OFDM系统中IFFT/FFT处理器的设计与实现

刘洪涛,杨红官,胡赞民
LIU Hongtao,YANG Hongguan,HU Zanmin

湖南大学 物理与微电子科学学院,长沙 410082
College of Physics and Microelectronics,Hunan University,Changsha 410082,China
E-mail:hnulht@163.com

LIU Hongtao,YANG Hongguan,HU Zanmin.Design and implementation of IFFT/FFT processor in OFDM system.Computer Engineering and Applications,2011,47(1):60-63.

Abstract: An optimized Radix-4 FFT algorithm is proposed firstly.Based on this algorithm,a pipelined 64-point IFFT/FFT processor,which can obtain 64-point output during 64 clock cycles only by three complex multipliers so as to improve its operation speed and save its hardware resources,is designed.The verification is carried out through the downloaded Xilinx XC2S300E Spartan2E series xc2s300e device,whose result is less than 0.5% difference compared with the result from the MATLAB calculation.The processor has been successfully used to an OFDM communication system.

Key words: Orthogonal Frequency Division Multiplexing(OFDM);Inverse Fast Fourier Transform/Fast Fourier Transform(IFFT/FFT);Radix-4;Field Programmable Gate Array(FPGA)

摘 要:提出了Radix-4 FFT的优化算法,采用该优化算法设计了64点流水线IFFT/FFT处理器,该处理器可以在64个时钟周期内仅采用3个复数乘法器获得64点处理结果,提高了运算速度,节约了硬件资源。通过Xilinx XC2S300E Spartan2E系列的xc2s300e器件进行下载验证,仿真结果与MATLAB计算结果误差小于0.5%,该处理器已经成功应用于某OFDM通信系统中。

关键词:正交频分复用(OFDM);反快速傅里叶变换/快速傅里叶变换(IFFT/FFT);基4算法;现场可编程门阵列(FPGA)
DOI:10.3778/j.issn.1002-8331.2011.01.017 **文章编号:**1002-8331(2011)01-0060-04 **文献标识码:**A **中图分类号:**TN492

1 引言

伴随着无线数据通信与多媒体应用的不断发展,正交频分复用(Orthogonal Frequency Division Multiplexing,OFDM)作为一种新型的物理层传输技术正越来越受到人们的重视,目前OFDM已是IEEE802.11a无线局域网(WLAN)的核心调制技术,被视为下一代移动通信(4G)中的关键技术[1]。

为了在OFDM系统中达到各个子载波之间的相互正交,通常采用IFFT/FFT作为调制解调的一部分[1],该方法中如何降低FFT处理器的复杂度,实现高速率、低功耗以及低硬件成本成为实现OFDM系统的关键技术之一。在阐述OFDM原理的基础上,利用提出的Radix-4优化算法设计了64点IFFT/FFT处理器并利用先进的FPGA硬件平台对设计进行验证,该处理器具有运算速度快、精度高、资源节省的优点。

2 OFDM系统的原理与算法

2.1 基于IFFT/FFT的OFDM系统基本原理

OFDM的基本思想是把高速率的信源信息流通过串并变换,变换成低速率的N路并行数据流,然后将这N路数据流分别调到N个相互正交的子载波上,再将N路调制后的信号相加即得发射信号[2]。设一个OFDM符号之内包含N个经过相移键控(PSK)或者正交幅度调制(QAM)的子载波。T表示OFDM符号的持续时间(周期),则OFDM的基带信号为:

$$s(t)=\sum_{n=0}^{N-1}d_n e^{j2\pi\frac{nt}{T}} \quad (1)$$

对于信号$s(t)$以T/N的速率进行抽样,即令$t=kT/N$(k=0,1,…,$N-1$),则可得:

$$s_k=s(kT/N)=\sum_{n=0}^{N-1}d_n e^{j2\pi\frac{nk}{T}} \quad (0\leq k\leq N\text{-}1) \quad (2)$$

可以看出,抽样值刚好为N点反离散傅里叶变换(IDFT)。同样在接收端,恢复原始数据符号d_n的处理就可以通过对S_k进行N点离散傅里叶变换(DFT)获得[2]。由此可见,OFDM系统的调制和解调可以分别通过IDFT/DFT来实现。图1为OFDM系统的调制与解调框图。

在OFDM系统的实际应用中,N点IDFT运算需要进行N^2次复数乘法和$N(N-1)$次复数加法,对于子载波数量非常大的OFDM系统而言,可以进一步采用Radix-2,Radix-4或分裂基FFT算法。N点Radix-2 FFT算法中需要$\frac{N}{2}\text{lb}N$次复数乘法和$N\text{lb}N$次复数加法;N点Radix-4算法仅需$3N\log_4^N$次复

作者简介:刘洪涛(1983—),男,硕士研究生,主要研究方向为电子系统及数字专用集成电路设计;杨红官(1968—),男,博士,副教授,主要研究方向为纳米级电子器件的理论、设计和测试;胡赞民(1982—),男,硕士研究生,主要研究方向为电子系统及数字专用集成电路设计。

收稿日期:2010-06-10 **修回日期:**2010-10-08

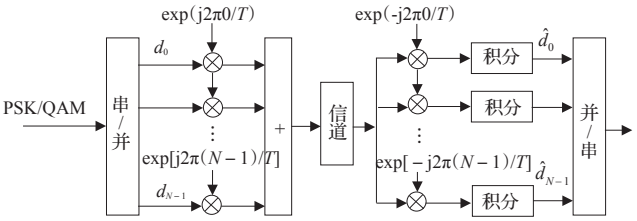

图1　OFDM系统调制解调框图

数乘法和$3N\log_4^N$次复数加法[3]，同Radix-2算法相比能够减少加法器和乘法器的使用数量，从而节省硬件资源、降低功耗、提高运算速度，满足OFDM技术的实时性要求。同时Radix-4 FFT算法的优点是只存在与{1，－1，j，－j}的相乘运算，因此可以不使用完整的乘法器，而通过简单的加减法以及交换实部和虚部（当与j或－j相乘时）就可以实现乘法运算。因此优先选用Radix-4 FFT算法。

2.2　Radix-4 FFT算法的优化

采用Radix-4 IFFT算法对信号进行抽样，则$x(n)$的IFFT可以表示为[4]：

$$x(n)=\frac{1}{N}\left[\sum_{k=0}^{3}X(\frac{N}{4}k)\exp[\frac{\mathrm{j}kn\pi}{2}]+\sum_{k=0}^{3}X(\frac{N}{4}k+1)\exp[\mathrm{j}(\frac{k}{2}+\frac{2}{N})n\pi]+\cdots+\sum_{k=0}^{3}X(\frac{N}{4}k+\frac{N}{4}-1)\exp[\mathrm{j}(\frac{k}{2}+\frac{2}{N}(\frac{N}{4}-1))n\pi]\right] \quad (3)$$

化简可得：

$$x(n)=\frac{1}{N}\left[\exp[\frac{\mathrm{j}2n\pi}{N}(0)]\sum_{k=0}^{3}X(\frac{N}{4}k+0)\exp[\frac{\mathrm{j}kn\pi}{2}]+\exp[\frac{\mathrm{j}2n\pi}{N}(1)]\sum_{k=0}^{3}X(\frac{N}{4}k+1)\exp[\frac{\mathrm{j}kn\pi}{2}]+\cdots+\exp[\frac{\mathrm{j}2n\pi}{N}(\frac{N}{4}-1)]\sum_{k=0}^{3}X(\frac{N}{4}k+\frac{N}{4}-1)\exp[\frac{\mathrm{j}kn\pi}{2}]\right] \quad (4)$$

式(4)可以简写为：

$$x(n)=\frac{1}{N}\left[\sum_{l=0}^{N/4-1}\exp[\frac{\mathrm{j}2ln\pi}{N}]\sum_{k=0}^{3}X(\frac{N}{4}k+l)\exp[\frac{\mathrm{j}kn\pi}{2}]\right] \quad (5)$$

令　$$P_l(n)=\exp(\frac{\mathrm{j}2ln\pi}{N}) \quad (6)$$

$$Q_l(n)=\sum_{k=0}^{3}X(\frac{Nk}{4}+l)\exp(\frac{\mathrm{j}kn\pi}{2}) \quad (7)$$

将公式(6)、公式(7)代入公式(5)得：

$$x(n)=\frac{1}{N}\sum_{l=0}^{N/4-1}P_l(n)Q_l(n) \quad (8)$$

令　$$R_l(n)=P_l(n)Q_l(n) \quad (9)$$

则式(8)化简为：

$$x(n)=\frac{1}{N}\sum_{l=0}^{N/4-1}R_l(n) \quad (10)$$

根据式(10)提出了优化Radix-4算法的IFFT运算框图，如图2所示。图2中产生一个输出值$x(n)$只需要进行$N/4$次$Q_l(n)\times P_l(n)$的复数乘法运算，其中$Q_l(n)$单元中4个输入值与$\exp(\mathrm{j}kn\pi/2)$的运算并不需要复数乘法器，因为根据k（4个端口对应k值分别为0，1，2，3）与n乘积的不同，$\exp(\mathrm{j}kn\pi/2)$分别等于{+1，－1，+j，－j}，可以采用简单的加减法以及交换实部和虚部来实现乘法运算，$P_l(n)$则可以通过访问查找表获得。因此每个输出值的计算时间减少为$N/4$个时钟周期，计算N个输出值的时间将缩短为$N\times N/4=N^2/4$个时钟周期。

考虑旋转因子的周期性，定义一个整型变量β（β为4的整

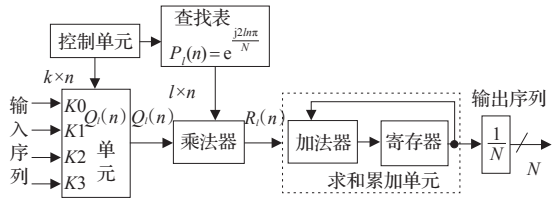

图2　优化Radix-4算法的IFFT运算框图

数倍），在函数$P_l(n)$、$Q_l(n)$的定义式(6)、式(7)中用$(n+\beta)$代替n，分别化简$P_l(n+\beta)$，$Q_l(n+\beta)$并推导$R_l(n+\beta)$可得：

$$P_l(n+\beta)=\exp[\frac{\mathrm{j}2l(n+\beta)\pi}{N}]=\exp(\frac{\mathrm{j}2\beta l\pi}{N})P_l(n) \quad (11)$$

$$Q_l(n+\beta)=\sum_{k=0}^{3}X(\frac{Nk}{4}+l)\exp[\frac{\mathrm{j}k(n+\beta)\pi}{2}]=Q_l(n) \quad (12)$$

$$R_l(n+\beta)=P_l(n+\beta)Q_l(n+\beta)=R_l(n)\exp(\frac{\mathrm{j}2\beta l\pi}{N}) \quad (13)$$

由式(13)可以看出，对于一个给定的l值，只需要计算$R_l(0)$、$R_l(1)$、$R_l(2)$、$R_l(3)$四个值，其余的$R_l(n)$均可由求出的四个值乘以相移因子$\exp(\mathrm{j}2\beta\pi/N)$获得，由图2的分析已知，计算一个$R_l(n)$需要$N/4$个时钟周期，所以产生4个$R_l(n)$并最终产生所有IFFT输出值共需要$4\times N/4=N$个时钟周期。

将式(13)带入式(10)可得：

$$x(n+\beta)=\frac{1}{N}\sum_{l=0}^{N/4-1}R_l(n+\beta)=\frac{1}{N}\sum_{l=0}^{N/4-1}R_l(n)\exp(\frac{\mathrm{j}2\beta l\pi}{N}) \quad (14)$$

由式(14)可以看出，只需要计算前面$N/16$个输出值$x(n)$，其余输出值可以利用前$N/16$个输出值乘以相应的因子获得。由于计算每个输出值需要一个复数乘法器并且产生$R_l(n)$本身需要一个复数乘法器，因此，完成N点IFFT输出值的计算需要$(N/16+1)$个乘法器。

进一步考虑复数乘法的一个特殊性质：对于在复平面上关于$y=x$对称的两个复数A与B，其中$A=x+\mathrm{j}y$，$B=y+\mathrm{j}x$，分别与第三个复数$Z=R+\mathrm{j}M$相乘，乘积内部包含的实数乘法均是xR，My，Ry，Mx四项，因此$A\times Z$与$B\times Z$只需要计算一组复数乘法，通过对内部四项实数乘法位置移动和加减符号调整即可获得另外一组。因此在式(14)中计算所有$R_l(n)\exp(\mathrm{j}2\beta\pi/N)$结果仅需要$N/32$个复数乘法器，最终所需要的乘法器数量为$(N/32+1)$个。

由以上分析可知：对于N点IFFT运算，在仅使用$(N/32+1)$个复数乘法器的条件下可以将运算时间从N^2缩减为N个时钟周期，实现了在最小化硬件资源开销的前提下最大化运算速度的目的，与通用的DSP处理器对比在硬件资源消耗与速度方面具有非常大的优势。

3　基于优化算法的64点IFFT/FFT处理器的设计

根据以上所分析的Radix-4 IFFT优化算法，当$N=64$时，式(11)可以写为：

$$x(n+\beta)=\frac{1}{64}\sum_{l=0}^{15}R_l(n)\exp(\frac{\mathrm{j}\beta l\pi}{32})$$

其中β为4的整数倍且$0<\beta\leq 60$。

由于$\exp(\mathrm{j}\beta l\pi/32)$的周期性，只需$\exp(\mathrm{j}4\pi/32)$、$\exp(\mathrm{j}8\pi/32)$、$\exp(\mathrm{j}12\pi/32)$三个复数因子与$R_l(n)$相乘，其余乘积可由3个乘积乘以{+1，－1，+j，－j}获得。

图3描述了一种基于优化算法的64点IFFT处理器实现框图。该处理器可以在16个时钟周期内实现IFFT的16个输出，

循环4次则可以在64个时钟周期内实现IFFT的64个输出。硬件结构主要包括$Q_l(n)$单元、$R_l(n)$计算单元、变换单元、求和运算单元、时序控制单元以及查找表1、查找表2。

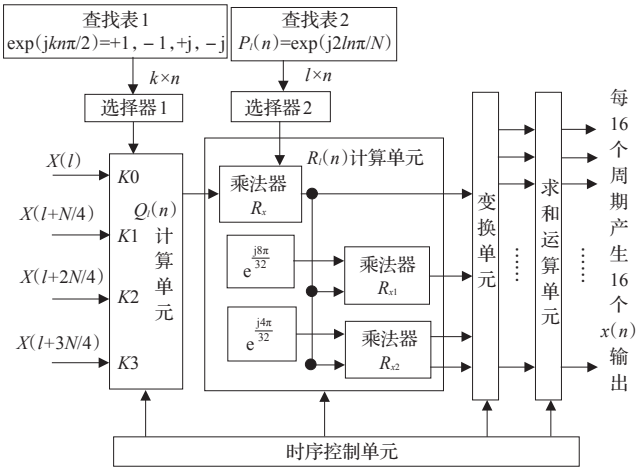

图3　基于优化算法的64点IFFT处理器实现框图

3.1　$Q_l(n)$计算单元

每个时钟周期$Q_l(n)$单元读入四个复数值，分别为$X(l)$，$X(l+N/4)$，$X(l+2N/4)$，$X(l+3N/4)$，选择器1根据输入值中$k\times n$（k=0,1,2,3）的不同从查找表1中选择相应的乘数因子{+1，−1，+j，−j}与输入值相乘并将结果相加产生输出$Q_l(n)$，下一时钟周期在输出$Q_l(n)$的同时流水读入下一组4个复数值。该过程将循环16次产生16个$Q_l(n)$（$0\le l\le 15$）。

3.2　$R_l(n)$计算单元

将$Q_l(n)$读入复数乘法器R_x，选择器2根据$l\times n$值的不同从查找表2中选择8个预定义的$P_l(n)$因子exp(jπ/32)~exp(j8π/32)中的一个完成$R_l(n)$的计算。

复数乘法器R_{x1}完成$R_l(n)\times$exp(j8π/32)的复数乘法运算，复数乘法器R_{x2}完成$R_l(n)\times$exp(j4π/32)，$R_l(n)\times$exp(j12π/32)的复数乘法运算，由于复常数exp(j4π/32)的实部与虚部对应exp(j12π/32)的虚部与实部，因此通过调整结果可节省一次乘法运算，减少硬件开销。

3.3　变换单元

变换单元利用$R_l(n)$计算单元产生的4个乘积，根据$\beta\times l$值的不同从{+1，−1，+j，−j}中选择一个适当的因子计算$R_l(n)\times$exp(jβlπ/32)的16个分立结果中剩余的12个并将产生的16个分立值输出到求和运算单元，该单元运算流程如图4所示。

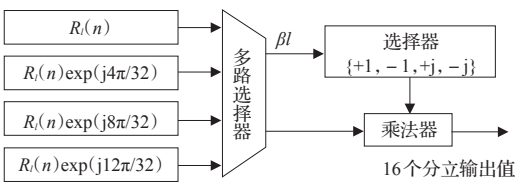

图4　变换单元运算流程图

3.4　求和运算单元

每个时钟周期将变换单元产生的16个分立输出值并行读入16组累加单元，分别进行累加运算并将结果存入寄存器中，每个周期判断l值是否为15，如果l值不等于15则加1并根据新的l值读入下一组16个分立值，如果l等于15，则16个时钟周期循环完毕，将各个寄存器中的值除以N，则在输出端口获得16个$x(n)$输出。图5为求和运算单元计算16个$x(n)$输出的运算流程图。

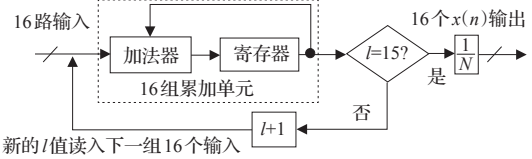

图5　求和运算单元运算流程图

根据以上分析，利用图3所示的64点IFFT实现图，每16个时钟周期将产生16个$x(n)$输出，当n=0时，第一次将产生$x(0)$，$x(4)$，$x(8)$，$x(12)$，$x(16)$，$x(20)$，$x(24)$，$x(28)$，$x(32)$，$x(36)$，$x(40)$，$x(44)$，$x(48)$，$x(52)$，$x(56)$，$x(60)$共16个输出样值，其下标间隔为4。重复以上过程4次（n分别取0，1，2，3），最终将在64个时钟周期产生64个$x(n)$输出。

4　64点FFT处理器的FPGA实现

4.1　设计验证

整个设计采用Top-Down的设计方法，利用Verilog HDL语言实现，便于移植和裁剪。设计中使用Pipeline流水线工作方式，内置端口RAM、ROM单元加快系统总体速度。通过Modelsim进行功能仿真，利用Xilinx公司的ISE10.1对设计进行综合，根据综合报告对设计进行了优化，降低关键路径的时延，系统最高工作频率可达100 MHz，利用Spartan-2E系列的XC2S300E器件完成FPGA验证。FFT处理器资源利用情况如下所示：Slice占用了36%，Flip Flop占用了25%，LUT占用了21%。用Modelsim进行功能仿真的局部时序如图6所示。

4.2　精度分析

为了验证所设计FFT模块的运算精确性，将64点FFT模块的FPGA仿真结果与MATLAB中对输入待测数据进行FFT精确运算的结果进行对比，实部及虚部的相对误差如图7所示。其中输入的一帧待测数据矢量为：[1+1i,2+2i,…,64+64i]。

图7中，仿真结果与MATLAB计算的高精度结果误差不超过0.5%，验证了所设计模块的高精度特点。

4.3　性能比较

表1为本文所设计FFT处理器与文献[5]中所提到FFT处理器性能比较，由对比可以看出，本文所设计FFT处理器可以较大限度地减少乘法器的使用数目，同时资源利用率获得近50%的提高。

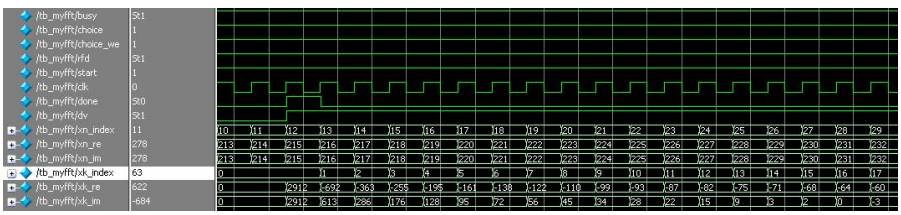

图6　64点FFT处理器功能仿真局部时序图

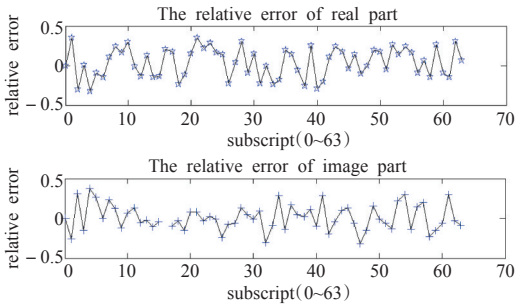

图7　FPGA仿真结果与MATLAB仿真结果相对误差图

表1　处理器性能比较表

FFT处理器	乘法器数目	64点情况	乘法器利用率/(%)	运算单元利用率/(%)	算法
R2MDC	$2(\log_4^N-1)$	4	50	50	Radix-2
R4MDC	$3(\log_4^N-1)$	6	25	25	Radix-4
R2SDF	$2(\log_4^N-1)$	4	50	50	Radix-2
R4SDF	$(\log_4^N-1)$	2	75	25	Radix-4
本文设计	$N/32+1$	3	75	100	优化Radix-4

4.4　设计应用

设计的FFT处理器，已经成功应用于某OFDM通信系统中，图8为该系统中通过发射机自环，得到的FFT模块解调后信号的星座图，由图8可以看到信号具有很好的汇聚度，便于后续信号的正确解调，其中中间零点处的信号点是OFDM信号中插入的保护零边带。

5　结论

根据OFDM系统中利用IFFT/FFT进行调制解调的原理，利用优化的Radix-4 FFT算法设计并实现了64点FFT处理器，该处理器仅仅采用3个复数乘法器并在64个时钟周期内获得64个处理结果，实现了处理器高速度、低成本的良好折中。当外围工作频率为100 MHz时，该处理器正常工作情况

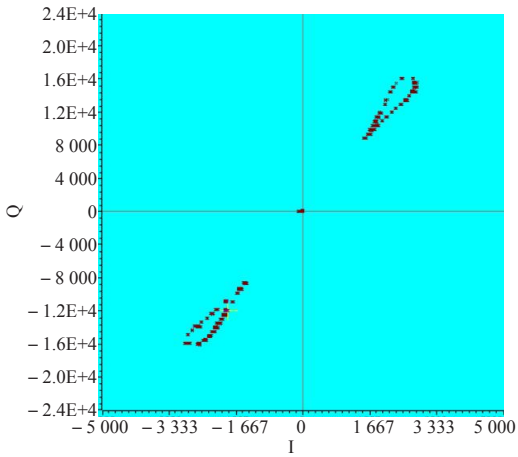

图8　FFT模块解调后信号星座图

下完成64点FFT运算的时间为0.64 μs，远小于IEEE 802.11a协议中64点FFT的时间参数t_{FFT}=3.2 μs，可实现数据的高速实时处理[6]。目前该处理器已成功应用于某OFDM通信系统中。

参考文献：

[1] 史治国，洪少华，陈抗生，等.基于XILINX FPGA的OFDM通信系统基带设计[M].杭州：浙江大学出版社，2009：1-5.

[2] 丁舒羽，陈健.OFDM的基本原理及FFT实现[J].山西电子技术，2004(5).

[3] 蒋青，吕翊.一种OFDM调制解调器的FPGA实现[J].信息技术，2006(4).

[4] 丁玉美，高西全.数字信号处理[M].2版.西安：西安电子科技大学出版社，2001：112-114.

[5] Swarlzlander E E，Young W K W，Joseph S J.A radix 4 delay commulator for fast Fourier transform processor implementation[J].IEEE J Solid-State Circuits，1984，SC-19(5)：702-709.

[6] 田继锋，江海宁，宋文涛，等.一种适用于OFDM系统的高效FFT处理器[J].系统工程与电子技术，2004(7).

(上接42页)

[6] Kennedy J，Eberhart R C.Particle swarm optimization[C]//Proceedings of IEEE International Conference on Neural Networks，Australia，1995：1942-1948.

[7] Bergh F，Engelbrecht A P.A cooperative approach to particle swarm optimization[J].IEEE Trans on Evolutionary Computation，2004，8(3)：225-239.

[8] Khalid N K，Basuk T.A model to optimize DNA sequences based on particle swarm optimization[C]//2nd Asia International Conference on Modeling & Simulation，2008：534-539.

[9] Wang W，Zheng X D，Zhang Q，et al.The optimization of DNA encodings based on GA/CA algorithms[J].Progress in Natural Science，2007，17(6)：739-744.

[10] Zhang K，Xu J，Geng X T，et al.Improved taboo search algorithm for designing DNA sequences[J].Progress in Natural Science，2008，18(5)：623-627.

[11] Soo-Yong S，Dong-Min K，In-Hee L，et al.Evolutionary sequence generation for reliable DNA computing[C]//Proc of Congress on Evolutionary Computation.[S.l.]：IEEE Computer Society Press，2002.

[12] Xu Chunxia，Zhang Qiang，Wang Bin.Research on the DNA sequence design based on GA/PSO algorithms[C]//Bioinformatics and Biomedical Engineering，2008：816-819.

[13] Reynolds R G.An introduction to cultural algorithms[C]//Proceedings of the 3rd Annual Conference on Evolutionary Programming，San Diego，California，1994.

[14] Feldkamp U，Raube H，Banzhaf W.Software tools for DNA sequence design[J].Genetic Programming and Evolvable Machines，2003，4(2)：153-171.

[15] Marathe A.On combinatorial DNA word design[J].DIMACS Series in Discrete Mathematics and Theoretical Computer Science，1999，44：75-88.

[16] 张凯，肖建华.基于汉明距离的DNA编码约束研究[J].计算机工程与应用，2008，44(14)：24-26.

[17] 马慧民，叶春明.背包问题的知识进化算法[J].计算机工程，2009，35(6)：208-211.

[18] 何洋林，叶春明.生产调度优化问题文化进化算法研究[J].计算机工程与应用，2007，43(36)：55-57.

[19] 许世明，张强.基于遗传粒子群算法的DNA编码优化[J].计算机工程，2008，34(1)：218-221.

[20] Shin S Y，Lee I H，Kim D.Multi-objective evolutionary optimization of DNA sequences for reliable DNA computing[J].IEEE Transactions on Evolutionary Computing，2005(2)：143-158.