用于SRAM PUFs的伪随机数发生器的FPGA实现

李 冰[1],周岑军[1],陈 帅[1],吉建华[2]
(1.东南大学微电子学院,江苏南京 210000;2.深圳大学信息工程学院,广东深圳 518060)

摘 要: 信息安全问题日益突出,而随机数则是信息安全系统的基石.本文以哈希算法为核心设计了一种伪随机数发生器,其以静态随机存储器物理不可克隆函数(Static Random Access Memory Physical Unclonable Functions,SRAM PUFs)为熵源,能够产生大量的伪随机序列.通过对熵源有效性的在线监测以及对种子的动态重播操作,本文提出的用于SRAM PUFs的伪随机数发生器提高了伪随机序列的安全性,可应用于各种高安全等级加密系统中.该发生器在FPGA开发平台上得到实现,其发生速度达598.1Mbps.随机数检测套件NIST分析结果表明:该伪随机数发生器的输出通过了所有测试项目,具有良好的随机性.

关键词: 伪随机数发生器;哈希算法;静态随机存储器物理不可克隆函数;熵检验;现场可编程门阵列
中图分类号: TP331 **文献标识码:** A **文章编号:** 0372-2112 (2017)09-2106-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.09.008

FPGA Implementation of Pseudo-Random Number Generator for SRAM PUFs

LI Bing[1],ZHOU Cen-jun[1],CHEN Shuai[1],JI Jian-hua[2]
(1. *School of Microelectronics,Southeast University,Nanjing,Jiangsu* 210000,*China*;
2. *School of Information Engineering,Shenzhen University,Shenzhen,Guangdong* 518060,*China*)

Abstract: The problem of information security is becoming serious,and the random numbers are the cornerstone of information security systems. This paper proposes a Hash-based pseudo-random number generator (PRNG) which takes static random access memory physical unclonable functions (SRAM PUFs) as entropy sources. This PRNG verifies the availability of entropy sources online and reseeds dynamically which improved the security of pseudo-random numbers. Therefore, it can be securely applied in high-level secure cryptographic protocols. This PRNG is implemented on FPGA development platform and the generation speed is up to 598.1Mbps. Experimental results of the NIST statistical test suite show that,the pseudo-random numbers generated by this PRNG pass all random tests and have good randomness.

Key words: PRNG;Hash algorithm;SRAM PUFs;entropy verification;field programmable gate array (FPGA)

1 引言

随机序列是一个重要的加密基元,广泛的用在密钥产生、认证协议系统中[1].通常使用真随机数发生器(True Random Number Generator,TRNG)从物理噪声源中导出种子,PRNG将该种子通过确定的算法扩展成为一个统计学上随机的周期性序列[2].种子的随机程度是衡量一个伪随机数发生器性能优劣的重要指标.

PUFs最主要的优点就是不需要存储任何保密信息.当加密系统需要使用这些信息时,只需要从PUFs的物理特性中导出即可[3],这使得攻击者想要获取保密信息变得非常困难.本文将SRAM PUFs作为TRNG向PRNG提供种子.该方法可以确保种子无法被攻击者预测,进而保证PRNG输出的伪随机序列的安全性.同时SRAM PUFs不需要搭建额外的电路来实现,因此也降低了系统的资源消耗[4].

本文实现的PRNG系统适用于广泛的SRAM芯片,能够在线检验熵源的有效性并进行动态重播操作.目

收稿日期:2016-05-11;修回日期:2016-08-19;责任编辑:覃怀银
基金项目:国家自然科学基金(No.61571116)

前大多数的加密系统中使用 LFSR 或者改进的 LFSR 来产生伪随机数,这种结构使得随机数容易被复原[5]. 本文根据 NIST SP800-90A 的说明设计了完整的 PRNG 硬件架构,其具有抗预测性和抗回溯性. 因此该设计相较于其他设计[6,7]具有较高的安全性和较快的发生速度,且能够实时检测、替换熵源,可被很好地应用于各种加密系统中.

2　SRAM PUFs

6 管 SRAM 单元是由图 1 中两个交叉耦合的 CMOS 反相器构成的双稳态电路[8],其利用两个对称正反馈环结构(图 1 中的 A、B)来使单元保持在逻辑 1 或逻辑 0 这两个状态上. 由于制造过程中产生的微小差异,SRAM 芯片中大部分单元的两个反馈环并不对称,导致出现以下两种类型的 SRAM 单元:(1)噪声单元:制造差异很小,以至于上电后偏向于逻辑 1 或逻辑 0 是一个受到环境变量影响的概率事件;(2)认证单元:制造差异很大,以至于上电后总是偏向于逻辑 1 或逻辑 0. 因此本文将噪声单元产生的不稳定值作为熵数据,压缩产生种子.

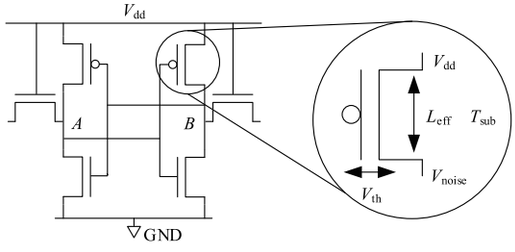


图1　SRAM单元结构

对于 SRAM PUFs,其激励为 SRAM 芯片的上电过程,其响应为 SRAM 芯片单元中未初始化的值. 因此,可以通过对 SRAM 芯片进行不断地上电、下电操作提取出种子. 为了描述种子的随机性,本文在这里引入最小熵 H_{min} 这个概念:H_{min} 表示一个随机变量随机性的最差情况[8],即该随机变量在混乱程度最小时的不可预测性. 1bit 随机变量的 H_{min} 的计算如式(1)所示:

$$H_{min} = -\log_2 \max(P_0, P_1) \tag{1}$$

其中 P_0 为随机变量为逻辑 0 的概率,P_1 为逻辑 1 的概率.

假设 SRAM PUFs 中所有单元都是独立的,且每个单元都可看作 1bit 的随机变量,则 n 个单元的 SRAM PUFs 所包含的最小熵和 $(H_{min})_{total}$ 的计算如式(2)所示:

$$(H_{min})_{total} = -\sum_{i=1}^{n} \log_2 (P_i)_{max} \tag{2}$$

其中 $(P_i)_{max}$ 为第 i 个单元的 $\max(P_0, P_1)$ 值.

3　最小熵提取模块设计

根据上一节中 $(H_{min})_{total}$ 的式(2),本文设计了最小熵提取模块,其具有两个关键参数:(1)SRAM PUFs 的下电时间 T;(2)SRAM PUFs 的上、下电次数 $N_{on\text{-}off}$.

参数 T 关系到 SRAM 芯片中的数据残留问题,即 SRAM 芯片下电后,SRAM 单元中的值仍会保留一段时间. 若在该段时间内对 SRAM 芯片上电,噪声单元将不会处于亚稳态状态,即只能提供很小的熵值. 根据文献[9]的实验结果,该实验使用的所有 SRAM 芯片数据残留时间都在 3s 以下. 为了针对最小熵模块设定最适合的参数 T,本文对 4 个系列的芯片(HM62256ALP-10, HY62256ALP-70, WS62256LLPG-70, UT62256CPC-70LL)进行了测试. 测试方法为:不同的 T 条件下,统计从 SRAM 芯片 0 地址开始能够产生 256bit 熵所需要的 SRAM 单元数量 Num,结果如图 2 所示. 从图中可以得出:当 T 大于 5s 时,SRAM 单元的数量趋于稳定,因此本文中最小熵提取模块的参数 T 被设置为 5s. 同时图 2 表明,实验中使用的所有 SRAM 芯片在常规环境下都能够产生足够的最小熵,即都能够作为 SRAM PUFs.

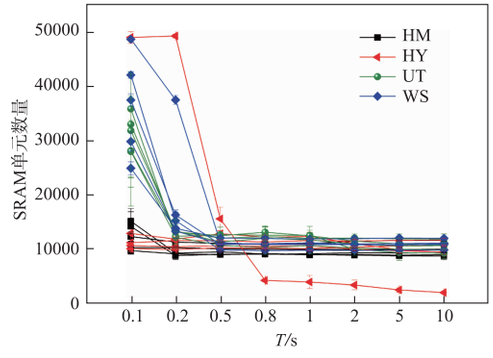


图2　参数T测试结果

参数 $N_{on\text{-}off}$ 关系到 SRAM 单元熵值的稳定性. 根据上一节中式(2),只有当 P_0 与 P_1 是稳定的,计算出的最小熵才符合真实情况. 若 $N_{on\text{-}off}$ 太小,会导致 PRNG 的安全性下降;若 $N_{on\text{-}off}$ 太大,将会延长数据处理时间并且浪费大量逻辑资源. 根据文献[6]的实验结果,当 $N_{on\text{-}off}$ 大于 100 后,最小熵将会趋于一个稳定的值. 由于该实验结果与芯片类型无关,因此本文初步将 $N_{on\text{-}off}$ 设置为 100.

本文提出的最小熵提取模块结构如图 3 所示,其在运行时分为两个阶段:检验阶段与压缩阶段,具体内部组成模块功能如下:

Switch 模块根据最小熵提取模块的运行阶段切换 PUF Initial 模块、PUF Process 模块与 SRAM PUFs 的接口.

Counter 模块控制 SRAM 芯片的上电与下电. 当 PUF Initial 模块和 PUF Process 模块发出下电信号时,Counter 模块将 SRAM 芯片的电源接地,并进行计时. 当计时达到 T,该模块将对 SRAM 芯片重新进行上电.

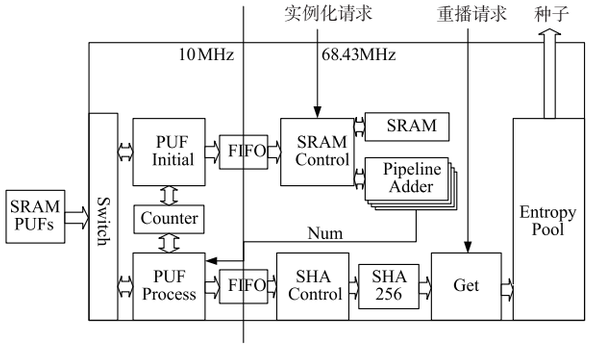


图3 最小熵提取模块结构框图

PUF Initial 模块从 SRAM PUFs 中遍历地址地读取 $N_{\text{on-off}}$ 次上、下电后 SRAM PUFs 中的值,并写入之后的 FIFO 中.

SRAM Control 模块将 FIFO 中的数据顺序写入 FPGA 内嵌 SRAM 中. 接收到安全强度请求后,该模块将相应的 $(H_{\min})_{\text{total}}$ 发送给 Pipeline Adder 模块,并将按照一定地址读取规则将 FPGA 内嵌 SRAM 中的数据输入 Pipeline Adder 模块. 该地址规则为:针对某单元第 i 次上、下电后的值与第 $i+1$ 次的值,由于两种 SRAM 芯片的位宽不同,储存在 FPGA 内嵌 SRAM 内实际间隔了 10000 个地址,因此读取 0 地址的值之后,应读取地址 10000 的值,以此类推.

Pipeline Adder 模块由 16 级流水线组成. 每一级流水将输入的 16 位数据中的固定位进行相加,以 $N_{\text{on-off}}$ 数据为一组,将相加结果通过一个 $\log_2$ 查找表,实现式(2)中 $\log_2(P_i)_{\max}$ 的计算. 该模块将 Num 组数据的最小熵和相加,当该数值大于 PRNG 模块所请求的安全强度时,表明该 SRAM PUF 可被用作 PRNG 的种子源,完成检验阶段,并发送出相应的 Num. 若接收完所有数据,最小熵和依然无法达到要求,该模块将发出更换 SRAM PUF 的信号.

PUF Process 模块根据 Num 从 SRAM PUFs 中读取上电后的初始值,通过 SHA Control 模块以及 SHA256 核心计算模块压缩成 256bit 的种子. 这些种子通过 Get Entropy 模块写入 Entropy Pool,等待 PRNG 模块的读取,完成压缩阶段. 为了防止环境噪声大幅变化导致 SRAM PUFs 熵减小,甚至失效,该模块在产生 50 个种子后将返回检验阶段,对 SRAM PUFs 再次检验. 当最小熵提取模块重新进入检验阶段,Entropy Pool 中的所有种子将被清空.

4 PRNG 模块设计

PRNG 模块根据 NIST SP800-90A 的说明[10]进行硬件实现,该模块的运行流程图如图 4 所示,共分为 3 个阶段:实例化阶段、发生阶段、重播阶段.

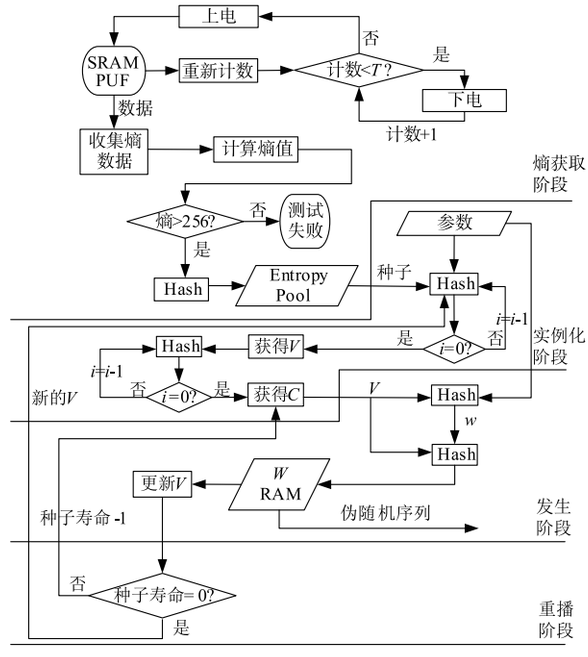


图4 PRNG模块运行流程图

该 PRNG 有 3 个额外输入:nonce、personalization string、additional input 以及 2 个主要参数:安全强度、种子寿命.

上述 3 个额外输入为系统提供了安全缓冲来抵挡一定的攻击,在本设计中分别将其设置为单向递增序列、一组随机序列、另一组随机序列.

该 PRNG 向下兼容的支持 112、128、192、256bit 安全强度,对应需要从熵源中获取种子的最小熵值. 种子寿命为一个种子能用于产生伪随机序列的次数. 当发生模块内部计数器递增到该值时,应立即停止产生伪随机序列,并启动重播模块重新获取 PRNG 系统的种子. 该参数的值在说明中并未规定,在本设计中初步将其设为 100,有待进一步实验证明其最优值.

PRNG 模块结构如图 5 所示,具体内部组成模块功能如下:

(1)数据控制模块

该模块控制 PRNG 的额外输入 nonce、personalization string、additional input 以及安全强度、种子寿命等参数输入.

(2)实例化模块

该模块获取种子与两个额外输入后,根据式(3)~(5)产生 PRNG 的初始内部状态 V_{ini}、C_{ini}.

$$\text{seed material} = \text{种子} \parallel \text{nonce} \parallel \text{personalization string} \tag{3}$$

$$V_{\text{ini}} = \text{leftmost}(\text{Hash}(0\text{x}01 \parallel \text{seed material}) \parallel \text{Hash}(0\text{x}02 \parallel \text{seed material}), 440) \tag{4}$$

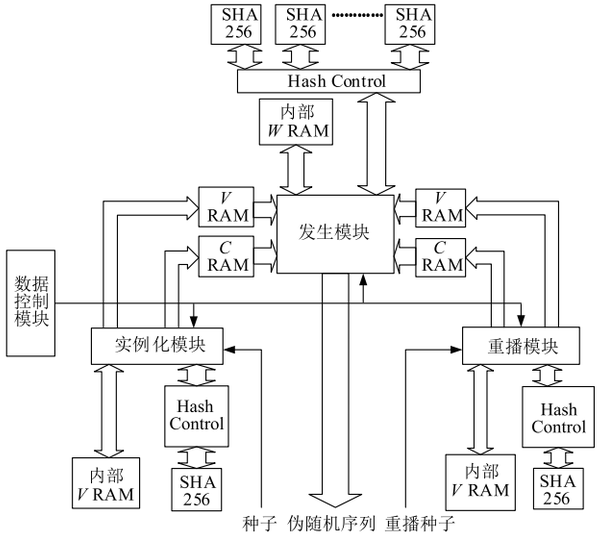


图5　PRNG模块结构框图

$$C_{\text{ini}} = \text{leftmost}\,(\text{Hash}\,(0\text{x}01 \parallel 34\text{'h}440 \parallel V_{\text{ini}}) \parallel \text{Hash}\,(0\text{x}02 \parallel 34\text{'h}440 \parallel V_{\text{ini}}),440) \tag{5}$$

其中 leftmost(a,b)的函数功能为从左至右截取序列 a 中前 b bit.

(3)发生模块

该模块获取实例化模块产生的初始内部状态 V_{ini}、C_{ini}后,根据一组式(6)~(8)产生伪随机序列 P-Numbers.

$$\omega = \text{Hash}\,(0\text{x}02 \parallel V_{\text{ini}} \text{or}\ V \parallel \text{additional input}) \tag{6}$$

$$V = V_{\text{ini}} \text{or}\ V + \omega \tag{7}$$

$$P\text{-Numbers} = \text{leftmost}\,(\text{Hash}\,(V) \parallel \text{Hash}\,(V+1) \parallel \cdots \parallel \text{Hash}\,(V+n), \text{request\_}n) \tag{8}$$

其中 request_n 为应用请求生成的伪随机序列的数量,该数量最大为 524288bit.

发生模块在输出产生的伪随机序列之前,必须先对内部状态进行更新,其根据一组式(9)、(10)更新 V.

$$H = \text{Hash}\,(0\text{x}03 \parallel V) \tag{9}$$

$$V = V + H + C_{\text{ini}} + \text{seed counter} \tag{10}$$

其中 seed counter 为种子寿命计数器,每生成一次伪随机序列,递减 1. 当种子寿命用尽或接收到重播请求时,发生模块会启动重播模块,进行重新播种操作. 之后发生模块将使用从重播模块获得的 V_{update} 和 C_{update} 来代替 V_{ini} 和 C_{ini}. 由于式(8)中对 V 的一系列哈希计算不依赖于之前的结果,因此为了加快伪随机序列的发生速度,设计中采用了并行的 SHA256 算法核,由 Hash Control 模块控制.

(4)重播模块

该模块重新从最小熵提取模块中获取种子,种子的熵值与初始种子熵值相同. 之后该模块从发生模块中获取当前的内部状态 V,根据式(11)~(13)产生新的内部状态 V_{updat} 和 C_{update},并发送清空 seed counter 计数器的指令.

$$\text{new seed material} = 0\text{x}01 \parallel V \parallel \text{重播种子} \parallel \text{additional input} \tag{11}$$

$$V_{\text{update}} = \text{leftmost}\,(\text{Hash}\,(0\text{x}01 \parallel \text{new seed material}) \parallel \text{Hash}\,(0\text{x}02 \parallel \text{new seed material}),440) \tag{12}$$

$$C_{\text{update}} = \text{leftmost}\,(\text{Hash}\,(0\text{x}01 \parallel 34\text{'h}440 \parallel V_{\text{update}}) \parallel \text{Hash}\,(0\text{x}02 \parallel 34\text{'h}440 \parallel V_{\text{update}}),440) \tag{13}$$

5　安全性讨论

第 3 节中设计的最小熵提取模块与第 4 节中设计的 PRNG 模块共同组成了 PRNG 系统. 图 6 描述了该系统运行时 PRNG 模块内部状态的转换. 伪随机数即从这些内部状态中推导出,一旦攻击者能够预测内部状态,那么该系统就处于不安全状态. 假设在 State_x 时系统发生了状态泄露,攻击者获得了相关信息并且假设其可获得系统的输出. 下面本文将在该假设条件下,就该系统所具有的抗回溯性与抗预测性进行相关讨论.

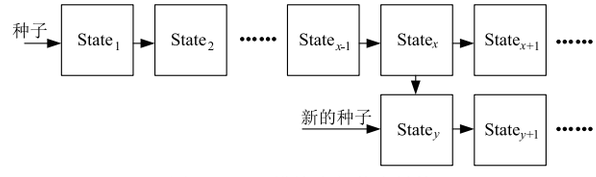


图6　PRNG模块内部状态转换

抗回溯性:一个内部状态的泄露无法对之前系统输出的伪随机序列的安全性造成影响[10]. 本文设计的系统采用 Hash 算法作为核心. 该算法的单向性特点可以确保在 State_x 状态已知的情况下,无法从输出中推导、复原出状态 State_1 到状态 State_{x-1},即通过式(8)与式(10)无法得出之前的 V.

抗预测性:一个内部状态的泄露无法对之后系统输出的伪随机序列的安全性造成影响[10]. 本文设计的系统能够进行周期性动态重播操作. 通过重播操作,系统获得新的种子能够将 State_x 的信息隐藏于 State_y 中,此时仅仅获得 State_x 信息的攻击者是无法计算出新的未来状态 State_{y+1},即通过式(12)无法得出之后的 V.

应用于高安全级别加密系统中的 PRNG 应能够在部分内部状态被攻击者窃取的情况下,保持功能的完整性. 通常 PRNG 将受到以下三种攻击[11]:密码分析攻击、边信道攻击、针对熵源的攻击.

(1)密码分析攻击

密码分析攻击又被称为直接密码攻击[12]. 针对本

文提出的系统实施该种攻击,攻击者需要对 SHA-256 算法进行密码分析破解. 就目前而言,该 Hash 算法还未得到有效的破解. 更换 Hash 算法也时抵御该种攻击的有效手段.

(2)边信道攻击

针对 PRNG 的边信道攻击可分为时间分析攻击[13]和功耗分析攻击[14],攻击者通常使用 PRNG 泄露出的额外信息来实施该类攻击. 根据上文所述该 PRNG 具有的抗回溯性与抗预测性,攻击者依然无法预测其输出.

(3)针对熵源的攻击

本文设计的 PRNG 系统以 SRAM PUFs 作为熵源. 目前针对其的攻击方法有:半侵入式攻击方法[15,16]以及基于数据残留衰退的边信道攻击方法[17].

文献[15]所述半侵入式攻击方法,主要针对 SRAM PUFs 中的认证单元,目的是获得认证单元中相对稳定的输出,从而破解认证协议、窃取密钥,相反地,噪声单元的存在将加大该类攻击实施的难度. 文献[16]通过侵入式方法修改 SRAM 电路来使其单元偏置,达到克隆出目标 SRAM PUF 中唯一的响应的目的. 该文献假设目标 SRAM PUF 中的噪声单元已被一个模糊提取器过滤,因此该方法同样是针对认证单元的输出. 本文设计的 PRNG 系统利用的是 SRAM PUFs 中的噪声单元,其输出受到温度、电压、芯片老化等因素影响为一个概率事件. 系统每次动态重播过程都会使噪声单元产生不可预测的输出差异. 同时该 PRNG 系统中种子具有最少 112bit 的熵值(最小支持安全强度),意味着预测出种子的概率将不会小于 1.93E-34,因此进行预测与克隆均存在很大难度.

文献[17]所述基于数据残留衰退的边信道攻击方法通过控制 SRAM 芯片的供电电压或下电时间向其注入错误,从而获得其上电模式,最终克隆出其中的认证单元. 在注入错误过程中,将极大程度改变 SRAM 单元中所包含的最小熵(如图 2 所示),文中的最小熵提取模块将会检测出该异常,并返回错误信号.

为了抵抗潜在的能够克隆、预测出完整 SRAM PUFs 的攻击方法,本文设计的 PRNG 系统将根据最小熵检测结果 Num 动态地从 SRAM PUFs 响应中截取不同数量的 SRAM 单元作为熵数据. 同时 SRAM PUFs 在本系统中处于不停振荡状态,检验阶段与压缩阶段对于攻击者来说是模糊的,实施攻击具有一定难度.

6 FPGA 硬件实现及测试

本文设计的 PRNG 系统在搭载了 Altera 公司的 Cyclone IV EP4CE115F29C7N 系列 FPGA 芯片的 DE2 开发板上实现,其实物图如图 7 所示. 图中编号 1 为最小系统板,其上搭载了用作 SRAM PUFs 的 SRAM 芯片. 编号 2 为由三极管组成的开关电路,用于最小熵控制模块中的 Counter 模块通过 CS 信号线控制 SRAM 芯片电源的上电与下电.

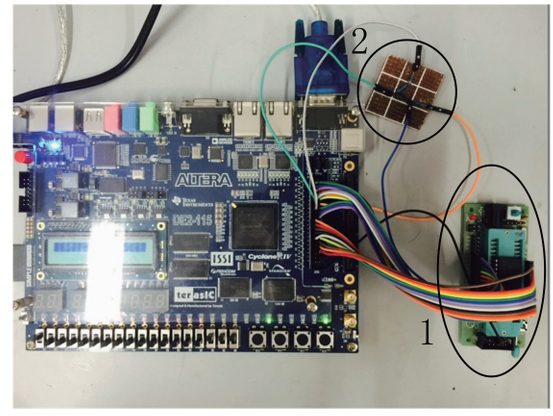


图7 PRNG系统的FPGA实现

表 1 为对该伪随机数发生器所消耗的逻辑资源以及存储资源的统计. 综合考虑资源消耗与伪随机数发生速度之间的平衡,本设计在发生模块中采用了四组并行的 SHA256 算法核心. 该种配置能够稳定运行在 68.43MHz 时钟频率下,产生伪随机数的速度为 598.1Mbps.

表 1 PRNG 系统消耗资源统计

	逻辑单元	存储单元
最小熵提取模块	5049	303104
配置 1:两组并行 SHA256 算法核心,吞吐量为 346.4Mbps		
PRNG 模块	15822	15488
系统	20874	318592
配置 2:四组并行 SHA256 算法核心,吞吐量为 598.1Mbps		
PRNG 模块	20694	15488
系统	25746	318592
配置 3:八组并行 SHA256 算法核心,吞吐量为 939.5Mbps		
PRNG 模块	30438	15488
系统	35487	318592

本文使用了 NIST SP800-22 中的随机性测试套件[18]对 PRNG 系统产生的伪随机序列进行了测试. 我们分别将 20 块 SRAM 芯片(HM62256ALP-10, HY62256ALP-70, WS62256LLPG-70, UT62256CPC-70LL 各 5 块)作为 SRAM PUFs,对每块芯片产生的 20 组伪随机序列(由 100 个长度为 1003520bit 的序列构成)进行了测试.

通常将显著性水平 α 与 P-Value 进行比较,以判断原假设是否正确. 若 P-Value $\geqslant\alpha$,那么被检测序列是随机的;若 P-Value $<\alpha$,那么被检测序列是非随机的. 本

文中将显著性水平 α 设置为 0.01,意味着当 P-Value $\geqslant$ 0.01 时,被检测序列在置信水平大于 99% 上通过了随机性检验. 测试结果表 2 所示.

表 2　NIST 随机性测试套件测试结果

	HM 系列	HY 系列	WS 系列	UT 系列
频率测试	0.51361	0.53490	0.48034	0.46219
块内频率测试	0.48528	0.53166	0.53351	0.47420
累积和测试	0.48009	0.49602	0.50913	0.51469
游程测试	0.53814	0.53834	0.48061	0.53129
块内最长连续 1 测试	0.54017	0.49063	0.40933	0.53411
二元矩阵秩测试	0.52678	0.49129	0.47355	0.47157
离散傅里叶变换测试	0.50007	0.51732	0.45894	0.51411
非重叠模板匹配测试	0.50093	0.50156	0.50060	0.49963
重叠模板匹配测试	0.50809	0.46529	0.49127	0.50305
全局通用统计测试	0.47161	0.54022	0.47734	0.49646
近似熵测试	0.51282	0.51590	0.52470	0.50611
随机偏移测试	0.39595	0.41117	0.43033	0.43780
随机偏移变量测试	0.41850	0.42028	0.43640	0.42626
串行测试	0.50267	0.52503	0.51067	0.52031
线性复杂度检测	0.48375	0.53405	0.47701	0.45746

7　结束语

本文设计了用于 SRAM PUFs 的 PRNG,在最小熵提取模块中采用了 16 级流水线,在发生模块中采用了 4 组 SHA256 算法核并行计算的设计方法,使其达到 598.1Mbps 的发生速度. 该设计以 SRAM PUFs 为种子源,能够在线检验其熵值并且动态进行重播操作,提高了 PRNG 的安全性. 本文通过分析该 PRNG 具有的抗回溯性、抗预测性以及对三类攻击具有的抗性,表明其可以应用于安全级别较高的加密系统中. 最后的测试结果表明:该 PRNG 的输出通过了 NIST 测试套件的所有随机性检测项目,其能够产生大量随机性良好的伪随机数.

参考文献

[1] Van Herrewege A. Lightweight PUF-Based Key and Random Number Generation[M]. Belgian:KU Leuven,2015. 19 – 26.

[2] Tsoi K H, Leung K H, Leong P H W. Compact FPGA-based true and pseudo random number generators[A]. Arnold J. Proceedings of Field-Programmable Custom Computing Machines[C]. New York:IEEE Computer Society, 2003. 51.

[3] Wachsmann C, Sadeghi A. Physically unclonable functions (PUFs): applications, models, and future directions[J]. Synthesis Lectures on Information Security Privacy & Trust,2014,9(1):1 – 91.

[4] Holcomb D E, Burleson W P, Fu K. Power-up SRAM state as an identifying fingerprint and source of true random numbers[J]. IEEE Transactions on Computers, 2009, 58(9):1198 – 1210.

[5] Cerda J C, Martinez C D, Comer J M, et al. An efficient FPGA random number generator using LFSRs and cellular automata[A]. Rafla N. International Midwest Symposium on Circuits and Systems[C]. New York:IEEE, 2012. 912 – 915.

[6] Leest V V D, Sluis E V D, Schrijen G J, et al. Efficient implementation of true random number generator based on SRAM PUFs[J]. Cryptography and Security, 2012, 6805: 300 – 318.

[7] Li D, Lu Z, Zou X, et al. PUFKEY: A high-security and high-throughput hardware true random number generator for sensor networks[J]. Sensors, 2015, 15(10): 26251 – 26266.

[8] Herder C, Yu M D, Koushanfar F, et al. Physical unclonable functions and applications: a tutorial[J]. Proceedings of the IEEE, 2014, 102(8):1126 – 1141.

[9] Fd C C, Skorobogatov S. Low Temperature Data Remanence in Static RAM[R]. Cambridge: University of Cambridge Computer Laboratory, 2002.

[10] Barker E B, Kelsey J M. SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators[M]. USA: National Institute of Standards & Technology, 2012. 11 – 42.

[11] Kelsey J, Schneier B, Ferguson N. Yarrow-160: notes on the design and analysis of the Yarrow cryptographic pseudorandom number generator[A]. Heys H. International Workshop on Selected Areas in Cryptography[C]. Berlin: Springer-Verlag, 1999. 13 – 33.

[12] Kelsey J, Schneier B, Wagner D, et al. Cryptanalytic attacks on pseudorandom number generators[J]. Lecture Notes in Computer Science, 2000, 1372: 168 – 188.

[13] Kocher P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems[A]. Koblitz N. Advances in Cryptology[C]. Berlin: Springer-Verlag, 2001. 104 – 113.

[14] Kocher P C, Jaffe J, Jun B. Differential power analysis[A]. Wiener M. Advances in Cryptology[C]. Berlin: Springer-Verlag, 1999. 388 – 397.

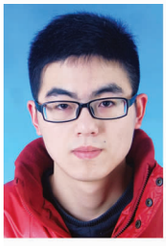
[15] Nedospasov D, Seifert J P, Helfmeier C, et al. Invasive PUF analysis[A]. Fischer W. Fault Diagnosis and Tolerance in Cryptography[C]. New York: IEEE Computer Society, 2013. 30 – 38.

[16] Helfmeier C, Boit C, Nedospasov D, et al. Cloning physically unclonable functions[A]. Karri R. Hardware-Oriented Security and Trust[C]. New York: IEEE, 2013. 1 – 6.
[17] Oren Y, Sadeghi A R, Wachsmann C. On the effectiveness of the remanence decay side-channel to clone memory-based PUFs[A]. Bertoni G. Cryptographic Hardware and Embedded Systems[C]. Berlin: Springer-Verlag, 2013. 107 – 125.
[18] Rukhin A, Soto J, Nechvatal J, et al. SP 800-22. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications[M]. USA: National Institute of Standards & Technology, 2010. 1 – 79.

作者简介

李 冰 男, 1968 年生于江苏南京, 博士, 现为东南大学教授, 博士生导师, 日本 IEICE 会员, 美国 IEEE 会员, 科技部火炬计划入库专家, 东南大学先进云系统联合研究中心主任, 目前主要从事安全信息数据交换及电路系统方面的研究工作.
E-mail: bernie_seu@seu.edu.cn

周岑军 男, 1991 年生于浙江杭州, 现为东南大学研究生, 主要从事信息安全及数字电路设计方面的研究工作.
E-mail: zhoucenjun@msn.cn



陈 帅 男, 1989 年生于山东邹平, 现为东南大学博士, 在国际期刊上发表多篇 SCI 论文, 主要从事信息安全、数据压缩及数字电路设计方面的研究工作.
E-mail: chenshuai_ic@seu.edu.cn

吉建华 男, 1970 年生于江苏江阴, 博士, 现为深圳大学教授, 硕士生导师, 目前主要从事光通信及信息安全方面的研究工作.