高技术通讯 2012 年 第22卷 第4期:340 ~ 347

doi:10.3772/j.issn.1002-0470.2012.04.002

# 多核虚拟可重构结构加速逻辑电路演化设计的研究①

王 进② 李丽芳 任小龙

(重庆邮电大学计算机科学与技术研究所 重庆 400065)

**摘 要** 提出了一种用基于多核虚拟可重构结构(MuViRaC)的内部演化硬件来加速组合逻辑电路演化设计过程的方法。其主要思想是依据增量演化中的输出函数分解策略,将一个组合逻辑电路分解为多个具有更少输出的子电路。每个子电路在 MuViRaC 上以两阶段并行演化的方式进行演化。MuViRaC 在 Celoxica RC1000 PCI 板上的 Xilinx Virtex xcv2000E FPGA 上实现。MuViRaC 分别被应用于演化 3 位乘法器和 3 位加法器。试验结果证明 MuViRaC 能够有效地减少组合逻辑电路的演化代数和演化时间。

**关键词** 数字电路,逻辑电路,演化硬件(EHW),演化算法(EA),并行算法

## 0 引 言

受自然进化启示,演化硬件(evolvable hardware, EHW)方法作为一种不同于传统的人工电路设计的方法,已成为自动化电路设计的一种新模式[1]。通过利用演化算法对更广的设计空间进行启发式搜索,演化硬件可以不依赖设计者的先验知识得到相对于传统人工设计更优的电路设计方案[2]。演化硬件方法为有效设计大规模电路带来了新的突破。但由于演化硬件规模的制约,该方法要取代人工设计还有一定的难度。相对较大的电路,由于演化设计的可扩展性问题,应用现今已有的演化方法和计算资源较难得到[3-10]。影响演化硬件可扩展性的因素主要有两个[1]:(1)染色体长度:复杂电路需要相对较长的染色体进行编码,导致在极大的搜索空间很难得到较优的解;(2)演化算法(evolutionary algorithm, EA)的计算复杂度:随着目标电路的输入输出数目不断增长,EA 所需评估的个体数目成指数增加,另外,评估每个个体适应值的时间也将大大延长。目前,对演化硬件的可扩展性问题,学界已从函数级演化[3]、演化发展[6]、并行 EA[11]、增量演化[7,9,12,13]、多目标 EA[8]等不同角度上进行了探索,但都未能得到完全解决。

为了使演化硬件能够演化较为复杂的组合逻辑电路,在虚拟可重构结构(virtual reconfigurable architecture, VRA)的基础上[14-17],本文提出了基于多核虚拟可重构结构(multi-virtual reconfigurable architecture cores, MuViRaC)的组合逻辑电路并行演化设计方法。MuViRaC 首先依据增量演化的原则将单个组合逻辑电路分解为多个子电路[9,12],然后将其作为演化目标电路,采用两阶段并行演化的方式进行演化。在第 1 阶段,每个子电路通过一个 VRA 核心实现独立的演化,正确的子电路演化完成后,其对应的 VRA 核心的硬件资源即可释放,用于演化临近的未演化完毕的其他子电路;在第 2 阶段,通过调用 2 个 VRA 核心的硬件资源,MuViRaC 执行类似于 global parallel EA[11]的并行算法以完成剩余子电路的演化。在该并行演化过程中,一个适应值评估周期内 MuViRaC 可同时评估两个个体。最终,将所有的演化完成的子电路进行整合可得到期望的完整顶层电路。演化 3 位乘法器和 3 位加法器的实验证明,相对于传统的直接演化[16]和增量演化方法[7,9],MuViRaC 具有更快的演化速度和相似的硬件代价。

## 1 算法描述

本文的主要目的是提出一种有效的演化硬件技术,以加速组合逻辑电路的设计过程。MuViRaC 可通过加速 EA 运算,缩短染色体编码长度,减少任务本身计算复杂度的方法,达到预期目的。本文解决

① 重庆市自然科学基金(2009BB2080)和教育部留学回国人员科研启动基金(教外司留[2010]1174 号)资助项目。

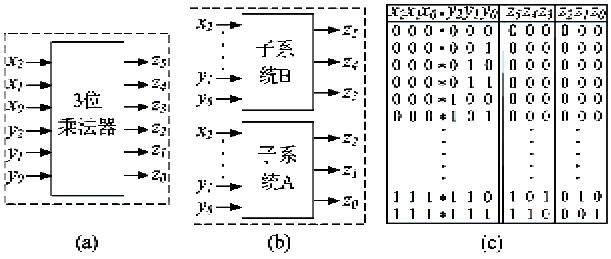
② 男,1979 年生,博士,教授;研究方向:演化硬件,模式识别,智能信息处理,生物信息处理;联系人,E-mail: wangjin_liips@yahoo.com.cn

(收稿日期:2010-09-08)

问题的核心思想是采用两阶段并行演化的策略演化由顶层系统拆分而成的子系统。组合逻辑电路的分解和整合依据增量演化策略[9, 12]。该方法有效地减少了 EA 的算法复杂度和染色体长度。本文使用名为 MuViRaC 的硬件结构实现在普通商用现场可编程门阵列(FPGA)电路上对分解后子系统的两阶段并行演化。FPGA 实现两阶段并行演化下层子系统是 MuViRaC 和传统的基于外部演化硬件的增量演化[7, 13]的主要不同。在传统增量演化过程中,子系统以成序列的方式通过软件仿真演化得到。

### 1.1　组合逻辑电路的分解

广泛用于 EHW 增量演化的组合逻辑电路分解策略包括训练集分解和系统输出函数分解[7, 13]。本文中,为了简化硬件实现,只在 MuViRaC 上应用了输出函数分解策略[9, 12]。通过使用输出函数分解策略,具有较多输出的顶层系统能够被分解为多个具有较少输出的子系统。图 1 说明了使用输出函数分解策略对一个 3 位乘法器进行拆分的过程。依据系统输出分解策略,一个 3 位乘法器的 6 位输出被划分为两组(如图 1(c)真值表中的竖线所示)。根据图 1(b),每一组 3 位输出被应用于演化其对应的子系统(子系统 A,子系统 B)。每个子系统均为具有 6 位输入/3 位输出的电路。演化后的子系统经整合后可以正确表达一个 3 位乘法器。尽管图 1 中只包含两个子系统,子系统的实际数目可依据具体情况进行划分。




(a)初始电路;(b)输出函数分解;(c)3 位乘法器真值表

**图 1　3 位乘法器的输出函数分解**

### 1.2　虚拟可重构结构(VRA)

作为实现内部演化硬件的一种有效途径,虚拟可重构技术近年来已被广泛用于不同的实际应用中[15, 16]。本文采用基于 VRA 的内部演化硬件[17]并行演化分解后的子电路。

我们使用基于传统的笛卡尔遗传程序(Cartesian genetic programming, CGP)[2, 16]的简化和改进模型对每个分解后的子电路进行演化算法基因编码。CGP 的表现型为 1 个二维的功能单元阵列结构。为了减少染色体长度,本文采用的 VRA 在传统 CGP 的二维功能单元阵列结构基础上加入了更多功能单元输入连接限制。如图 2 所示,在 MuViRaC 中,每个分解后的子系统都被表达为 1 个固定大小的 $n$ 列 $m$ 行的功能单元(function element,FE)阵列。每个 FE 包含 2 个输入和 1 个输出,可根据基因的功能选择设定执行 8 种不同的功能处理 FE 输入。所有的 FE 输入都是前反馈的。同 Sekanina 等人提出的的基于 CGP 的虚拟可重构电路(virtual reconfigurable circuit, VRC)[16]比较(VRC 中每个 FE 的输入

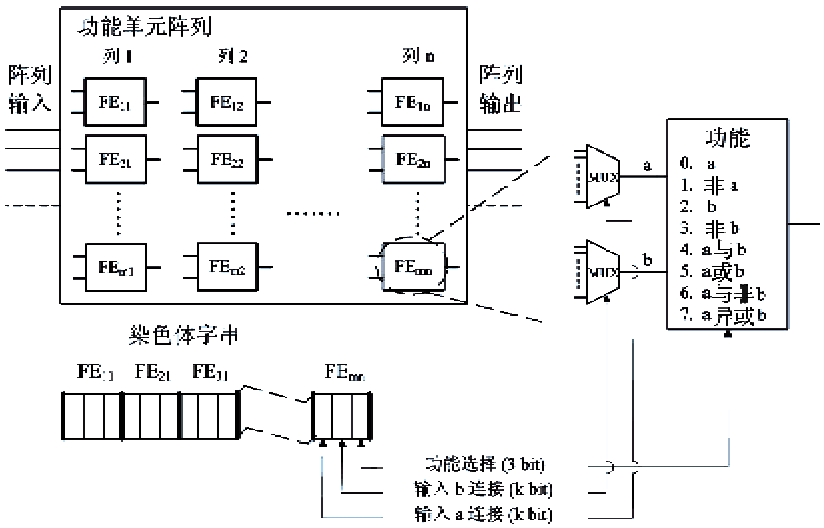
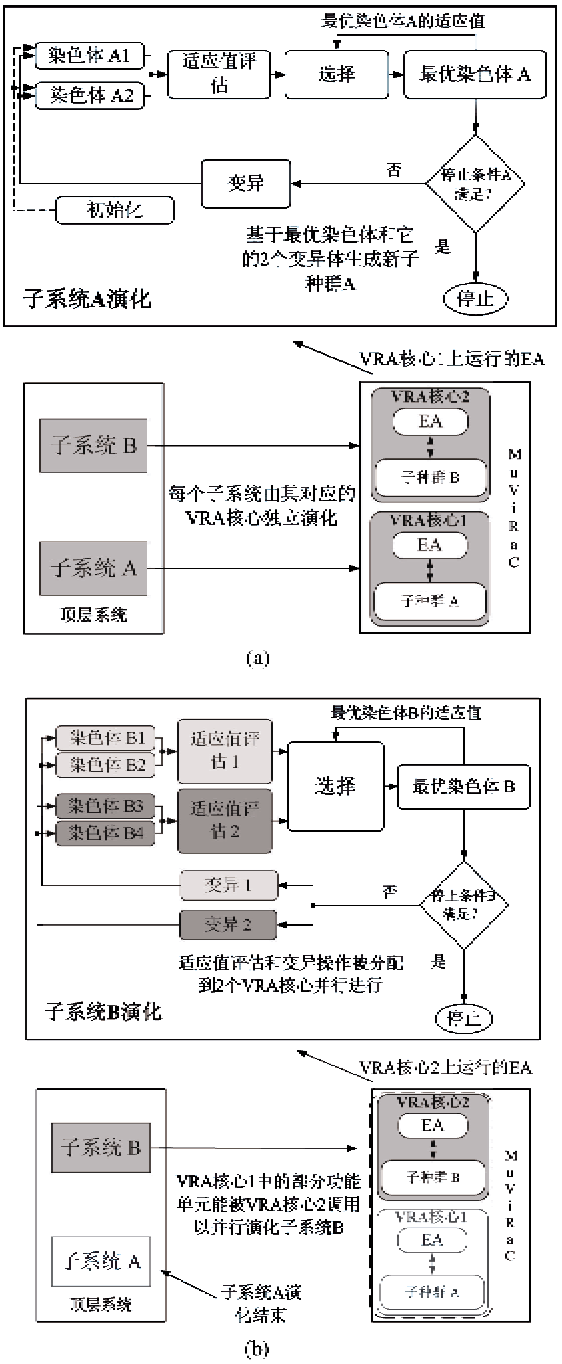



**图 2　VRA 的基因型-表现型映射**

可和 FE 阵列的外部输入或其先前列的 FE 输出进行连接),VRA 中每个 FE 的输入仅限于连接其前一列的 FE 输出,以达到减少基因型长度的目的。VRA 的基因型为固定长度的比特位串,其作用是定义 FE 阵列的内部连接和各个 FE 所执行的功能。VRA 的基因型和基因型到表现型的映射如图 2 所示。

### 1.3 双阶段并行演化算法

双阶段并行演化的算法模型由图3进行说明。




(a)第 1 阶段;(b)第 2 阶段

**图 3 用于演化 2 个子系统的两阶段并行演化算法流程**

注意算法描述中子系统的数目可以根据实际需要调整。在第 1 个并行演化阶段,EA 的种群被分为两个子种群,由两个独立的 VRA 核心执行。每个子系统的演化依据 1 + λ 演化策略( λ = 2 )。演化操作是基于精英选择和变异。第 1 阶段的 EA 运行流程如图 3(a)所示。

在我们早期的 MuViRaC 算法模型中[9],每个 VRA 核心只能对其对应的子系统进行独立演化。当某个子系统演化完毕后,它所对应的 VRA 核心的计算资源不能用于其它未演化完毕的子系统。然而在大部分实际应用中,每个分解后的子系统的计算复杂度都是不对等的。这就造成了每个子系统所需的演化时间有较大差异。所以,在我们早期的模型中存在较大的硬件资源浪费。为了提高 MuViRaC 的性能,本文在早期单一并行演化模式的基础上引入了额外的并行演化模式。在第 2 个并行演化阶段,其主要思想为运行中的 VRA 核心可以调用演化完子系统对应的 VRA 核心的硬件资源。VRA 核心的调用是以单向环拓扑(one-way ring topology)方式进行,即每个 VRA 核心仅能调用其邻近的一个 VRA 核心的资源。

第 2 个并行演化阶段的 EA 流程如图 3(b)所示。在该阶段,MuViRaC 以类似于 global parallel EA[11]的方式工作。EA 中只存在一个单一的种群(子种群 B,包含 4 个个体),但是个体适应值的评估和染色体的变异能被两个 VRA 核心同时执行。即 MuViRaC 能同步对 2 个个体进行适应值评估。EA 的选择操作作用于整个子种群 B。EA 的演化停止条件为:(1)EA 找到每个子系统的期望解;或(2)达到预先设定的演化代数。

在 MuViRaC 中,顶层系统的输出被分为几组以对应不同的子系统。因此在各个子种群中,每个个体的适应值

$$Fitness = \sum_{i=0}^{2^l-1} \sum_{j=0}^{k-1} (w_{ij} \ \mathrm{e} \ v_{ij}) \qquad (1)$$

通过对比子系统的实际输出和其对应的真值表中的期待输出得到。式中 $k$ 和 $l$ 分别表示子系统的输出和输入数目,$w_{ij}$ 和 $v_{ij}$ 分别为输出矢量和其对应期待输出矢量中的 1 位。每个输出矢量的每一位和其对应的期待输出相等,则适应值(*fitness*)增加 1。对于演化 1 个 6 输入/3 输出的 3 位乘法器的子系统而言,该系统包含 $2^6 = 64$ 三位输出矢量,其最大适应值为 $64 \times 3 = 192$。

## 2　MuViRaC 的 FPGA 实现

本文以演化3位乘法器(包含2个具有6输入/3输出的子系统)为例,说明MuViRaC在FPGA上的实现。如图4所示,使用的Celoxica RC1000 PCI板卡包括了1个Xilinx Virtex xcv2000E FPGA和8Mbyte板载SRAM。在Xilinx Virtex xcv2000E FPGA上实现的MuViRaC演化系统主要包含控制界面和两个等同的VRA核心。控制界面主要用于和主机及板载SRAM通信以及控制VRA核心的运行过程。每个对应6输入/3输出子系统的VRA核心都包括1个EA单元,1个适应值单元,以及1个功能单元阵列。EA单元执行演化操作以产生配置位串(染色体)对功能单元阵列进行配置。功能单元阵列用于处理系统输入。适应值单元通过将功能单元阵列的输出与系统期待输出进行对比,计算个体适应值。

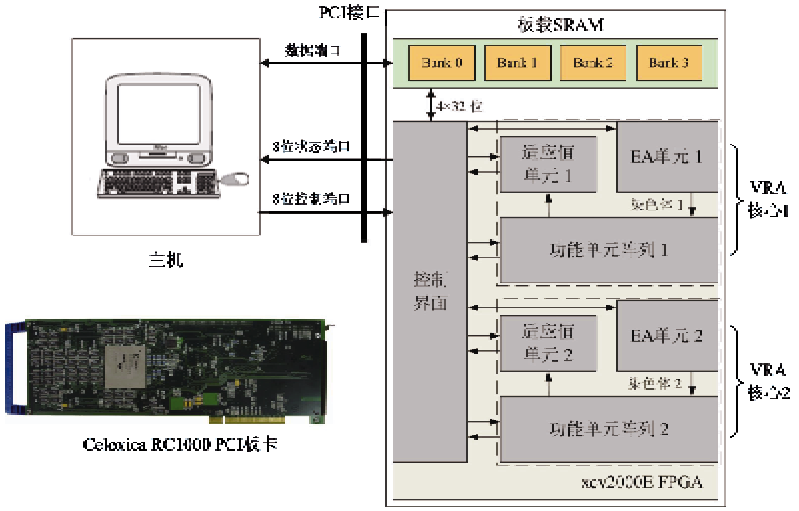



**图4　MuViRaC的组织结构图**

VRA核心中的功能单元阵列的具体FPGA实现过程已在[9,17]中进行了详细描述。与文献[9]不同,为了实现VRA核心间的硬件资源共享以进行第2阶段并行演化,作者对适应值单元和EA单元进行了相应改进。增加了多路选择器,以实现使用不同子系统期望输出值作为适应值单元输入。

图5为VRA核心2中的EA单元2的框图。每个EA单元都包括1个2×6×72位的种群内存,1个6×72位的最优染色体内存,1个8位适应值内存,1个随机数产生器,和一个变异操作器。在第1个并行演化阶段,MuViRaC中存在2个运行的EA单元(EA单元1和EA单元2)。每个EA单元在其对应的VRA核心中按传统的序列EA的方式运行[9,17]。在该阶段,每个EA单元在每个适应值评估周期内只能生成和评估一个个体。

当子系统A演化完毕后,EA单元1将停止工作,而EA单元1中的种群内存1,随机数产生器1,变异操作器1(图5中虚线框表示)等部件能被EA单元2调用以实现第2阶段并行演化。在该并行演化阶段,EA单元2能在每个适应值评估周期内通过

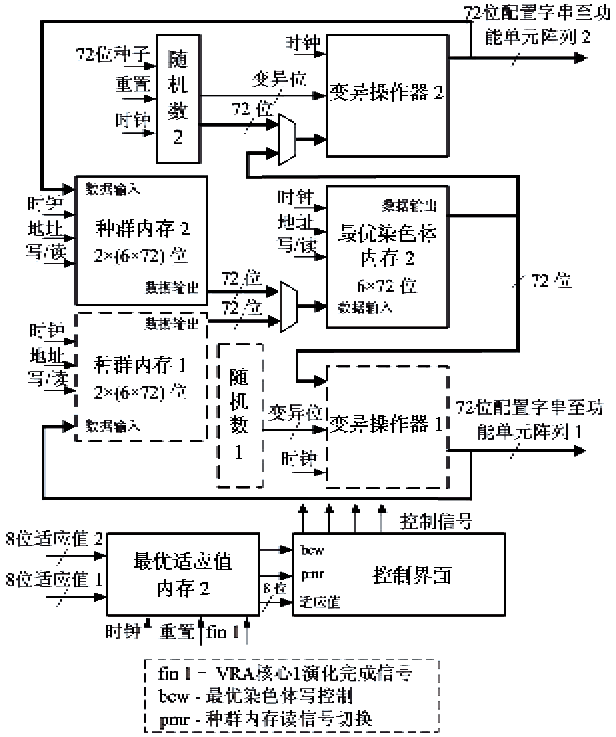



**图5　VRA核心2中的EA单元2及相关功能部件**

变异操作器 1 和变异操作器 2 生成 2 个最优染色体的变异体,并通过对 2 个功能单元阵列同步进行配置后计算 2 个变异体适应值。两个并行演化状态的切换是通过控制界面操作多个多路选择器实现。

## 3 实验结果

通过演化 3 位乘法器[7,9,16]和 3 位加法器[9,16]这 2 个 EHW 研究中常用的基准测试电路,对 MuRiVaC 的有效性进行验证。表 1 为 EA 参数和实验设定。为了证明相对于传统方法 MuRiVaC 可以用更短的演化时间和更少的演化代数演化出结果电路,我们在 FPGA 上对 4 种演化策略进行了对比试验:(1)直接演化[16];(2)基于输出函数分解的增量演化[12];(3)只包含第 1 个并行演化阶段的 MuViRaC[9](单阶段 MuViRaC);(4)本文提出的包含两个并行演化阶段的 MuViRaC(双阶段 MuViRaC)。

本文使用 VHDL 语言对 MuViRaC 进行设计。Xilinx ISE 6.3 工具对设计进行 Virtex xcv2000E FPGA上的综合与实现。表2和表3分别给出了3位乘法器和 3 位加法器的包括硬件实现代价(FPGA CLB Slice 占用)和 FPGA 最高运行频率($f_m$)等项的综合结果。根据 Xilinx ISE 6.3 给出的综合报告,MuViRaC 在所有应用中都能以大于 80 MHz 的 FPGA 频率运行。然而为了与 Celoxica RC1000 板卡上的 PCI 接口同步,所有演化系统的运行频率都被限制在了 33 MHz。

**表 1 EA 参数和实验设定**

| 参数 | | 3 位乘法器/3 位加法器 |
|---|---|---|
| FE 阵列大小 | 直接顶层系统演化 | 6 列,12 行 |
| | 子系统演化 | 6 列,8 行 |
| 变异比率 | | 0.8% |
| 变异操作方式 | | 按位变异 |
| 个体选择策略 | | 精英策略 |
| 每个实验设定下的 EA 运行次数 | | 100 次 |
| 种群大小(1+λ) | | 1+2 或 1+4 |
| 演化终止条件 | | 1)达到预先设定的演化代数,134,217,728;或 2)EA 得到期待结果电路。 |

**表 2 不同分解和演化策略设定下的 3 位乘法器演化结果**

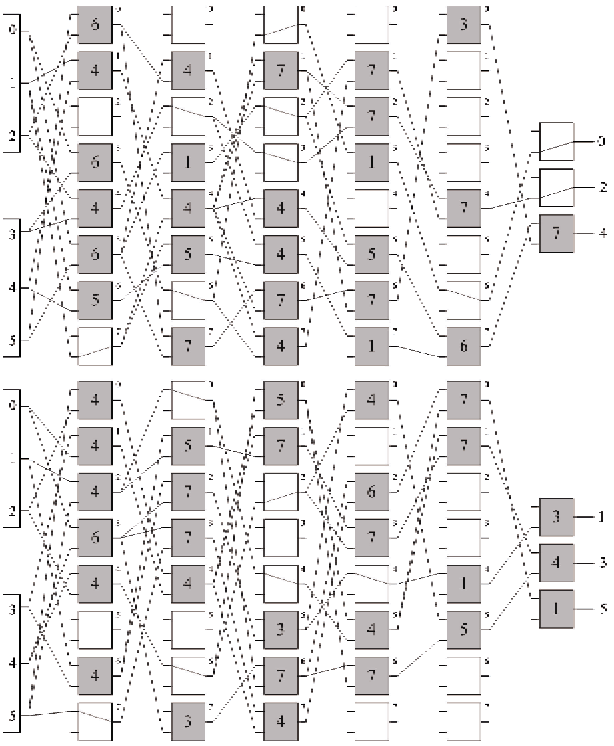
| 演化策略 | 系统分解 | 子系统类型 | 染色体(bits) | 硬件代价(slices) | $f_m$(MHz) | 平均演化代数 | 平均演化时间(s) | 演化成功率 |
|---|---|---|---|---|---|---|---|---|
| 直接演化 | 无 | 输出(0-5) | 702 | 4272 | 97.713 | 17283102 | 67.037 | 49% |
| 增量演化 | 方案 1 | 输出(0,2,4) | 387 | 2984 | 98.889 | 1621737 | 15.009 | 59% |
| | | 输出(1,3,5) | | | | 2247813 | | 61% |
| 单阶段 MuViRaC | | 输出(0,2,4)(1,3,5) | | 4505 | 98.348 | 2846295 | 11.040 | 56% |
| 双阶段 MuViRaC | | | | 4592 | 81.914 | 2043642 | 7.927 | 49% |
| 增量演化 | 方案 2 | 输出(0,1,2) | 387 | 2984 | 98.889 | 99584 | 17.829 | 49% |
| | | 输出(3,4,5) | | | | 4496935 | | 53% |
| 单阶段 MuViRaC | | 输出(0,1,2)(3,4,5) | | 4505 | 98.348 | 6658183 | 25.826 | 42% |
| 双阶段 MuViRaC | | | | 4592 | 81.914 | 3365791 | 13.055 | 46% |
| 增量演化 | 方案 3 | 输出(0,3) | 378 | 3423 | 98.889 | 527068 | 4.058 | 71% |
| | | 输出(1,4) | | | | 378962 | | 66% |
| | | 输出(2,5) | | | | 140277 | | 58% |
| 单阶段 MuViRaC | | 输出(0,3)(1,4)(2,5) | | 6422 | 98.348 | 688360 | 2.670 | 47% |
| 双阶段 MuViRaC | | | | 6547 | 81.914 | 497707 | 1.930 | 50% |
| 增量演化 | 方案 4 | 输出(0,1) | 378 | 3423 | 98.889 | 2481 | 3.109 | 58% |
| | | 输出(2,3) | | | | 631937 | | 50% |
| | | 输出(4,5) | | | | 167183 | | 68% |
| 单阶段 MuViRaC | | 输出(0,1)(2,3)(4,5) | | 6422 | 98.348 | 867483 | 3.365 | 38% |
| 双阶段 MuViRaC | | | | 6547 | 81.914 | 531025 | 2.600 | 46% |

表 3　不同分解和演化策略设定下的 3 位加法器演化结果

| 演化策略 | 系统分解 | 子系统类型 | 染色体(bits) | 硬件代价(slices) | $f_m$(MHz) | 平均演化代数 | 平均演化时间(s) | 演化成功率 |
|---|---|---|---|---|---|---|---|---|
| 直接演化 | 无 | 输出(0-3) | 680 | 4130 | 97.713 | 394470 | 1.530 | 46% |
| 增量演化 | 方案 1 | 输出(0,2) | 378 | 2948 | 98.889 | 47351 | 0.417 | 57% |
| | | 输出(1,3) | | | | 60133 | | 57% |
| 单阶段 MuViRaC | | 输出(0,2)(1,3) | | 4460 | 98.348 | 70315 | 0.273 | 53% |
| 双阶段 MuViRaC | | | | 4521 | 81.914 | 56563 | 0.219 | 58% |
| 增量演化 | 方案 2 | 输出(0,1) | 378 | 2948 | 98.889 | 1815 | 0.355 | 50% |
| | | 输出(2,3) | | | | 89763 | | 57% |
| 单阶段 MuViRaC | | 输出(0,1)(2,3) | | 4460 | 98.348 | 90992 | 0.353 | 41% |
| 双阶段 MuViRaC | | | | 4521 | 81.914 | 46801 | 0.182 | 41% |

### 3.1　3 位乘法器演化结果

图 6 所示为双阶段 MuViRaC 演化的一个 3 位乘法器(方案 1,包括 54 个逻辑门)。图中功能单元所执行的函数标号依据图 2 给出第 1 个实验为演化 6 输入/6 输出的 3 位乘法器。为证明结果的普适性,我们试验了 4 种不同系统分解方案:(1)方案 1:两个子系统,其输出分别为(0,2,4),(1,3,5);(2)方案 2:两个子系统,其输出分别为(0,1,2),(3,4,5);(3)方案 3:三个子系统,其输出分别为(0,3),(1,4),(2,5);(4)方案 4:其输出分别为(0,1),(2,3),(4,5)。在表2中,我们分别对比了不同实验设定下的子系统染色体长度、平均演化代数、平均演化时间及 EA 在 100 次运行中搜索出期待电路的次数(演化成功率)。所有的平均值都来自每个实验设定下 100 次 EA 运行的结果。图 6 给出了方案 1 系统分解设定下,双阶段 MuViRaC 演化的一个 3 位乘法器结果。



(方案 1,包括 54 个逻辑门)。
图中功能单元所执行的函数标号依据图 2 给出

**图 6　双阶段 MuViRaC 演化的一个 3 位乘法器**

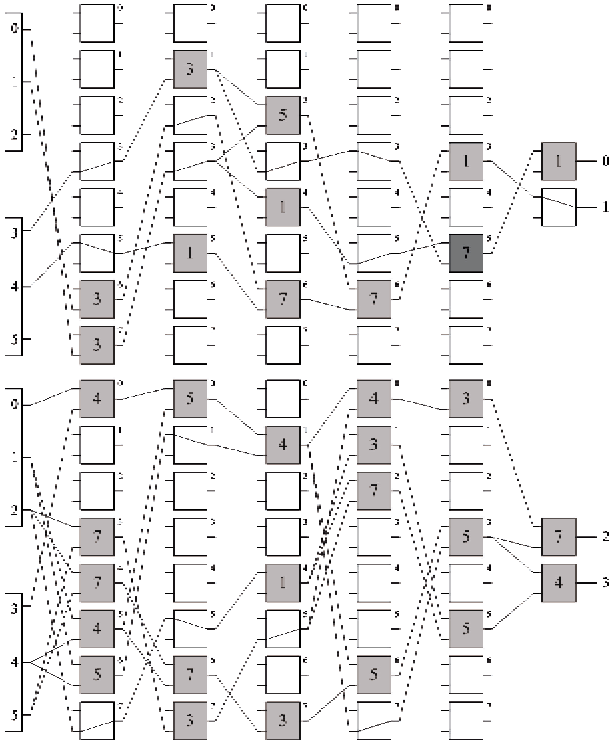
### 3.2　3 位加法器演化结果

第 2 个实验为演化 6 输入/4 输出的 3 位加法器(不含输入进位)。我们测试了 2 种不同系统分解方案:(1)方案 1:两个子系统,其输出分别为(0,2),(1,3);(2)方案 2:两个子系统,其输出分别为(0,1),(2,3)。表 3 对 3 位加法器在不同实验设定下的对比试验结果进行了总结。所有的平均值都来自每个实验设定下 100 次 EA 运行的结果。图 7 给出了方案 2 系统分解设定下,双阶段 MuViRaC 演化的一个 3 位加法器结果。

## 4　讨　论

本文的主要目的在于提出一个有效的演化硬件系统以加速组合逻辑电路演化。通过对比不同实验设定下的平均演化时间我们可以看到,双阶段 MuViRaC 方法优于其他传统的演化方法。MuViRaC 技术是并行演化技术和增量演化技术的一种结合。MuViRaC 能够有效缩短电路演化时间主要得益于以下两点:(1)在 FPGA 上并行执行多个 VRA 核心带来的硬件效能提升,以及 VRA 核心本身强大的计算能力;(2)增量演化策略有效减少了每个子系统的计算复杂度和 EA 搜索空间。

在 MuViRaC 上,每个 VRA 核心都能对其对应的子系统实现粗粒度的双阶段并行演化。同时,通过在FPGA上实现MuViRaC系统,可大大减少软件



（方案 2，包括 31 个逻辑门）。

图中功能单元所执行的函数标号依据图 2 给出

**图 7 双阶段 MuViRaC 演化的一个 3 位加法器**

执行中常见的函数调用等待，方便数据流水线处理，提高 VRA 核心的运算能力[17]。

另一方面，通过增量演化中的输出函数分解策略，分解后的各子系统相对顶层系统而言具有更少的系统输出。因此，硬件实现各个子系统所需的逻辑门数就可相对减少，进而减少了其对应的染色体长度，缩小了 EA 搜索空间。例如演化 3 位乘法器实验方案 1 中，子系统的染色体长度就由分解前的 702 中减少到了 387 个。另外，相对直接演化完整的顶层系统，通过输出函数分解也降低了每个子系统的计算复杂度。在我们的实验中，所有的实验结果都证明了系统演化时间和演化代数与顶层系统的分解程度密切相关。以演化 3 位乘法器为例，当顶层系统被分解为 2 个子系统时，双阶段 MuViRaC 相对于直接演化在演化时间性能上的提升分别为 5.1 倍（方案 2）和 8.5 倍（方案 1）。当顶层系统被分解为 3 个子系统时，双阶段 MuViRaC 相对于直接演化在演化时间性能上的提升提高到了 25.8 倍（方案 4）和 34.7 倍（方案 3）。

在单阶段 MuViRaC 中，顶层系统的总演化时间将取决于最耗时的一个子系统的演化时间。通常每个子系统具有越对等的计算复杂度，那么相对而言顶层系统的演化时间越短。例如使用单阶段 MuViRaC 演化一个 3 位乘法器：方案 1 中，其相对于直接演化和增量演化的演化时间性能提升分别为 6.1 和 1.4；方案 2 中，其演化时间性能提升分别为 2.6 和 0.7。在方案 2 中，由于 2 个子系统的计算复杂度差异极大，演化输出为（3，4，5）的子系统时间过长，造成单阶段 MuViRaC 方法的效能甚至低于序列执行的增量演化策略。为了解决以上单阶段 MuViRaC 的局限，我们设计了双阶段 MuViRaC。通过灵活地在 VRA 核心之间调配闲置的硬件资源，双阶段 MuViRaC 在所有实验设定下都得到了最佳的结果。特别是在分解后的子系统计算复杂度严重不均衡时（如演化 3 位乘法器中的方案 2，演化 3 位加法器中的方案 2 等），双阶段 MuViRaC 较单阶段 MuViRaC 在演化时间上有了明显性能提升。

相对于基于单 VRA 核心的直接演化，MuViRaC 使用了更多的 VRA 核心，然而其硬件代价的增长并不明显。根据表 2、表 3 中的结果，基于双 VRA 核的 MuViRaC 相对于直接演化只有 7% ~9% 的硬件代价增长，基于三个 VRA 核的 MuViRaC 也只有 53% 的硬件代价增长。这主要是因为对应子电路较小的系统输出，可以使用相对直接演化更小的功能单元阵列对子电路进行演化。

## 5 结 论

本文提出一种名为 MuViRaC 的演化硬件方法以加速组合逻辑电路演化设计。该方法应用增量演化策略将一个顶层系统分为几个子部分，而各个分解后子系统的演化由其对应的 VRA 核心通过双阶段并行演化独立进行。作者通过演化 3 位乘法器和 3 位加法器对 MuViRaC 技术进行了系统验证。实验证明 MuViRaC 和直接演化、增量演化等传统方法相比，在 FPGA 资源消耗变化不大的基础上，大大减少了 EA 演化时间。3 秒内，MuViRaC 可完成较复杂的 3 位乘法器演化。将来的工作主要集中于将 MuViRaC 作为硬件引擎和其他演化策略相结合，演化更复杂的组合逻辑电路。

**参考文献**

[1] Yao X, Higuchi T. Promises and challenges of evolvable hardware. *IEEE Trans SMC-Part C*, 1999, 29(1): 87-97

[2] Miller J F, Job D, Vassilev V K. Principles in the evolu-

tionary design of digital circuits - part I. *J Genetic Programming and Evolvable Machines*, 2000, 1(1): 8-35
[3] 平建军,王友仁,高桂军等. 数字演化硬件的函数级在线进化技术研究. 高技术通讯,2009, 19(1):61-65
[4] 李康顺,梁九生,张文生等. 一种基于GEP的演化硬件复杂电路优化算法. 计算机工程与应用,2008,44(18): 83-86
[5] 何国良,李元香,史忠植. 基于精英池演化算法的数字电路在片演化方法. 计算机学报,2010,33(2): 365-372
[6] Bidlo M. Evolutionary design of generic combinational multipliers using development. In: Proceedings of the 7th International Conference on Evolvable Systems: From Biology to Hardware, Wuhan, China, 2007. 77-88
[7] Stomeo E, Kalganova T, Lambert C. Generalized disjunction decomposition for evolvable hardware. *IEEE Trans SMC-Part B*, 2006, 36(5): 1024 - 1043
[8] Liu R, Zeng S Y, Ding L X, et al. An efficient multi-objective evolutionary algorithm for combinational circuit design. In: Proceeding of the 1st NASA/ESA Conference on Adaptive Hardware and Systems, Istanbul, Turkey, 2006. 215-221
[9] Wang J, Piao C H, Lee C H. Implementing multi-VRC cores to evolve combinational logic circuits in parallel. In: Proceedings of the 7th International Conference on Evolvable Systems: From Biology to Hardware, Wuhan, China, 2007. 23-34
[10] Zhao S G, Jiao L C. Multi-objective evolutionary design and knowledge discovery of logic circuits based on an adaptive genetic algorithm. *J Genetic Programming and Evolvable Machines*, 2006, 7(3): 195-210
[11] Cantu-Paz E. A survey of parallel genetic algorithms. *J Calculateurs Paralleles*, 1998, 10(2): 141-171
[12] Wang J, Jung J K, Lee Y M, et al. Using reconfigurable architecture-based intrinsic incremental evolution to evolve a character classification system. In: Proceedings of the Internation Conference Computational Intelligence and Security, Xi'an, China, 2005. 216-223
[13] Torresen J. Evolving multiplier circuits by training set and training vector partitioning. In: Proceedings ot the 5th International Conference Evolvable Systems: From Biology to Hardware, Trondheim, Norway, 2003. 228-237
[14] 朴昌浩,王进, 孙志华等. 自适应变异比率控制在虚拟可重构结构中的应用. 高技术通讯,2010,20(4): 398-402
[15] 丁国良,原亮,褚杰等. 内进化演化硬件平台的设计与实现. 军械工程学院学报,2007,19(1): 66-68
[16] Sekanina L, Friedl S. An evolvable combinational unit for FPGAs. *J Computing and Informatics*, 2004, 23(5): 461-486
[17] Wang J, Chen Q S, Lee C H. Design and implementation of a virtual reconfigurable architecture for different applications of intrinsic evolvable hardware. *IET Computers & Digital Techniques*, 2008, 2(5): 386-400

# Using MuViRaC to accelerate evolutionary design of combinational logic circuits

Wang Jin, Li Lifang, Ren Xiaolong
(Institute of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065)

**Abstract**

This paper presents a multi-virtual reconfigurable architecture cores (MuViRaC)-based intrinsic evolvable hardware (EHW) to speedup the evolutionary design of combinational logic circuits. The basic concept of the proposed scheme is to divide a combinational logic circuit into several sub-circuits according to the output function decomposition strategy for incremental evolution. Each sub-circuit is then evolved separately as a subcomponent through a two-stage parallel evolution process implemented on the MuViRaC. In this study, the MuViRaC was realized on a Xilinx Virtex xcv2000E FPGA that was fitted in a Celoxica RC1000 PCI board. The performance of the proposed scheme was evaluated on the evolution of a 3-bit multiplier and a 3-bit adder, respectively. The experimental results show that the MuViRaC approach could significantly reduce the number of generations and computational time in evolving a combinational logic circuit.

**Key words**: digital circuits, logic circuits, evolvable hardware(EHW), evolutionary algorithm(EA), parallel algorithm