为标识位, 用于标识是否存在后续字节, 而另外 7 位用于存储数据. 例如, 对于十六进制数据 0x1647(二进制码为 0001011001000111), 用 32 位整数类型表示需要 4 个字节, 而通过变长编码后只需要 2 个字节, 其结果如图 4 所示. 该方法往往可以将数据从 4 个字节编码到 1~2 个字节, 适用于数值二进制表示中高位为 0 的情况. 为了让信号值编码的二进制表示中出现尽可能多的 0, 下面采用 2 种预测方案. 第 1 种方案是基于历史信号值的预测方法, 该方法需要一个历史数据表(history table, HT)存储信号值的历史数据. 先考虑对 SCALAR 类型信号的预测方法. 如果在当前仿真时间点的信号值为 0, 则下一个时间点会大概率变为 1; 如果在当前时间点信号值为 1, x, z, 则预测下一个时间点会变成 0. 再考虑对 VECTOR 类型信号的预测方法, 可以推测下一个时间点的信号值即为最低有效位进行翻转后的历史信号值. 将预测值与原始信号值进行异或运算(XOR)后, 写入信号值编码数据流 S_{data}, 同时更新历史数据表. 最后可以将 S_{data} 经过基础编码压缩后再存储. 第 2 种方案的思想是对一个 VTR 经过第 1 种方案异或运算步骤后得到了数据 S_{XOR}, 将之与 VTR 对应的历史信号集数据再进行异或, 选取异或结果和 S_{XOR} 中数据位为 0 较多的一个作为 S_{data}, 并同时写入一个标记位到 S_{data} 头部.

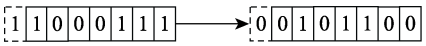

图 4　0x1647 变长编码结果

最后介绍文献[8]中对信号名 ID 的压缩方案以及对应于 $CTRB_i$ 的存储格式. 此处先约定 $TRSB=\{ID_0, ID_1, \cdots, ID_k\}$ 表示 TRB 中信号值发生变化的信号名的集合, 经过观察发现不同的 TRB 所对应的 TRSB 往往是一致的. 针对这样的特性, 可以借助一个对照表 $(TRSB_I_{access}_table, TRSBIT)$ 进行压缩处理. 最初表是空的, 开始遍历 TRSB, 去对照表中查询, 如果查询未命中, 则插入 TRSB 到表中, 并返回一个访问索引值 I_{access}, 同时将 TRSB 和 I_{access} 写入 $CTRB_i$ 中; 如果在表中查询命中, 则返回对应的 I_{access} 并写入 $CTRB_i$ 中. 同时, 也可以将仿真时间点 t 连同 I_{access} 一起写入, 最终得到了如图 5 所示 $CTRB_i$ 存储格式. 仅当在对照表中查询缺失时, 图 5 中的 TRSB 需要写入存储格式中. 文献[8]认为可以先将 TRSB 的信号名转化成整数后, 按照从小到大排序, 然后互相作差并存储差值, 最后利用变长编码进行压缩. 在测试中发现对 TRSB 的压缩方案并不适用于实际情况. 首先, 排序会有额外时间开销; 其次, 因为最后对图 3 的存储格式分批运用了二次压缩, 使 TRSB 的压缩方案对整体压缩率贡献极少; 最后, 发现对照表 TRSBIT 在实际测试中会因表项过多而占用大量运行内存.

图 5　信号值变化 TRB 的压缩存储格式

2　适用于仿真器的高效数字波形压缩和解压缩方法

第 1.2 节主要介绍了信号值变化部分信号值的压缩方案、信号名的压缩方案和相对应的压缩存储格式. 在实际情况下, 逻辑仿真器输出的数据更加复杂, 如 VCD 文件不仅包含了波形信号值变化数据, 还包含了波形信号名、信号类型、信号位宽等仿真过程中十分重要的辅助信息, 文献[8]没有针对辅助信息的处理和压缩方案. 本文方法先全部读取这部分信息并将之存储在一个字符串中, 再对字符串运用通用压缩方法(gzip/bzip2)压缩存储. 读取辅助信息时, 要注意识别在辅助信息里信号名和对应的信号位宽, 将其作为一个映射表存储在内存, 方便压缩算法和解压缩算法的进行. 由于

辅助信息整体占波形文件很少一部分, 运用 bzip2 进行压缩后的辅助信息通常占用的大小只有几千字节或几万字节, 因此可以将对辅助信息的压缩处理作为压缩算法开始运行之前的预处理过程.

文献[8]没有设计出针对数字波形信号更高效的压缩存储格式, 第 1.2 节的信号名压缩方案和对应的存储格式存在很大缺陷. 在实际情况下, 图 5 中对 TRSB 的压缩方案压缩率几乎没有提升, 并且要想解压缩得到与原来顺序一致的信号值变化存在困难; 在实际测试中还发现 TRSBIT 会造成内存开销过大, 从而使实际运行内存难以控制的问题. 另外, CTRB_i 由 t, I_{access}, TRSB 和 S_{data} 组成, 造成数据不规则, 不利于二次压缩, 因此并不实用. 文献[8]没有给出完整的压缩和解压缩算法, 更没有能够适用于仿真器的具体方案. 因此, 本文逐一解决以上问题, 得到在实际应用中表现良好的数字波形压缩和解压缩算法.

2.1 高效的数字信号压缩存储格式

仿真器分段地输出信号值变化信息 TRB={t, VTR}, 可以将 t 依次存储到一个时间数据流 S_t 中. 针对 S_t 的压缩方案, 只需要存储第 1 个时间点的数值以及后续时间点之间的差值, 由于仿真时间点之间差值较小, 从而采用变长压缩方法能进行高效的压缩, 最后再对其使用二次压缩的方法进行压缩存储. 针对 VTR 的处理, 可以借鉴第 1.2 节的技术, 先从 TRB 中得到 TRSB, 再按照信号名压缩方案进行处理, 对 TRSB 信息的压缩存储方案主要依赖对照表 TRSBIT 进行, 然而该表在实际情况下往往占用过大内存. 因此, 可以考虑限制该表的大小为 s_1, 当前 TRSB 查表若没有命中, 则将 TRSB 信息插入表中以及写入数据流 S_{TRSB} 中, 并生成一个 I_{access} 写入访问索引值数据流 S_{I} 中; 当前 TRSB 查表若命中, 则返回一个 I_{access} 写入到 S_{I} 中, 并且如果此时表的大小超出了 s_1, 则将表中内容按照命中率进行排序, 删除掉命中率较低的一半. S_{I} 整体可以用变长压缩后再二次压缩的方法进行压缩存储. 考虑到信号名是由大小为 33~126 的 ASCII 字符组成, 因此对 S_{TRSB} 的压缩调用变长压缩和二次压缩也可以达到较好的压缩率. 以上是针对第 1.2 节信号名压缩方案的改进方法, 这样既增加了对数据的压缩能力也方便了压缩和解压缩. 最后根据第 1.2 节信号值的第 1 种预测方案, 对仿真器输出的每个 TRB 中的信号值进行相应的预测编码压缩得到 S_{data}, 并拼接多块 S_{data} 得到最终的 $S_{\mathrm{data_}m}$. 对 $S_{\mathrm{data_}m}$ 采用基础编码后再二次压缩的方法压缩存储. 最终得到图 6 所示的信号值变化的压缩存储格式 (compressed VTRB, CVTRB), n 表示 S_t 所包含的仿真时间点个数. 在压缩过程中, n 由预设的 $S_{\mathrm{data_}m}$ 可以包含的字节数上限参数 s_2 决定. 假设整个压缩过程中共生成了 m 个压缩数据块, 对应每块的 n 值为 $n_0, n_1, \cdots, n_{m-1}$, 则 $n_0+n_1+\cdots+n_{m-1}=N$, 其中 N 为整个波形数据中包含的仿真时间点的数目.

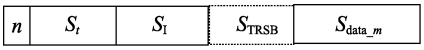

图 6 第 1 种预测方案下信号值变化压缩存储格式

根据第 1.2 节中第 2 种预测方案的信号值压缩方法, 每个 TRB 处理后可以得到一个 S_{data}, 其中 S_{data} 头部都带有一个标记位. 将标记位全部写入标记数据流 S_{token} 中, 而 S_{data} 存储的是 S_{XOR} 或与 TRB 对应的历史信号集数据, 以 S_{token} 存储的标记位作为区分, 因此容易得到如图 7 所示的压缩存储格式. $S_{\mathrm{data_}m}$ 的压缩存储方式同上, 再考虑对 S_{token} 的压缩, 无法保证提取一段 4 个字节数据, 其高位都为 0, 因此直接采用二次压缩方法压缩存储.

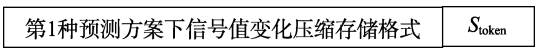

图 7 第 2 种预测方案下信号值变化压缩存储格式

综上, 本文总结了 2 种适用于数字波形的高效压缩存储格式, 自然也容易得到对应 VCD 文件的压缩文件存储格式, 如图 8 所示. 压缩文件的第 1 部分是经过 bzip2 压缩后的辅助信息; 第 2 部分是整数类型数值 m, 为了方便解压缩算法的运行, m 指代第 3 部分用 m 批存储的 CVTRB 数据; 第 3 部分就是由 $\{\mathrm{CVTRB}_0, \mathrm{CVTRB}_1, \cdots, \mathrm{CVTRB}_{m-1}\}$ 组成的全部信号值变化的压缩存储数据.

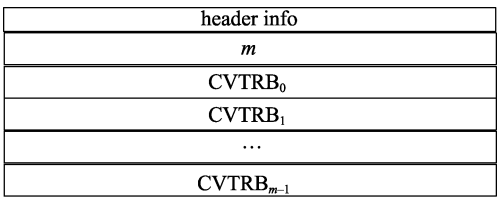

图 8 压缩文件的存储格式

2.2 压缩与解压缩过程的算法描述

结合第 2.1 节的数字波形的高效压缩存储格式以及第 1.2 节的压缩技术, 得到高效数字波形压缩

算法 1.

算法 1. 高效数字波形压缩算法.

输入. 从逻辑仿真器输出信号值变化的数据 $D=\{\mathrm{TRB}_i,\mathrm{TRB}_{i+1},\cdots,\mathrm{TRB}_j\}$.

输出. 数字波形文件对应的压缩文件 F.

Step1. 初始化 $S_{\mathrm{data_}m}$, S_{predict}, S_{original}, S_{XOR}, S_1, S_t, S_{TRSB}, m, n.

Step2. while 仿真没有结束 do {

Step3. for $k=i, i+1,\cdots, j$ do {

Step4. 从信号值变化数据的 TRB_k 中得到时间 t 和 VTR_k. 清空临时变量 TRSB.

Step5. for $l=0, 1,\cdots, \mathrm{num}(\mathrm{VTR}_k)$ do {

Step6. 从 VTR_k 中得到 TR_l.

Step7. 从 TR_l 得到信号名 ID 并添加到 TRSB 中.

Step8. 从 TR_l 得到信号值 v 编码并写入 S_{original}，查询历史数据表 HT，如果查询命中，则将历史信号值对应的预测值写入 S_{predict} 中；如果查询未命中，则将信号值 v 写入 S_{predict}. 最后更新 HT. }

Step9. 在对照表 TRSBIT 中查询对应的 TRSB，如果查询命中，则记录对应的 I_{access} 为索引值；如果查询未命中，则插入 TRSB 到对照表 TRSBIT 中，同时生成一个 I_{access}. 维护限制大小为 s_1 的对照表 TRSBIT.

Step10. 对 S_{predict} 和 S_{original} 进行异或运算并将结果写入 S_{XOR}，清空 S_{predict}，S_{original}.

Step11. if $\mathrm{size}(S_{\mathrm{XOR}})+\mathrm{size}(S_{\mathrm{data_}m})< s_2$ then {

Step12. 将 S_{XOR} 写入 $S_{\mathrm{data_}m}$，将时间 t 写入 S_t，将得到的 TRSB 与 I_{access} 分别写入 S_{TRSB} 和 S_1, n++ }

Step13. else {先对 S_t, S_1 和 S_{TRSB} 进行变长编码再二次压缩方法压缩存储，对 $S_{\mathrm{data_}m}$ 进行基础编码再二次压缩方法压缩存储.

Step14. 拼接压缩后的 n, $S_{\mathrm{data_}m}$, S_t, S_1 和 S_{TRSB} 形成压缩数据块并写入压缩文件 F 中, m++, n=0.

Step15. 清空 $S_{\mathrm{data_}m}$, S_t, S_1 和 S_{TRSB} 后重新写入当前对应数据, n++ }

Step16. 清空 S_{XOR} }}

Step17. 将对照表 TRSBIT 经过变长压缩后再通过二次压缩后写入压缩文件 F 中，移动文件指针，写入 m.

在运行算法 1 之前，已经通过预处理往压缩文件中写入了辅助信息、整型数值 m，因此可以看到算法 1 的 Step2 是不断接收来自仿真器的输出数据. 算法 1 的 Step3 和 Step5 的循环则是遍历输出数据的 TRB 和 TR，其中 $\mathrm{num}(\mathrm{VTR}_k)$ 指 VTR_k 包含的 TR_l 的个数. 算法 1 的 Step7 和 Step9 就是信号名的压缩方法，其中 s_1 是预设的限制对照表大小的参数. Step8 和 Step10 就是信号值的第 1 种预测压缩方案. 算法 1 的 Step11~Step15 主要是将数据压缩并拼接成数据块再写入压缩文件中，其中 size() 指求数据流所占字节数，参数 s_2 是预设的 $S_{\mathrm{data_}m}$ 最大字节数.

针对第 2 种预测压缩方案以及图 7 所对应的压缩数据块存储格式，只需要再考虑 S_{token}. 可以在 Step10 和 Step11 之间插入第 1.2 节第 2 种预测方案，即将 Step10 得到的 S_{XOR} 和历史信号集数据表中对应 I_{access} 的历史信号集数据流进行异或运算得到 S_{XORXOR}. 最后比较 S_{XOR} 和 S_{XORXOR}，得到数据位为 0 较多的结果，记为 S_{data}，将之拼接并写入 $S_{\mathrm{data_}m}$ 中，同时更新标识符到 S_{token}. 因此，可以再改写算法 1 的 Step11~Step16，将 S_{XOR} 替换成 S_{data}，并在 Step13~Step15 加上 S_{token} 并对之进行相应的压缩处理.

下面介绍算法 1 所对应的解压缩算法，解压过程要注意对辅助信息的处理以及对照表的解压缩，具体的算法描述见算法 2.

算法 2. 高效数字波形解压缩算法.

输入. 压缩文件 F.

输出. 解压得到波形信号值变化的数据 $D=\{\mathrm{TRB}_i,\mathrm{TRB}_{i+1},\cdots,\mathrm{TRB}_j\}$.

Step1. 对辅助信息进行读取解压缩处理获得信号长度信息、信号类型，读取 m. 清空对照表 (I_{access}_TRSB_table, ITRSBT).

Step2. while m != 0 do {

Step3. 读取压缩数据块并使用对应的变长解压缩和二次解压缩 (ungzip/unbzip2) 方法解压，得到 n, $S_{\mathrm{data_}m}$, S_t, S_1 和 S_{TRSB}.

Step4. for k=0, 1,…, n do {

Step5. 从 S_t 中得到时间 t，并添加 t 到 TRB_k 中. 从 S_1 中得到访问索引值 I_{access}.

Step6. 根据 I_{access} 查表 ITRSBT，若查表未命中，则从 S_{TRSB} 中得到 TRSB，将 TRSB 和 I_{access} 插入表中，同时维护限制大小为 s_1 的对照表；若查表命中，则取出对应的 TRSB 信息. 最后根据 TRSB 中信号总位宽从 $S_{\mathrm{data_}m}$ 提取数据到 S_{final} 中.

Step7. for l=0, 1,…, num(TRSB) do {

Step8. 从 TRSB_l 中获得信号名 ID.

Step9. 根据读取辅助信息获得的信号位宽映射表，从 S_{final} 中取出对应长度的位数据，存储为编码值 v_{XOR}.

Step10. 查询历史数据表 HT，如果查询命中，则取出对应的历史信号值 v_{pre}，并将其与 v_{XOR} 进行预测解码后得到信号值 v；如果查询未命中，则解码后的信

号值为 v, 更新 HT. 由信号值 v, 再结合信号名 ID 恢复成 TR_l, 添加 TR_l 到 TRB_k 中}

Step11. 形成 TRB_k }

Step12. m--}

Step13. 形成数据 D= {TRB_i,TRB_{i+1},…, TRB_j} .

算法 2 的输入是压缩文件 F, 故算法 2 的 Step1 按照压缩文件的存储格式读取并解压得到辅助信息和 m. Step3 是读取并解压压缩数据块, Step4 和 Step7 是遍历求得 TRB_k 的循环, Step10 则是对应预测值的解压缩方法. 具体分析对于 SCALAR 类型信号, 可以按照表 1 解压, 如果 v_{pre} = 0, v_{XOR} = 1, 则表示预测失败, 实际信号值 v = 0, 其他情况可以依次分析得到, 而最后 v_{pre} = 1/x/z 指的是当上一个时间点信号值为 1 或 x 或 z 时, 下一个信号值都应该预测为 0, 任何实际值 d 和 0 异或运算结果都是 d. VECTOR 类型信号解码值可以通过对最低有效位翻转后的历史信号值以及编码值做异或运算得到.

表 1 基于历史信号值的 SCALAR 信号解压方法

v_{pre}	v_{XOR}	v
0	0	1
0	1	0
0	x	z
0	z	x
1/x/z	d	d

算法 2 是对应信号值的第 1 种预测方案的解压缩算法, 而对应信号值的第 2 种预测方案的解压缩算法只需要对算法 2 的 Step3 压缩数据块的解码进行修改, 具体做法就是按照图 7 的格式进行解码, 并得到 S_{token}. 另外, 还需要在 Step6 和 Step7 之间插入第 2 种预测解压方案, 在 Step6 之后, 根据 S_{token} 得到当前 TRB_k 对应的标识符. 当标识符为 1 时, 则将 Step6 得到的 S_{final} 与历史信号集数据表中对应 I_{access} 的数据流进行异或运算的结果作为后续 Step9 的 S_{final}; 当标识符为 0 时, 则将 Step6 得到的 S_{final} 直接作为后续 Step9 的 S_{final}, 如此便可完成对应第 2 种预测方案的解压缩算法.

2.3 基于 3 阶段流水线的并行压缩与解压缩

根据第 2.1 节和第 2.2 节提出的数字波形文件压缩存储格式和适用于仿真器的数字波形压缩算法、解压算法, 可以将整个压缩、解压缩过程按照 3 阶段流水线的模式连起来, 如图 9 所示.

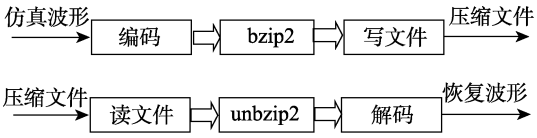

图 9 3 阶段流水线

使用流水线技术可以进一步减少算法 1 和算法 2 的运行时间. 本文发现, 波形压缩的主要过程是逐个数据块进行的, 每个数据块的处理可以简单地视为 3 阶段流水线, 如图 9 所示. 在第 1 阶段中, 先使用算法 1 的 Step3~Step15 的压缩编码过程生成相应的数据流, 然后在第 2 阶段使用二次压缩算法压缩数据块, 最后将压缩数据块写入磁盘文件. 在并行计算中, 2 个相邻阶段之间的关系可以描述为典型的生产者-消费者模型, 因此, 算法 1 可以很容易地用 3 个线程并行化. 与电路仿真相比, 波形压缩所消耗的时间要少得多, 因此来自仿真器的信号值变化数据的生成比执行压缩流水线中的每个阶段要慢得多, 这意味着基于流水线技术的并行压缩方法几乎不会增加整个仿真的时间. 同样, 可以设计出类似的并行解压程序, 利用它也可以减少解压缩的时间.

3 实验及结果分析

本文提出了数字波形压缩和解压缩算法, 其主要对比对象是常见的压缩算法 bzip2 以及文献[8]方法, 实验中用到的二次压缩方法主要是 bzip2. bzip2 相比较于 gzip, 虽然运行时间更长, 但是压缩率更高. 本文的实验设置 bzip2 的压缩参数使它达到最高压缩率, 另外设置算法 1 中的参数 s_1 =800 000, s_2 =20 000 000. 文献[8]方法和本文方法均使用了第 1.2 节对应的 2 种预测方案, 并取 2 种方案中最好的实验结果作为最终结果. 本文方法使用 C++实现, 并且同样用 C++复现了文献[8]方法.

所有实验均运行于一个装有 Intel Xeon E5-2630 CPU(2.4 GHz)的 Linux 服务器上, 操作系统是 Ubuntu16.04.

3.1 测试例子

测试例子是由工业界合作者提供的, 都是来自逻辑仿真器的输出波形. 一共有 9 个实际波形数据, 这些波形数据均按照 VCD 文件的格式输出, VCD 文件大小从几十兆字节到几十千兆字节不等.

3.2 验证压缩算法的有效性

对于每个测试例子, 算法 1 用于波形的压缩,

算法 2 用于波形的解压缩. 测试例子压缩的结果如表 2 所示，R_i (i=1,2,3)分别对应 3 种方法的压缩率. 从表 2 中可以看到，本文方法在所有测试例子上的压缩率都比文献[8]方法好，与 VCD 原文件、bzip2 的压缩文件和文献[8]方法的压缩文件相比，本文方法的压缩文件压缩率最大分别可以达到 720 倍、28 倍和 23 倍. 观察测试例子 3, 4, 9 和表 2 对应的压缩率结果，本文方法的压缩率比文献[8]方法的压缩率要高很多，这主要是因为这 3 个测试例子中的仿真时间点之间差值非常小，非常适合于使用本文提出的压缩存储格式(包含 S_t)进行变长压缩和二次压缩. 从实验结果可知，本文方法比文献[8]方法能够更有效地进行压缩，这主要得益于第 2.1 节提出的针对信号名压缩部分的改进以及更有效的数字波形文件压缩存储格式.

3.3　验证压缩算法和解压缩算法的高效性

第 2.3 节表明，本文方法可以通过 3 阶段流水线的方法进行加速. 因此，本文用C++标准库实现了一个并行版本的数字波形压缩和解压缩程序. 流水线的每个阶段可以被实现为一个线程，其中线程的实现可以使用 std::async 函数. 一个线程移送处理过的数据到下一个线程的过程可以用一个阻塞队列实现，其中阻塞队列可以用 std::condition_variable 和 std::mutex 实现. 基于流水线技术的并行数字波形压缩和解压缩程序也在 9 个测试例子上进行测试，本文方法以及其他对比对象的运行时间都在表 3 中. 表 3 中的 T_{comp} 表示从波形数据编码压缩到写入压缩文件的实际运行时间，T_{decomp} 表示从压缩文件读取到恢复波形数据的实际运行时间. 从表 3 中本文方法的运行时间结果可以看到，信号值编码和解码过程的时间在 3 个阶段中占据了主要部分，又由于逐块进行数据处理和并行计算，因此总体压缩过程或解压缩过程运行时间仅比编码或解码过程多一点. 文献[8]方法的压缩时间以及解压时间均比本文的方法要长一些，在测试例子 9 上，本文方法在压缩和解压缩过程分别用了 1 411 s 和 2 381 s，而文献[8]方法则分别用了 2 125 s 和 7 509 s.

表 2　不同方法对测试例子的压缩比结果比较

测试例子	原始文件/MB	bzip2[13]		文献[8]		本文			
		压缩文件/MB	R_1	压缩文件/MB	R_2	压缩文件/MB	R_3	R_3/R_1	R_3/R_2
1	42	4.5	9.3	4.4	9.5	3.4	12.4	1.3	1.3
2	98	5.5	17.8	7.8	12.6	3.0	32.6	1.8	2.6
3	216	6.0	36.0	3.5	61.7	0.3	720.0	20.0	11.7
4	519	56.0	9.3	46.0	11.3	2.0	259.5	28.0	23.0
5	2 458	214.0	11.5	175.0	14.0	124.0	19.8	1.7	1.4
6	8 090	1 229.0	6.6	391.0	20.7	255.0	31.7	4.8	1.5
7	10 035	1 331.0	7.5	200.0	50.2	98.0	102.4	13.6	2.0
8	16 384	3 379.0	4.8	1 638.0	10.0	1 126.0	14.5	3.0	1.5
9	29 696	1 843.0	16.1	732.0	40.6	96.0	309.3	19.2	7.6

表 3　不同方法对测试例子的压缩与解压缩速度比较　　s

测试例子	原始文件/MB	bzip2[13]		文献[8]		本文							
		T_{comp}	T_{decomp}	T_{comp}	T_{decomp}	编码时间	二次压缩时间	写文件时间	T_{comp}	读文件时间	二次解压缩时间	解码时间	T_{decomp}
1	42	7.0	1.0	5.6	5.9	1.2	0.8	0.0	1.5	0.1	0.4	1.7	1.9
2	98	10.0	2.0	9.3	20.5	5.3	4.7	0.0	6.1	0.0	2.8	7.4	7.5
3	216	20.0	5.0	14.6	44.1	10.0	10.5	0.0	12.0	0.1	3.1	15.7	15.9
4	519	57.0	11.0	52.4	193.0	45.3	46.8	0.1	47.0	0.1	13.1	46.5	47.0
5	2 458	150.0	57.0	215.0	454.0	106.0	81.0	0.1	117.0	0.1	58.0	158.0	159.0
6	8 090	697.0	240.0	2 233.0	4 156.0	1 048.0	385.0	0.2	1 128.0	0.7	233.0	1 040.0	1 044.0
7	10 035	1 392.0	256.0	2 210.0	3 615.0	1 015.0	109.0	0.2	1 112.0	0.3	64.0	1 168.0	1 169.0
8	16 384	1 430.0	501.0	4 070.0	5 906.0	2 256.0	897.0	0.3	2 387.0	1.6	520.0	1 929.0	1 935.0
9	29 696	2 364.0	791.0	2 125.0	7 509.0	1 248.0	1 066.0	0.1	1 411.0	1.2	483.0	2 380.0	2 381.0

4　结　语

逻辑仿真的输出波形主要包含信号跳变时刻信号值，还包含全部信号名、信号类型、信号位宽等辅助信息. 本文首先提出对辅助信息的压缩处理方法，然后针对信号值数据的特点改进了现有

工作中的信号名压缩方案, 并据此提出了一种更高效的数字波形压缩存储格式以及对应于 VCD 文件的压缩文件存储格式. 所提出的数字波形压缩存储格式有利于变长编码压缩的同时, 可以使用通用压缩算法进行二次压缩, 进一步提升了压缩率. 结合以上技术, 得到了适用于集成电路逻辑仿真器的高效数字波形压缩和解压缩算法. 通过引入并行策略, 压缩和解压缩算法能够以 3 阶段流水线模式运行, 在缩短压缩和解压缩时间的同时, 能更好地与逻辑仿真器相结合. 实验显示, 本文方法的压缩率比现有工作的压缩率要更高, 压缩和解压缩速度要更快.

参考文献(References):

[1] Liu Jun, Han Yinhe, Li Xiaowei. Test data compression technique using dynamic updating reference slices for IP cores[J]. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(11): 2013-2020(in Chinese)
(刘军, 韩银和, 李晓维. 动态更新参考切片的 IP 核测试数据压缩方法[J]. 计算机辅助设计与图形学学报, 2010, 22(11): 2013-2020)

[2] Zhan Wenfa, Wu Qiong, Cheng Yifei, *et al*. Integrated circuit test data compression scheme built-in generalized folding technology[J]. Journal of Computer-Aided Design & Computer Graphics, 2017, 29(8): 1542-1548(in Chinese)
(詹文法, 吴琼, 程一飞, 等. 嵌入广义折叠技术的集成电路测试数据压缩方案[J]. 计算机辅助设计与图形学学报, 2017, 29(8): 1542-1548)

[3] Livinston F, Magotra N, Stearns S, *et al*. Real time implementation concerns for lossless waveform compression[C] //Proceedings of the IEEE International Symposium on Circuits and Systems. Los Alamitos: IEEE Computer Society Press, 1995: 1267-1270

[4] Hatami S, Feldmann P, Abbaspour S, *et al*. Efficient compression and handling of current source model library waveforms[C] //Proceedings of Design, Automation & Test in Europe Conference & Exhibition. Los Alamitos: IEEE Computer Society Press, 2009: 1178-1183

[5] Hatami S, Pedram M. Efficient representation, stratification, and compression of variational CSM library waveforms using robust principle component analysis[C] //Proceedings of Design, Automation & Test in Europe Conference & Exhibition. Los Alamitos: IEEE Computer Society Press, 2010: 1285-1290

[6] Liu Yi, Yang Fan, Zeng Xuan. An efficient compression method for analog waveforms[J]. Journal of Fudan University: Natural Science, 2013, 52(4): 72-77(in Chinese)
(刘毅, 杨帆, 曾璇. 一种高效的模拟波形数据压缩方法[J]. 复旦学报: 自然科学版, 2013, 52(4): 72-77)

[7] Saurabh S, Mittal P. A practical methodology to compress technology libraries using recursive polynomial representation[C] //Proceedings of the 31st International Conference on VLSI Design. Los Alamitos: IEEE Computer Society Press, 2018: 301-306

[8] Naroska E, Ruan S J, Ho C L, *et al*. A novel approach for digital waveform compression[C] //Proceedings of the Asia and South Pacific Design Automation Conference. New York: ACM Press, 2003: 712-715

[9] Deutsch P. RFC 1951-1996: DEFLATE compressed data format specification version 1.3[S]

[10] Yazdanpanah A, Hashemi M R. A new compression ratio prediction algorithm for hardware implementations of LZW data compression[C] //Proceedings of the 15th CSI International Symposium on Computer Architecture and Digital Systems. Los Alamitos: IEEE Computer Society Press, 2010: 155-156

[11] IEEE Standard Association. IEEE Std 1800TM-2005 IEEE standard for systemverilog-unified hardware design, specification, and verification language[S]

[12] Deutsch P. Gzip file format specification version 4.3[OL]. [2020-11-23]. https://dl.acm.org/doi/pdf/10.17487/RFC1952

[13] Seward J. bzip2 and libbzip2, version 1.0.5: a program and library for data compression[OL]. [2020-11-23]. https://www.sourceware.org/bzip2/

[14] WAPTM WSP WAP-203-WSP-2000 wireless application protocol, wireless session protocol specification[S]

(上接第 1785 页)

[32] Kelkar B A, Rodd S F, Kulkarni U P. A novel parameter-light subspace clustering technique based on single linkage method[J]. Journal of Information & Knowledge Management, 2019, 18(1): 1950007

[33] Yamamoto R, Sako S, Nishimoto T, *et al*. On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar[OL]. [2020-11-23]. https://hal. inria.fr/inria-00104743/document

[34] Mendes A, Backhouse R, Ferreira J F. Structure editing of handwritten mathematics: Improving the computer support for the calculational method[C] //Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces. New York: ACM Press, 2014: 139-148

[35] Le A D, Nakagawa M. Speedup of parsing for recognition of online handwritten mathematical expressions[C] //Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition. Los Alamitos: IEEE Computer Society Press, 2017: 896-901

[36] Young G, Zavelina L, Hooper V. Assessment of workload using NASA task load index in perianesthesia nursing[J]. Journal of Perianesthesia Nursing, 2008, 23(2): 102-110

[37] Kajler N, Soiffer N. A survey of user interfaces for computer algebra systems[J]. Journal of Symbolic Computation, 1998, 25(2): 127-159

[38] Brown C M. Comparison of typing and handwriting in “two-finger typists”[J]. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 1988, 32(5): 381-385

[39] Le A D, Indurkhya B, Nakagawa M. Pattern generation strategies for improving recognition of handwritten mathematical expressions[J]. Pattern Recognition Letters, 2019, 128: 255-262