计算机科学
COMPUTER SCIENCE

Shor 整数分解算法的线路优化

刘建美，王洪，马智

引用本文

刘建美，王洪，马智. Shor 整数分解算法的线路优化[J]. 计算机科学, 2022, 49(6A): 649-653.

LIU Jian-mei, WANG Hong, MA Zhi. Optimization for Shor's Integer Factorization Algorithm Circuit[J]. Computer Science, 2022, 49(6A): 649-653.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210600149

Shor 整数分解算法的线路优化

刘建美　王　洪　马　智
数学工程与先进计算国家重点实验室　郑州 450001
河南省网络密码技术重点实验室　郑州 450001
(modumingdi@foxmail.com)

摘　要　借助加窗技术和模整数的陪集表示技术,在加法的近似编码表示基础上给出 Shor 算法量子线路的整体优化和资源估计,并对设计的量子线路进行了仿真实验。借助加窗技术和模整数的陪集表示技术可以有效减少 Toffoli 门的数目以及降低整个量子线路的深度,其中 Toffoli 门数目为 $0.18n^3+0.000465n^3\log n$,线路深度为 $0.3n^3+0.000465n^3\log n$。由于采用加窗的半经典傅里叶变换,使得空间资源代价为 $3n+O(\log n)$个量子比特。在增加少量近似误差(误差可以随着填充数目的增加呈指数减小)的前提下,实现了时间空间资源代价的折衷。

关键词:整数分解;量子算法;量子线路
中图法分类号　TP301

Optimization for Shor's Integer Factorization Algorithm Circuit

LIU Jian-mei, WANG Hong and MA Zhi
State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China
Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China

Abstract　With the help of techniques such as windowed arithmetic and the coset representation of modular integers, the overall optimization and resource estimation for the quantum circuit of Shor's algorithm has been shown. What's more, the simulation experiment of the designed quantum circuit has been carried out. The Toffoli gate and the depth of the overall circuit can be reduced by techniques such as windowed arithmetic and the coset representation of modular integers. The Toffoli count is $0.18n^3+0.000465n^3\log n$ and the measurement depth is $0.3n^3+0.000465n^3\log n$. Due to the windowed semiclassical Fourier transform, the space usage includes $3n+O(\log n)$ logical qubits. A tradeoff for resources consume between the time and the space has been made at the cost of adding some approximation errors.

Keywords　Integer factorization, Quantum algorithm, Quantum circuits

1　引言

1994 年,Shor 提出了用量子计算机进行整数分解和离散对数求解[1]。Shor 整数分解量子算法的基本介绍如下:设 N 为要分解的大整数,随机选择一个与 N 互质且小于 N 的自然数 a,求函数 $f(x)=a^x \bmod N$ 的周期 r。显然 $f(x)$所取的值属于正整数集合$\{1,2,\cdots,N-1\}$,而且是一个周期性的函数。求出其周期后,可通过计算 $gcd(a^{r/2}\pm1,N)$得出 N 的因子 p 和 q。该算法最关键之处是利用量子傅里叶变换求 $f(x)$的周期。只要求得 $f(x)$的周期,就可以对 N 进行分解。在 Shor 整数分解量子算法中,函数周期的信息是通过量子傅里叶变换(Quantum Fourier Transform,QFT)来提取的。首先,对两组存有量子比特的寄存器进行幺正变换得到纠缠状态:$\sum|x\rangle\otimes|f(x)\rangle$,其中 $f(x)=a^x \bmod N$。对寄存器 x 进行傅里叶变换:$2^{-n}\sum e^{2\pi ikx/2^{2n}}|k\rangle$。可以看出,QFT 就是将态前面的叠加系数变为原叠加系数的离散傅里叶变换。由于 $f(x)$的周期性,最后得出只有 k 取下列各值时系数(概率幅)明显不为 0:$k=\lceil 2^{2n}\cdot s/r\rceil$。其中括号表示取不小于括号中值的最小正整数,因此除了 $k=0$ 外,有 $2^{2n}/k=r/s(s=0,1,\cdots,r-1)$。对 k 寄存器进行测量,可以测到 k 的本征值。当测到这些值后,再通过上式计算出 $f(x)$的周期 r。得到周期后,按照经典算法,可以推导出整数 N 的因子。

Shor 提出的整数分解算法需要一个输入寄存器和输出寄存器。由于在该量子算法的经典后处理部分用到了连分数逼近,受此限制,算法中的输入寄存器至少耗费 $2n$ 个量子比特。此后,人们提出了各种各样的量子算法优化和线路实现来优化整数分解的量子资源开销[2-10]。

1995 年,Barenco 等[2]指出"所有单量子比特门和两比特异或门(即 CNOT 门)构成的门集合是通用的,可以用来表示作用在任意多的量子比特上的所有幺正操作",这对于用量子线路模型来描述量子计算有着重大意义。1996 年,Griffiths 等提出了只需 1 个量子比特就可实现量子傅里叶变换的半经

基金项目:国家自然科学基金(61972413,61701539,61901525);国家密码发展基金(mmjj20180107,mmjj20180212)
This work was supported by the National Natural Science Foundation of China(61972413,61701539,61901525) and National Cryptography Development Fund(mmjj20180107,mmjj20180212).
通信作者:王洪(redwang@meac-skl.cn)

典量子傅里叶变换方法[3]，可以将指数寄存器中的量子比特数目降到1个，但其并未证明利用半经典量子傅里叶变换能够实现量子傅里叶变换，Fu等从几率幅的角度证明了两种变换后得到的同一量子态的几率幅完全相同[4]。2006年，Zalka指出，可以用整数的陪集表示来执行加法操作，近似地达到模加的效果[5]。此前一般用几个(基本上是5个)不同形式的加法来实现模加，如文献[6]。这种近似表示方法可大大减少Toffoli门的数目以及降低整个线路的深度，但需要使原来的n量子比特寄存器再增加$O(\log n)$个量子比特。2016年，Häner等提出使用半经典傅里叶变换、未初始化辅助比特和增量门的方法结合递归分治的思想可以将模幂需要的量子比特数目降为$2n+2$，缺点是Toffoli门数目和线路深度会增加，尽管其规模仍分别为$O(n^3\log n)$和$O(n^3)$[7]。如何使得时间花费和空间花费达到很好的折衷是一个非常有研究意义的问题。

2017年，Ekerå等提出Shor算法来求解离散对数的一个变体[8]，与Shor算法的量子部分类似，主要区别如下：Ekerå-Håstad算法有两个指数寄存器来储存e_1和e_2的信息，其长度分别为$2m$个量子比特和m个量子比特，总共的指数长度为$n_e=3m=1.5n+O(1)$个量子比特，低于Shor算法中的$2n$个量子比特。这导致了需要执行的乘法数目显著减少。

Gidney[9]提出可将执行n比特加法所需的Toffoli门数目降为Cuccaro加法线路[10]的一半(即渐近规模为n)而深度仍为$2n$。

2019年，Gidney将窗口法与量子实现线路相融合，进行量子运算的加窗操作，通过该方法可减少模乘操作和模加操作中的Toffoli门数目以及降低线路深度。

本文在Ekerå-Håstad算法的基础上结合半经典傅里叶变换方法以及加窗、模整数的陪集表示技术来优化整数分解的求阶(即求周期)量子线路。结果表明：在增加一定近似误差的前提下，实现了时间空间资源代价的折衷。其中近似造成的误差可以随着填充比特数目的增加呈指数级减小，以至于小于目前其他因环境噪声引起的误差。

2 优化的模幂线路

采用基于格技术的经典后处理算法，可以将第一寄存器量子比特数目由原始的(受连分数逼近条件限制)$2n$变为$1.5n$。

实现n维量子傅里叶变换需要的量子比特数为n。在半经典量子傅里叶变换的线路实现中，存放指数比特的寄存器在完成测量之后，可用于存放半经典傅里叶变换下一步运算的输入值，即该量子寄存器可循环使用。因此，实现半经典量子傅里叶变换所需量子比特数为1。使用加窗方法的半经典傅里叶变换，可将第一寄存器需要的量子比特限制在n的对数规模(即$O(\log n)$)。

在Gidney加法器[9]和Babbush等提出的QROM读取器(量子只读存储器)[11]的基础上，本文使用模整数的陪集表示和加窗查表操作对模幂逻辑线路中需要的乘法、加法的Toffoli门数目和线路深度进行优化。

2.1 模整数的陪集表示

在量子计算机中，通常用计算基态$|b\rangle$来表示一个整数b，而模整数的陪集表示方法是用周期为N、偏移量为b的周期性叠加态来表示一个整数。其关键思想是：陪集态表示的叠加周期性使得非模加法器能够执行近似的模加运算，并且近似的偏差随着寄存器中填充比特的增加而指数倍地降低。Zalka指出，使原来的n量子比特寄存器再增加$O(\log n)$个量子比特就足够了。

定义1 模整数的陪集表示定义了一个近似的编码加法族

$$COM_{k,N,m}=(G,u_k,E,v_k,C,L,f)$$

其中，$m\in N$是一个填充(比特数目)参数，$N\in N^+$是一个模(模量)，$k\in Z/NZ$是一个偏移量。七元组中元素定义如下：

(1)$G=Z/NZ$

(2)$u_k(g)=(g+k)\bmod N$

(3)$E=Z/2^{m+\lceil\lg N\rceil}Z$

(4)$v_k(e)=(e+k)\bmod 2^{m+\lceil\lg N\rceil}$

(5)$C=Z/2^mZ$

(6)$f((g,c))=g+cN$。

反之，$f^{-1}(e)=(e\bmod N,\lfloor e/N\rfloor)$。

(7)$L=\{f^{-1}(l)\mid 2^mN\leqslant l<2^{m+\lceil\lg N\rceil}\}$

定义2 输入$g\in G$关于近似编码加法族$COM_{k,N,m}$的编码$Encodings_g(COM_{k,N,m})$为：

$$Encodings_g(COM_{k,N,m})=\{f((g,c))\mid c\in C\}$$

定义3 输入$g\in G$关于近似编码加法族$COM_{k,N,m}$的陪集偏移量$Deviated_g(COM_{k,N,m})$为使得$v(f((g,c)))$未落在$Encodings_{\mu(g)}(COM_{k,N,m})$中的那些陪集值$c\in C$的集合。

定义4 近似编码表示$COM_{k,N,m}$造成的偏差$Dev(COM_{k,N,m})$定义为：

$$\max_{g\in G}|Deviated_g(COM_{k,N,m})|/|C|$$

引理1[12] 由模整数的陪集表示定义的每个近似的编码加法$COM_{k,N,m}$的最大偏差为2^{-m}。

证明：假设k是偏移量，$k\in Z/NZ$；g是输入，$g\in G$；c是陪集值，$c\in C$；x和y为编码表示后的结果$(x,y)=f^{-1}(v_k(f((g,c))))$。陪集值$c$落在$Deviated_g(COM_{k,N,m})$中当且仅当$x\neq(g+k)\bmod N$或$y\notin C$。令$x=e_2\bmod N$和$y=\lfloor e_2/N\rfloor$是解码后的结果。当$c<2^m-1$时，有$e_1<N\cdot 2^m-N$。同时由于$k<N$，因此$e_2<e_1+N\leqslant N\cdot 2^m$。只有$e_2\geqslant N\cdot 2^m$时产生的解码陪集值落在$C$的外面，从而使得$y\in C$。此外，由于$e_2<N\cdot 2^m$，故$x\equiv g+k(\bmod N)$。因此，$c$的值不满足成为$Deviated_g(COM_{k,N,m})$中元素的两个充分性条件中的任何一个。在$C$中只有一个值$c$不满足$c\neq 2^m-1$，即$c$为$2^m-1$时。因此$Deviated_g(COM_{k,N,m})\subseteq\{2^m-1\}$，即$|Deviated_g(COM_{k,N,m})|\leqslant 1$。

从而，近似表示造成的偏差：

$$Dev(COM_{k,N,m})=\max_{g\in G}|Deviated_g(COM_{k,N,m})|/|C|\leqslant 1/|C|=2^{-m}$$

由于近似编码表示技术之间的偏移量满足次可加性[12]，因此以模整数的陪集表示的近似编码加法为基本组件构成的整个模幂线路的最大偏移量为$k\cdot 2^{-m}$，其中k为模幂线路中的加法数目。

2.2 加窗查表操作

模幂运算$a^x\bmod N$是Shor算法实现中最耗资源的。加窗操作是一种在经典计算中被广泛使用的用于减少操作数目的技术，它用“查表法”将多个操作合并在一起[13]。查表法是通过增加预计算量来提高模幂运算的实现速度，是以空间换

时间来提高模幂运算的实现速度。加窗操作节省的是查表操作中的预计算量而非查表次数，其大体思想是：对 x 的二进制表示式，用固定长度 t 按一定的规则[14]进行划分，即将 x 表示为如下形式：$x=\sum_{i=0}^{v-1}2^{it}u_i$，$0\leqslant u_i\leqslant 2^t-1$，$0\leqslant i\leqslant v-1$。这样，如果计算模幂之前先将 a，a^{2^t}，$a^{2^{2t}}$，…，$a^{2^{(v-1)t}}$ 求解并存储起来，则在进行模幂运算时就可以减少加法运算的次数，而且每次运算时 u_i 的所有比特信息均已输入。若要将窗口法与量子实现线路融合，那么每个 u_i 的所有比特信息需同时输入。

之前需要的 n_e 个受控乘法操作变为 n_e/c_{exp} 个非受控乘法操作，之前每个乘法需要的 $2n$ 个受控加法操作现在变为 $2n/c_{\text{mul}}$ 个非受控加法操作。这两个优化所需要的代价是：要查找一个大小为 $c_{\text{exp}}+c_{\text{mul}}$ 的表。如果用 Cuccaro 等提出的加法器，n 比特加法的 Toffoli 门数目和测量深度都为 $2n$；如果用 Gidney 等提出的加法器，n 比特加法的 Toffoli 门数目为 $n-1$，测量深度为 $2n-1$。用 Babbush 等提出的 QROM 使得每个查表需要的 Toffoli 门数目和测量深度都为 $2^{c_{\text{mul}}+c_{\text{exp}}}$，而撤销查表计算的操作是因为用了基于测量的撤销计算，其成本也可以忽略。因此，如果在 Cuccaro 加法器和 QROM 的基础上进行加窗优化运算，Toffoli 门数目的渐近首项变为：$\frac{2n_en}{c_{\text{mul}}c_{\text{exp}}}(2n+2^{c_{\text{mul}}}+c_{\text{exp}})$；如果令 $c_{\text{mul}}=c_{\text{exp}}=\frac{1}{2}\log_2 n$，则得 Toffoli 门数目的渐近首项为 $\frac{24n_en^2}{\log_2^2 n}$，当 n 很大时，该渐近首项比不使用加窗操作时 Toffoli 门数目的渐近首项小。注：本文中 $\log n=\log_2 n$。类似地，加法器可换成 Gidney 加法器，其资源估计见第 3 节。

本文借助模整数的陪集表示技术，在文献[9]和文献[11]的基础上对整个模幂线路所需的 Toffoli 门数目和线路深度都进行了优化。

3 线路资源估计

Ekerå-Håstad 算法中需要的模幂操作 $|e\rangle|1\rangle\mapsto|e\rangle|g^e \bmod N\rangle$（初始的输入寄存器状态为 $|e\rangle$，最终的输出寄存器状态为 $|g^e \bmod N\rangle$）可以被分解为 n_e 次乘法，每个乘法又可以被分解为 $2n$ 次（寄存器长度为 n 时）受控加法和 1 次受控交换操作（位于两个 n 比特长度寄存器之间的交换），因此总共需要 $2n_en$ 次受控加法。考虑到陪集表示使寄存器的长度增加了，这意味着实现 Ekerå-Håstad 算法时如果未使用加窗优化术，则需要 $2n_e(n+c_{\text{pad}})$ 次受控加法。使用加窗优化技术后，受控加法的数目为：

$$\begin{aligned}\frac{n_e}{c_{\text{exp}}}\cdot 2\cdot\frac{n+c_{\text{pad}}}{c_{\text{mul}}}&\approx\frac{n_e}{c_{\text{exp}}}\cdot 2\cdot\frac{n+(3\log n+10)}{c_{\text{mul}}}\\&=\frac{2nn_e}{c_{\text{exp}}c_{\text{mul}}}+O\left(\frac{n_e\log n}{c_{\text{exp}}c_{\text{mul}}}\right)\\&\approx 0.08n_en\end{aligned}$$

一个查表加法中的 Toffoli 门数目为：

$$(n+c_{\text{pad}})-1+2^{c_{\text{exp}}+c_{\text{mul}}}-1=n+c_{\text{pad}}+2^{c_{\text{exp}}+c_{\text{mul}}}-2$$

测量深度为：

$$2(n+c_{\text{pad}})-1+2^{c_{\text{exp}}+c_{\text{mul}}}-1=2n+2c_{\text{pad}}+2^{c_{\text{exp}}+c_{\text{mul}}}-2$$

因此总共的 Toffoli 门数和测量深度分别为：

$$0.08n_en\cdot(n+c_{\text{pad}}+2^{c_{\text{exp}}+c_{\text{mul}}}-2)\approx 0.12n_en^2+0.00031n_en^2\log n$$

$$0.08n_en\cdot(2n+2c_{\text{pad}}+2^{c_{\text{exp}}+c_{\text{mul}}}-2)\approx 0.2n_en^2+0.00031n_en^2\log n$$

所耗费的量子比特分布如表 1 所列。

表 1 本文模幂量子线路量子比特耗费分布

Table 1 Distribution of quantum bit consumption in modular exponentiation

比特分布位置	需要的比特数目
加窗半经典傅里叶变换的指数寄存器	$O(\log n)$
存储乘积的乘法寄存器	$n+O(\log n)$
存储加法结果的寄存器	$n+O(\log n)$
查表寄存器	$n+O(\log n)$

4 仿真实验

本文对 Gidney 的加法线路使用陪集表示的加法，经数据拟合发现，得到的加法线路构成的模幂具有最小的量子体积，可达到很好的时间空间折衷。

本节利用 Q# 量子程序开发工具，对第 3 节构造的量子模乘线路进行仿真并对构造线路所需的 Toffoli 门数量进行统计。

4.1 加法量子线路数值模拟

2019 年，Ekerå 等提出一种方法可在 8 h 内用两千万个带噪声的量子比特来分解 2048 比特的 RSA 整数[15]，该文用的加法器是 Cuccaro 提出的加法器[9]。本文用 Gidney 提出的加法器[9]结合模整数的陪集表示[12]和加窗量子查表技术[12-13]，得出了比文献[15]时空资源折衷更优的量子逻辑线路。其结论对比如表 2 所列。

表 2 模幂量子线路资源消耗量对比

Table 2 Comparison of resource consumption in modular exponentiation

	本文	Gidney
Qubits 数目	$3n+O(\log n)$	$3n+O(\log n)$
Toffoli 门数目	$0.18n^3+0.000465n^3\log n$	$0.3n^3+0.0005n^3\log n$
线路深度	$0.3n^3+0.000465n^3\log n$	$500n^2+n^2\log n$

从表 2 可以看出，本文的量子逻辑线路比文献[15]的量子逻辑线路在 Toffoli 门数目和线路深度上更有优势。

为了便于区分，图 1－图 4 中表格的每个量除以寄存器的长度 n 以及模幂操作中整个加法的数目 $k=\lambda n^2$[12]。在本文中，我们取 $\lambda=1.5\cdot 2=3$。Shor 整数分解算法原始版本需要执行的加法数目为 $2n\cdot 2n=4n^2$，Ekerå-Håstad 算法改进后需要执行的加法数目为 $1.5n\cdot 2n=3n^2$。应用文献[12]中的分析方法，得到的对比结果如图 1－图 4 所示。

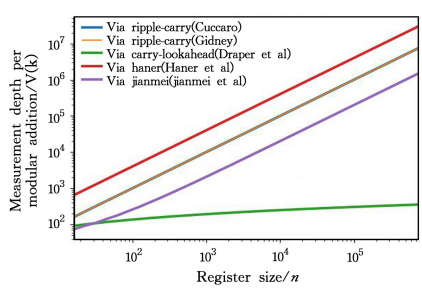

图 1 模幂量子线路深度对比

Fig. 1 Comparison of depth per modular addition in modular exponentiation

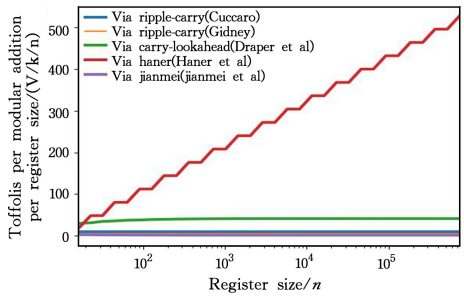

图 2 模幂量子线路 Toffoli 门消耗量对比

Fig. 2 Comparison of Toffoli per modular addition per register size in modular exponentiation

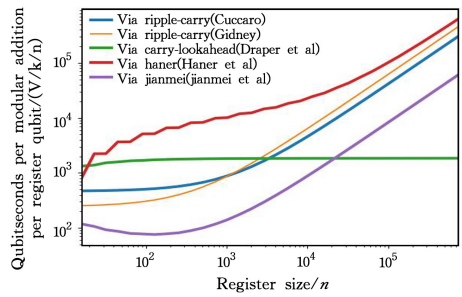

图 3 模幂量子线路体积对比

Fig. 3 Comparison of volume per modular addition per register size in modular exponentiation

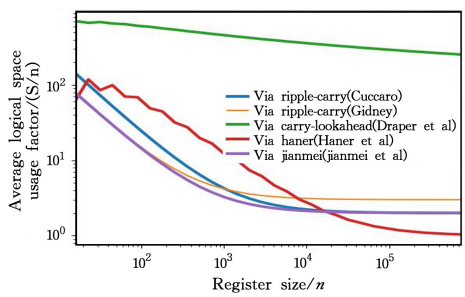

图 4 模幂量子线路逻辑比特消耗量对比

Fig. 4 Comparison of average logical qubits during modular addition per register size in modular exponentiation

从图中可以看出，本文的量子逻辑线路在时间空间资源花费折衷方面更有优势。采用模整数的陪集表示技术优化后的数值模拟结果显示，这种技术大大减少了量子门数目（以 Toffoli 门为指标）并大幅降低了线路深度，所付出的代价是增加了对数规模的填充比特、引入了一定的错误率，但错误率随着填充比特数目增加呈指数级降低。

4.2 整数分解模乘量子算法线路仿真结果

本节给出加窗的模乘与不加窗的模乘（Legacy）以及 Karatsuba 乘法[16]之间的量子资源（Toffoli 门数目、量子比特数目、线路深度）耗费对比具体的实例，如表 3 所列。其中 $n=\lceil \log N \rceil$，N 为要分解的整数。表 3 列出 n 为 8，15，16，30，60，64 时 3 种乘法逻辑线路的量子资源消耗情形。表 3 中的数据显示，使用加窗技术的模乘相比其他两种乘法量子线路，随着 n 的规模的增加，Toffoli 门数大大减小，线路深度大幅降低，只是增加了一定的量子比特数目。

表 3 模乘量子法线路资源消耗量对比

Table 3 Comparison of resource consumption in modular multiplication

n		Window	Legacy	Karatsuba
8	Toffoli 门数目	216.0	116.6	97.4
	量子比特数目	43.0	32.9	32.3
	线路深度	952.0	529.5	442.3
15	Toffoli 门数目	720.0	367.5	337.0
	量子比特数目	78.0	61.0	60.9
	线路深度	3119	1662	1524
16	Toffoli 门数目	480.0	342.0	387.8
	量子比特数目	84.0	63.3	65.5
	线路深度	1752.8	1546.1	1752.8
30	Toffoli 门数目	1530.0	1404.5	1229.0
	量子比特数目	154.0	121.5	120.3
	线路深度	6274.8	6335.5	5544.3
60	Toffoli 门数目	4180	6172	17346
	量子比特数目	305	241	634
	线路深度	16524.5	27807.5	54049.0
64	Toffoli 门数目	4064	6321	18414
	量子比特数目	326	257	642
	线路深度	14271.0	28476.5	56301.0

结束语 本文基于 Gidney 加法器，综合其他文献关于量子线路的设计工作，优化了 Shor 提出的整数分解问题量子算法的线路结构。结果显示：通过增加一定的近似误差（误差可以随着填充比特数目的增加呈指数级降低），实现了时间空间资源代价的折衷。其中近似造成的误差可以通过增加 n 的对数级规模的填充比特得到控制，以至于能低于目前其他因环境噪声引起的误差。

后续可针对椭圆曲线离散对数问题中的“点加”运算构造基于未初始化辅助比特、加窗技术以及模整数的陪集表示方法的量子线路，设计求解椭圆曲线离散对数问题的量子算法线路，并对线路需求的量子资源进行估测和分析。

参 考 文 献

[1] SHOR P. Algorithms for quantum computation: discrete logarithms and factoring[C]// Proceeding from the 35th Annual Symposium on Foundations of Computer Science. Santa Fe, NM, IEEE Computer Society Press, 1994: 124-134.

[2] BARENCO A, BENNETT C H, CLEVE R, et al. Elementary gates for quantum computation[J]. Physical Review A, 1995, 52(5): 3457.

[3] GRIFFITHS R B, NIU C S. Semiclassical Fourier transform for quantum computation [J]. Physical Review Letters, 1996, 76(17): 3228.

[4] FU X Q, BAO W S. Research on the quantum algorithm with high probability[D]. Zhengzhou: PLA Information Engineering University, 2011.

[5] ZALKA C. Shor's algorithm with fewer (pure) qubits [J]. arXiv: quantph/0601097, 2006.

[6] BEAUREGARD S. Circuit for Shor's algorithm using $2n+3$ qubits[J]. Quantum Information & Computation, 2003(2): 175-185.

[7] HÄNER T, ROETTELER M, SVORE K M. Factoring using $2n+2$ qubits with Toffoli based modular multiplication[J]. arXiv: 1611.07995, 2016.

[8] EKERÅ M, HÅSTAD J. Quantum algorithms for computing short discrete logarithms and factoring RSA integers[C]// Post

Quantum Cryptography(PQCrypto 2017). LNCS, Cham: Springer, 2017: 347-363.
[9] GIDNEY C. Halving the cost of quantum addition[J]. arXiv: 1709.06648, 2018.
[10] CUCCARO S A, DRAPER T G, KUTIN S A, et al. A new quantum ripple-carry addition circuit[J]. arXiv: ph/0410184, 2004.
[11] BABBUSH R, GIDNEY C, BERRY D W, et al. Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity [J]. arXiv: 1805.03662, 2018.
[12] GIDNEY C. Approximate encoded permutations and piecewise quantum adders[J]. arXiv: 1905.08488, 2019.
[13] GIDNEY C. Windowed quantum arithmetic [J]. arXiv: 1905.07682, 2019.
[14] KENJI K, YUKIO T. Speeding up elliptic crypto-systems using a signed binary window method[C]// Proceeding of the 12th Annual International Cryptology Conference on Advances in Cryptology. Berlin: Springer-Verlag, 1993: 345-357.
[15] GIDNEY C, EKERÅ M. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits[J]. arXiv: ph/1905.09749, 2019.
[16] GIDNEY C. Asymptotically efficient quantum karatsuba multiplication[J]. arXiv: 1904.07356, 2019.

LIU Jian-mei, born in 1988, doctoral candidate. His main research interests include quantum computation and quantum information.

WANG Hong, born in 1985, Ph.D, professor, is a member of China Computer Federation. His main research interests include quantum computation and quantum information.

(上接第 494 页)

[15] ZHU Y W, ZUO Z Q, WANG L Z, et al. C Program Memory Leak Intelligent Detection Method[J]. Journal of Software, 2019, 30(5).
[16] FURR M, FOSTER J S. Checking type safety of foreign function calls[C]// PLDI. 2005: 62-72.
[17] FURR M, FOSTER J S. Polymorphic Type Inference for the JNI[C]// ESOP. 2006: 309-324.
[18] JIANG T Y, WANG P, YANG S, et al. JNI Memory Leak Detection Based on Intermediate Language[J]. Journal of Computer Research and Development, 2015, 52(4).
[19] TAN G, APPEL A W, CHAKRADHAR S, et al. Safe Java Native Interface[J]. IEEE International Symposium on Secure Software Engineering, 2006: 97-106.
[20] LI S L, TAN G. Finding bugs in exceptional situations of JNI programs[C]// CCS. 2009: 442-452.
[21] LI S L, TAN G. JET: exception checking in the Java native interface[C]// OOPSLA. 2011: 345-358.
[22] LI S L, TAN G. Exception analysis in the Java Native Interface [J]. Science Computer Program, 2014, 89: 273-297.
[23] TAN G, CROFT J. An Empirical Security Study of the Native Code in the JDK[J]. USENIX Security Symposium, 2008: 365-378.
[24] LI S L, TAN G. Finding Reference-Counting Errors in Python/C Programs with Affine Analysis[C]// ECOOP. 2014: 80-104.
[25] MAO J J, CHEN Y, XIAO Q X, et al. RID: Finding Reference Count Bugs with Inconsistent Path Pair Checking[C]// ASPLOS. 2016: 531-544.
[26] HU M Z, ZHANG Y. The Python/C API: Evolution, Usage Statistics and Bug Patterns[C]// SANER. 2020: 532-536.
[27] TAN G, MORRISETT G. Ilea: inter-language analysis across java and c[C]// OOPSLA. 2007: 39-56.
[28] GHANAVATI M, COSTA D, SEBOEK J, et al. Memory and resource leak defects and their repairs in Java projects[J]. Empirical Software Engineering, 2020, 25(1): 678-718.
[29] FÜLÖP E, PATAKI N. A DSL for Resource Checking Using Finite State Automaton-Driven Symbolic Execution[J]. Open Computer Science, 2021, 11(1): 107-115.
[30] SINGH S, KHURSHID S. Distributed Symbolic Execution using Test-Depth Partitioning[J]. CoRR, abs/2106.02179. 2021.
[31] YAN H, SUI Y L, CHEN S P, et al. Automated memory leak fixing on value-flow slices for C programs[C]// SAC. 2016: 1386-1393.
[32] ROYCHOUDHURY A, XIONG Y F. Automated program repair: a step towards software automation[J]. Science China Information Science, 2019, 62(10): 200103: 1-200103: 3.
[33] GUPTA R, PAL S, KANADE A, et al. DeepFix: Fixing Common C Language Errors by Deep Learning[C]// AAAI. 2017: 1345-1351.
[34] SCOTT A, BADER J, CHANDRA S. Getafix: Learning to fix bugs automatically[J]. CoRR, abs/1902.06111, 2019.
[35] LI Y. Improving bug detection and fixing via code representation learning[C]// ICSE (Companion Volume). 2020: 137-139.

JIANG Cheng-man, born in 1995, master. His main research interests include network and information security.

HUA Bao-jian, born in 1979, Ph.D, assistant professor, graduate supervisor. His main research interests include programming language theory and implementation, computer and network security, etc.