doi: 10.3969/j.issn.1006-1576.2010.09.020

基于 FPGA 的盲图像复原算法

郝源源[1,2]，张耀[1,2]，祁小平[1]
（1. 中国科学院 光电技术研究所，四川 成都 610209；2. 中国科学院 研究生院，北京 100080）

摘要：为提高盲图像复原的处理速度，以 Xilinx 公司的 Virtex-4 LX80 FPGA 芯片为平台，使用 Verilog 硬件描述语言，在合理利用硬件资源的条件下，实现了 SeDDaRA 盲图像复原算法。该算法结构简单、易于实现，能提高处理速度，较好的解决资源、速度与功耗三者之间的矛盾，具有较好的可扩展性。

关键词：盲图像复原；二维FFT；对数运算；指数运算；FPGA
中图分类号：TN911.73; TP301.6　　**文献标识码**：A

Blind Image Restoration Algorithm Based on FPGA

Hao Yuanyuan[1,2], Zhang Yao[1,2], Qi Xiaoping[1]
(1. Institute of Optics & Electronics, Chinese Academy of Sciences, Chengdu 610209, China;
2. College of Graduate, Chinese Academy of Sciences, Beijing 100080, China)

Abstract: In order to speedup the process of blind image restoration, implement SeDDaRA image restoration algorithm using Verilog hardware description language on Xilinx Virtex-4 LX80 platform. Through optimizing algorithms and structures, with conditions of efficiently using of hardware resource, the utility of the parallel algorithm is exploited more efficiently in the design, and the trade-off between speed, area and power has been well handled. The algorithm is also easy to enlarge.

Keywords: blind image restoration; 2-D FFT; logarithm operation; exponent operation; FPGA

0　引言

图像复原[1-6]是依据图像退化模型，对观测图像进行处理来恢复实际景物的图像处理技术。在图像复原领域，当图像退化系统的点扩展函数已知时，对退化图像进行的复原通常称为经典图像复原。但在实际应用中，图像退化系统的点扩展函数往往是未知的，只能从退化图像中以某种方式提取退化模型的信息，找出图像复原的方法，称为盲图像复原，或图像盲反卷积。

由于盲图像复原是运算密集型计算过程，提高其处理速度一直是研究热点。由于采用专用集成电路来执行图像复原算法是提高处理速度的一种有效手段，故对图像复原处理算法和与其相适应的电路结构进行研究。

1　SeDDaRA 算法原理

对于观测所得的降质图像，通常可表示为：

$$g(x,y)=d(x,y)\otimes o(x,y)+n(x,y) \tag{1}$$

式中，$\otimes$ 为卷积算子，$g(x,y)$ 为观测所得的降质图像，$d(x,y)$ 为降质过程，$o(x,y)$ 为未受降质过程影响的目标原始图像，$n(x,y)$ 为系统噪声。而盲图像复原即为从观测所得降质图像 $g(x,y)$ 获取原始图像 $o(x,y)$ 和降质过程 $d(x,y)$ 的最佳估计。显然，从式 (1) 可知，盲图像复原作为病态的逆问题求解，由于其信息的匮乏和待估计量很多而非常困难，为此Caron[4]等提出了SeDDaRA（Self-Deconvolution Data Reconstruction Algorithm）盲图像复原算法，该算法利用指数定律（Power Law）直接从所观测降质图像的功率谱对降质过程进行估计，并以此为基础运用经典的图像复原算法实现原始图像的估计。其原理为：首先，从降质图像的功率谱进行降质过程的估计，故对式 (1) 进行傅立叶（Fourier）变换有其频域的表示为：

$$G(u,v)=D(u,v)O(u,v)+N(u,v) \tag{2}$$

在频域，SeDDaRA 算法利用指数定律可提取降质过程 $D(u,v)$ 为：

$$D(u,v)=KS\left\{\left[S_G(u,v)-S_N(u,v)\right]^{\alpha/2}\right\} \tag{3}$$

式中，$S(\cdot)$ 为平滑算子，α 为幂指数采用试错的方法进行选取，K 为常系数由 $\max[D(u,v)]=1$ 确定，式中 $S_F(u,v)=\dfrac{|F(u,v)|^2}{N^2}$。

估计的 $D(u,v)$，SeDDaRA 算法采用经典的维纳

收稿日期：2010-03-04；修回日期：2010-04-01
基金项目：中科院知识创新工程重大项目基金（YYYJ-0815）
作者简介：郝源源（1984-），男，河南人，硕士研究生，从事实时图像处理技术研究。

（Wiener）滤波图像复原算法的目标原始图像估计的频谱为：

$$O(u,v)=\frac{D^c(u,v)G(u,v)}{|D(u,v)|^2+k} \tag{4}$$

式中，$D^c(u,v)$ 表示 $D(u,v)$ 的共轭，k 为规整化因子，可通过试错的方法进行选取，对式 (4) 进行反傅立叶变换，即可得目标原始图像的估计。

显然，SeDDaRA算法通过利用指数定律直接提取降质过程的估计，而非迭代的实现了降质图像的盲复原，具有好的实时性。

根据式 (1)～式 (4) 可知，SeDDaRA 算法的 FPGA 实现主要包括二维 FFT、幂运算、平滑操作、维纳滤波和二维 IFFT 等模块。其实现框图如图 1。

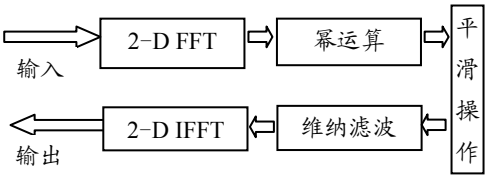

图 1 SeDDaRA 算法的实现框图

其中，平滑操作可采用 3*3 的中值滤波实现。维纳滤波只需进行取共轭等简单操作。式 (4) 中的二维傅里叶反变换可通过对正变换的输入输出数据分别取共轭来实现，不需专门设计 IFFT 模块。

2 二维 FFT 处理器的设计与实现

由于二维傅里叶变换核的可分离性，二维 FFT 是可分解的。对于 N*N 的二维 FFT，可先计算 N 行 N 点的行变换，然后对行变换的结果进行扫描方式转置，最后进行 N 行 N 点的列变换。行列分解法算法结构简单，数据流也很规则，非常适用硬件实现。其实现框图如图 2。

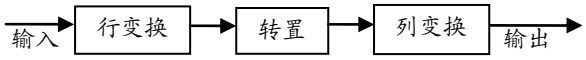

图 2 2-D FFT 实现框图

其中，转置模块可用 DPRAM 实现。即先将二维数据按逐行扫描的方式写入 DPRAM，然后按照逐列扫描的方式读出。

2.1 一维 FFT 架构设计

采用图 3 的架构完成一维 FFT[7-10]运算。

为了实现数据的流水输入输出，该设计的输入输出模块采用乒乓结构来实现。对输入部分，其中一组存储器用于寄存片外的输入数据，另一组向片内蝶形运算器输入数据；对于输出部分，其中一组存储器用于寄存片内蝶形运算器输出的数据，另一组向片外输出数据。基-2 蝶形单元分别完成 FFT 对应级上的蝶形运算。设计了 128 点 FFT，需要 7 级蝶形运算。为了节省资源，对 2 个蝶形运算器之间的存储器，没有使用乒乓 RAM 而采用双端口的 RAM。因为蝶形运算为同址运算，蝶形运算器可以在前一个时钟周期接收上一级的双端口 RAM 输入数据，后一个时钟周期向下一级 RAM 输出数据。旋转因子实现向蝶形运算器输入所需的因子值，该因子可事先计算出来，并存储在 ROM 中。地址产生单元，负责完成对所有存储单元的读写控制。

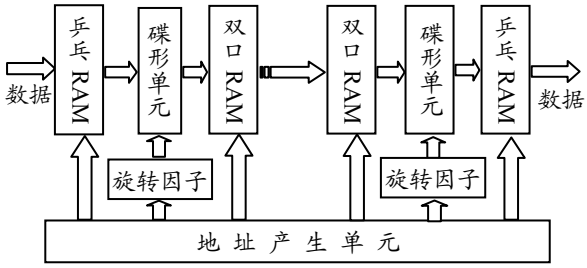

图 3 一维 FFT 架构

2.2 基-2 碟形单元设计

基-2 蝶形运算决定了 FFT 的速度，是 FFT 实现的关键部分。根据 FFT 的迭代运算公式，可看出每个蝶形运算单元计算 2 个值，其运算公式如式 (5)：

$$\begin{aligned} X_{m+1}(p)&=X_m(p)+W_N^P X_m(q)\\ X_{m+1}(q)&=X_m(p)-W_N^P X_m(q) \end{aligned} \tag{5}$$

显然，蝶形运算中最占时间的是乘以旋转因子时的复数乘法，下面介绍高效复数乘法器的设计思想。假设其中一条复数乘为：

$$R+jI=(X_1+jY_1)*(X_2+jY_2) \tag{6}$$

则：

$$\begin{aligned} R&=X_1*X_2-Y_1*Y_2\\ &=(X_2-Y_2)*Y_1+X_2*(X_1-Y_1) \end{aligned} \tag{7}$$

$$\begin{aligned} I&=X_1*Y_2-Y_1*X_2\\ &=(X_2+Y_2)*X_1-X_2*(X_1-Y_1) \end{aligned} \tag{8}$$

采用表达式共享的方式可以把原来需要 4 次乘法才能完成的复数乘降为 3 次，而且实部和虚部可以并行实现；既节约了乘法器资源，又提高了速度。

2.3 地址产生单元与低功耗设计

地址产生单元负责产生输入输出数据的地址和旋转因子的地址，主要由计数器和相应控制逻辑组成。由于涉及大量的 RAM/ROM 访问，所以该部分显得尤为重要。

地址产生单元包含大量的地址总线信号，而且其所控制的存储单元也涉及大量数据总线信号。由

于总线会带来大负载、长连线、大电阻等效应，所以，总线的功耗要占整个设计总功耗的15%～20%。为降低该部分的功耗，可利用一些特定的编码来减少信号的开关活动率[10]，以降低由数据传输而造成的功耗。

由于地址总线相对于数据总线更具规律性，故可根据地址数据源的规律进行编码。格雷码（Gray）在地址总线低功耗编码中具有广泛的应用。如果前后数据连续时，格雷码相邻码字之间只有一位数据跳变，有效地降低了地址总线上的功耗。

数据总线上的数据随机性比较强，使得数据总线低功耗编码的发展受到了很大的限制，但也出现了一些经典的通用编码方法，如 BI[11]（Bus Invert）编码。该编码方法是在总线上附加一条额外的标志信号线 INV，用来标志总线前后两段连续时间内的信号翻转情况。这种编码方法通过判断 t−1 时刻总线上编码的数据与 t 时刻总线上需要传输的数据之间的 Hamming 距离来实现编码过程。当 Hamming 距离（H）大于 $N/2$（N 为总线位宽）时，对总线上的数据取反并将 INV 置 1；否则情况相反。BI 码可表示为：

$$(B^{(t)},INV^{(t)})=\begin{cases}(b^{(t)},0) & H^{(t)}\le N/2\\ (\bar{b}^{(t)},0) & H^{(t)}>N/2\end{cases} \tag{9}$$

式中，$H^{(t)}=(B^{(t-1)}\,|\,INV^{(t-1)},b^{(t)}\,|\,0)$。

在接收端，当 INV 为“1”时，将接受数据取反；当 INV 为“0”时，数据不作任何处理。

格雷码、BI 码的编译码电路只需简单的移位、异或、与和取反运算即可实现，二者增加的硬件开销微不足道，但却降低了设计的总功耗。

2.4　块浮点防溢出单元

为防止数据溢出，从设计复杂性、运算精度与运算速度等方面折中考虑，采用了 Fialho 等提出的块浮点防溢出[12]算法。其主要思路是在原有 16 数据表示的基础上增加 2 位标识位，以保证运算过程中不发生溢出，这样，运算过程中的数据用 18 位来表示。由有限状态机构成的溢出检测单元对蝶形运算输出数据的高三位进行溢出检测，蝶形运算结果中绝对值最大的数值的高三位决定了该级蝶形运算完成后有限状态机的状态，由此可确定下一级运算如何从 18 位数据中选择 16 位数据进入蝶形单元，并用移位累加单元实现对每一级移位次数的累加，以便给出 FFT 运算结束后总的移位次数，由此可确定最后结果的指数位。蝶形输出数据的高三位与“溢出检测”的状态之间关系分为 3 种情况：

1) 111（或000），表示无溢出，状态S1；

2) 110（或001），表示1比特溢出，状态S2；

3) 10x（或01x），表示2比特溢出，状态S3。

状态转换图如图 4。

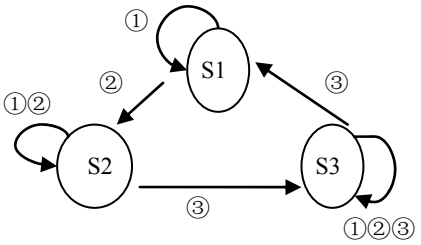

图 4　防溢出状态机

实验表明这种算法实现简单，适合硬件实现，可以最大限度地降低截断误差的影响。

3　幂运算的实现

式 (3) 中的幂运算可以一般化的表示为：

$$y=x^{a} \tag{10}$$

式中，x 为自变量，a 为常参数。一般而言，该运算可以通过泰勒公式来实现，即：

$$y=x^{a}=\sum_{n}a_{n}x^{n} \tag{11}$$

该方法可将幂运算转化为基本的乘加操作，以便通过 FPGA 内部的 DSP 核来实现。但当调整参数 a 时，系数 a_n 发生了变化，其所对应的硬件电路也发生了相应的改变，其结果为后续的调试带来了不必要的麻烦，特别是在进行外场实验时（需根据实验场景的变化，调整参数 a）该方法可操作性严重下降。在实现时将的幂运算转化为自然对数与指数运算，即：

$$y=x^{a}=e^{a\ln x} \tag{12}$$

式中，当参数 a 变化时，只需调整指数运算的输入端口，不需调整硬件电路，大幅度降低了调试复杂度，使其应用变得可行。

3.1　对数运算的实现

对数运算的算法研究和硬件实现都比较复杂，常见的方法有查找表法、流水线查找表法、多项式拟合[13]等。查找表法是将一个函数所有可能参数的运算结果保存为一个表，其大小随数据表示的位数成指数级变化。使用查找表是直接的方法，适用于任何函数的计算，对于 $y=F(x)$，设其中 x 和 y 长

度都是 n bits，则需要一个 $2^n \times n$ bits的ROM。该方法适用于 n 比较小的时候，当 n 较大时其所需的存储空间是难以估量的。笔者结合自然对数运算的特点以及算法对运算精度及器件资源的要求，采用切比雪夫多项式逼近来实现对数运算。

3.1.1 对数的切比雪夫逼近

切比雪夫多项式是以递推方式定义的一系列正交多项式序列，在逼近理论中有重要的应用。因为切比雪夫多项式的根（被称为切比雪夫节点）可以用于多项式插值，相应的插值多项式能最大限度的降低龙格现象，并且提供多项式在连续函数的最佳一致逼近。对数函数的切比雪夫多项式逼近为：

$$\begin{aligned}\ln x = {} & 0.160\ 149x^7 - 0.980\ 021x^6 + 2.622\ 149x^5 - \\ & 4.028\ 442x^4 + 3.927\ 299x^3 - 2.562\ 525x^2 + \\ & 1.227\ 934x - 0.367\ 009\end{aligned} \tag{13}$$

直接计算式 (13) 需要的乘法和加法次数过多，这样会加大实现的复杂度。可利用霍纳[10]（Horner）方法来求解多项式的值。即将 N 阶的多项式改写为如下嵌套乘法的形式：

$$\begin{aligned}\ln x = {} & ((((((0.160\,149x - 0.980\,021)x + 2.622\,149)x - \\ & 4.028\ 442)x + 3.927\ 299)x - 2.562\ 525)x + \\ & 1.227\ 934)x - 0.367\ 009 \qquad (0.5 \le x \le 1)\end{aligned} \tag{14}$$

与直接计算式 (13) 比较，利用霍纳（Horner）方法求解的多项式可以明显的减少乘法和加法的次数，从而简化了计算。

3.1.2 数据范围压缩与恢复

由于逼近多项式中 $x \in [0.5,1]$，要计算这个区间外的 X 的对数值，就必须采用某种方法对 X 的值进行变换，使变换结果落入[0.5,1]区间内，在完成计算后再次对结果进行调整，恢复出原来输入值的真实值。这里，将这种数据变换过程称为范围压缩与恢复。

对于输入，X 有关系式如下：

$$\ln X = \ln(2^k x) = \ln x + k \ln 2 \tag{15}$$

按这一关系，可将区间[0.5,1]外的 X 变换到区间内进行计算，再将多项式计算结果根据这个关系式恢复出原始输入数据的函数值。可见，这一过程的关键是确定 k 的值。对输入 X 的处理是将其除以2，直至结果落入区间[0.5,1]内，而除以 2 的运算可以通过移位操作完成。计算 $\ln x$ 完成后只需将此结果与 $k \ln 2$ 相加即可恢复出 $\ln X$ 的值。

3.2 指数运算单元的实现

利用关系式 (16) 完成指数函数的计算：

$$e^z = \sin hz + \cos hz \tag{16}$$

而双曲函数的求值采用Cordic[14]（Coordinated Rotation Digital Computer）算法来实现。在实现Cordic算法时，使用Xilinx公式提供的IP核。用FPGA厂家提供的IP核来完成设计，不仅可获得优异的性能，且可减少用户的开发时间。

而在实现Cordic算法时，由于算法的限制，输入的值有范围的限制。Xilinx的Cordic核将输入变量的值限制在 $-\pi/4 \sim \pi/4$ 之间。因此，要计算在这个区间之外的函数值，就必须采用某种方法对x的值进行变换，使结果落入 $-\pi/4 \sim \pi/4$ 区间内，在完成计算后再次对结果进行调整，恢复出原来输入值的真正结果。对于 e^x，存在如下关系：

$$e^X = 2^k e^x = e^{k\ln 2} e^x = e^{x + k\ln 2} \tag{17}$$

因此，将输入值X减去 $k\ln 2$，k=$-n$,…-1,0,1…n，使得 $-\pi/4 < x = X - k\ln 2 < \pi/4$，再利用Cordic算法求的 $\sin hx$ 和 $\cos hx$，然后由式 (16) 得到 e^x 的值。

要从 e^x 恢复 e^X 的值，可再利用的关系式 (17) 来完成。该运算是一个与 2 的幂相乘的运算，可通过对数据进行移位操作来代替乘法运算，减少实现的复杂度。在前面的范围压缩过程中已经确定的 k 值决定了移位的位数。

4 中值滤波的实现

中值滤波是一种非线性空间滤波器，其响应由模板区域中像素的中位值决定，其实现公式为：

$$g(x,y) = \mathrm{med}\{f(x-i, y-j)\}, (i,j) \in S \tag{18}$$

其中，$g(x,y)$、$f(x,y)$ 为图像像素值，S 为模板窗口。笔者选用的模板窗口尺寸为 3*3。

由式 (18) 可以看出，3*3 中值滤波的输出只与输入值相应像素的相邻 2 行、2 列有关。即只需通过缓存 3 行 3 列的数据[15]，再进行 9 个像素的排序便可算出相应的中值。其实现过程如下：先构建一个 3 点的比较器，为后续的工作提供基础。该比较器对输入的 3 个像素进行排序后按灰度高低输出。

中值滤波排序算法的原理框图如图 5，输入图像的 3 行 3 列共 9 个数据经过并行的三级的三点排序以后，得到相应的中值。具体步骤如下：

图 5 中，输入值为原始图像中以 $x(i,j)$ 为中心的 3 行 3 列数据。A1、A2、A3、B1、B2、B3 及 C1 为构建 3 点比较器，至上而下输出的是最大值、

中值、最小值。

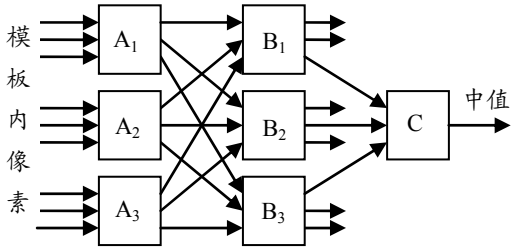

图 5　中值滤波实现框图

1) 通过 3 个比较器 A1、A2、A3，分别对各自的 3 个输入进行排序，将得到的最大值送至比较器 B1，中值送至比较器 B2，最小值送至比较器 B3；

2) 比较器 B1、B2、B3 运算规则与第 I 级的 A1、A2、A3 相同。此时，中值在 B1、B2、B3 输出的三值之中；

3) 通过比较器 C 求出第 II 级的 3 个输出值的中间值，此即为输入 9 个值的中值。

为提高速度和时序性能，可考虑在各级中间加一些寄存器，以流水线的方式进行处理。该实现方法稍加修改即可实现最大值和最小值滤波，具有较好的可扩展性。

5　盲图像复原结果

笔者采用流水线、并行处理等结构，充分利用 FPGA 内部 BlockRAM、DSP 核等资源，实现了 SeDDaRA 盲复原算法。在完成综合、布局布线和时序仿真后，将位图文件下载到 Xilinx 公司的 Virtex-4 LX80 器件上，占用 Slice 约 19 500（25%）片、BlockRAM 约 65（47%）块及 DSP48 核 50（63%）个，较好地处理了速度与资源间的矛盾。最后，在包含相机、FPGA 处理板及监视器的平台上进行了实验，针对 128*128*16 bit 图像，可达到 400 fps 的处理速度，满足工程应用需求。其处理结果如图 6。

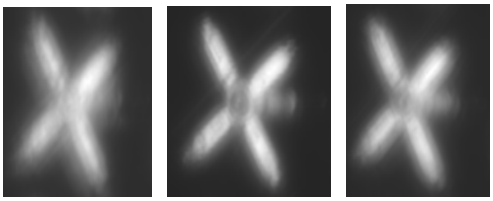

图 6　盲图像复原结果

其中，左侧为原始退化图像，中间为 Matlab 复原结果，右侧为处理结果。通过对比发现，FPGA 处理结果和 Matlab 相当，达到了实时复原的目的。

6　结束语

该算法通过数据范围压缩与恢复实现了大动态范围的对数、指数运算，具有占用资源少，处理精度高的特点；通过乒乓结构、流水线与并行处理、总线编码、块浮点等方法实现了一种高精度、低功耗的二维 FFT 模块，该方法结构简单、易于实现，具有较好的可扩展性。

参考文献：

[1] J. Zhang, Q. Zhang, Noniterative blind image restorationbased on estimation of a significantclass of point spre-ad functions [J]. Opt.Eng. 46(7), 077005 2007.

[2] J. Zhang, Q. Zhang, G. He, Blind deconvolut-ion: mult-iplicative iterative algorithm [J]. Opt.Lett. 2008, 33(1): 25−27.

[3] R.Banhamand, A.K.Katsaggelos. Digital Image Restoration[J]. IEEE Signal Proeessing Magazine, 1997, 14(3): 24−41.

[4] J. N. Caron. et al. ,Blind data restoration with an extracted filter function[J]. Optics Letters, 2001, 26(3): 27−35.

[5] Charles L.Maston, et al. Fast and optimal multiframe blind deconvolution algorithmfor high-resolution ground-based imaging of space objects[J]. Optical Society of America, 2009, 48(1): 129−138.

[6] James N. Caron, Nader M. Namazi, et al. Noniterative blind data restoration by use of an extracted filter fun-ction[J]. Applied Optics, 2002, 41(32): 277−285.

[7] J.W.Cooley and J.W.Tukey, An algorithm for machine calculation of complex Fourier series[J]. Math. Compute, 1966, 20(6): 429−430.

[8] B. M. Baas. An Approach to Low-Power, High Performance Fast Fourier Transform Processor Design, NGT-70340-1999-1[R]. California: Stanford University Technical Report, 1999.

[9] Tom Chen. Glen Sunada. Jian Jin.: A 100-MOPS Si-gle-chip Programmable and Expandable FFT[J]. IEEE, 2004, 15(3): 326−329.

[10] Keshab K.Parhi. VLSI Digital Signal Processing Syst-em Design and Implementation[M]. Minnesota: John Wiley & Sons, INC 1999.

[11] Stan M R, Burleson W P. Bus Invert Coding for Low Power I/O[J]. IEEE Transactions on VLSI Systems, 1995, 3(1): 49−58.

[12] Fialho. I. J. Georgiou, T.T., On stability and performance of sampled-data systems subject to wordlength constraint[J]. IEEE Trans Automatic Control, 1994, 39(12): 2476−2481.

[13] 李庆杨, 等. 数值分析[M]. 北京: 清华大学出版社, 2001.

[14] J. Volder, The CORDIC Computing Technique[J]. IRE Trans. Computers, 1959, 5(2): 330−334.

[15] Maya Gokhals et al. Reconfigurable Computing Accelerating Computation with Field-Programmable Gate Arrays[M]. Netherlands: Springer, 2007.