软硬件协同的遗传算法设计

聂 鑫[1,2],殷若兰[2],刘海峰[3]
(1.智能机器人湖北省重点实验室,湖北 武汉 430205;
2.武汉工程大学 计算机科学与工程学院,湖北 武汉 430205;
3.华为技术有限公司,广东 深圳 518000)

摘 要:针对软件和硬件实现方式各自的优点及不足,提出了遗传算法的软硬件协同设计方法,并且将这种方法在FPGA上进行了具体的实现。首先对遗传算法流程中的各个模块进行了详细的分析,根据软硬件的不同特点以及设计实现的目标,对遗传算法的功能模块进行了软硬件划分,然后对硬件实现的部分进行了详细的介绍,包括模块之间的连接,模块的内部状态机,模块的端口,所有硬件模块的功能仿真。同时,为了保证软硬件之间的正常通讯,提出了一种新的软硬件交互通讯协议。最后将硬件实现部分做成通用IP核,方便其他设计者使用,并给出了软硬件协同的遗传算法在二进制问题和0-1背包问题两个实例中的具体应用数据。通过与纯软件实现方式的实验数据进行比较,提升了算法运行时间50%的效率,且算法的收敛性保持一致,进一步验证了该算法的适用性及高效性。

关键词:遗传算法;FPGA;软硬件协同;软硬件划分;IP核;0-1背包问题
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2021)11-0114-08
doi:10.3969/j.issn.1673-629X.2021.11.019

Design of Genetic Algorithm Based on Software and Hardware Cooperation

NIE Xin[1,2],YIN Ruo-lan[2],LIU Hai-feng[3]
(1.Hubei Key Laboratory of Intelligent Robot,Wuhan Institute of Technology,Wuhan 430205,China;
2.School of Computer Science and Engineering,Wuhan Institute of Technology,Wuhan 430205,China;
3.Huawei Technologies Co.,Ltd.,Shenzhen 518000,China)

Abstract:In view of the advantages and disadvantages of software and hardware implementation,a software hardware co-design method based on genetic algorithm is proposed and implemented on FPGA. We analyze each module of the genetic algorithm flow chart,divide the function module of genetic algorithm in hardware and software according to their different features and design purpose. After that,we elaborate the implementation of hardware part,including connection between modules,state machine in a module and interfaces of a module,function simulation of all hardware modules. We also provide a protocol to guarantee the communication between hardware and software. Ultimately,the hardware implementation is designed as common IP kernel to be reused by other designers. Then we give the application data of software hardware cooperation genetic algorithm in binary problem and 0-1 knapsack problem. Compared with the experimental data of pure software implementation mode,the efficiency of the algorithm running time is improved by 50%,and the convergence of the algorithm is consistent,which further verifies the applicability and high efficiency of the algorithm.

Key words:genetic algorithm;FPGA;software and hardware cooperation;software and hardware division;IP core;0-1 knapsack problem

0 引 言

遗传算法是Holland在1975年提出的一种概率搜索算法[1]。遗传算法通过有组织地然而是随机的信息交换来重新结合那些适应性好的串。类似于生物的进化,遗传算法作用于类似于基因的二进制串上,通过寻找好的二进制串来求解问题。在每一代中,算法使用上一代适应值较好的个体通过杂交变异的方式生成一个新的种群。由于它不是直接作用于解空间,所以不受搜索空间的限制,同时也不需要丰富的先验知识,以及其内含的天然并行性。因此遗传算法作为一种有效

收稿日期:2020-12-08 **修回日期**:2021-04-13
基金项目:国家自然科学基金资助项目(61672391);智能机器人湖北省重点实验室引导基金资助项目(HBIRL202009)
作者简介:聂 鑫(1983-),男,博士,副教授,硕导,CCF会员(E0047M),研究方向为演化算法、演化硬件;殷若兰(1997-),女,硕士研究生,研究方向为软件工程。

的优化算法在多个领域得到了应用。

目前遗传算法的应用研究中存在若干问题:算法的执行效率低下、算法收敛速度慢以及算法无法找到最优解(即算法过早收敛)。对于后两个问题,不少学者通过不断的改进算法的因子以及算法的整体结构,或者是将算法与其他新算法进行融合来提高算法的收敛速度。针对算法执行效率低下的问题,现在也有不少学者试图使用硬件化的方式来实现遗传算法[2-4],或者在大型的工作站中利用遗传算法所具有的天然并行性来解决,这些方法都取得了一定的效果。这些方法在一定程度上也代表了今后遗传算法的发展方向,同时也为遗传算法在更多领域甚至在实际场合种应用创造了条件。

软硬件协同设计(hardware/software co-designing)的思想是在硬件和软件设计过程中尽最大限度地利用其协同作用来满足系统的要求。自从软硬件协同思想提出以后,一直备受国内外研究者的关注,关于软硬件协同设计领域的研究也十分活跃。到目前为止,国内外学者已经在此方面做过很多研究[5-7],比如在遥感影像的实时效应,音频编码算法,Lattice 译码算法,数字电路仿真,系统的模拟、仿真和调试等方面都使用过软硬件协同设计方法,并且获得比使用传统的设计方法更好的效果。因此利用软硬件协同设计方法不仅可以提高求解问题的效率,同时可以扩宽其应用领域,进一步推动软硬件协同设计的发展等。FPGA 的快速发展,也为软硬件协同工作搭建了平台[8],使得软硬件协同处理成为了可能。

国内外在软硬件协同处理遗传算法方面的研究还很少。考虑到纯硬件或者纯软件实现的遗传算法在各自的优点上面可以互补,通过软硬件协同工作的遗传算法同时具有硬件的高效性以及软件的通用性。这种遗传算法部署方便,开发成本低,效率高,功耗小,具有可移植性。为遗传算法在更多领域应用提供了一定的参考价值。

1　遗传算法参数设计

1.1　遗传算法操作

遗传算法通过“适者生存”这种指导思想对种群进行操作。具体就表现为遗传算法不断地通过杂交操作、变异操作以及选择操作使得适应值较坏的个体逐渐被淘汰。最后种群会逐渐地向最优解的方向收敛。下面就对文中采用的遗传算法中的一些基本操作进行介绍[9-11]。

(1)编码操作。

遗传算法不是直接作用于解空间,而是作用于一种编码方式。编码是使用遗传算法时要解决的首要问题,也是设计遗传算法时的关键步骤,因为设计遗传算子是建立在编码基础之上的。不同的编码方式所对应的遗传算子是完全不同的。遗传算法中一般使用位串编码与实数编码两种方式。众所周知在所有生物中,基因决定了一个生物的种类以及生物的形态。而这种基因就好比是一类数据的集合。使用位串编码就可以很好地模拟基因。

考虑到二进制编码在硬件系统中实现方便的特性,在本设计中使用了该编码方式。同时为了确保算法的精度,二进制串的长度 i 设定为 50。

(2)杂交操作。

杂交操作是遗传算法中遗传操作的一部分。同生物界中一样,杂交操作可以在一定程度上保持父代个体所具有的适应值。遗传算法中的杂交操作一般有点式杂交与均匀杂交两种方式。

在本设计中,考虑到所使用的种群大小为 128 以及方便 FPGA 的实现,设计使用单点式杂交操作。

(3)变异操作。

变异操作较杂交操作相对简单。在一般的遗传算法中,变异操作是按照一定的概率 n 发生的。变异操作在整个遗传算法中起到辅助性搜索的作用。当变异概率 n 过大时,可能就会破坏种群较好的模式。当 n 过小时,就会使得算法产生新个体能力下降与过早成熟。当变异操作发生时,随机的在基因片段中选取一点或者多点进行编译操作。具体操作就是按位取反。

本设计中由于采用了单点式的杂交方式,并且种群并不是很大,所以这里采用的变异方式为在种群的 128 个个体中每次选取 1 个个体进行变异操作。

(4)选择策略。

在遗传算法中,选择策略也起着相当重要的作用。不同的选择策略导致了不同的选择压力。较大的选择压力使得较优的个体能在种群中获得更多的复制数目。使得种群更快收敛。而较小的选择压力则使得种群收敛速度较慢,但使得算法获得全局最优解的概率增大。

比较常见的选择有繁殖池策略、轮盘策略、精英选择策略等。繁殖池策略就是根据个体的适应值计算出其在种群中的相对适应值,根据相对适应值进行复制操作。适应值越高的个体复制的个数越多。然后在复制个体中进行遗传操作。并且使用子代完全替换父代。

在本设计中,使用 $\mu+\gamma$ 的选择策略:在父代中选择 μ 个产生 γ 个子代,然后从 $\mu+\gamma$ 个个体中选择 μ 个最优个体替换父代。具体操作是当子代生成进行完评价之后,每次都用子代和父代中的最好个体去替换旧个体,从而增大选择压力,使得种群更快收敛。由于本

设计中变异概率较大,从而也在一定程度上缓解了种群陷入局部最优解的状况。

(5)适应值评价。

遗传算法中,适应值是评价种群个体好坏的标准,是遗传算法中收敛的驱动力。在遗传算法中,具有优秀适应值的个体将在种群中获得更多的生存机会。不同的问题有着不同形式的适应值评价函数。但是一般来说适应值评价方式有两种,一种是选取结果的最大值,另外一种则是选取结果的最小值。

在本设计中,使用选取最大值的方式。

(6)算法终止。

遗传算法的终止是通过提前设定的参数来确定的。一般使用遗传算法所解决的问题都是运算复杂度高的类型。可能只知道解的空间范围。一般来说,遗传算法的终止条件有多种。一种是无论算法是否找到最优解,当算法执行 N 代后则停止。使用此种方法的遗传算法运行时间比较稳定,因为它运行的代数是个确定的数字。数字 N 的大小在该方法中比较重要。首先,如果 N 较小,算法可能得不到最优解就停止。另一方面,如果 N 较大,虽然找到最优解的概率变大,但是如果算法在早期就达到稳定状态,那么就浪费了后续运行的时间。另外一种方法是跟踪每代运行的最优个体,如果该最优个体在 N 代内不发生变化,算法就会停止。数字 N 的大小在该方法中同样比较重要,如果 N 较小,可能算法只是在局部的最优解收敛,并不是全局的最优解。如果 N 较大,也会造成运行时间的浪费。该方法的好处是至少可以确定获得局部最优解。缺点则是算法的运行时间不确定。

在本设计中,为了保证寻找到局部最优解,采用第二种方法。

1.2 软硬件平台简介

在本设计中,使用 Xilinx 公司的 FPGA 作为开发平台。Xilinx 公司 FGPA 芯片主要由 6 部分组成,即可编程输入输出单元、时钟管理、基本可编程逻辑单元、布线资源、块 RAM、底层功能单元和硬核。

Virtex 系列是 Xilinx 的高端产品,这个系列的产品一般性能好,速度快,并且板载更多硬核。Virtex-II Pro 系列是在 Virtex-II 的基础上增强了嵌入式处理功能,内嵌了 Power PC 405 内核,还包括了先进的主动互联技术,以解决高性能系统所面临的挑战。Virtex-II Pro 系列的主要特征如下:

(1)采用了 1.5 V 核电压,4 输入 LUT。

(2)420 MHz 的始终技术,内置多达 12 个 DCM 模块。

(3)支持 20 多种 I/O 接口标准。

(4)增加多个 3.125 Gb/s 速率的 Rocket 串行收发器。

(5)内置 18x18 位乘法器模块。

(6)内嵌 PowerPC 405 硬核处理器。

本设计中使用的 Virtex-II Pro 系列型号为 XC2VP30,因为该型号的 FPGA 提供了板载 CPU,配合 Xilinx 公司的 EDK 工具可以很方便地进行软件开发,方便软硬件协同设计的实现。该型号的 FGPA 的主要性能特征如表 1 所示。

表 1 Virtex-II Pro XC2VP30

资源名称	数量
Slice	13 686
分布式 RAM/kb	4 280
块 RAM/kb	2 448
DCM	8
18x18 乘法器	136
PowerPC 405	2
Rocket I/O	8
最大可用 I/O	644

2 软硬件协同的系统设计

2.1 硬件划分

使用软硬件协同的方式可以结合软硬件各自的优点,对设计进行进一步的优化。综合软硬件各自的优点,采用软硬件协同的工作方式实现的系统,将系统中一些底层简单而重复,特别是能够并行化的工作交由硬件完成,将具有通用性,串行的工作交由软件完成,不仅可以提升系统的效率,缩短系统的开发周期,并且使得系统具有可重用性。在软硬件协同中,如何划分软硬件具体工作是一件重要的事[12]。因为这涉及到设计的运行效率以及具体实现的功能。对遗传算法软硬件划分确定,硬件部分为总控模块、初始化模块、交叉选择模块、变异模块、评价模块。软件部分为随机数模块和适应值评价模块。整个系统的设计图如图 1 所示。

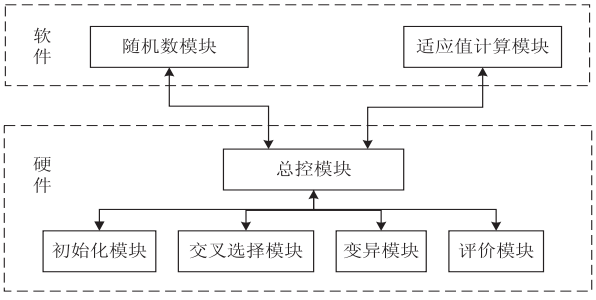

图 1 系统设计图

2.2 交互协议

在设计平台中,CPU(PowerPC)时钟频率(300 MHz)与 FPGA 提供的时钟频率(100 MHz)不一致,并

且在 CPU 上运行的软件程序完成所需要的时钟周期数是不确定性的。由于在本设计中存在许多软件层面与硬件层面的信息交互,为了保证信息交互的同步性、可靠性,必须设计一个通信协议来确保数据的正确性。本设计中软硬件信息交互中可能存在的问题大致如下:

(1)由于 PowerPC 时钟频率较快,如果软件端的请求只发送一次,而在时序控制的系统中硬件响应事件仅在时钟上升沿或者下降沿的时刻响应信号的变化,这就有可能导致硬件无法获取启动信号。

(2)同样由于 PowerPC 时钟频率较快,如果一直发送请求,直到硬件返回数据停止,那么就有可能使硬件响应软件多次,出现硬件无法正常工作等错误。

(3)由于 FPGA 处理数据较快,当硬件完成工作之后,如果在软件未就绪的情况下通知软件,就会出现信号丢失,导致整个系统出错。

(4)由于系统 PowerPC 时钟频率与 FPGA 时钟频率不一致,可能会使信号无法采集,从而使某一方面无限等待,导致系统的死机。

基于以上种种可能发生的异常,必须设计一种可靠的协议[13-14]。鉴于以上描述问题与网络通信中出现的问题有一定程度的相似性,并且网络通信中的复杂程度要远大于这里,故考虑借鉴网络通信中的可靠性传输协议,即 TCP 协议。TCP 协议通过 3 次握手协议确保通信双方数据传输的可靠性。本设计借鉴 3 次握手的方式,设计通信协议,如图 2 所示。

在本设计中,在杂交模块、变异模块、初始化模块以及总控模块中都使用了该协议来确保软硬件双方信息交互的正确性。

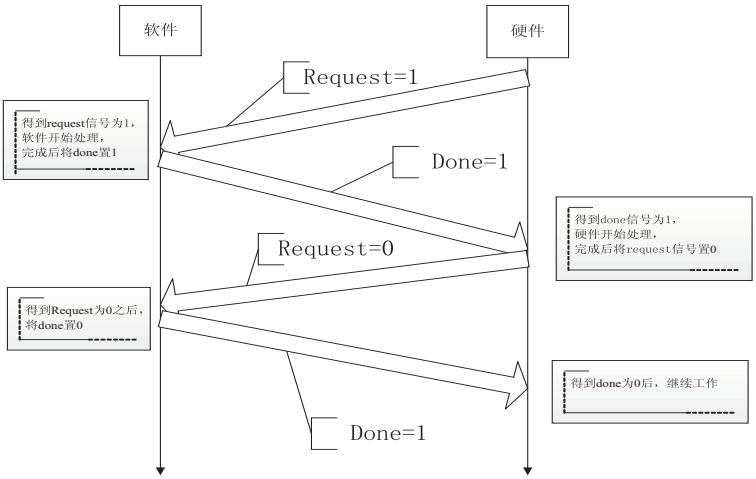

图 2　通信协议

2.3　通信端口设计

为了让软件与硬件能够正常通信,除了协议之外还需要有一个共同的通道。因此需要在 FPGA 上建立部分寄存器来模拟通信的通道。依据通信协议中所需要的状态信号以及通信数据的需要,共建立 10 个 32 位的寄存器。寄存器作用分别描述如下:

(1)寄存器 0:该寄存器保存着软件端给硬件端的所有信号。信号依次如下:

Start:启动信号,通知硬件启动,该信号使用寄存器 0 中的第 0 位即 reg0[0];

Stop:停止信号,当系统有特殊需要时可能所发出的停机信号,通过该信号通知硬件停机。该信号使用寄存器 0 中的第 1 位即 reg0[1];

Cal_done:适应值评价完成信号。通过该信号来告知硬件适应值评价结束,并且已经存放在相应位置。该信号使用寄存器 0 中的第 2 位即 reg0[2];

ini_sg_done:初始化完成信号。通过该信号来告知硬件初始化工作完成。该信号使用寄存器 0 中的第 3 位即 reg0[3];

rand_done:随机数产生完成信号。通过该信号告知硬件随机数产生完成。该信号使用寄存器 0 中的第 4 位即 reg0[4]。

(2)寄存器 1:该寄存器保存着硬件端给软件端的所有信号。信号依次如下:

rand_reg:随机数请求信号,该信号使用寄存器 1 中的第 5 位即 reg1[5];

cal_req:适应值评估请求信号,该信号使用寄存器 1 中的第 3 位即 reg1[3];

ini_reg:初始化请求信号,该信号使用寄存器 1 中的第 1 位即 reg1[1];

done:算法运行结束信号,该信号使用寄存器 1 中的第 4 位即 reg1[4]。

(3)寄存器 2 ~ 3 保存需要进行适应值计算的个体。

(4)寄存器 4 ~ 5 保存初始化生成的个体。

(5)寄存器 6 ~ 7 保存最优个体。

(6)寄存器 8 保存软件评价出的适应值结果。

(7)寄存器 9 保存由软件生成的随机数。

3 软硬件协同的平台实现

3.1 硬件模块

如图 1 所示,需要实现的硬件模块为总控模块、杂交模块、变异模块、评价选择模块、片上内存模块、片上内存选择读取模块、初始化模块。对平台设计中的硬件化模块进行功能仿真。仿真工具选择使用 Xilinx ISE 自带仿真器。

使用 Xilinx ISE 自带仿真器进行仿真之前,必须要建立测试硬件功能的激励文件。ISE 提供了两种不同的方式。第一种是基于 HDL 测试代码,建立一个 Verilog Test Fixture 类型的文件,将这个文件与待测试文件相关联,然后根据不同的测试目的编写激励代码。这种方法工作量较大,并且修改较复杂。另外一种方式就是基于波形的测试代码,建立一个 testbench 波形文件,一样也将这个文件与待测模块进行关联。系统会根据模块的输入输出显示一个波形文件,用户可以通过修改这个波形文件的输入信号直观地改写测试的激励文件。最后在系统的 Behavioral Simulation 状态中运行 Xilinx ISE 自带仿真器,就可以显示波形文件。本设计中采用第二种测试方法。

本设计硬件部分的工作流程是由总控模块来控制的。软件层面的随机数模块和适应值计算模块,硬件层面的初始化模块、交叉选择模块、变异选择模块、评价模块都与总控模块进行着信息的交互。因此总控模块是整个设计的核心部分,就好比计算机内部的 CPU,控制着整个设计的流程以及数据的流向,各模块也都是在总控模块的控制信号下有序地工作。对本设计硬件部分的设计而言,整个系统的工作状态都体现在总控模块上。各个子模块通过总控模块的控制信号进行信息的交互。

该模块提供各个模块之间调用的控制信号,从而控制整个遗传算法的流程以及数据的流向,协调各个模块有序地工作。总控模块由一个大状态机构成,状态分别为 IDLE、INIT、CROSS、MUT、VALUE 和 STOP。总控模块通过状态机的方式来控制其他模块。状态机如图 3 所示。

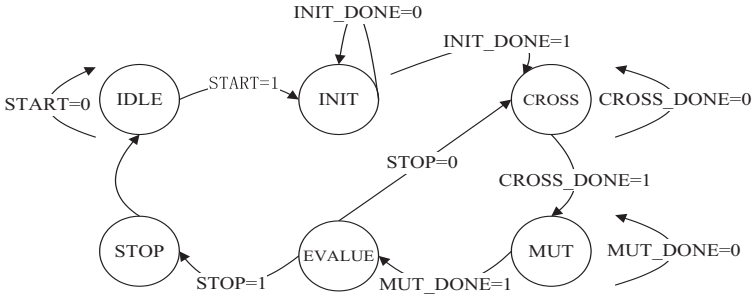

图 3 系统运行状态机

如图 3 所示,本模块中共有 6 个状态。

整个系统的工作流程为:系统复位或者上电后进入 IDLE 状态,系统在 START 信号为 0 的时候则在 IDLE 状态保持等待。当系统得到 START = 1 后,转入 INIT 初始化状态;INIT 状态工作未完成时,INIT_DONE 一直等于 0,并且保持在 INIT 状态继续工作。当初始化工作完成后发出 INIT_DONE = 1 信号,系统转入 CROSS 状态;在杂交操作未完成之前,CROSS_DONE 一直保持为 0,并且保持在 CROSS 状态继续工作。当交叉操作完成后发出 CROSS_DONE = 1 信号,系统转入 MUT 状态;在变异操作未完成之前,MUT_DONE 一直保持为 0,并且保持在 MUT 状态继续工作。当变异操作完成后发出 MUT_DONE = 1 信号,系统转入 EVALUE 状态;评价操作完成后进行停止状态判断,如果满足停止条件,则发出 STOP = 1 信号,系统转入 STOP 状态,否则发出 STOP = 0 信号并转入 CROSS 状态,继续进行循环操作;系统完成 STOP 状态要做的工作后无条件转入 IDLE 状态。

3.2 软件模块

(1)随机数模块。

该模块提供了遗传算法流程中初始化模块中个体生成、杂交选择模块的个体选择和杂交点选择、变异选择模块的个体选择和变异点选择。该模块采用 Xilinx 公司扩展的 C 语言实现。具体实现表现为一个函数。函数有一个控制参数。函数返回为所需求的随机数值。软硬件交互协议的部分由调用该函数的部分控制实现。函数的内部具体实现为根据控制参数来判断硬件所需的随机数的范围。

(2)适应值评价模块。

该模块的功能为完成遗传算法整个流程中的适应值评估工作,也就是所需要解决问题的具体描述。该模块采用 Xilinx 公司扩展的 C 语言实现。具体实现表现为一个函数。函数的参数为控制参数以及需要进行适应值评价的个体(具体表现为一个 50 维数组)。

函数返回为该个体的适应值。软硬件交互协议的部分由调用该函数的部分控制实现。函数的内部具体实现为针对所求问题对二进制编码形式的个体进行相应的计算。

3.3　硬件算法 IP 核建立

由于设计中所有硬件化模块之间存在种种的外在以及内在联系,本设计中将所有的硬件化模块打包生成一个硬件 IP 核。IP 核的生成使用 Xilinx 公司的 EDK 工具提供的 IPIF 接口。

使用 EDK 工具的硬件 IP 核生成向导新建一个 IP 核,在生成过程中同时选择生成使用该 IP 核时所需要的寄存器,共 10 个 32 位寄存器。使用总控模块作为整个 IP 核的次顶层模块。在向导提供的顶层模块中将总控模块进行添加,并且将寄存器与总控模块的输入输出端口进行连接。IP 核资源使用情况如图 4 所示。

Post Synthesis Device Utilization			
Resource Types	Used	Available	Percent
Slices	2 572	13 696	18
Slices Flip Flops	2 463	27 392	8
4 input LUTs	4 754	27 392	17
IOs	109	NA	NA
Bonded IOBs	0	556	0
BRAMs	3	136	2

图 4　遗传算法 IP 核资源使用情况

从图 4 中可以看到,硬件 IP 核占用资源较少,占用整个系统资源不到 18%。说明硬件部分实现体积较小并且功耗较小。IP 核的使用可以极大程度地简化开发者的工作量,并且由于 IP 核可配置,所以为后续设计提供了可扩展空间。

3.4　FPGA 工程平台搭建

使用 EDK 软件新建工程,在新建工程向导时选择开发板型号以及速度等信息。然后再根据本设计中的具体需求,添加 PowerPC、内存、开关等基本设备。然后再往设计中添加新建的遗传算法 IP 核。并且将 IP 核的数据通讯绑定在 OPB 高速总线上。因为本设计中采用了更加直观的视频输出结果,所以还需要在平台中添加视频输出 IP 核,并加入相应的驱动程序。最后根据开发板用户使用手册将管脚进行绑定。至此硬件开发环境搭建完成。

在平台完成综合布局布线生成可下载文件之后,启动 SDK 工具进行软件端程序模块的搭建。在 SDK 工作环境下新建一个 C 语言环境的工程,将随机数模块、适应值评价模块以及软硬件交互协议软件部分在该工程下整合。系统所需要的内存地址数据由 xparameters. h 文件提供。在系统停机之后将算法停机之后的结果通过视频输出。最后编译整个工程,并且将程序代码片段及数据片段存放位置指定与片上内存,将程序使用堆栈等其他数据设置内存中存储。将 SDK 工程产生的 ELF 文件与硬件平台产生的下载文件使用系统自带工具进行联合,生成新下载文件。使用 IMPACT 工具将生成的文件下载到开发板中或者使用 IMPACT 工具新建 CF 卡启动引导文件完成整个设计。

4　数值计算应用实例

在科学研究中,数值计算[15]是一类经常遇到的问题,通常这类问题的复杂度较高,使用普通的算法计算时间较长。使用遗传算法解决数值问题可以起到较好的效果。但是软件实现的遗传算法执行效率低下,所以可以使用软硬件协同工作的方式在不改变算法通用性的基础上加快算法的收敛速度。

4.1　二进制问题中的应用

二进制问题是一类可以有效验证遗传算法功能正确性的基本问题。设计中使用的问题为计算二进制串中“1”的个数。1 的个数越多,效果越优。由于初始化个体为随机初始化,所以在初始化个体中 0 与 1 的比例接近 1∶1,必须通过不断的杂交和变异才可以获得最优解。因为个体串长为 50,所以最优解为 50 个 1。使用软硬件协同的遗传算法解决该问题只需要改写软件端的适应值函数,无需对硬件进行修改。

二进制问题运行效果如图 5 所示。

从图 5 可以看出,算法可以正确找到最优解,运行的代数为 1 272,去除用于判断终止条件的 500 代,算法共运行 700 多代。整个算法运行时间为 4.2 秒。

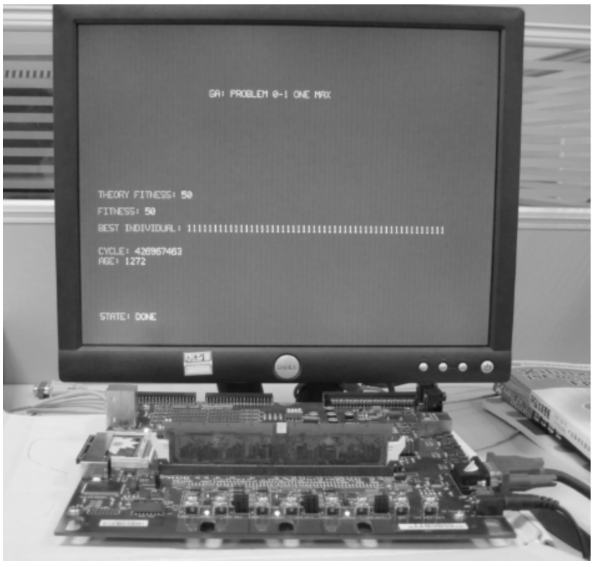

图 5　二进制问题运行效果

4.2　背包问题中的应用

0-1 背包问题是生活中比较常见的一类问题。该类问题描述为有一体积无限大的背包,但是该背包只

能容纳一定重量的货物。如果你有一组货物,每种货物有着不同的重量以及价格,所求问题是如何在这些货物中选取适当的货物(货物个数没有限制),使这个背包在指定的重量范围内,商品的价值最高。在本设计中,个体二进制串长为 50。所以使用的货物总数为 50 个。在适应值评价时,50 个货物与二进制串的二进制位相对应,如果该二进制位为“1”则该货物被选取,为“0”则不选取。最后累加选取货物的重量与价值。首选判断货物总重量,如果总重量大于设定的最大重量,就将适应值设置为-1。否则就将适应值设置为货物的总价值。

设物品的价值为 P,重量为 S,背包容量为 C,分别有:

P = {220,208,198,192,180,180,165,162,160,158,155,130,125,122,120,118,115,110,105,101,100,100,98,96,95,90,88,82,80,77,75,73,70,69,66,65,63,60,58,56,50,30,20,15,8,5,3,1,1}

S = {80,82,85,70,72,70,66,50,55,25,50,55,40,48,50,32,22,60,30,32,40,48,50,32,22,60,30,32,40,38,35,32,25,28,30,22,50,30,45,30,60,50,20,65,20,25,30,10,20,25,15,10,10,4,4,2,1}

C = 1 000

使用软硬件协同的遗传算法解决该问题同样只需要改写软件端的适应值函数,无需对硬件进行修改。本问题经过优化的遗传算法所能获得的最优值为 3 103。而使用普通遗传算法(即本设计所采用的遗传算子)所能获得的最优解为 3 031。

背包问题运行效果如图 6 所示。

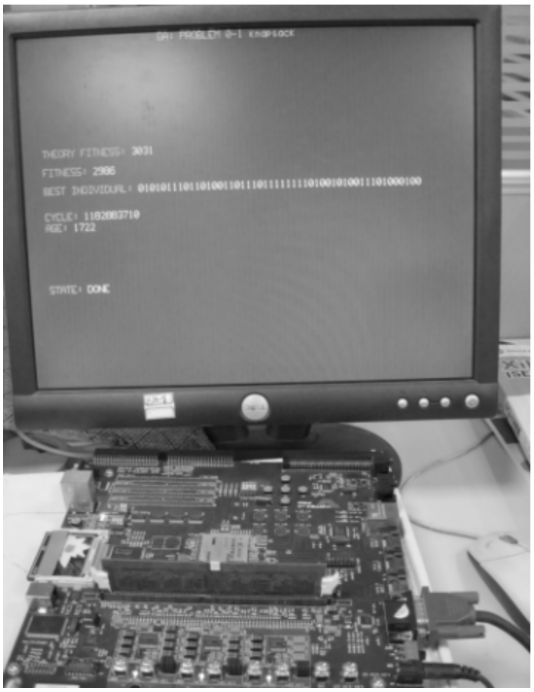

图 6 0-1 背包问题运行效果

从图 6 中可以看出,算法获得的最优解为 2 906,基本接近解 3 031,运行的代数为 1 722,去除用于判断终止条件的 500 代,算法共运行 1 000 多代。整个算法运行时间为 11 秒。

4.3 实验分析

因为遗传算法是一个随机搜索算法,算法的性能受初始化种群的好坏以及杂交变异的随机性影响较大。而以上实验截图均是实验中随机获取的数据,并不能代表设计的性能以及效果。

为了对软硬件协同遗传算法平台的性能进行分析,本设计还利用开发板上 PowerPC 实现了一种纯软件实现的遗传算法。为了验证本设计的性能,这种实现方式的遗传算法在流程和遗传算子上设计都是和本设计完全一致的。为了避免由于随机数带来的偶然性误差,本实验得到大量数据之后,再对数据平均进行数据统计。其中二进制问题的对比数据如表 2 所示。

表 2 二进制问题算法运行效率对比

指标	软件实现	软硬件协同实现
时钟频率	CPU 300 MHz	CPU 300 MHz FPGA 100 MHz
运行时间	4.2 s	0.42 s
最佳价值	50	50
收敛代数	1 223	1 246

0-1 背包问题的对比数据如表 3 所示。

表 3 0-1 背包问题算法运行效率对比

指标	软件实现	软硬件协同实现
时钟频率	CPU 300 MHz	CPU 300 MHz FPGA 100 MHz
运行时间	15.2 s	8.3 s
最佳价值	3 019	3 014
收敛代数	1 491	1 486

从表 2 中可以看到,本设计实现的算法和其他方式实现的算法都可以获得最优解,并且收敛速度基本一致。本设计相比软件实现性能却有了 10 倍的提高。从表 3 中同样可以看到,本设计实现的算法在处理 0-1 背包问题时同样和其他方式实现的算法可以获得近似的最优解,并且在收敛代数上也相差不多。同样可以发现,在实验二中,本设计和软件实现的遗传算法运行时间减少将近一半,已经有了近 100% 的提高。因此可以得出结论,使用软硬件协同方式的遗传算法在保留软件可移植性的基础上还可以大幅提升运行效率。

5 结束语

文中介绍了算法平台设计中的软硬件划分的依

据、软硬件划分的结果以及设计中模块之间的连接关系。随后对设计中的各个硬件模块的具体实现做了详细的介绍。设计了软硬件之间进行信息交互所需要的协议。最后分别介绍了将硬件化模块整合成整体遗传算法 IP 核、FPGA 开发平台搭建、软件平台搭建、整个系统整合。

该方法具有软件的通用性以及硬件的运算高效性,并且通过使用 IP 核节约了开发成本以及开发时间。理论上使用到遗传算法的地方均可以使用该方法提高运行效率,尤其适合在实时性要求较高的场合(例如工业控制)应用。

参考文献:

[1] 潘正君,康立山,陈毓屏. 演化计算[M]. 北京:清华大学出版社 &广西科学技术出版社,2000:1-20.

[2] 王玉体. 基于 FPGA 并行遗传算法的硬件实现技术研究[D]. 南昌:南昌大学,2008.

[3] SUN X,LI J,TIAN F,et al. Design of FPGA hardware based on genetic algorithm[C]//Proceedings of the 3rd international conference on computer engineering,information science & application technology (ICCIA 2019). Nanchang,China:[s. n.],2019.

[4] 付海龙. 遗传算法的 FPGA 实现与加速研究[D]. 长春:吉林大学,2019.

[5] 王法臻,崔少辉,王　成,等. FPGA 可重构仪器 USB 通信接口的设计与实现[J]. 测试技术学报,2020,34(5):451-456.

[6] ZHANG L,TANG Z. Application of FPGA-based genetic algorithm in traffic control[J]. Modern Electronics Technique,2015(15):153-157.

[7] DAMODARAM D, VENKATESWARLU T. FPGA implementation of genetic algorithm to detect optimal user by cooperative spectrum sensing[J]. ICT Express,2019,5(4):245-249.

[8] 潘　松,黄继业. EDA 技术实用教程:VHDL 版[M]. 北京:科学出版社,2010.

[9] MEHRI H,ALIZADEH B. Genetic-algorithm-based FPGA architectural exploration using analytical models[J]. ACM Transactions on Design Automation of Electronic Systems,2016,22(1):1-17.

[10] 王晓霞,宋学君,郭振兴,等. 基于矩阵编码的 CGP 电路进化设计[J]. 河北师范大学学报:自然科学版,2018,42(4):312-317.

[11] FANG Mengxu,TANG Bin. FPGA implementation of an adaptive genetic algorithm[C]// Proceedings of the 12th international conference on service systems and service management (ICSSSM 2015). Guangzhou,China:IEEE,2015.

[12] 范舒颜. 数字电路的模块化优化设计[J]. 电子世界,2019(9):194-195.

[13] 李金洋. 软硬件划分若干算法研究及工具实现[D]. 上海:华东师范大学,2018.

[14] 李胜旺,李伟峰,于　洁. 改进遗传算法在数字电路中的应用[J]. 信息通信,2017(10):115-117.

[15] 文　艺,潘大志. 用于求解 TSP 问题的改进遗传算法[J]. 计算机科学,2016,43(z1):90-92.

(上接第 113 页)

tional architecture for fast feature embedding[J]. arXiv:1408.5093,2014.

[15] CENTER B V A L. Alex's CIFAR-10 tutorial,Caffe style[EB/OL]. 2014. https://github.com/BVLC/caffe/tree/master/examples/cifar10.

[16] KRIZHEVSKY A,HINTON G. Learning multiple layers of features from tiny images[M]//Handbook of systemic autoimmune diseases. [s. l.]:[s. n.],2009.

[17] 张　榜,来金梅. 一种基于 FPGA 的卷积神经网络加速器的设计与实现[J]. 复旦学报:自然科学版,2018,57(2):236-242.

[18] KRIZHEVSKY A,SUTSKEVER I,HINTON G E. ImageNet classification with deep convolutional neural networks[J]. Communication of ACM,2017,60(6):84-90.

[19] IOFFE S,SZEGEDY C. Batch normalization: accelerating deep network training by reducing internal covariate shift[J]. arXiv:1502.03167,2015.

[20] SRIVASTAVA N, HINTON G, KRIZHEVSKY A,et al. Dropout:a simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research,2014,15(1):1929-1958.

[21] CONG J,XIAO B. Minimizing computation in convolutional neural networks[C]//Artificial neural networks and machine learning-ICANN. Hamburg,Germany:Springer,2014:281-290.

[22] SHAWAHNA A,SAIT S M,EL-MALEH A. FPGA-based accelerators of deep learning networks for learning and classification:a review[J]. IEEE Access,2018,7:7823-7859.