第 36 卷 第 4 期 计 算 机 工 程 2010 年 2 月
Vol.36 No.4 Computer Engineering February 2010
·人工智能及识别技术· 文章编号：1000—3428(2010)04—0166—03 文献标识码：A 中图分类号：TP301.6

粒子群优化算法的硬件实现及其性能分析

蔡 瑞，须文波，柴志雷，王 斌，刘 凡
(江南大学信息工程学院，无锡 214122)

摘 要：介绍量子粒子群优化(QPSO)算法的硬件实现方法并对其进行性能分析。将 QPSO 算法应用于现场可编程门阵列开发板，并对比了不同硬件实现方式的运算速度和资源耗费。采用硬件并行和流水技术缩短算法的运算时间，仿真结果表明，硬件化 QPSO 的运算时间为原 Matlab 中运算时间的 0.032%。

关键词：量子粒子群优化；现场可编程门阵列；硬件实现

Hardware Implementation and Capability Analysis of Particle Swarm Optimization Algorithm

CAI Rui, XU Wen-bo, CHAI Zhi-lei, WANG Bin, LIU Fan
(School of Information Engineering, Jiangnan University,Wuxi 214122)

【Abstract】 This paper introduces the hardware implementation of Quantum-behaved Particle Swarm Optimization(QPSO) algorithm and analyses its capability. The algorithm runs in FPGA. The operation speed and resource using with hardware implementation methods are compared. The pipeline technology shortens the runtime enormously. Simulation result indicates that runtime of Field Programmable Gate Array(FPGA) based QPSO achieves about 0.032% of runtime on Matlab.

【Key words】 Quantum-behaved Particle Swarm Optimization(QPSO); Field Programmable Gate Array(FPGA); hardware implementation

1 概述

在科研和日常生活的诸多领域中经常需要用到一些优化算法。影响优化算法实用性的一个重要因素是其运算速度，为了提高算法的运算速度，越来越多的算法被用硬件来实现，如遗传算法的现场可编程门阵列(Field Programmable Gate Array, FPGA)实现[1]，椭圆曲线密码加密算法的硬件实现[2]等。FPGA 同复杂可编程逻辑器件(Complex Programmable Logic Device, CPLD)一样都是可编程逻辑器件，但逻辑资源更丰富，拥有完整的输入工具和仿真工具[3]，具有很强的可编程能力，用 FPGA 作为硬件平台，弥补了传统特定用途集成电路(Application Specific Intergrated Circuits, ASIC)灵活性不足的缺点。

粒子群优化(Particle Swarm Optimization, PSO)算法[4]是一种进化计算技术，其算法简单、容易实现、并且没有许多参数调整，是一种有效的优化算法，拥有众多应用领域，如多目标优化、模式识别、决策支持等。目前，研究人员还在不断地对 PSO 算法进行改进优化，其中，量子粒子群优化(Quantum-behaved Particle Swarm Optimization, QPSO)算法[5]是在 PSO 的基础上结合量子理论提出的新的微粒群算法，提高了全局优化搜索能力。但目前 QPSO 多用软件实现，当面对大规模和复杂计算问题时，软件方式往往难以满足系统的实时性需求。针对此问题，采用串行方式实现了 QPSO 的硬件加速，取得了较好的效果[6]。本文采用并行流水技术，进一步提高 QPSO 算法的运算速度。

2 PSO 与 QPSO 算法简介

2.1 PSO 算法

在 PSO 算法中，粒子通过不断调整自己的位置 X 来搜索新解。每个粒子都能记住自己搜索到的最好解，记作 P_{id}；整个粒子群经历过的最好的位置，即目前搜索到的最优解，记作 P_{gd}。每个粒子都有一个速度，记作 V，于是有

$$V_{id}=\omega_{id}V_{id}+\eta_1 rand()(P_{id}-X_{id})+\eta_2 rand()(P_{gd}-X_{id}) \quad (1)$$

其中，V_{id} 表示第 i 个粒子在第 d 维上的速度；ω 为惯性权重；η_1,η_2 为调节 P_{id} 和 P_{gd} 相对重要的参数；$rand()$为随机函数。这样，可以得到粒子移动的下一位置：

$$X_{id}=X_{id}+V_{id} \quad (2)$$

2.2 QPSO 算法

QPSO 是一种改进的 PSO 算法，结合量子行为理论，对以前的 PSO 算法的缺点进行了改善。它们的区别主要体现在进化公式上，QPSO 算法的进化公式如下：

$$mbest=\frac{1}{M}\sum_{i=1}^{M}P_i=\left(\frac{1}{M}\sum_{i=1}^{M}P_{i1},\frac{1}{M}\sum_{i=1}^{M}P_{i2},\cdots,\frac{1}{M}\sum_{i=1}^{M}P_{id}\right) \quad (3)$$

$$p_{id}=\varphi P_{id}+(1-\varphi)P_{gd},\ \varphi=rand() \quad (4)$$

$$x_{id}=p_{id}\pm\alpha\left|mbest_d-x_{id}\right|\ln\left(\frac{1}{u}\right),\ u=rand() \quad (5)$$

其中，α 为活力系数；M 为粒子的数目；d 为粒子的维数；$mbest$ 是粒子群 $pbest$ 的中间位置；p_{id} 为 P_{id} 和 P_{gd} 之间的随机点。

QPSO 算法拥有很多优点，其参数少，全局优化效果好，算法本身具有很高的潜在可并行性，非常适合在 FPGA 上实

基金项目：国家自然科学基金资助项目“高可靠实时系统的计算平台(SoPC)研究”(60703106)
作者简介：蔡 瑞(1985－)，男，硕士研究生，主研方向：嵌入式系统，现场可编程门阵列；须文波，教授、博士生导师；柴志雷，副教授、博士；王 斌、刘 凡，硕士研究生
收稿日期：2009-09-30 **E-mail：**chbcrcss@163.com

现其并行计算。

3 算法的硬件实现

3.1 系统结构设计

QPSO 算法拥有良好的可并行性，因为粒子之间在进化时互相之间无数据相关，而是可以单独更新自己的位置。可以将全部粒子分组分别进化，采用上述方法实现 QPSO 算法，能够在更大程度上提高算法的运行效率。首先，算法的整体结构设为并行的，将所有粒子分为 N 个小的子群，每个子群单独进化。在本实验中，使用适应函数

$$f(x)=\sum_{i=1}^{n}x_i,\ 0<x_i<128$$

采用 8 个粒子，3 维维数，将粒子分成 2 个子群，每个子群有 4 个粒子进行运算，然后在每一代的最后结果中选取最优的全局值作为结果输出。同时，子群内部可以采用流水线进行优化。

3.2 系统原理与数据通路

本算法的子群原理图如图 1 所示。因为整个算法的数据量比较大，数据流向比较复杂，所以在实现时将整个算法进行模块化，这样可以使数据流动更加明确，仿真调试时更方便。

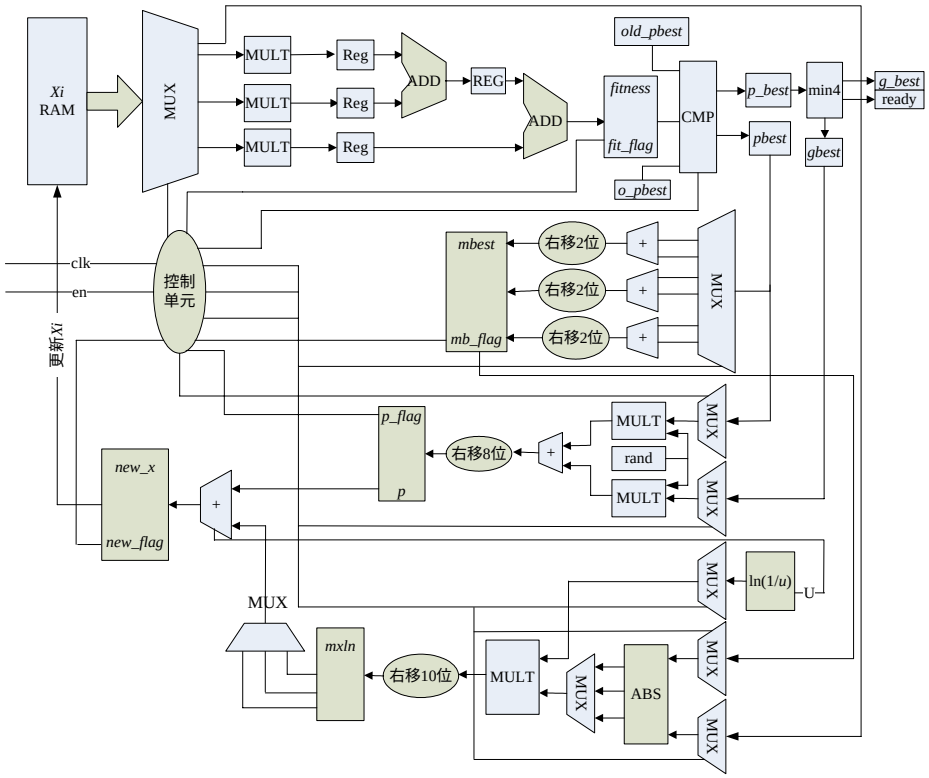

图 1 子群系统原理图

对于一个子群模块来说，一共分成了 8 个模块，下面就在对模块进行介绍的同时对数据通路进行描述。

(1)粒子适应值求取模块。计算粒子适应值是一个复杂的过程，也是占用整个算法时间最多的模块，在考虑到 FPGA 开发板的资源的情况下，尽可能地对此模块使用并行设计，以此来提高该模块的运行速度，从而提高整个算法的运行效率。因此，对每个粒子的多维空间进行并行运算，另外，乘法器的输出结果输出到流水线寄存器组，使得从整体上构成流水线。

(2)粒子局部最优值计算模块。这个模块比较简单，实际上就是一个比较模块，因为这个过程就是将粒子的当前局部最优值 *p_best* 与它当前计算出的适应值 *fitness* 进行比较，根据它们的大小来更新局部最优值 *p_best*，并得出当前粒子最优位置向量。

(3)全局最优值 *g_best* 更新模块：这个过程就是在所有粒子的局部最优值 *p_best* 中找到一个更好的值 *g_best*，对于本实验就是要找最小值。为了能及时应对每个粒子更新后产生新的 *p_best* 对全局最优值的影响，本模块采用数据流描述，以在一个时钟找到最优值，图 2 为仿真结果。

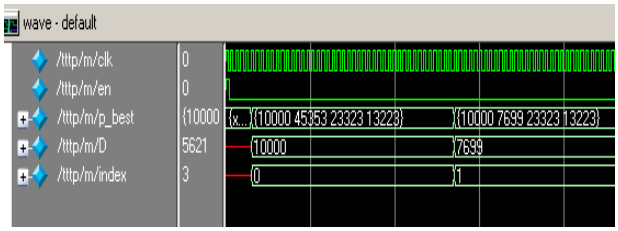

图 2 *g_best* 模块仿真图

(4)*mbest* 求取模块。这个模块是求取所有粒子的局部最优值 *pbest* 的中间值 *mbest*，如果要将所有运算展开的话，会消耗很多加法器，所以在此使用累加器进行计算，使用 3 个累加器就可以使所有粒子的 *pbest* 的三维同时进行计算，较好地考虑了速度与资源消耗的平衡。

(5)随机值 P 的求取模块。此模块是计算 *pbest* 和 *gbest* 之间的随机值 P，这个过程也用到大量计算，如果要全部并行的话要用 6 个乘法器和 3 个加法器。为了能达到平衡，采用了 2 个乘法器、1 个加法器和流水线技术用来应对整个算法的需要，这个过程还要用到随机小数。为了避免实现复杂的浮点数运算，本实验采用的是先扩大再缩小的定点数方法来计算。

(6)ln(1/u)求取模块。本模块结合 FPGA 的特点，采用查找表(LUT)法来实现对数的求取，先用 Matlab 求出 1/u 对应的对数值，把它们以地址和数据的形式存放在 RAM 中，当需要使用时就可直接调用。

(7)粒子位置更新模块。本模块就是将上述模块中得到的数据进行汇总和计算，得到粒子的新位置 *Xi*，从而实现周期循环运算。该模块中主要运算有绝对值求取、乘法、加法和移位。

(8)随机数产生模块。在本算法中，随机数使用了很多次，为了能方便地使用随机数，单独设计了一个随机数产生模块，因为编程工具里的随机系统函数只能在功能仿真中使用，不能进行综合，所以本模块采用线形反馈移位寄存器(LFSR)来产生随机数。

最后，在顶层模块中调用上述模块，添加控制信号使所有模块正常协调运行，构成整个算法的运行。为了防止早熟，还在每个子模块内部设置通信信号，在每循环一次后，子模块间选取最好粒子位置进行交换，加快优化进度。

4 仿真结果与讨论

本实验使用上述函数和参数，并用硬件描述语言 verilog

HDL 实现了并行 QPSO 算法 PQPSO。为了便于说明，在 ModelSim 6.2b 中获得仿真结果，如图 3 所示。

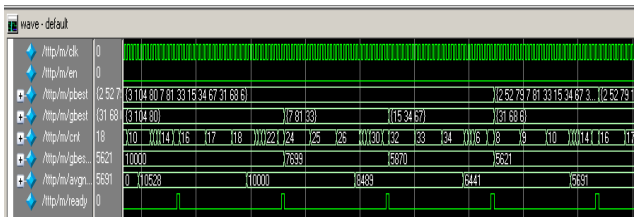

图 3　实验仿真结果

从仿真结果可以看出，算法在进行搜索化时比较均匀，没有出现早熟问题。在使用并行和流水线技术之后，运行速度也大大地提高了，算法经过一次循环迭代只需 960 ns。为了能更好地说明并行的优越性，本实验还设计了一个串行的 QPSO 算法的硬件实现。在相同硬件环境下，对并行 QPSO 算法(PQPSO)和串行 QPSO 算法(SQPSO)在搜索到 0 时的运行数据及综合情况进行比较，比较结果如表 1 所示。

表 1　PQPSO 与 SQPSO 在同等条件下的比较

算法	运行时间/μs	频率/MHz	Slices 使用率/(%)	LUT 使用率/(%)	MULT18×18 使用率/(%)	优化效果
PQPSO	44.58	78.8	48	45	12	搜索均匀，无早熟现象
SQPSO	764.5	49.5	35	33	6	前期搜索均匀，后期较慢

从比较中可以看出，PQPSO 无论搜索效果还是运行时间都比串行的 SQPSO 有较大的优势。从综合的结果来看，PQPSO 的资源消耗会多一些，但是除了乘法器是 SQPSO 的 2 倍以外，其余的资源使用指标都没有高出太多。所以，当开发板有足够的资源进行开发时，就可以用资源来换速度。

为了更好地说明算法在软硬件上的差别，下面与 QPSO 算法在 Matlab 里实现时得出的数据进行比较，Matlab 的运行环境是 windows XP 操作系统，处理器为 P4 2.66 GHz，内存为 512 MB。为了能得到运行时间的差异，表 2 是对同一函数在相同迭代次数下的比较。需要说明的是，Matlab 中采用浮点数，数据精度较高，随机优化时范围波动较大，表中的数据是在 30 次迭代次数下的平均值。

表 2　Matlab 与 ModelSim 的测试结果

测试	运行时间	最优适应值
硬件测试(PQPSO)	44.58 μs	0
软件测试(Matlab)	140.6 ms	0.012 2

从表 2 中可以看出，硬件方式的 QPSO 计算性能可达软件方式的 3 140 倍。而且上述例子规模较小，当遇到规模复杂的问题时，更能体现出硬件实现的优越性。最后搜索到的最优适应值不同，是因为两者使用的数据精度不同。不过上表是在相同的迭代次数下进行的比较，影响不大，而且从仿真的结果来看，搜索优化时更加均匀，达到了算法的要求。

前文已经提到，对数 ln(1/u)是通过查找表实现的，是将它对应的值放在一个 RAM 时，这样就难免造成误差。为了能得到更好的结果，在实验中对 u 的临界值进行了测试，其中是把 u 的值扩大映射到(0, 128)进行计算。表 3 是几组临界值对实验运行结果影响的测试，条件是都找到最优适应值 0。

表 3　u 临界值与实验结果的关系

临界值	运行时间/μs	搜索效果
40	66.70	搜索均匀，无早熟现象
50	52.80	搜索均匀，后期稍慢
60	44.58	搜索均匀，无早熟现象
70	55.40	搜索均匀

5　结束语

本文采用流水和并行方法在 FPGA 上实现了 QPSO 算法，通过实验表明，硬件化后的 QPSO 算法比在软件上运行速度有很大的提高，同时在资源和速度间达到了很好的平衡。下一步将重点研究如何采用浮点数来提高算法的运算精度，以推进 QPSO 算法最终在实际领域中得到具体的应用。

参考文献

[1] 肖宏峰, 谭冠政. 并行遗传算法的 FPGA 硬件实现研究[J]. 小型微型计算机系统, 2008, 29(6): 1179-1184.

[2] Chelton W N, Benaissa M. Fast Elliptic Curve Cryptography on FPGA[J]. IEEE Transactions on Very Large Scale Integration Systems, 2008, 16(2): 198-205.

[3] Wolf W. 基于 FPGA 的系统设计[M]. 闫敬文, 译. 北京: 机械工业出版社, 2006.

[4] Jin Nanbo, Rahmat-Samii Y. Advances in Particle Swarm Optimization for Antenna Designs: Real-number, Binary, Single-objective and Multiobjective Implementations[J]. IEEE Transactions on Antennas and Propagation, 2007, 55(3): 556-567.

[5] Sun Jun, Feng Bin, Xu Wenbo. Particle Swarm Optimization with Particles Having Quantum Behavior[C]//Proc. of Congress on Evolutionary Computation. [S. l.]: IEEE Press, 2004: 225-331.

[6] He Yinghui, Xu Wenho, Chai Zhilei. A Parallel Platform for QPSO's High Performance Computing[C]//Proc. of International Symposium on Distributed Computing and Applications for Business Engineering and Science. [S. l.]: IEEE Press, 2008: 201-205.

编辑　顾逸斐

(上接第 165 页)

6　结束语

本文提出了一种可用于智能移动设备的掌纹识别算法，设计了智能移动设备下掌纹图像的采集方式和定位分割方法，确保了掌纹识别用于智能移动设备的有效性。在特征提取中，对 Gabor 特征点进行优化选取，在保证识别精度的同时显著地提高了识别的效率。在本文算法的基础上，成功开发了联想 ET980 智能手机下的掌纹验证系统，测试结果表明该系统在验证精度和效率上均满足实时验证系统的要求。

参考文献

[1] Jain A K, Ross A, Prabhakar S. An Introduction to Biometric Recognition[J]. IEEE Transactions on Circuits and System for Video Technology, 2004, 14(1): 4-20.

[2] Han Chin-Chuan, Chen Hsu-Liang, Lin Chih-Lung, et al. Personal Authentication Using Palmprint Features[J]. Pattern Recognition, 2003, 36(2): 371-381.

[3] Johson D J, Rahman Z, Woodell G A. Properties and Performance of a Center/Surround Retinex[J]. IEEE Transactions on Image Processing, 1997, 6(3): 451-462.

[4] Lee T S. Image Representation Using 2D Gabor Wavelets[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1996, 18(10): 959-971.

编辑　顾逸斐