文章编号：1672-2892(2012)01-0107-04

# 并行化改进遗传算法的 FPGA 高速实现方法

张妮娜，窦　衡

(电子科技大学 电子工程学院，四川 成都　610054)

**摘　要：**为提高硬件运行速度和资源利用率，利用硬件并行化的思想改进传统算法的处理模式，将遗传算法传统实现方法的控制部分分解到各模块内部，按照流水线模式，应用现场可编程逻辑门阵列(FPGA)高速实现。综合后时钟频率达到 137.08 MHz，演化 1 代需 64 个时钟周期，即 0.467 μs。实现结构节约硬件资源，效率高，使大规模遗传算法的高速硬件实现成为可能。

**关键词：**遗传算法；硬件并行化；现场可编程逻辑门阵列；演化

**中图分类号：**TN74　　**文献标识码：**A

# Methodology of realizing FPGA for improved parallel genetic algorithm

ZHANG Nina, DOU Heng

(School of Electronic Engineering, UESTC, Chengdu Sichuan 610054, China)

**Abstract:** To enhance the operation speed and utilize the resource, according to the idea of hardware parallel method, one traditional implementation of genetic algorithms is improved by separating controlling part into other components and using Field Programmable Gate Array(FPGA) to realize control in pipelining mode. The result of synthesis shows its frequency can reach 137.08 MHz for evolution of a generation needing 64 cycles(namely 0.467 μs). With optimized hardware resources and high efficiency, the realized structure demonstrates the possibility of large-scale and high-speed hardware realization of genetic algorithms.

**Key words:** genetic algorithms; hardware parallel; Field Programmable Gate Array; evolution

遗传算法能够模拟自然界优胜劣汰适者生存的进化过程，搜索出最优解，被广泛用于实现各种复杂的应用场合，如图像处理、自适应控制、通信系统、人工神经网络等[1-5]。一般情况下，遗传算法都需要大量的运算进而达到优化的过程。目前，人们大都采用软件实现遗传算法，这使遗传算法的实现受到计算机运行速度的影响，在实时性要求高的场合的应用受到限制，因此研究基于硬件实现的遗传算法十分必要。新一代现场可编程逻辑门阵列(FPGA)的出现，使遗传算法的硬件实现变为现实，且更加方便。本文采用 Verilog 硬件描述语言，设计了一种基于 FPGA 的遗传算法硬件系统，并分别采用 Modelsim 和 Quartus II对设计进行了功能仿真和综合后时序仿真。仿真结果验证了设计的正确性，并且大大缩短了运算时间，较传统的遗传算法 FPGA 实现在效率上有所提升。

## 1　硬件实现概述

现代的生物是经过一代一代的进化发展起来的，其基本特征大致可以包括生长、繁殖、新陈代谢、遗传和变异。而遗传算法是对这种生物的遗传进化过程的模拟，为人工自适应系统的开发设计提供了明朗的前景。

图 1 给出了遗传算法的计算流程，可以看出一个遗传算法分为群体初始化、适应度评估和遗传操作(选择、交叉、变异)3 部分，其中选择、交叉和变异是整个算法流程的主要部分。图 1 中的适应度计算、选择、交叉和变异操作的执行方式类似于流水线操作，且具有并行性，

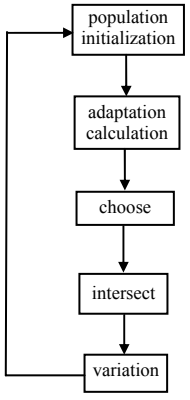


Fig.1 Flow chart of analyzing genetic algorithm
图 1 遗传算法计算流程

**收稿日期：**2011-02-25；**修回日期：**2011-06-21
**基金项目：**国家自然科学基金资助项目(61001032/F010501)

使其非常适合于用硬件实现。

一般的基于 FPGA 的遗传算法实现都设计了一个控制模块来协调整个算法实现的各个模块的运作，如图 2 所示。它控制整个设计的数据流向，使各模块在控制模块发出的控制信号下有节奏、有顺序地工作。然而这样却使各模块并不能得到并行的执行，如选择模块工作时，不能进行交叉的操作。因此整个遗传算法的收敛时间会比较长，不能较好体现出硬件实现的优势。本设计省去了控制模块的模块任务分配控制，把具体的控制分解到模块内部，并且按照流水线处理模式，使各个模块同时进行，极大提高了种群的进化速度和收敛速度。

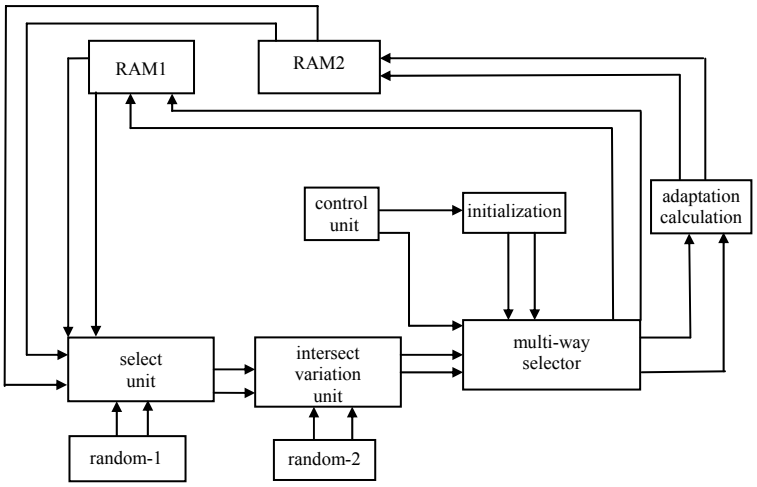


Fig.2 Conventional realization of genetic algorithms

图 2 传统的遗传算法实现框图

图 3 给出的各个功能模块复位后便同时工作，采用流水线技术，如 2 个选择模块并行执行选择，与此同时，上一个周期的数据按照指定概率被执行交叉和变异，依此类推。由图 3 可以看出，遗传算法的实现被分为了 9 个模块，依次为初始化模块、2 个选择模块、随机数产生、交叉/变异模块、多路选择/地址发生模块、适应度计算模块、存储器模块和随机读地址发生模块。没有控制模块，每个模块同时执行，节省了进化的时间，并且所使用的芯片资源也会得到节省。各模块功能如下。

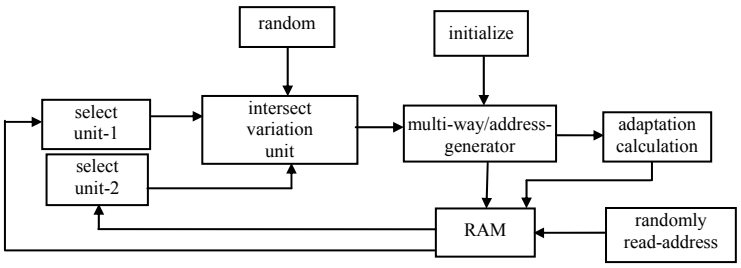


Fig.3 Improved genetic algorithm diagram

图 3 改进的遗传算法实现框图

### 1.1 初始化模块

当复位信号有效时，本模块开始工作，多路选择器此时也选中初始化模块。每个时钟随机产生 2 个个体，直到装满存储模块时再让多路选择器选择初始化模块 3 个周期，这样的目的是保证当多路选择器选择到交叉/变异模块时一定有数据，因为 1 组数据从存储模块读出，通过选择模块和交叉/变异模块各需 1 个周期。此后多路选择器不再选择此模块，并长期选中交叉变异模块。

### 1.2 随机数模块

2 个随机数模块所产生的随机数将供给另外 2 个模块使用。一个在交叉变异模块中用于判定是否达到交叉和变异的比率，用随机数来决定是否进行交叉或变异操作，以及在哪个比特位进行交叉和变异；另一个随机模块产生的数据用来产生随机的读地址，即让选择是自然选择。这 2 个模块每个时钟均输出 2 个随机数。本模块在 FPGA 内利用线性反馈移位寄存器(Linear Feedback Shift Registers，LFSR)结构实现伪随机数发生器。这种方法不仅结构简单，易于实现，而且所产生的伪随机序列具有周期长，随机特性好的特点[6-8]。

### 1.3 交叉变异模块

每个时钟周期得到 1 对来自选择模块的个体，同时接收到来自随机模块产生的 2 个随机数，一个用来判定是否进行交叉操作；另一个用来决定是否进行变异操作。当 1 个随机数小于交叉概率，则将 2 个来自选择模块的个体按照指定的交叉点进行交叉，否则保持不变；再判定另一个随机数，若小于变异概率，则将上一操作结果的指定位取反，否则保持不变。这里取交叉概率为 0.875，变异概率为 0.073。即随机数 1 小于 8'b11100000 则进行交叉，随机数 2 小于 8'b00010010 则进行变异操作。

### 1.4 选择模块

2 个选择模块，功能一模一样。由于存储模块里存储了当代所有的个体及其适应度值，选择模块每个时钟都

从存储单元中取出 2 个个体，选择适应度大的个体，淘汰掉小适应度的个体。这样才会使较优秀的个体两两结合，将其优势繁衍至下一代，进而更快地实现优化。

### 1.5 多路选择/地址发生模块

系统复位进入初始化群体阶段，多路选择器/地址发生模块，选择初始化模块，当个体数量达到要求时，选择交叉变异模块。同时每个时钟顺序产生 2 个地址以供给存储目标个体。

### 1.6 适应度计算模块

本模块完成对个体的适应值进行评估计算。这里适应度函数选取为 $f(x)=\sum_{n=1}^{2}x_n^{\,2}$，适应度值选取长度为 17 bit，由于本设计将传统的遗传算法实现简化，因此有足够的资源用以实现更复杂的适应度函数。

### 1.7 存储模块

用于存储个体及其适应度值，每个时钟可同时写入和存入 2 个个体及其适应度值。

## 2 仿真与综合实现

设计选用 Altera 的 Statix Ⅱ芯片，采用 Quartus Ⅱ进行综合，采用 Modelsim 进行波形仿真。仿真的参数设置为：每一代的个体数为 32 个，每个个体含有 2 个基因，且每个基因长度为 8 bit 二进制码，即个体长度为 16 bit 的二进制码。适应度函数选取为 $f(x)=\sum_{n=1}^{2}x_n^{\,2}$，适应度值选取长度为 17 bit。


Fig.4 Simulated evolution-waveform

图 4 进化波形仿真图

由图 4 可以看出，第 3 代最后个体染色体值发展为 1111111011111011，到第 4 代的时候，最优个体已经能够稳定达到 1111111111111011。能否进化到最优，与交叉和变异的具体操作有关，个体到这个程度已经进入局部收敛，离全 1 的最优值还有一定距离，这还需要进一步研究算法和更多的人参与。综合后，系统资源使用情况如下：

| | |
|---|---|
| Flow Status | Successful-Mon Nov 29 21:32:13 2010 |
| Quartus II Version | 9.0 Build 132 02/25/2009 SJ Full Version |
| Revision Name | ea_top |
| Top-level entity name | ea_top |
| Family | Stratix II |
| Met timing requirements | Yes |
| Logic utilization | 42% |
| Combinational ALUTs | 2,812/12,480(23%) |
| Dedicated logic registers | 2,557/12,480(20%) |
| Total registers | 2557 |
| Total pins | 41/343(12%) |
| Total virtual pins | 0 |
| Total block memory bits | 0/419,328(0%) |
| DSP block 9-bit elements | 4/96(4%) |
| Total PLLs | 0/6(0%) |
| Total DLLs | 0/2(0%) |
| Device | EP2S15F484C3 |
| Timing Models | Final |

时钟频率可以达到 137.08 MHz，演化 1 代需要 64 个时钟周期，即需用时 0.467 μs。由此可以得出进化稳定仅需 1.868 μs(0.467 μs×4=1.868 μs，进化 4 代进入稳定阶段)。对比基于传统结构的遗传算法 FPGA 实现(进化稳定所需时间约为 300 μs[9])具有明显优势。虽然适应度函数的选择在很大程度上影响进化时间，不过这组数据证明了改进的遗传算法实现结构能大大提高算法的执行效率。

(下转第 117 页)