可重构阵列的同步性能优化算法

张元瑞 武继刚 段新明
(天津工业大学计算机科学与软件学院 天津 300387)

摘 要 可重构多处理器阵列上的容错技术可用来重构含有故障单元的处理器阵列,以便获得最大可用的目标阵列。现有的研究成果主要侧重于重构算法的构造,还没有涉及对重构后目标阵列的同步通讯性能的研究。提出了一种改善目标阵列同步通讯性能的电路优化算法,用来降低目标阵列行与行之间通讯的延时,使得相邻两行处理器的通讯尽可能达到同步。实验结果表明,提出的算法对不同大小、不同故障率的阵列都有相应的同步通讯性能的改善。

关键词 超大规模集成电路(VLSI)处理器阵列,重构算法,容错,同步优化算法
中图法分类号 TP303 **文献标识码** A

Improved Algorithm for Communication Synchronization on Reconfigurable Mesh with Faults

ZHANG Yuan-rui WU Ji-gang DUAN Xin-ming
(School of Computer Science and Software, Tianjin Polytechnic University, Tianjin 300387, China)

Abstract Fault-tolerant technique for reconfigurable multiprocessor array deals with the issue of reconstruction of the processor array which contains fault units to get the largest available target array. Previous research focused primarily on the reconfiguration algorithm, which does not involve in the study of the synchronous communication performance for reconstructed target array. This paper proposed an optimization algorithm which can improve the performance of the synchronous communication on target array as it reduces the communication delay between neighboring rows for the target array. Experimental results show that the proposed algorithm achieves improvement on communication synchronous performance on processor arrays with different scales and different fault densities.

Keywords VLSI array, Reconfiguration algorithm, Fault-tolerance, Synchronous optimization algorithm

1 相关工作

二维网状(mesh)的处理器阵列具有规整的结构,能够高效地对图像和数据等信息进行处理。近年来,超大规模集成电路(Very Large Scale Integrated circuites, VLSI)和晶片规模集成电路(Wafer Scale Integration, WSI)的集成技术和工艺发展得越来越成熟,VLSI和WSI阵列集成密度不断提高,单一芯片上集成的处理单元数量呈指数倍增长,芯片生产使用过程中内部处理单元出现故障的概率将会大大增加。这些故障单元将会影响整个系统的可靠性。因此,有必要使用有效的容错技术对含有故障处理器的VLSI阵列进行重构从而充分发挥剩余的处理器的功效,以提高芯片的可靠性。现有的容错技术主要有两种:冗余方法[1,2]和降阶重构方法[3,4]。本文的工作主要基于降阶重构方法来展开。关于降阶重构方法的研究文献有很多。S. Y. Kuo 和 I. Y. Chen [5,6]提出了3种不同的选路约束条件:1)行、列穿越方式;2)行穿越、列选路方式;3)行、列选路方式。通过研究,他们证明了大多数基于这3种约束条件而产生的问题都是NP完全问题,同时他们针对这类问题提出了启发式的解决算法。

C. P. Low 和 H. W. Leong[7,8]对文献[5]所提算法进行改进,提出贪心的列路选算法(GCR算法),其在多项式时间内找到近似最优解,并给出了算法的详细描述和证明过程。由于第二种选路约束在重构时具有更大的灵活性,后续的很多研究都偏向于以此为模型。文献[11]基于启发式策略和动态规划思想对文献[8]中的算法构造的目标子阵列进行优化,提出了基于动态规划策略的局部最优算法(LDP算法),其在贪心的列选路算法运行结果的基础上,采用动态规划的策略,对运行结果进行再次优化,减少了重构后VLSI内部的开关使用数目和连接通路数,从而减少了VLSI处理器阵列的能耗,提高了性能。

当今对VLSI阵列的容错技术的研究主要侧重于VLSI阵列的重构算法的构造,还没有涉及对重构后目标阵列的同步通讯性能的研究。本文从这一新的角度入手,提出了一种改善目标阵列同步通讯性能的电路优化算法,用来降低目标阵列行与行之间通讯的延时,使得相邻两行处理器的通讯尽可能达到同步。我们进行了大量实验,来检验算法的性能。

本文第2节介绍一些基本定义以及本文的研究目标;第3节就本文提出的同步性能优化算法进行论述;第4节展示实验结果并对其进行分析;最后进行总结。

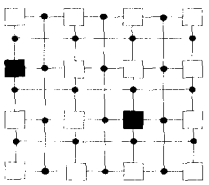
到稿日期:2011-04-16 返修日期:2011-07-20 本文受国家自然科学基金项目(60970016)资助。
张元瑞(1986—),男,硕士生,主要研究方向为高性能算法,E-mail:zyr2006a@163.com;武继刚(1963—),男,教授,CCF会员,主要研究方向为理论计算机科学并行分布计算;段新明(1976—),男,博士,副教授,硕士生导师,主要研究方向为并行计算机互连网络、无死锁路由算法。

2 基本定义与研究目标

本文中的大部分定义和一些术语来自相关参考文献。

对于生产出的 VLSI/WSI 芯片，原始处理器阵列称为主阵列[5-9]，文中用 H 表示。阵列中的每个处理器都被称作单元或元素，用 e 表示。主阵列中包含正常工作的处理单元和故障单元。由主阵列的正常单元重组得到的子阵列被称为目标阵列或者逻辑阵列，用 T 表示。主阵列中的行(列)被称为物理行(物理列)，目标阵列中的行(列)称为逻辑行(逻辑列)。N 表示主阵列中故障元素的数量。

本文沿用文献[6-10]中提出的假设：芯片内部连接线路，控制器和开关都不会出现故障，发生故障的只有处理器；并采用与文献[5-9]中相同的 VLSI 体系结构模型。内部结构如图 1 所示。图 1(a)中，方块代表处理器或者处理单元。圆圈代表连接相邻处理器之间的开关，其结构和功能是同等的，均为 4 口的开关，开关的状态如图 1(b)所示，处理器和开关之间的连线为 VLSI 内部链接通道/导线。



(a)基于 4 口开头联接的 4×4 规模的处理器阵列

(b)阵列内部开关状态

图 1 VLSI 阵列结构模型

文献[11]中对长连接和短连接的概念也进行了相关描述。

如图 2 所示，原始物理阵列是一个 5×5 的阵列，其内部结构与图 1 所示相同，这里为了突出显示，省略了部分处理单元之间的导线。其中随机分布着故障处理单元(黑色实心方框)，图中由列选路的相关重构算法得到了包含 3 个逻辑列的目标阵列。图中对处理单元进行了编号，如 12 表示第一行第二个处理单元。阵列中两个处理器之间的连接方式根据使用的开关数目分为两类，第一类如 13-23，或 15-25 等连接(本文中用两个处理器的编号加"-"表示连接这两个处理器的开关和导线)，只使用了一个开关，称为"短连接"，定义其连接长度为一个单位；另一类如 12-21 等的连接，使用两个开关，称为"长连接"，长度为两个单位。图中可以看到，"长连接"所使用的导线长度大致为"短连接"的两倍，而且要多用一个开关，这样就会增加功耗，并且信息的传递将产生延时。

下面对本文中涉及的同步和延时的概念进行阐述。

使用 VLSI 处理器阵列进行并行处理时，位于同一行的处理器需要保持高度同步。如图 2 所示，第五行的处理器可以认为能够同步接收到第四行处理器传来的信息，因为各逻辑列中可用处理器之间的连接都是短连接。但其他如第一行与第二行之间，由于长连接 12-21 的存在，造成目标阵列中信息由第一行传递到第二行时不同步，这两行之间的信息延迟必须按照长连接计算，这样就存在一个延时。第二行与第三行，第三行与第四行由于分别存在长连接 23-32 和 35-44，信息传递也不能同步，分别存在一个延时。这样从上往下整个目标阵列的延时为 3。

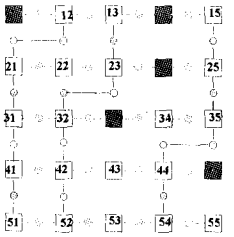


图 2 调整前的目标阵列

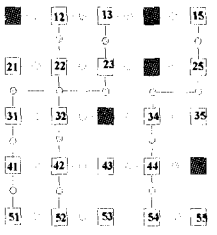


图 3 调整后的目标阵列

定义 1 目标阵列中相邻两行处理单元之间存在一个以上长连接，则这两行处理单元并行传递信息时，通信时间要按长连接计算，时间开销明显大于短连接，则该两行之间通信延时记为 1。若这两行处理单元之间都是短连接，则延时记为 0。整个阵列的总延时是各行通讯延时之和。

定义 2 对于一个给定大小的目标阵列，通过调整目标阵列中各长连接的位置，来减少整个阵列从上到下的通讯延时，这个过程称为阵列的同步性能优化。

图 3 是经过同步性能优化后的目标阵列。通过调整目标阵列中长连接 12-21，35-44 的位置，使长连接分布在同一行中，这样，目标阵列的延时变成了 1。由此可见，为了最小化整个阵列从上到下的通讯延迟，调整各长连接(斜边)的位置是必要的。本文对长连接的移动是指在阵列中上下平行移动。例如图 2 中的长连接 12-21 平行移动后变成图 3 中的 22-31。

经过对一些相关的研究方法的回顾以及体系结构的选择[12]和相关术语的定义，我们可以提出本文的研究目标：

1. 针对 VLSI 阵列重构问题考查新的研究方向——目标阵列的同步性能优化；并提出目标阵列的同步性能优化算法。

2. 对新算法进行软件模拟，收集实验数据，检验并分析新算法的性能。

3 目标阵列的同步性能优化算法

本节将对提出的可重构逻辑列的同步优化算法进行描述。其基本设计思想是：通过调整重构后的逻辑阵列(目标阵列)中长连接的位置，来减少目标阵列的延时，达到同步效果。该同步算法可以对相关重构算法，如 GCR，LDP 等构造的目标阵列进行同步性能优化，输出同步后延时减少的目标阵列。

在同步算法中，实现减少延时的主要方法是调整目标阵列长连接的位置，使长连接的分布尽可能地集中。算法对目标阵列中处理单元行与行之间的间隔从上往下进行编号为 $1,2,\cdots,n-1$(n 为目标阵列的总行数)。首先是扫描目标阵列，标记出所有的长连接，并确定每个长连接的活动范围，用集合表示；这里将长连接看作斜边，其活动范围是指在相同的逻辑列上能够平行的上下连续移动的范围，例如图 2 中，斜边 12-21 移动后变为图 3 中的 22-31，则其范围为{2,3}(范围集合中的元素即为斜边可以移动到的行间隔号)，这样每个斜边对应一个集合。其次是对所有集合进行求交运算，使尽可能多的集合合并到尽可能少的交集，有交集说明斜边能够移动到同一行间隔中。最后是根据求交运算的结果来移动斜边，若斜边当前位置与求交后的位置不一致，则移动斜边。不难看出，同步性能优化算法具有平方阶的时间复杂度。算法的描述如下：

Algorithm SYN_OPT;/* 同步性能优化算法 */

输入：重构的阵列 T，包含 $n\times n$ 个处理器和 m 个长连接；
输出：经过同步优化后的阵列 T；
Begin
1 对目标阵列中处理单元行与行之间的间隔从上往下进行编号为 1，2，…，$n-1$；
2 for $i_1:=0$ to m do Loc(a_i)：=(x_i，y_i)；/* 扫描阵列 T，找出所有长连接，进行编号 a_i 并记录其位置信息 */
3 for $i_1:=a_1$ to a_m do
begin
$R_i:=$ {长连接 a_i 可以移动到的行间隔的编号}；
/* 检测长连接 a_i 可移动的行间隔，用集合 R_i 存储这些行间隔编号 */
$C_1=\{R_1,R_2,R_3,\cdots,R_m\}$；/* C 为所有长连接集合的集合 */
$S:=\emptyset,S':=\emptyset$；
end
/* 检测行间隔编号 i 是否包含在长连接移动范围集合 R_j 中 */
4 for $i_1:=1$ to $n-1$ do
begin
for $j:=1$ to m do
begin
count:=0；/* 用于统计 S 中集合的个数 */
/* 如果长连接 a_j 能够移动到位置 i，则将 R_j 的链接加入集合 S */
if $i\in R_j$ then $R_j\rightsquigarrow S$，count++
end
if $S\neq\emptyset$ then /* 若 S 不空，则说明有多个斜边集合含同一行间隔号 */
对 S 中的集合求交集得 S'
for $i_1:=1$ to count do
令 $R_i=S'$ /* 修改 S 中包含的斜边集合，即是修改 C 中的对应集合 */
end
5 for $i_1:=1$ to m do /* 根据求交结果来移动长连接 */
begin
$x:=$ Loc(a_i)；/* x 是 a_i 当前位置——长连接 a_i 当前位于的行间隔 */
if $x\notin R_i$ then
移动斜边 a_i 到 R_i 中第一个元素所代表的行间隔位置
end
end

为了更好地理解算法，下面举例进行分析说明：如图 4 所示，主阵列是 8×10 的物理阵列，其中有规模为 8×6 的重构后的目标阵列，现要将此目标阵列进行同步性能优化。具体操作步骤如下。

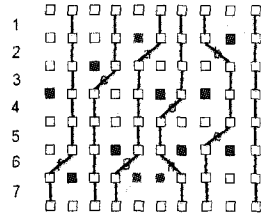


图 4　同步前阵列图

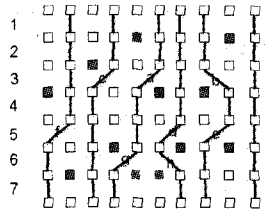


图 5　同步后阵列图

第一步　如图所示，将行间隔编号为 1，2，3，…，7，对于每个长连接，以行为序从上到下编号为 $a,b,c,\cdots,h$，同时记录所有斜边的当前位置。然后，将每个长连接的活动范围用集合表示。集合表示如下：

a：{2，3}，b：{2，3}，c：{3，4，5}，d：{4，5}，e：{4，5}，f：{4，5，6}，g：{6}，h：{6}

此处每个斜边编号对应一个集合，集合元素为斜边可以移动到的行间隔的编号。

第二步　按行间隔号从小到大的顺序检查每个集合。从 1 开始，检查每个集合，检查完所有集合，没有斜边对应集合包含 1。进入下一轮，检查 2，照例将所有集合检查一遍，找到斜边编号 a,b 对应的集合含有 2，对其求交，得[a,b]：{2，3}，将参与求交的集合 a：{2，3}，b：{2，3}，替换为[a,b]：{2，3}，当前的集合变为：

[a,b]：{2，3}，c：{3，4，5}，d：{4，5}，e：{4，5}，f：{4，5，6}，g：{6}，h：{6}。

进入第三轮，检查 3，并将含 3 的斜边对应的集合求交，图中为[a,b]：{2，3}和 c{3，4，5}求交。结果为[a,b,c]：{3}。

当前的集合变为：

[a,b,c]：{3}，d：{4，5}，e：{4，5}，f：{4，5，6 }，g：{6}，h：{6}。

依此类推，直到检查完所有行间隔号(检查到 7)。

最后集合变为：[a,b,c]：{3}，[d,e,f]：{4，5}，[g,h]：{6}。

第三步　根据以上结果，对斜边进行平行移动(看斜边当前位置是否位于求交后的集合中，若没有，则移动斜边)：将 a,b 移动到第 3 个空档(c 当前位于第 3 个空档，不需要移动)，将 d,f 移动到第 5 个空档(e 不需要移动)，g,h 位于第 6 个空档，不需要移动。

同步优化过程结束。图 5 是图 4 中的阵列同步后的效果图。可以看出，延时从图 4 中的 5 个降至了图 5 中的 3 个，最大限度地达到了同步的效果。

我们已经知道，通过 GCR 算法对物理阵列进行重构，能够得到一个规模(指得到的阵列的行列个数)最大的目标阵列[8]，但是有些情况并不要求得到最大规模的目标阵列，而是在不超过阵列可重构的最大规模的情况下，给出一个特定规模的目标阵列。本文采用分治的策略，对相关重构算法(如 GCR 或 LDP)的目标阵列进行删除逻辑列操作，以此来满足对阵列规模的要求。此处删除逻辑列指释放该逻辑列所占的处理单元，使其变为可再被利用状态。即用分治策略删除逻辑列，释放掉逻辑列占的处理单元，使剩余逻辑列均匀分布在芯片中。这样就能得到规模符合要求的目标阵列。我们也对这种特定规模的目标阵列进行了同步性能优化，以更好地展现同步性能优化算法的效果。

本文为了考察算法的性能，对 LDP 算法构造的目标阵列进行了同步实验，并考查了用分治策略对 GCR 算法构造的阵列进行删除逻辑列，再进行 LDP 算法优化后得到的目标阵列的同步性能优化效果。

4　实验结果及分析

在实验数据的采集过程中，为了客观地展示算法的性能优劣，构造逻辑阵列时沿用了以往算法的假设和模型。使用随机错误发生器对故障单元进行定位，这就保证了实验结果是算法在故障单元随机分布的原始阵列上的执行结果。这种情况通常发生在系统实时运行的过程中，从而，本文的实验结果能够较好地模拟现实中的实例。为了减小外界环境的影

响，本文的实验结果是30个随机例子的平均值。实验主要收集了不同大小的主阵列在不同故障率下的实验结果。

我们用C语言实现了特定阶的目标阵列的构造及可重构逻辑列同步优化算法，并用同步优化算法对特定阶的高性能目标阵列进行同步优化，来考察同步算法的性能。运行环境为PentiuDual-Core 2.59 GHz 1.0 GB RAM的处理器。

图6—图8分别给出了在规模32×32，64×64，128×128 3种情况下，故障率变化时，同步前GCR算法得到的阵列、LDP算法得到的阵列以及对LDP算法阵列运行同步算法后阵列中各自的延时情况曲线。例如图6所示是在阵列规模为32×32时，故障率分别为1%，2%，3%，…，10%时的GCR算法、LDP算法，以及同步算法的延时变化曲线。

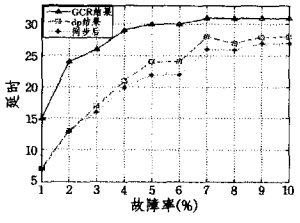


图6　阵列规模为32×32

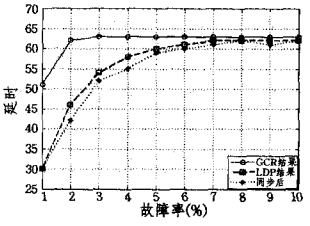


图7　阵列规模为64×64

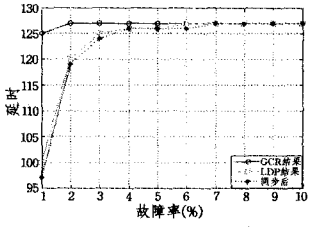


图8　阵列规模为128×128

该组实验是针对LDP算法重构后的目标阵列采用同步算法进行同步优化。我们收集了不同规模下的GCR算法延时、LDP算法延时，以及同步优化后的延时。可以看到LDP算法在GCR算法运行结果的基础上，采用动态规划的策略对运行结果进行优化，减少了重构后VLSI内部的开关使用数目和连接通路数，间接地降低了目标阵列的延时，而同步优化算法则是针对LDP算法的结果进行同步优化，达到了一定的效果。如图所示，特定规模下，当故障率较低的时候LDP算法延时改进很大，随着故障率的增加，改进率逐渐变小。同步算法也是如此。并且，结合图6—图8分析可知，改进率和阵列规模也有关系，当规模较小时，改进效果较为明显。对同步算法与LDP算法的比较可知，同步算法对LDP算法的改进有限，LDP算法得到的是局部最优的目标阵列，对它的同步优化效果并不明显。当阵列规模较大、故障率也较大时，逻辑阵列的分布情况越复杂，长连接很多，单个斜边可移动的范围也将受到影响，无论怎么调整，每一个行间隔中仍然存在长连接，所以同步算法在这种情况下几乎没有效果。

为了全面地展示同步算法的性能，我们对特定规模的目标阵列进行了同步优化实验，前文已经提到，对GCR重构后的目标阵列采用分治策略均匀删除逻辑列得到特定规模的目标阵列，然后对该阵列先后进行LDP算法优化和同步算法优化，同步算法仍然是直接对LDP算法进行同步优化。图9展示了部分实验结果。实验中，采集了阵列规模为64×64、故障率分别为5%，10%，20%时删除逻辑列占不同百分比的延时变化曲线。例如图(a)是故障率在5%时，删除逻辑列5%，10%，15%，20%，25%，30%，35%，40%，45%后的GCR算法、LDP算法、同步算法的延时变化曲线。

如图9所示，可以看到阵列规模和故障率一定时，对LDP算法的延时改进率(同步后的效果)随着删除逻辑列的增多而变大，当删除阵列占45%时，改进效果能够达到20%。这是因为删除逻辑列增多，目标阵列的斜边数大为减少，同时释放的处理单元也增加，即可用处理单元增加，从而使斜边活动范围变大，同步效果就会越明显。另外，结合图中(a)、(b)、(c)可知，同步算法也受故障率的影响，故障率越小，改进效果越好。

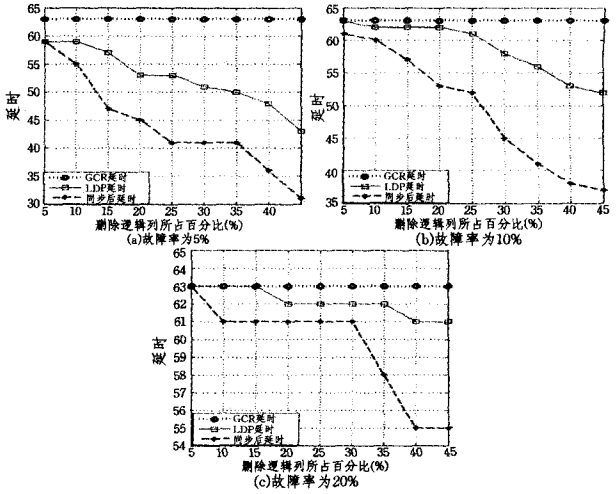


图9　阵列规模为64×64不同故障率下的同步效果

由此可见，同步算法受故障率、阵列规模，及同一规模下删除逻辑列个数等因素影响。同一规模下，故障率一定时，删除逻辑列占百分比越大，改进效果越好；同一规模下，删除相同百分比的逻辑列，故障率越小，改进效果越好。相同故障率时，阵列规模越小，改进效果越好。

结束语　本文提出了一种可重构逻辑列同步优化算法，用来降低目标阵列同步通讯的延时，并且用同步优化算法对LDP构造的目标阵列进行同步性能优化，减少了阵列的延时，并通过实验展示了同步算法的性能。

参考文献

[1] Chen Y Y, Upadhyaya S J, Cheng C H. A comprehensive reconfiguration scheme for fault-tolerant VLSI/WSI array processors [J]. IEEE Trans. Computers, 1997, 46(12): 1363-1371

[2] Horita T, Takanami I. Fault-tolerant processor arrays based on the 1.5-track switches with flexible spare distributions [J]. IEEE Trans on Computers, 2000, 49(6): 542-552

[3] Kung S Y, Jean S N, Chang C W. Fault-Tolerant Array Processors Using Single-Track Switches [J]. IEEE Trans. on Computers, 1989, 38(4): 501-514

[4] Roychowdhury V P, Bruck J B, Kailath T. Efficient Algorithms forReconfiguration in VLSI/WSI Arrays [J]. IEEE Trans. Computers, 1990, 39(4): 480-489

[5] Kuo S Y, Chen I Y. Efficient Reconfiguration Algorithms for Degradable VLSI/WSI Arrays [C] // Proc. Int'l Conf. Wafer Scale Integration. 1991: 120-126

(下转封三)

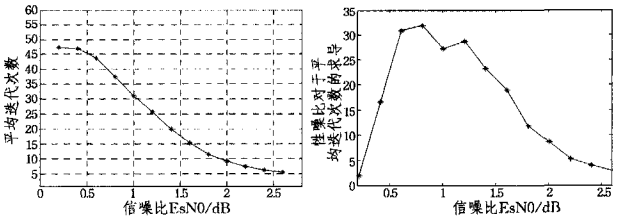


图5 平均迭代次数

本仿真平台在大幅度增加最大迭代次数时，其误码率性能改善虽不明显，但仍然可以看出改善的趋势。如果增大码长的同时较大幅度地增加最大迭代次数，可以进一步改善此仿真结果及性能。但是由于最大迭代次数的增加，会导致运算时间的增加，可能对一些时效性比较强的通信造成影响。

4.3 结论

本仿真平台的构建与模拟只进行了码长 $n=576$，码率 $Rate=1/2$ 下的调试运行。一些重要参数诸如最大迭代次数等都有待进一步研究得出更多的对比结论。通过分析，可得出以下一些结论。

(1)关于码长，从理论上得知 LDPC 码通常要在码长较长时才能发挥出优越性能，当码长为中短长度时，由于编码中会不可避免地出现一定的短长度环，就会在某种程度上降低编码性能，因此，如果条件允许，适当地增大码长，此仿真结果及性能应该有一定程度的改观。

(2)关于最大迭代次数，从仿真结果上可以得知，每一次迭代都会带来校验节点和信息节点之间对数似然比信息的传递，迭代次数越多，译码结果与信源发送码字之间的差别就越小，则译码性能就会越好。

(3)关于运算复杂度，由前文可知，LDPC 码的性能要在长码长的条件下才能发挥出来，但是从编译过程来看，LDPC 码运算复杂度较高，其间有大量的迭代循环运算，而在长码长的情况下又必须接收到所有的信息比特后才能开始编译，这就对时效性产生了很大的影响。

结束语 通过本仿真实现说明，MiMAX 下的 LDPC 码最为关键的还是改进译码算法和提高编码效率，并以此减少运算复杂度，减少通信时延。运用 Microsoft Visual C++语言进行 LDPC 编译码器仿真，进一步说明了解决 LDPC 码应用在移动通信技术中的关键问题，同时也说明了 VC 技术应用于 LDPC 编译码仿真的可行性、实用性和可扩展性。

参 考 文 献

[1] 蔡少春. LDPC 码及其在 WiMAX 中的应用[D]. 广州：中山大学，2007

[2] 樊昌信，曹丽娜，等. 通信原理(第 6 版)[M]. 北京：国防工业出版社，2006

[3] 王新梅，肖国镇. 纠错码-原理与方法[M]. 西安：西安电子科技大学出版社，1996

[4] 翁静兰，张世庆. 适用于 WiMAX 标准的 LDPC 码的仿真分析[J]. 电视技术，2008(z1)：117-119

[5] 胡潘，李珊珊. 基于 WiMAX 的 LDPC 码性能研究[J]. 通信技术，2011(04)：76-79

[6] 贺玉成. 基于图模型的低密度校验码理论及应用研究[D]. 西安：西安电子科技大学，2002

[7] Shannon C E. A mathematical theory of communication[R]. Bell Syst. Tech. J，27：379-423/623-656. July/Oct. 1948

[8] Gallager R G. Information Theory and Reliable Communication [M]. New York，Wiley，1968

[9] Garcia-Frias J，Wei Z. Approaching Shannon performance by iterative decoding of linear codes with low-density generator matrix [J]. IEEE Communications Letters，2003，7(6)：266-268

[10] IEEE Std 802. 16e. Air Interface for Fixed and Mobile Broadband Wireless Access Systems[S]. NY 10016-5997. IEEE，3 Park Avenue，New York，USA，February 2006

[11] 钟州. 基于 LDPC 码校验节点度的分类修正最小和算法[J]. 清华大学学报：自然科学版，2009，49(1)：45-48

[12] 刘德保. 低密度奇偶校验码及其应用研究[J]. 电子科技，2007(3)：78-81

[13] 徐勐. 用于 WiMAX 系统中 LDPC 译码器的改进型 Benes 网络结构[J]. Journal of Southeast University：English Edition，27(2)：140-143

[14] 胡新桂，孙刚. 一种 LDPC 码校验矩阵消短环算法[J]. 计算机工程与科学，2009，31(9)：156-158

[15] 冯云飞，李建平. 一种构造低密度奇偶校验码校验矩阵的方法[J]. 中国传媒大学学报：自然科学版，2008(12)：52-56

(上接第 298 页)

[6] Kuo S Y，Chen I Y. Efficient Reconfiguration Algorithms for Degradable VLSI/WSI Arrays[J]. IEEE Trans. Computer-Aided Design，1992，11(10)：289-301

[7] Low C P，Leong H W. On the Reconfiguration of Degradable VLSI/WSI Arrays[J]. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems，1997，16(10)：1213-1221

[8] Low C P. An Efficient Reconfiguration Algorithm for Degradable VLSI/WSI Arrays[J]. IEEE Trans. Computers，2000，49(6)：553-559

[9] Wu Ji-gang，Srikanthan T. An Improved Reconfiguration algorithm for Degradable VLSI/WSI arrays[J]. Journal of Systems Architecture，2003，49：23-31

[10] Wu Ji-gang，Srikanthan T. Partital Rerouting Algorithm for Reconfigurable VLSI Arrays[C]//Proc. of IEEE International Symposium on Circuits and Systems. Bangkok，Thailand，2003：641-644

[11] Wu Ji-gang，Srikanthan T. Reconfiguration Algorithms for Power Efficient VLSI Sub-Arrays with 4-port Switches[J]. IEEE Trans. on Computers，2006，55(3)：243-253

[12] 刘耿耿，王小溪，陈国龙，等. 求解 VLSI 布线问题的离散粒子群优化算法[J]. 计算机科学，2010，37(10)：197-201