doi:10.11887/j.cn.201703012 http://journal.nudt.edu.cn

使用位流重定位与差异配置在线演化数字系统*

姚 睿,何 坤,朱 萍,李增武,羊宇中
(南京航空航天大学 自动化学院,江苏 南京 210016)

摘 要:利用位流重定位与差异配置技术对现有基于动态部分重构的演化硬件实现方法进行改进,以解决其演化复杂电路时位流存储开销大和演化速度慢的问题。利用Xilinx早期获取部分重构技术,定制能实现位流重定位的可演化IP核。原始位流文件经设计形成算子核位流库存于外部CF卡上,方便系统调用。将现场可编程门阵列片内软核处理器MicroBlaze作为演化控制器,采用染色体差异配置技术,在线实时调节可演化IP核的电路结构,构成基于片上可编程系统的自演化系统。以图像滤波器的在线演化设计为例,在Virtex-5现场可编程门阵列开发板ML507上对系统结构和演化机制进行验证,结果表明,所提演化机制能有效节省位流存储空间,提高演化速度。

关键词:演化硬件;现场可编程门阵列;位流重定位;差异配置;自演化
中图分类号:TP302 **文献标志码**:A **文章编号**:1001-2486(2017)03-069-08

Online evolution of the digital system on bitstream relocation and discrepancy configuration

YAO Rui, HE Kun, ZHU Ping, LI Zengwu, YANG Yuzhong
(College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: To break through the limitations of huge memory space and low evolution speed for complex circuits' evolution, the bitstream relocation and the discrepancy configuration were adopted to improve the efficiency of the evolvable hardware implementation approach based on dynamic partial reconfiguration. Firstly, an evolvable IP core capable of bitstream relocation was customized by using the technology of early stage accession to partial reconfiguration provided by Xilinx. Then the original bitstream files were pre-synthesized to form a partial bitstreams library stored in the CF memory for the system to call. Next, a self-evolving system based on a programmable chip system was built, in which the soft processor, MicroBlaze, was utilized as the evolution controller. And the discrepancy configuration was adopted for the real-time adjustment of the circuit topology of the evolvable IP core. Finally, the system structure and the self-evolving mechanisms were verified by the online evolution of digital image filters implemented on the Xilinx Virtex-5 FPGA(field programmable gate array) development board ML507. Experimental results show that the proposed evolutionary mechanisms can reduce the storage space of bitstream files and can accelerate the speed of evolution significantly.

Key words: evolvable hardware; field programmable gate array; bitstream relocation; discrepancy configuration; self-evolution

在航空航天应用中,嵌入式系统必须承受恶劣的空间环境[1],要求在提供复杂功能的同时,还需满足高可靠性、低功耗、低资源面积开销等需求。这些要求互相制约,使系统复杂度以指数增长,为系统设计带来极大挑战。迫切需要能随环境和需求变化自动调节自身结构和行为以实现任务目标的自适应系统[2-3]。然而,传统自适应系统的适应能力有限(如算法结构固定、仅参数可变等),且无法实现故障情况下的自主修复,因而已无法满足现代自适应系统的需求。演化硬件(Evolvable HardWare, EHW)的出现为构建自适应系统提供了一种解决方案。以演化算法(Evolutionary Algorithm, EA)为全局搜索的主要工具,以现场可重构器件为评估平台和实现载体,寻求在不依赖先验知识和人工干预的情况下,通过演化来获得满足给定要求的电路和系统结构[4-5],进而根据环境变化自主调节自身结构及功能,达到从故障中恢复、在运行生命周期内提高性能等目的。

早期的EHW通过软件仿真离线演化,仅将最终所得最优染色体配置于可编程硬件器件上进行验证,称作外部演化[6]。该方法比较适合研究演化方法,探索新型EHW结构模型,但不能实时调整硬件电路结构,无法满足系统自适应需求。

* **收稿日期**:2016-01-12
基金项目:国家自然科学基金资助项目(61402226);中央高校基本科研业务费专项资金资助项目(NS2014036)
作者简介:姚睿(1974—),女,河南邓州人,副教授,博士,硕士生导师,E-mail: yaorui@nuaa.edu.cn

随着技术的发展,出现了内部演化方式,直接将每代种群的每条染色体分别下载到可重构器件中进行评估,因而可实时调整硬件结构,为实现自适应硬件提供了可能。

20 世纪 90 年代中期,EHW 的思想第一次在 Xilinx 的 XC6200 系列现场可编程门阵列(Field Programmable Gate Array,FPGA)上实现。该芯片内部结构位串(位流)格式公开,可直接对位流操作,很适合实现演化,但已于 1998 年停产。此后,由于担心随机修改位流威胁器件的完整性,商用 FPGA 芯片位流格式不再公开;厂商提供的重构技术也不够成熟,故无法在商用 FPGA 上直接操作位流进行无约束演化。为此,Sekanina 提出了基于虚拟可重构电路(Virtual Reconfigurable Circuit,VRC)的 EHW 实现方式[7-9]。该方法在 FPGA 上实现由处理节点矩阵组成的虚拟可重构层,每个节点包含所有需求功能,可通过多路选择器选择;同时,利用 FPGA 片上微处理器核运行 EA,实现了基于片上可编程系统(System On a Programmable Chip,SOPC)的自演化系统。该方法能实现电路结构与功能的自调整,以适应外部环境变化和故障的自恢复[10-12],成为实现自适应系统的一种解决方案。然而,由于 VRC 中每个节点同时实现了该节点所有可能实现的功能,资源开销较大,且多路选择器加大了电路延时[13-14]。

因此,基于动态部分重构(Dynamic Partial Reconfiguration ,DPR)的方法应运而生。该方法利用 Xilinx 早期获取部分重构技术,在 FPGA 上使用规则的二维可重构分区(Reconfigurable Partition,RP)阵列取代 VRC 定制可演化 IP 核[15]。每个 RP 可配置为多种功能,每种功能可用一个函数描述。设计阶段预先产生包含该函数信息的位流文件,称作算子核位流;所有位流存放于同一存储空间,形成算子核位流库。演化过程中,RP 间连线固定,由 EA 控制改变各 RP 配置的算子核位流,以实现不同电路结构。与 VRC 方法相比,该方法未使用多路选择器,减少了电路延时;也未同时在每个单元上实现所有可能的功能,每个 RP 所占用芯片面积取决于其中最复杂的功能,减少了面积开销。然而,文献[15]中建立算子核位流库时,要为每个 RP 的每种功能生成位流并进行存储,因此存储空间开销大;且评估每一条染色体的适应度时,无论各 RP 模块功能改变与否,均对其重新配置,演化速度较慢。

因此,本文采用位流重定位和差异配置技术改进现有基于 DPR 的 EHW 实现方式,以降低位流存储量和提高演化速度。

1 系统的 SOPC 体系结构

1.1 总体结构与工作原理

系统的 SOPC 体系结构如图 1 所示,包含一个片上软核处理器 MicroBlaze,所有 IP 核以外设形式挂接于处理器本地总线(Processor Local Bus,PLB)上。MicroBlaze 作为演化控制器,运行 EA 控制演化过程;自定制可演化核 MATH_IP_Core 可根据演化命令调节内部处理功能,实现系统功能的自适应;Input_data 和 Idealout_data 是定制的 ROM IP 核,分别存放演化区域的输入数据和期望输出数据;演化过程所需算子核位流与系统全局初始化位流一起存放于外部 CF 卡上;系统高级配置环境(Advanced Configuration Environment, ACE)控制器负责从 CF 卡读取位流文件;重构引擎硬件内部配置访问端口(HardWare Internal Configuration Access Port,HWICAP)负责 FPGA 重构的实现;通用异步收发传输器(Universal Asynchronous Receiver/Transmitter, UART)用于实现 FPGA 开发板与 PC 机超级终端的通信。

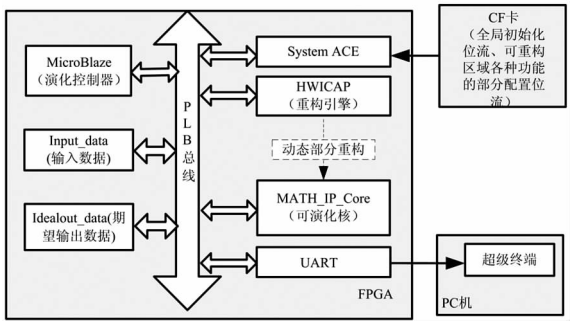


图 1 自演化数字系统的总体结构框图

Fig. 1 General structure of the self-evolving system

开发板上电后,System ACE 通过联合测试工作组(Joint Test Action Group, JTAG)接口读取 CF 卡上的 ace 文件,实现系统的全局初始化。演化开始时,MicroBlaze 根据染色体配置向 HWICAP 发送重构命令;HWICAP 通过 System ACE 从 CF 卡读取相应算子核位流,配置到对应 RP 中,实现染色体到 FPGA 底层硬件的映射;然后将输入只读存储器(Read Only Memory, ROM)核中的数据作为MATH_IP_Core的输入,并将运算结果反馈给 EA,与期望输出数据比较,评估个体适应度;接着 EA 产生下一代种群;重复以上操作,即可实现系统的自演化。演化结果可通过 PC 机超级终端观察。

1.2　可演化核结构

只要硬件资源允许,可演化核可以设计为任意 $m \times n$ 的 RP 阵列,每个 RP 可根据需要配置为任意多种功能,如图 2 所示。每个 RP 模块有两个输入端口和两个完全相同的输出端口,阵列中每个 RP 可配置任意 p 种功能。与 VRC 方法相比,该方法未使用大量的多路选择器,也未在每个节点同时实现所有可能的功能,各 RP 占用芯片面积取决于其中最复杂的功能,减少了电路延时和面积开销。此外,该结构也可根据目标电路的复杂度方便地扩展,提高了系统设计的灵活性。

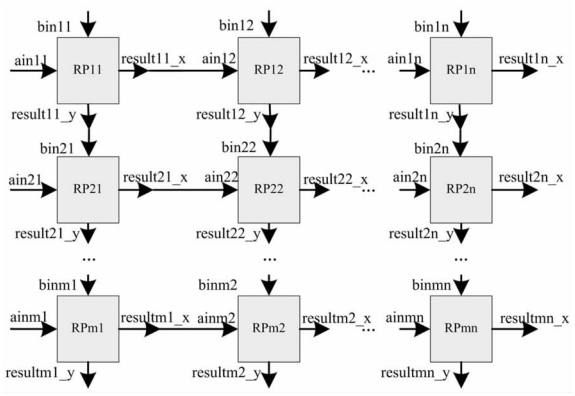


图 2　可演化核的二维阵列结构

Fig. 2　2D structure of the evolvable core

目前基于 DPR 思想演化时,需要预先为每个 RP 的每种功能生成位流文件。若每个 RP 可实现 p 种功能,则需为其生成 p 个位流文件;这样 $m \times n$的 RP 阵列需要 $m \times m \times p$ 个位流,大大增加了位流存储空间。因此,本文仅针对某一 RP 生成实现 p 种功能所需的 p 个位流文件并存储;演化过程中,通过位流重定位技术,实现其他 RP 的功能,降低了对存储空间的要求。

另外,目前基于 DPR 技术演化过程中,无论每个 RP 的功能是否改变,均对演化区域中所有 RP 的位流进行重新配置,大大增加了演化时间,降低了演化速度。实际上,每次演化需要改变的 RP 的个数有限,尤其在演化后期,仅很少一部分 RP 的功能需要改变。因此,没有必要每次对所有的 RP 进行完全配置。为此,本文提出了差异配置技术,配置过程中通过对比新染色体与原染色体的差异,仅重新配置需要改变的 RP,大大减少了配置时间,提高了演化速度。

2　位流重定位技术

2.1　Virtex－5 FPGA 位流格式

位流文件(bit 文件)是 FPGA 的配置数据流,其中包含了配置命令字和配置数据。它是一个二进制文件,包括文件头和 FPGA 的有效配置数据[21],其构成如图 3 所示。文件头主要表示 FPGA 的类型及文件生成的时间信息。尽管不同位流文件的文件头长短和内容不同,但是有效数据总是以同步字“AA995566”开始。

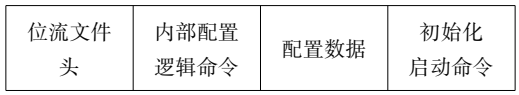


图 3　bit 文件的组成示意图

Fig. 3　Diagram of the bit file

位流文件中有效数据可分为 3 个功能区:配置命令字区、配置数据区和循环冗余校验(Cyclic Redundancy Check,CRC)区。配置命令字区的作用是利用内部配置逻辑来加载数据帧;配置数据区是实现功能的配置数据帧;CRC 校验区的作用是完成初始化启动及 CRC 校验。只有 CRC 校验完成之后,位流才能配置到 FPGA 中。

2.2　位流重定位的原理

利用传统 DPR 技术设计可重构系统时,即使实现同一种功能,每个 RP 的部分位流均不同,故一个 RP 的位流不能直接配置到其他 RP 上。图 4 所示系统包含 3 个 RP,每个 RP 均可实现加法(adder)和减法(sub)两种功能。若采用传统 DPR 技术,PRR1 的位流文件 adder_1 只能用于配置 RP1,而不能配置 RP2 和 RP3。因此,每个 RP 需要 2 个位流文件,一共需要 6 个位流文件,存储空间的开销较大。

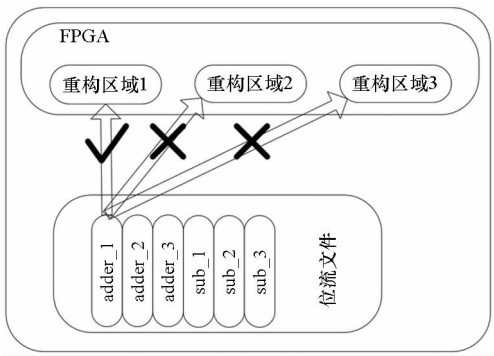


图 4　DPR 系统位流配置示意图

Fig. 4　Diagram of the DPR system′s bitstream configuration

位流重定位是指在传统 DPR 设计流程的基础上,通过增加 RP(可重构区域)设计约束,并对位流进行适当修改,允许一个部分位流(部分重构模块)从一个 RP 配置到资源、区域大小相同的另外一个 RP。

RP 满足设计约束时,比较不同位置 RP 上实现相同功能的位流发现,其配置数据区相同,只有帧地址寄存器(Frame Address Register ,FAR)和 CRC 值不同。因此,实现位流重定位时,只需修改配置位流中的 FAR 和 CRC 的数据。如若需将图 4 中 RP1 的位流文件 adder_1 配置到 RP2,首先应要求 RP2 和 RP1 满足设计约束,其次需要根据 RP2 与 RP1 的相对位置信息修改 adder_1 中 FAR 和 CRC 的值。

2.3 位流重定位的实现

若各 RP 满足设计约束,要将某 RP 的某种功能的位流文件重新定位到其他 RP 模块,只需修改 FAR 和 CRC 的值,配置数据区无须改变。

2.3.1 FAR 结构

FAR 用于存储位流配置的起始地址,其有效数据包括 5 部分:配置块类型、上半部/下半部、行地址、列地址、次地址,如图 5 所示。Virtex - 5 FPGA 包含可编程输入/输出块,可配置逻辑块,块 RAM、CLK、DSP 等资源,分别对应不同编码("001"代表块 RAM 互连,"000"代表其他资源)。资源对称分成上下两部分,行编号分别从中间向顶部和底部递增,列编号从左向右递增,次地址为每列包含的地址。

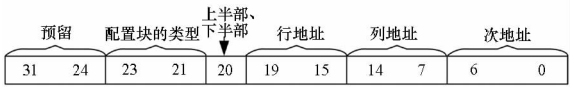


图 5 FAR 寄存器的结构

Fig. 5 Structure of the FAR

2.3.2 CRC 值的计算

CRC 是为了检查配置文件的合法性,进而保护 FPGA 设备的安全而对配置数据进行的循环冗余校验。虽然重定位位流中配置数据区内容未改变,但是 FAR 的值发生了改变,因此必须重新计算其 CRC 的值。Virtex - 5 FPGA 配置位流的 CRC 值可根据式(1)计算。

$$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{6}+1 \quad (1)$$

CRC 值的计算可以在线或离线进行。为了减少重构时间,本文采用离线计算的方式,根据式(1)计算重定位位流中 FAR 值改变后对应的 CRC 值并存储。演化过程中实现位流重定位时不用在线计算 CRC 值,从而提高演化速度。

3 自演化系统的设计实例与实现结果

本节以在 Virtex - 5 FPGA 开发板 ML507 上实现图像滤波器的在线演化设计为例,说明系统的软、硬件设计方法与实现结果。

3.1 自演化系统的硬件设计

3.1.1 系统硬件平台的设计

首先在赛灵思平台工作室(Xilinx Platform Studio, XPS)中创建系统硬件平台,然后将生成的 system. ngc 文件导入 PlanAhead 中,实现 DRP 设计。

3.1.2 可演化 IP 核的设计

设计图像滤波器在线演化系统时,可演化核为 3 × 3 二维阵列结构。每个 RP 有两个 8 bit 输入和两个完全相同的 8 bit 输出,可配置为表 1 所示的 8 种功能。PR 间连线固定,改变各 RP 的配置可实现不同电路结构和系统功能。采用 3 × 3 窗口采集图像数据,作为 RP 阵列的输入;最后一个 RP 的输出作为最终运算结果。

表 1 功能单元及其对应编码

Tab. 1 Functional units and their encoding

名称	功能	对应编码		
		bit[3i + 2]	bit[3i + 1]	bit[3i]
adder	ain + bin	0	0	0
absolute	| ain - bin |	0	0	1
aiden	ain	0	1	0
ainver	255 - ain	0	1	1
binver	255 - bin	1	0	0
average	(ain + bin)	1	0	1
max	max(ain, bin)	1	1	0
min	min(ain, bin)	1	1	1

注:$i = 0,1,\cdots,8$;$n = i + 1$。

3.1.3 可重定位位流设计

位流重定位技术对各 RP 区域的设计,包括各 RP 模块区域划分、各 RP 与静态区域接口以及布线路径等,均有特殊要求。要求重构区域设计必须满足以下规则:①重构区域所占的逻辑资源相同;②重构模块端口引脚的数目一致;③重构模块代理逻辑的相对位置一致;④静态区域的布线不能穿过动态区域。这些规则均可在 PlanAhead 中通过修改约束文件实现。

设计过程主要包括:规划 RP 区域、约束接口代理逻辑位置和统一接口布线信息。

规划重构区域时,必须保证每个 RP 的高度为整行,最小 RP 为一行一列。若 RP 高度小于 1 行,则位流将会包含静态区域的配置信息。一个 RP 所需资源和资源利用率如图 6 所示。

Physical Resource Estimates

Site Type	Available	Required	% Util
LUT	160	0	0
FD_LD	160	14	9
SLICEL	40	5	13

图 6　1 个 RP 资源利用情况

Fig. 6　Resource usage of one RP

可重构区域与静态区域的接口采用代理逻辑(proxy logic)实现。代理逻辑可通过实现工具自动插入。要实现位流重定位,各 RP 区域代理逻辑的相对位置必须一致,故需修改约束文件来修改代理逻辑的位置。

代理逻辑放置于可重构区域,它与静态区域之间的布线路径信息也包含在重构位流信息内。为保证所有 RP 的代理逻辑与静态区域之间的布线路径信息一致,还需对所有 RP 的代理逻辑和静态模块之间的布线信号路径进行分析和修改。首先提取所有 RP 模块的路径信息,然后选取其中一个 RP 的信号路径信息作为标准,对其他 RP 区域的路径信息进行修改。

3.1.4　图像数据的存储

演化图像滤波器时,需要存储待滤波图像数据、理想图像数据和滤波后图像数据。为方便管理,本文将待滤波和理想图像数据存储于 FPGA 片内块随机存取存储器(Block Random Access Memory, BRAM)中,并将其定制为 ROM IP 核,挂接于 PLB 总线上,作为外设调用;滤波后的图像数据作为变量,临时分配存储空间。演化过程中,将待滤波图像数据作为可演化核的输入。读取滤波后图像数据,按照一定的准则与理想图像数据比较,即可得到适应度值。

3.2　自演化系统的软件设计

系统软件设计的主要任务是在 MicroBlaze 上运行演化算法实现演化过程的控制。本文选择遗传算法进行系统软件设计。

3.2.1　染色体编码方式

为降低演化复杂度,本文采用加、减、求均值等函数功能模块作为演化积木块,采用间接编码方式。3×3 RP 阵列的分段二进制编码方案如图 7 所示,RP1 ~ RP9 分别对应 9 个 RP 的编码。每个 RP 的 8 种功能需 3 位二进制数描述,故与 RP 阵列功能相关的染色体长度为 3×9 = 27 bit。各 RP 编码与功能对应关系如表 1 所示。3×3 窗口每次需要 9 个像素值,故每个输入端口的选择需 4 位二进制数描述。由于 3×3 RP 阵列共有 6 个输入端口,故与输入选择相关的染色体长度为 4×6 = 24 bit。因此染色体总长度为 27 + 24 = 51 bit。

RP[]	RP9			RP8			RP7			RP6			RP5			RP4			RP3			RP2			RP1		
bit[]	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

图 7　3×3 RP 阵列功能的染色体编码方案

Fig. 7　Chromosome encoding of the 3×3 RP array

3.2.2　染色体差异配置技术

评价染色体时,需将其解码后配置到可演化核的重构区域中。目前基于 DPR 的方法采用整体配置,即对每个 RP 功能和所有输入端口均进行重新配置,增加了配置时间。尤其是改变各 RP 功能时,从外部读取位流和通过内部配置访问端口(Internal Configuration Access Port, ICAP)端口配置耗时较多。因此,为了减少配置时间,提高演化速度,本文提出了染色体差异配置技术。配置新染色体时,调用相应算子位流之前,首先逐位区比较待配置染色体与当前染色体,找出不同的位区,并重新配置对应的 RP 区域,而功能不变的 RP 区域则不进行配置。

实现过程中,首先利用配置矩阵描述新、老染色体,然后对各矩阵元素进行异或,得到差异矩阵,最后根据差异矩阵和染色体编码方法对需改变配置的 RP 区域进行重构,实现差异配置。

染色体差异配置技术的采用大大减少了需要配置区域的数目,进而减少了演化的时间。

3.2.3　适应度计算

可演化核有 6 个输入 1 个输出。滤波过程与典型的图像卷积滤波器类似,采用 3×3 窗口选取每个像素及其相邻像素作为可演化核的输入,可演化核输出即为中心像素滤波后的输出。本文采用的滤波图像像素为 125 列 124 行,除了第 1 行、第 124 行、第 1 列及第 125 列等“图像边框”中像素外,其他像素点均可作为中心点。衡量滤波效果时,采用式(2)所示的平均每像素误差(Mean Difference Per Pixel, MDPP)作为评价标准,

$$MDPP = \frac{\sum_{i=0}^{R-1}\sum_{j=0}^{C-1} |ideal(i,j) - filtered(i,j)|}{C \times R} \tag{2}$$

式中,C 为列数,R 为行数,$ideal(i, j)$ 为理想无噪图像像素,$filtered(i, j)$ 为滤波后的像素。$MDPP$ 越小表明滤波图像效果越好。计算每一代种群中每个个体对应的 $MDPP$ 值,并保留 $MDPP$ 值最小的个体,参与下一代种群的产生。

3.2.4　遗传算子设计

遗传算法通过选择、交叉和变异等遗传算子

产生新个体。本文为了降低复杂度,对算法进行了简化,仅采用了选择算子和变异算子。

1)选择算子

选择算子采用联赛选择方式。每次联赛选择时,首先从父代种群中随机选取一定数量的个体,选择其中适应度最大者进入下一代种群;进行 P(种群规模)次选择即可产生新的种群。该过程中,适应度较高的个体可能多次被选中,而适应度较低的个体可能一次未被选中,充分体现了自然界生物进化过程中"优胜劣汰"的原则。

2)变异算子

变异算子以一定的概率对染色体位串进行翻转。设变异概率为 P_M,则种群中任意个体被选中参与变异操作的概率为 P_M。本文设计采用简单均匀随机变异操作。为保证个体变异后与其父体的差异不会过大,变异率一般取值较小,为 0.1 或更小,以保证种群发展的稳定性。

3.2.5 自演化操作过程

自演化操作流程图如图 8 所示,有两个演化终止条件:一是找到达到最大适应度的个体,即找到最优解;二是达到设定的最大演化代数。

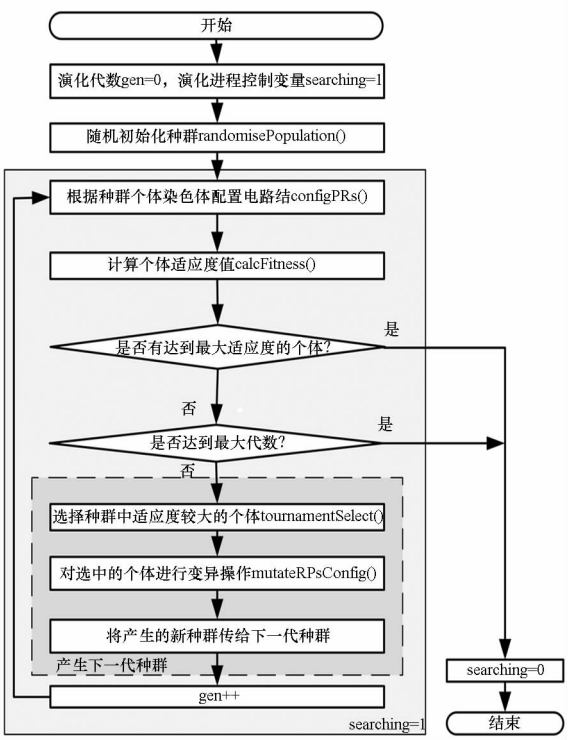


图 8 自演化操作流程图

Fig. 8 Operation flow of self-evolution

4 实验结果与分析

4.1 位流重定位技术对位流存储空间的影响

采用位流重定位技术可以减少需要存储的位流个数,节省存储空间。表 2 所示为实现图像滤波器在线演化时,采用位流重定位技术(本文方法)与不采用位流重定位技术(传统方法)所需位流个数和存储空间大小的对比。由表 2 可见,本文方法所需位流文件数目比传统方法减少了 64 个。每个位流文件大小为 7 KB,故与传统非重定位方法相比,采用位流重定位技术可以节约 448 KB的存储空间,比传统方法节省 88.9%。

表 2 所需位流文件的个数和存储空间对比

Tab. 2 Comparisons of number of bit files and memory space

逻辑功能	位流文件个数		所需存储空间大小	
	本文方法	传统方法	本文方法	传统方法
adder	1	9	7 KB	63 KB
absolute	1	9	7 KB	63 KB
average	1	9	7 KB	63 KB
aiden	1	9	7 KB	63 KB
max	1	9	7 KB	63 KB
min	1	9	7 KB	63 KB
binver	1	9	7 KB	63 KB
ainver	1	9	7 KB	63 KB
总计	8	72	56 KB	504 KB

事实上,采用位流重定位技术所节约存储空间的大小取决于 RP 的个数及 RP 区域的大小。本文 3×3 RP 阵列中共有 9 个 RP,采用位流重定位技术仅需为一个 RP 生成 8 种逻辑功能的位流文件;而非重定位方法则需为 9 个 RP 各自生成 8 种位流文件(共 $8\times9=72$ 个),故存储空间节约率为 8/9。若采用 $m\times n$ 的 RP 阵列,每个 RP 可实现 p 种功能,则非重定位方法需存储 $m\times n\times p$ 个位流,假设每个位流为 s KB,则需 $m\times n\times p\times s$ KB存储空间;而位流重定位技术仅需存储 p 个位流,即 $p\times s$ KB 的存储空间,为传统方法的 $1/(m\times n)$,即存储空间节约率为 $(m\times n-1)/(m\times n)$。

4.2 差异配置技术对演化时间的影响

种群规模 $P=64$,联赛规模 $T=10$,变异率 $P_M=2/256$ 时,采用本文的差异配置与完全配置的演化时间与演化代数关系对比如图 9 所示。由图 9 可见,完全配置方法的演化时间与演化代数成正比;原因是该方法对每代的每条染色体均进行完全配置,故配置所需时间基本固定。

而差异配置在评估时将新染色体与原染色体进行比较,仅重新配置功能改变了的 RP,故大大节省了每条染色体的配置时间,提高了演化速度。而且,演化的代数越多,差异配置的优势越明确。

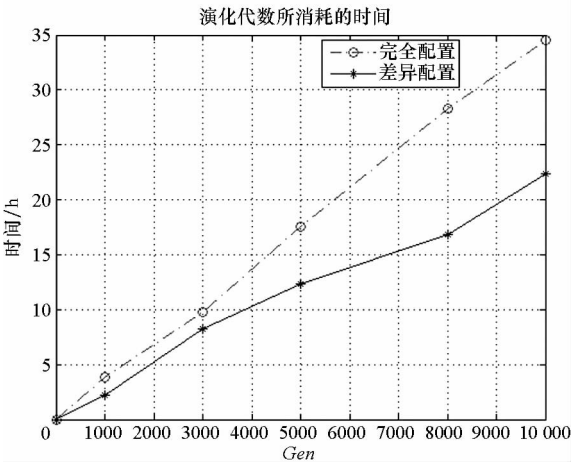


图 9　差异配置和完全配置演化时间对比

Fig. 9　Comparison of consumed time between discrepancy configuration and complete configuration

4.3　图像滤波的效果

采用 125 × 124 具有 $L = 256$ 级灰度的标准“Lena”图像作为测试图像,分别注入均值为 0,方差为 0.03 的高斯噪声和噪声强度为 5% 的椒盐噪声,进行图像滤波器在线演化设计,滤波前后图像如图 10 所示,滤波前后 *MDPP* 值对比如表 3 所示。实验中遗传算法的参数:种群规模 $P = 64$,联赛规模 $T = 10$,变异率 $P_M = 2/256$,演化终止代数 $Gen = 10\ 000$。

(a) 理想图像

(a) Ideal image

(b) 加高斯噪声的图像

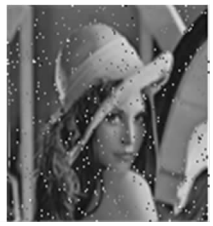
(b) Image polluted with Gaussian noise

(c) 滤波后的图像

(c) Filtered image

(d) 加椒盐噪声的图像

(d) Image polluted with salt and pepper noise



(e) 滤波后的图像

(e) Filtered image

图 10　滤除图像中高斯噪声的结果

Fig. 10　Experimental results of filters Gaussian noise

表 3　滤波前后 *MDPP* 值对比

Tab. 3　Comparison of *MDPP* before and after filtering

噪声	滤波前 *MDPP*	滤波后 *MDPP*
均值 0、方差 0.03 的高斯噪声	33.08	14.86
5% 的椒盐噪声	6.55	1.21

由图 10 和表 3 可见,本文系统可以对高斯噪声和椒盐噪声进行有效滤波。尤其是对椒盐噪声滤波效果更为明显。滤波前图像的 *MDPP* 值为 6.40,滤波后仅为 1.21,有效降低了 *MDPP* 值。

5　结论

为克服现有演化硬件实现方式存在的不足,本文采用位流重定位和差异性配置技术实现自演化数字系统,研究了其体系结构与在线自主演化技术。与目前基于 VRC 的 EHW 实现方式相比,本文方法未利用大量的多路选择器,也未同时在每个可重构模块上实现所有可能的功能,减少了电路延时,降低了器件资源面积开销。与现有的基于 DPR 的实现方式相比,本文方法大大减少了存储位流文件所需要的存储空间,并大大缩短了演化每代所需要的时间,提高了演化速度。

参考文献(References)

[1] Vipin K, Fahmy S A. Mapping adaptive hardware systems with partial reconfiguration using CoPR for Zynq [C]// Proceedings of Adaptive Hardware and Systems (AHS), NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2015), 2015: 1-8.

[2] Mora J, Gallego A, Otero A, et al. A noise-agnostic self-adaptive image processing application based on evolvable hardware[C]//Proceedings of Design and Architectures for Signal and Image Processing (DASIP), 2013: 351-352.

[3] Gallego A, Mora J, Otero A, et al. A self-adaptive image processing application based on evolvable and scalable hardware[C]//Proceedings of Field Programmable Logic and Applications (FPL), 2013.

[4] 莫宏伟, 徐立芳. 基于 Memetic 算法的电路演化设计研究[J]. 电子学报, 2013, 41(5): 1036 – 1040.
MO Hongwei, XU Lifang. Research on evolvable hardware design based on Memetic algorithm [J]. Acta Electronica Sinica, 2013, 41(5): 1036 – 1040. (in Chinese)

[5] 刘少腾, 来金梅, 陈利光, 等. 可进化可重构图像滤波器的设计[J]. 复旦学报: 自然科学版, 2010, 49(6): 709 – 715.
LIU Shaoteng, LAI Jinmei, CHEN Liguang, et al. Design of an evolvable reconfigurable image filter[J]. Journal of Fudan University: Natural Science, 2010, 49(6): 709 – 715. (in Chinese)

[6] Haddow P C, Tyrrell A M. Challenges of evolvable hardware: past, present and the path to a promising future[J]. Genetic Programming and Evolvable Machines, 2011, 12(3): 183 – 215.

[7] Sekanina L. Virtual reconfigurable circuits for real-world applications of evolvable hardware[M]//Tyrrell A M, Haddow P C, Torresen J. Evolvable Systems: From Biology to Hardware, Germany: Springer Berlin Heidelberg, 2003: 186 – 197.

[8] Sekanina L. Evolutionary design of gate-level polymorphic digital circuits[M]//Rothlauf F, Branke J, Cagnoni S, et al. Applications of Evolutionary Computing, Germany: Springer Berlin Heidelberg, 2005: 185 – 194.

[9] Sekanina L, Harding S L, Banzhaf W, et al. Image processing and CGP [M]. Cartesian Genetic Programming, Germany: Springer Berlin Heidelberg, 2011: 181 – 215.

[10] 朱继祥, 李元香, 邢建国. 可重构系统的演化修复机制[J]. 计算机学报, 2014, 37(7): 1599 – 1606.
ZHU Jixiang, LI Yuanxiang, XING Jianguo. The evolvable recovery of reconfigurable system [J]. Chinese Journal of Computers, 2014, 37(7): 1599 – 1606. (in Chinese)

[11] Zhang X, Luo W. Evolutionary repair for evolutionary design of combinational logic circuits [C]//Proceedings of IEEE Congress on Evolutionary Computation (CEC), 2012: 1 – 8.

[12] 姚睿, 王友仁, 于盛林, 等. 具有在线修复能力的强容错三模冗余系统设计及实验研究[J]. 电子学报, 2010, 38(1): 177 – 183.
YAO Rui, WANG Youren, YU Shenglin, et al. Design and experiments of enhanced fault-tolerant triple-module redundancy systems capable of online self-repairing [J]. Acta Electronica Sinica, 2010, 38(1): 177 – 183. (in Chinese)

[13] Salvador R, Otero A, Mora J, et al. Self-reconfigurable evolvable hardware system for adaptive image processing[J]. IEEE Transactions on Computers, 2013, 62(8): 1481 – 1493.

[14] Salvador R, Otero A, Mora J, et al. Implementation techniques for evolvable HW systems: virtual vs dynamic reconfiguration[C]//Proceedings of International Conference on Field Programmable Logic and Applications (FPL), 2012: 547 – 550.

[15] 孙艳梅. 基于 FPGA 动态部分重构的数字在线演化技术研究[D]. 南京: 南京航空航天大学, 2015.
SUN Yanmei. Research on online evolution technology of digital system based on dynamic partial reconfiguration of FPGA[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2015. (in Chinese)