Journal of Computer Applications
计算机应用,2017,37(5):1341 - 1346
ISSN 1001-9081
CODEN JYIIDU
2017-05-10
http://www.joca.cn

文章编号:1001-9081(2017)05-1341-06
DOI:10.11772/j.issn.1001-9081.2017.05.1341

基于高层次综合的 AES 算法研究与设计

张　望[1,2*],贾　佳[3],孟　渊[1,2],白　旭[1,2]
(1.中国科学院 信息工程研究所,北京 100093;　2.信息内容安全技术国家工程实验室,北京 100093;
3.北京特种工程设计研究院,北京 100028)
(*通信作者电子邮箱 zhangwang@iie.ac.cn)

摘　要:由于对广泛使用的 AES 算法的性能要求越来越高,基于软件的密码算法已经越来越难以满足高吞吐量密码破解的需求,因此越来越多的算法利用现场可编程逻辑门阵列(FPGA)平台进行加速。针对 AES 算法在 FPGA 硬件上存在的开发复杂度高且开发周期长等问题,采用高层次综合(HLS)设计方法,使用高级程序语言描述并设计 AES 硬件加速算法。首先利用循环展开等提高运算并行度;其次使用资源平衡技术进行优化,充分利用片上存储和电路资源;最后添加全流水结构,提高整体设计的时钟频率和吞吐量,同时也详细对比分析基准设计、利用结构展开、资源均衡以及流水线优化方法的设计。经过实验表明,在 Xilinx xc7z020clg484 FPGA 芯片上,最终 AES 算法的时钟频率最高达到 127.06 MHz,而吞吐量达到了 16.26 Gb/s,较之基准的 AES 设计,性能提升了三个数量级。

关键词:对称密钥加密算法;高级加密标准;高层次综合;现场可编程逻辑门阵列
中图分类号:TP309.7　　**文献标志码**:A

Research and design of AES algorithm based on high-level synthesis

ZHANG Wang[1,2*], JIA Jia[3], MENG Yuan[1,2], BAI Xu[1,2]
(1. *Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093, *China*;
2. *National Engineering Laboratory of Information Security Technologies, Beijing* 100093, *China*;
3. *Beijing Special Engineering Design and Research Institute, Beijing* 100028, *China*)

Abstract: Due to the increasingly high performance requirements on the Advanced Encryption Standard (AES) algorithm which was widely used, software-based cryptographic algorithms have been increasingly difficult to meet the demands of high-throughput ciper cracking. As a result, more and more encryption algorithms have been accelerated by using Field-Programmable Gate Array (FPGA) platform. Focused on the issue that the development of AES algorithm based on FPGA has high complexity and long development cycle, with High-Level Synthesis (HLS) design methodologies, AES hardware acceleration algorithm was designed by using high-level programming language. Firstly, loop unrolling, etc were used to improve operation parallelism. Secondly, to make full use of on-chip memory and circuit resources, the resource balance optimization technology was used. Finally, the full pipeline structure was added to improve the clock frequency and throughput of the overall design. The detailed analysis and comparison of the benchmark design and different optimized designs with structural expansion, resource balance and pipeline were decribed. The experimental results show that the clock frequency of AES algorithm is up to 127.06 MHz and the throughput eventually achieves 16.26 Gb/s on Xilinx xc7z020clg484 platform, compared with the benchmark AES design, performance increases by three orders of magnitude.

Key words: symmetric key encryption algorithm; Advanced Encryption Standard (AES); High-Level Synthesis (HLS); Field-Programmable Gate Array (FPGA)

0　引言

随着计算机和 Internet 技术的迅速发展,网络信息技术日益深入社会、经济、政治、民生的方方面面,使人们的生活和工作越来越离不开网络通信;然而,现代社会的人们面临着信息窃听、信息伪造、信息重放等多种信息安全方面的巨大隐患,因此,伴随而来的信息传输安全问题也引起了学者们的高度关注。信息安全的主要内容是保护信息和承载信息的载体,避免信息和信息系统遭受任意形式的破坏和攻击;一旦信息和信息系统受到攻击和破坏,应能迅速响应、阻断破坏、修复信息和信息系统。在实际应用中,信息安全的解决主要依靠两种方式:以“防火墙”技术为核心的被动防御系统和以“密码”为基础的主动保障系统。密码技术作为保护信息安全的最重要方式之一,在保护信息系统安全方面有着举足轻重的地位。信息加密以某种特殊的算法改变原有的信息数据,使得未授权的用户即使获得了已加密的信息,但因不知解密的方法,仍然无法了解信息的内容,从而对信息进行保护。

美国国家标准与技术研究院(National Institute of

收稿日期:2016-09-12;**修回日期**:2016-11-27。
基金项目:国家自然科学基金青年科学基金资助项目(61402475);新疆自治区科技专项(201230123)。
作者简介:张望(1992—),男,江西南昌人,博士研究生,主要研究方向:信息安全、计算机体系结构;　贾佳(1981—),男,北京人,工程师,博士,主要研究方向:通信与信息工程;　孟渊(1983—),男,甘肃高台人,硕士研究生,主要研究方向:信息安全;　白旭(1990—),男,辽宁喀左人,博士研究生,主要研究方向:信息安全、计算机体系结构。

Standards and Technology, NIST)在 1997 年便宣布了一个高级加密标准(Advanced Encryption Standard, AES)的开发工作以代替已经力不从心的数字加密标准(Data Encryption Standard, DES)。而最终,由 Joan Daemen 和 Vincent Rijmen 设计的 Rijndael 算法[1]被选择为 AES 算法。对于目前密码算法的高速性能要求,尤其是数据传输速度达到 GB 数量级时,密码算法的硬件实现便显得尤为重要。以现场可编程门阵列(Field-Programmable Gate Array, FPGA)为代表的可重构硬件以其自身所固有的特点——既具有硬件的安全性和高速性又有软件的灵活性和易维护性,已经成为分组密码算法硬件实现的热点研究方向[2]。

文献[3]总结了目前传统开发方法遇到的问题以及使用高层次综合(High-Level Synthesis, HLS)技术的优势,即由于在片上系统(System on a Chip, SoC)上创建更多更复杂的应用,其功能设计的复杂度也随之增加,而且还要考虑功耗控制以及工艺变化等问题。然而利用 HLS 技术可以自动化管理这些日益增加的复杂设计,让设计开发人员专注于高层功能的设计以及非常便捷地探索设计空间以优化设计。目前的一些高层次综合工具支持从C/C++代码进行硬件综合。使用这些高层次综合工具的优势也是十分明显的,这将给硬件设计开发带来更高的效率,并能够充分利用已有的可靠软件程序,减少开发成本与时间[4];然而,并不是所有的C/C++代码都能够直接用于硬件设计,因此需要在C/C++设计层面上作出改变以达到高效的硬件设计实现,比如文献[5]探讨了基于 HLS 技术,在低资源情况下随机存取存储器(Random Access Memory, RAM)不同分割数目对于 AES 算法的性能的影响;文献[6]则探讨了算符、操作相关带来的影响;文献[7]基于 HLS 技术利用 C 到寄存器传输级(C to Register-transfer level, C2R)设计方法探索设计空间,以设计不同硬件结构的 AES 算法来满足不同的性能资源需求的实现;文献[8-11]探讨了传统设计方法下对于 AES 算法的硬件设计,其中的文献[8]利用了 $GF(2^4)$ 上的多项式表示来替换 AES 原始的 $GF(2^8)$ 上的计算以提高硬件流水吞吐量。还有应用于主干网的 AES 高性能硬件设计研究[12-15]。

利用 HLS 技术可以非常方便地实现与优化 AES 算法硬件设计;然而,目前却缺乏对于全自动优化与手动优化对于最终性能的影响的分析,也缺乏对于代码映射出的硬件结构的分析,这对于进一步改善性能,减少资源消耗是不利的。在本文中描述了关于 AES 算法基于 C 的软件设计以及其硬件实现结构,本文依次给出了三种不同代码设计以及优化方法,并分别添加流水线优化得到其最终性能资源情况,分析其映射的总体硬件结构同时也对比分析不同的优化策略效果,最终得到了高性能、资源利用低的设计版本。

1 背景介绍

1.1 高级加密标准

AES 高级加密标准,在密码学中又称 Rijndael 加密算法,是美国联邦政府采用的一种区块加密标准。这个标准用来替代原先的 DES,已经被多方分析且广为全世界所使用。其分组长度为 128 位,而使用的密钥长度可以为 128 位、192 位或 256 位,即 AES 算法分为 AES-128、AES-192 和 AES-256,由于彼此之间的相似性,本文只描述并实现了 AES-128 加密算法,更多细节可以参考文献[16]。

大多数 AES 计算是在一个特别的有限域完成的。AES 加密过程是在一个 4×4 的字节矩阵上运作,这个矩阵又称为“状态(state)”矩阵,其初值就是一个明文区块(矩阵中一个元素大小就是明文区块中的一个 Byte)。加密时,各轮 AES 加密循环(除最后一轮外)均包含 4 个步骤:

轮密钥加(AddRoundKey) 矩阵中的每一个字节都与该次轮密钥(Round Key)做 XOR 运算;每轮子密钥由密钥生成方案产生。

字节代换(SubBytes) AES 中的字节代换往往用查找表的方式把每个字节替换成对应的字节。即定义了一个 S 盒与逆 S 盒。S 盒用于加密过程的字节代换,逆 S 盒用于解密操作的逆字节代换。对于状态矩阵中的元素,将其高 4 位作为行值,低 4 位作为列值,取出 S 盒或逆 S 盒中对应的元素作为输出。

行移位(ShiftRows) 将矩阵中的每行进行循环式移位。即状态矩阵的第 0 行左移 0 字节,第 1 行左移 1 个字节,第 2 行左移 2 个字节,第 3 行左移 3 个字节。这就使得每一列完全重排了。

列混合(MixColumns) 为了充分混合矩阵中各个直行的操作。这个步骤使用线性转换来混合每列的四个字节。状态矩阵中的第 j 列($j = 0, 1, 2, 3$)的列混合如下表示为:

$$\begin{cases} s'_{0,j} = (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} = s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} = (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{cases} \tag{1}$$

AES 算法中还包括密钥拓展(KeyExpansion)操作。其为每一个轮密钥加操作提供轮密钥。首先,将初始密钥输入到一个 4×4 的字节矩阵上,然后将每一列 4 个字节的元素组成一个字,于是便形成了数组 w。对于 AES-128 算法而言,要将 w 扩充 40 个新列,构成 44 列的拓展密钥数组。其计算过程为:

$$\begin{cases} w[4i] = w[4i-4] \oplus (RotWord(SubWord(w[4i-1])) \oplus Rcon[i]) \\ w[4i+1] = w[4i] \oplus w[4i-3] \\ w[4i+2] = w[4i+1] \oplus w[4i-2] \\ w[4i+3] = w[4i+2] \oplus w[4i-1] \end{cases} \tag{2}$$

其中:*RotWord* 表示字循环的功能,即将这个字的循环左移 1 个字节;*SubWord* 是对字进行字节代换操作;*Rcon* 为轮常量数组。

1.2 现场可编程逻辑门阵列

FPGA 是一类高集成度的可编程逻辑器件,起源于美国的 Xilinx 公司,可以用硬件描述语言(Hardware Description Language, HDL)来编程,它灵活性强,不仅能够进行编程、除错、可重复擦写操作,还可以结合算法的特点和硬件的结构进行重构,提高了系统的可升级性和产品的市场寿命。

FPGA 由基本可编程的逻辑单元、可编程 I/O 的单元、嵌入式块 RAM、布线资源等重要部分构成。FPGA 器件及其系统开发利用计算机软件,绘制出实现用户逻辑的原理图或用硬件描述语言等方式作为设计输入,然后经过一系列转换程序、自动布局布线、模拟仿真的过程,最后生成配置 FPGA 器件的数据文件,下载到 FPGA 器件中,从而实现了满足用户需求的专用集成电路,真正达到了用户自行设计集成电路的目的。其一般通过自顶向下的设计方法,实现设计的结构化,使一个复杂的系统设计可由多个设计者分工合作,同时还可以

实现层次化的管理[17]。其自顶向下的设计流程包括设计定义、HDL 实现、功能仿真、逻辑综合、前仿真、布局布线、后仿真、静态时序分析、系统测试 9 个步骤[18]。

1.3　高层次综合

HLS 是指采用高层次语言,比如 C 或者C++描述底层的数字设计。显然,它能让设计师和工程师在较高的抽象层面上完成底层设计。利用 HLS 能快速探索各种方案的可能性,分析资源面积和性能特点,最终确定硬件芯片上算法实现方案。

如图 1HLS 流程,用户先创建一份 C、C++ 算法设计文件,以及一个用于测试系统行为的 Test Bench; 随后用高层语言的仿真器验证设计的系统行为, 一旦算法设计的运行结果正确,就可以通过 HLS 运行设计,生成 RTL(Register-Transfer Level)级别的设计,代码可以是硬件描述语言 Verilog 或者 VHDL。有了 RTL 设计之后,随即可以执行设计的硬件仿真。

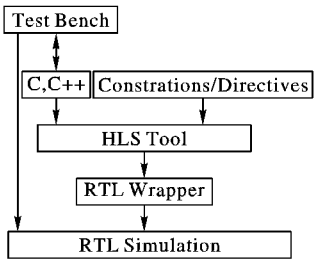

图 1　HLS 流程

Fig. 1　Flow of HLS

不仅如此,HLS 提供的抽象层次有助于开发人员从高层设计硬件逻辑,解决了常见硬件系统设计难题,最终简化系统汇编,简化 FIFO(First Input First Output) 和存储器访问; 此外,HLS 提供了指令完成架构研究功能,也就是 Constrains/Directives 部分,设计者可以通过添加插入程序指令,把设计所需特性传递给综合工具,这样就可以在不修改设计代码本身的情况下研究大量备选架构方案。

算法的设计工作基于 HLS,主要完成 Test Bench、C/C++ 以及 Constrains/Directives 部分的设计。Test Bench 模拟未来系统中的软件部分,也充当测试硬件设计的角色; 而C/C++ 部分经过 RTL 设计后,就是未来系统中的硬件部分; Constrains/Directives 是开发人员在设计C/C++ 部分时,考虑采用的优化手段,能够指引硬件代码资源的利用、并行的设计等。这样从整体上完成了软硬件的协同设计,统筹设计了整个系统,有助于挖掘硬件更大的计算能力。

2　算法实现

2.1　基准代码设计

利用 HLS 技术就可以像软件设计人员开发软件一样通过编写C/C++ 代码来达到 FPGA 硬件设计的目的。此时,像平时编写 C 程序一样,首先,编写出第 1 版 AES-128 加密模块的 C 代码, 其伪代码简要描述如下:

```
方法名　aes_enc_v1。
输入　indata,initkey;
输出　outdata。
1)　if 未进行过密钥拓展
2)　　keyExpansion( initkey);
3)　endif
4)　初始化状态矩阵 state 并对其对做轮密钥加;
5)　for round = 1 to 9
6)　　subByte( state);
7)　　shiftRow( state);
8)　　mixColumn( state);
9)　　对状态矩阵 state 做轮密钥加;
10)　endfor
11)　subByte( state);
12)　shiftRow( state);
13)　对状态矩阵 state 做轮密钥加;
14)　输出结果 outdata.
```

首先,大体上说明一下代码的实现。对于函数 keyExpansion,其输入初始密钥 initkey,然后进行密钥拓展的操作,再将 44 列的结果放入数组 key 中。当然,这个密钥拓展操作只需要在开始时计算即可,对于后续需要加密的 128 比特块可以一直使用先前的密钥直到需要更改密钥时。

密钥拓展操作完成以后,就是 AES-128 加密算法的具体流程。首先,在第 0 轮首先对状态矩阵 state 对每一个元素与此轮轮密钥进行异或操作,很显然,状态矩阵 state 在代码中是以数组形式存在的,而与之异或的轮密钥也是存储在数组中,很自然地轮密钥加的过程是用一个循环来实现的。循环在软件程序设计中是十分常用的结构,而且正因为循环的使用,软件开发人员可以编写十分简练的代码,大大提高软件开发效率,并且,本研究在这一版代码中像平时编写 C 代码一样许多地方使用循环操作。

第 0 轮操作以后,就是第 1 轮到第 9 轮的加密过程,除了轮密钥加的操作以外,还包含字节代换、行移位以及列混合操作。这三个操作都是通过函数调用的形式完成的。函数调用也是平时常用的方法,利用函数调用可以逐步地解决算法任务,而且还可以利用许多高效的系统函数来帮助编写C/C++ 程序,进而提高程序开发的效率。因此,本文在这一版的程序中也将一些加密操作以函数调用的形式实现。

对于字节代换函数 subByte,其输入是状态矩阵 state,在这个过程中,本研究使用查表的方式实现的,即函数中通过查找 S 盒来完成字节代换操作,而不是通过计算得到的。行移位函数 shiftRow,输入是 state,进行的操作如同上一章节描述的那样进行。列混合函数 mixcolumn 输入为 state,其实现的方式与上一章节描述的过程略有不同,其并没有乘法运算过程,而是将乘法运算的结果都放入到两个数组 mm02 和 mm03 中,前者是元素与 0x02 相乘的所有结果的集合,后者是元素与 0x03 相乘的结果集合。因此,实际上列混合操作只有异或运算了。当然,在这个函数中也用到了循环结构。

第 10 轮调用函数 subByte 完成字节代换,接着调用函数 shiftRow 完成行移位,最后再做一次轮密钥加的操作完成了一个 128 比特块的整个加密过程。

完成了软件设计以后,就要对软件代码进行 C 语言功能仿真以验证功能的正确性,完成高级语言功能验证以后,就可以利用 HLS 工具综合出 RTL 级设计,即本文所设计的硬件部分。首先,对于第 1 版的代码主要加密操作对应的结构如图 2 所示。在 HLS 工具综合过程中,数组时常被映射为 RAM 结构,运算式则利用 LUT(Look Up Table) 构造。在 aes_enc_v1 这个顶层模块中,HLS 工具一般会将子函数单独生成一个子模块,若是子函数非常简单,HLS 工具会自动将其内联到顶层函数中。对于方法 aes_enc_v1 所映射出的结构,可以看出,受限于循环和子函数的限制,整体并行运算程度不高,HLS 工具也就不会分配更多的可访问资源,因此,子模块与顶层模块之

间以及子模块与 RAM 之间的通信会是十分频繁的,这就带来更多的延迟了。

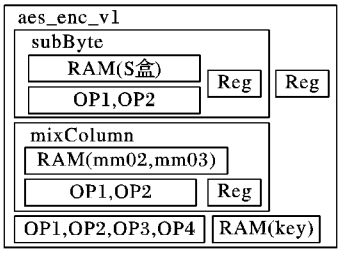

图 2 aes_enc_v1 方法硬件模块结构

Fig. 2 Hardware module structure of aes_enc_v1

2.2 结构展开

上述的第 1 版的代码可以说是一个非常常见的 C 代码设计。然而,事实上从硬件设计的角度而言是并非是一个很高效的方法。因此,可以对代码进行改写,而第 2 版 AES-128 加密模块的 C 代码的伪代码如下所示:

方法名 aes_enc_v2。

输入 indata,initkey;

输出 outdata。

```
1)  if 未进行过密钥拓展
2)      keyExpansion(initkey);
3)  endif
2)  初始化状态矩阵 state;
2)  对状态矩阵 state 进行轮密钥加;
2)  for round = 1 to 9
2)      对状态矩阵 state 进行行移位;
2)      对状态矩阵 state 进行字节代换;
2)      对状态矩阵 state 进行列混合与轮密钥加;
2)  endfor
2)  对状态矩阵 state 进行行移位;
2)  对状态矩阵 state 进行字节代换与轮密钥加;
13) 输出结果 outdata.
```

在这一个版本的代码中,与前一个版本最大的不同在于把加密操作的函数代码都写在顶层函数中,并且,除此之外,还把加密操作中的循环全部展开了,只保留有 1 到 9 轮加密操作的主循环。将循环展开是一种非常自然的优化方法,将循环展开之后,就可以发现表达式之间其实并不存在任何相关关系,此时就可以并行处理了,而不像先前受限于循环只能串行操作;而且又打破了函数之间界限,可以进一步提高并行处理的能力以提高数据吞吐量,而且也能优化资源的利用。

对于字节代换过程而言,状态矩阵中的每一个元素通过 S 盒来进行替换。事实上,每个元素的替换过程没有任何依赖关系,可以完全并行执行。而对于行移位操作而言,其在传统设计中只需要改变连线就可以达到效果,而在本研究的设计中,只需要在字节代换过程中改变元素的赋值顺序就可以了。对于列混合操作,从式(1)可以看出,状态矩阵每一个新元素依赖于同一列的四个旧元素。因此,每一个新元素的计算是可以完全并行执行的。因此,循环展开是完全可以提高性能的。

aes_enc_v2 方法所映射的总体结构如图 3 所示。正如之前所言,在方法 aes_enc_v2 中,打破了循环与函数之间界限,而且,事实上 AES 算法本身具有非常良好的并行执行能力,许多表达式之间并没有相关关系,完全可以并行处理,之前循环的串行执行则会强迫产生不必要的等待时间,增加整个加密过程的延时。由于此时打破了循环与子函数的限制,HLS 工具可以在更广的范围内对一系列的操作进行调度优化,可以进一步提高 AES 加密算法的并行执行能力,从而进一步降低整个加密过程的延时与资源占用情况。

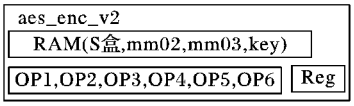

图 3 aes_enc_v2 方法硬件模块结构

Fig. 3 Hardware module structure of aes_enc_v2

2.3 资源均衡

上述的两个版本的代码在很大程度上是依赖于表的查找,因为已经事先就计算好了一些运算操作的结果并存储在数组中,而数组往往会被映射成 RAM 结构。对于 RAM 块的需求会很大,而且频繁访问 RAM 块会增加路径延时,从而影响硬件的工作时钟频率[15],这对于进一步提升算法实现的性能是不利的。事实上,本文还可以对 aes_enc_v2 的列混合部分的代码进行改写。在域上的乘法乘以 0x02 等价于下列的结果:

$$\begin{cases} a_6a_5a_4a_3a_2a_1a_00, & a_7 = 0 \\ a_6a_5a_4a_3a_2a_1a_00 \oplus 00011011, & a_7 = 1 \end{cases} \tag{3}$$

因此,基于方法 aes_enc_v2 所做的改进而得到新一版的 C 语言算法设计。将这个第 3 版方法命名为 aes_enc_v3。

对于 aes_enc_v3 方法映射成的硬件结构与图 3 并没有什么太大的不同,只是资源占用的情况不太一样,其显然要比 aes_enc_v2 方法的 RAM 少,但是由于运算增加,对 LUT 资源需求会增加一些。

2.4 流水线优化

本文的目标是追求高性能算法实现,因此有必要进一步提高数据吞吐率,故必须要给硬件结构增加流水线设计以进一步提高吞吐量。而此时,可以利用 HLS 工具自带的命令来帮助完成想要的硬件结构设计。本文利用 HLS 工具的流水线命令对 aes_enc_v1、aes_enc_v2 与方法 aes_enc_v3 进行流水线的优化;然后,增加流水线的设计可以大幅地增加硬件设计的吞吐量,利用流水线优化会自动展开循环进行优化调整,这样可以进一步提高硬件实现的并行执行的能力,因此整个加密过程的延迟也可以进一步降低,当然,这会导致更大的资源占用情况。

图 4 是设计 aes_enc_v1 第 1 轮到第 9 轮轮内的结构,而第 10 轮结构与其类似,只是少了列混合操作。

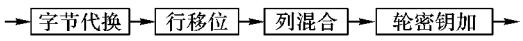

图 4 aes_enc_v1 轮内流水结构

Fig. 4 Pipeline structure of aes_enc_v1 in one round

图 5 是设计 aes_enc_v1 整个实现结构。对于方法 aes_enc_v2与方法 aes_enc_v3 而言,其轮内轮外操作已经都糅合在一起。虽然总体操作过程与原始方法 aes_enc_v1 相同,然而其某些操作过程的顺序已经被高层次综合工具自动调度了。

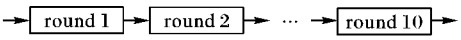

图 5 aes_enc_v1 整体流水结构

Fig. 5 Overview of pipeline structure of aes_enc_v1

3 实验结果与分析

本研究使用 Vivado 集成开发环境中的 HLS 工具完成本

文中的设计。实验采用的 FPGA 器件是 xc7z020clg484-1。首先对 aes_enc_v1 方法、aes_enc_v2 方法和 aes_enc_v3 进行硬件综合以了解其性能和利用情况。表 1 描述的是三种设计的性能评估。表 2 描述的是三种设计的资源利用情况(注:BRAM 为 Block RAM,FF 为 Filp-Flop,DSP 为 Digital Signal Processing,LUT 为 Look-Up Table)。

实验中 FPGA 的时钟约束均为 100 MHz。方法 aes_enc_v1 受限于循环与子函数的限制,导致并行执行的能力没有很好地发挥出来,因此比后两者的延时要大很多。从表 1 可以看出,aes_enc_v1 的延时是后两者的 11.7 倍,而方法 aes_enc_v1 与方法 aes_enc_v2 的延迟此时是一样的。

表 1　三种设计性能评估　　时钟周期

Tab. 1　Performance evaluation of three designs　clock cycle

设计	Lantency	Interval	设计	Lantency	Interval
v1	1 156	1 157	v3	99	100
v2	99	100			

此时再来查看三种设计对于资源的占用情况:方法 aes_enc_v1 的对资源的占用从总体上看是最高的;方法 aes_enc_v2 比方法 aes_enc_v1 在 BRAM 方面少用 3 块 RAM,在 FF 方面比其减少 12.2% 的资源使用,在 LUT 方面比其减少 24.2% 的资源使用;方法 aes_enc_v3 比方法 aes_enc_v1 在 BRAM 方面少用 7 块 RAM,在 FF 方面比其减少 16.4% 的资源使用,在 LUT 方面比其减少 3.8% 的资源使用;而方法 aes_enc_v2 与方法 aes_enc_v3 在有相同的资源利用。

表 2　三种设计资源占用情况

Tab. 2　Resource occupancy of three designs

设计	BRAM	DSP	FF	LUT	设计	BRAM	DSP	FF	LUT
v1	10	1	957	1 570	v3	3	1	800	1 511
v2	7	1	840	1 187					

综合表 1 与表 2 的实验结果分析,方法 aes_enc_v1 的设计效果并不理想,不仅延时很高,而且资源的占用也是 3 种设计之中最高的。对于方法 aes_enc_v2 与方法 aes_enc_v3 来说,其无论是性能上还是资源占用上并没有太大的不同,而之前预计方法 aes_enc_v3 要比方法 aes_enc_v2 延时要更低一些。这是因为受限于主循环的限制,加密模块整体并行执行能力依旧受到了许多限制,访问 RAM 所带来额外的延时依旧有许多可以缓和的额外空间。总的来看,对于方法 aes_enc_v1 所进行的优化设计符合要求,达到了预期的效果。

然后,此时来查看对于方法 aes_enc_v1、aes_enc_v2 与 aes_enc_v3进行流水线优化后其性能情况以及资源利用情况。表 3 描述的是三种设计进行流水优化的性能情况。表 4 描述的是三种设计进行流水优化后的资源利用情况。

从表 3 可以看出,添加了流水线之后,加密模块的延时与吞吐量得到了很大的改进。方法 aes_enc_v1 流水线优化后的延时为原先的 1.6%,方法 aes_enc_v2 流水线优化后的延时为原先的 28.3%,而方法 aes_enc_v3 流水线优化后的延时仅为原先的 19.2%。从吞吐量方面来看,对于方法aes_enc_v2 与 aes_enc_v3 而言,若时钟频率均相同的情况下,进行流水线优化后其吞吐量是之前的 100 倍。对于方法aes_enc_v1而言则是 1 157 倍,而这也意味着与原始的基准设计相比,最终的性能提升了 3 个数量级。另外,可以看到方法 aes_enc_v2 与方法 aes_enc_v3 在延时上的区别,后者的延时比前者减少了 32.1%。虽然方法 aes_enc_v2 申请了大量 RAM 资源以提高并行执行效率,减少资源冲突带来性能损失,然而频繁访问 RAM 会增加延迟,从而影响加密模块的工作频率,这降低了硬件的吞吐量。其次,三种设计 Interval 均为 1,也就意味着每个时钟周期都可以输入数据进行运算处理。事实上,当达到全流水线的架构时,吞吐量仅仅取决于时钟频率[8],其计算公式为:

$$throughput = blocksize \times frequency \tag{4}$$

表 3　流水线优化后性能评估　　时钟周期

Tab. 3　Performance evaluation with pipelining　clock cycle

设计	Lantency	Interval	设计	Lantency	Interval
v1_pipeline	19	1	v3_pipeline	19	1
v2_pipeline	28	1			

从表 4 可以看出,进行流水线优化后,对资源利用需求大幅提高。对于方法 aes_enc_v1 而言,进行流水线优化后对于 RAM 块的需求是原先的 24.4 倍,对 FF 资源的需求是原先的 17.7 倍,对 LUT 的需求是原来的 5.6 倍;对于方法 aes_enc_v2 而言,进行流水线优化后对于 RAM 块的需求是原先的 34.9倍,对 FF 资源的需求是原先的 4.3 倍,对 LUT 的需求是原来的 5.7 倍;对于方法 aes_enc_v3 而言,进行流水线优化后对于 RAM 块的需求是原先的 33.3 倍,对于 FF 的需求是原先的 3.0倍,而对于 LUT 资源的需求是原来的 7.0 倍。

表 4　流水线优化后资源占用情况

Tab. 4　Resource occupancy with pipelining

设计	BRAM	DSP	FF	LUT
v1_pipeline	244	0	16 931	8 806
v2_pipeline	244	0	3 577	6 769
v3_pipeline	100	0	2 376	10 545

对于完全依赖于高层次综合工具进行优化的方法 aes_enc_v1 而言,其最终性能非常优越。其延时与方法 aes_enc_v3相同,比方法 aes_enc_v2 还要低。这也说明过多地将逻辑内联进顶层模块事实上会降低性能。当然,此时可以看到方法 aes_enc_v1 对于资源的占用从总体上看是最大的。

综合来看,虽然此时资源需求提高很多,然而,性能的提升更为显著。最终,对于方法 aes_enc_v1 进行流水优化后时钟频率达到了 127.06 MHz,吞吐量达到了 16.26 Gb/s;对于方法 aes_enc_v2 进行流水优化后,时钟频率达到121.65 MHz,吞吐量达到了 15.57 Gb/s;而对于方法 aes_enc_v3 进行流水优化后时钟频率达到了 127.06 MHz,吞吐量达到了 16.26 Gb/s。因此,可以看出,方法 aes_enc_v3 在本文所设计方法之中不仅具有最高的性能,而且还具有最优的资源占用。

表 5 为与具有代表性的传统硬件设计的横向比较。由于不同的设计基于的 FPGA 芯片有差异,为公平起见,采用性能面积比(吞吐率/电路资源数量)这一指标进行对比。

从表 5 可以看出,在性能面积比上,本文的设计比之前最好的实验结果提升了 44%。因此,本文的设计达到了较好的性能与资源需求之间的平衡,性能面积比是占有优势的。事实上,对于算法的设计要考虑到实际应用的场景,不能一味地追求性能而不考虑到对于资源的需求、器件的成本等因素,需

要综合考虑各种因素,从而达到最优的结果。

表 5 本文设计与传统设计的比较

Tab. 5 Comparison of v3_pipeline and traditional hardware designs

设计	器件	时钟频率/MHz	吞吐量/(Gb·s^{-1})	Slices	性能面积比/(Mbps·Slice^{-1})
v3_pipeline	xc7z020clg484-1	127.06	16.26	5214	3.11
文献[8]算法	xc2v2000-5	139.10	17.80	10750	1.65
文献[9]算法	virtex7	456.00	5.30	2444	2.16
文献[10]算法	xcv2600e	34.20	4.12	11354	0.36
文献[11]算法	xcv812e	54.35	6.96	4444	1.57

4 结语

在本文的设计中利用高层次综合工具并采用高层次语言(C 语言),描述底层的硬件设计,从而在较高的抽象层面上完成底层设计。在文中不断探讨如何优化C/C++层面的程序设计,并且也讨论了其反映的硬件结构之间的区别。最终,本文完成了一个吞吐量高达 16.26 Gb/s、资源需求仅为 5 214 Slices 的算法设计与实现,与原始的基准设计相比,最终的性能提升了 3 个数量级。

经过实验表明,利用高层次综合工具完成的设计可以与传统 FPGA 硬件设计所达到的效果相媲美,而最关键之处在于,本文利用C/C++的高层次综合工具来进行硬件设计的工作效率比传统 FPGA 硬件设计的工作效率高了许多,而且入门门槛也大大降低。而这也意味着开发人员可以更加专注于应用的功能设计,减少对于底层的关注,并且借助于高层次综合工具探索不同的方法,从而找到最优化的解决方案。

目前对于网络安全性的要求也越来越高,更多的数据流量需要通过加密技术来保证传输。因而,高效、安全、稳定的加密算法的实现研究依旧是一个研究重点,这也是下一阶段研究工作所重点追求的目标。

参考文献(References)

[1] DAEMEN J, RIJMEN V. The Design of Rijndael[M]. New York: Springer-Verlag, 2002: 31 – 50.

[2] 周轶男, 李曦, 冯朝阳. 高速全并行的 AES 加解密算法在单片 FPGA 上的实现[J]. 计算机应用, 2004, 24(S2): 102 – 106. (ZHOU Y N, LI X, FENG C Y. An implementation of high-speed parallel AES algorithm on a single chip FPGA[J]. Journal of Computer Applications, 2004, 24(S2): 102 – 106.)

[3] COUSSY P, TAKACH A. Guest editors' introduction: raising the abstraction level of hardware design[J]. IEEE Design & Test, 2009, 26(4): 4 – 6.

[4] 党宏社, 王黎, 王晓倩. 基于 Vivado HLS 的 FPGA 开发与应用研究[J]. 陕西科技大学学报(自然科学版), 2015, 33(1): 155 – 159. (DANG H S, WANG L, WANG X Q. Development and application of FPGA based on Vivado HLS[J]. Journal of Shaanxi University of Science & Technology (Natural Science Edition), 2015, 33(1): 155 – 159.)

[5] MEURER R S, MÜCK T R, FROHLICH A A. An implementation of the AES cipher using HLS[C]// Proceedings of the 2013 Ⅲ Brazilian Symposium on Computing Systems Engineering. Washington, DC: IEEE Computer Society, 2013: 113 – 118.

[6] 孙桂玲, 纪永鑫, 张潺潺, 等. 基于 HLS 技术的 Rijndael 算法 IP 核实现与优化[J]. 微电子学与计算机, 2010, 27(4): 205 – 208. (SUN G L, JI Y X, ZHANG C C, et al. Implementation and optimization of Rijndael arithmetic IP core based on HLS technology [J]. Microelectronics & Computer, 2010, 27(4): 205 – 208.)

[7] AHUJA S, GURUMANI S T, SPACKMAN C, et al. Hardware coprocessor synthesis from an ANSI C specification[J]. IEEE Design & Test of Computers, 2009, 26(4): 58 – 67.

[8] JÄRVINEN K U, TOMMISKA M T, SKYTT J O. A fully pipelined memoryless 17.8 Gbps AES-128 encryptor[C]// Proceedings of the 2003 ACM/SIGDA 11th International Symposium on Field Programmable Gate Arrays. New York: ACM, 2003: 207 – 215.

[9] HUSSAIN U, JAMAL H. An efficient high throughput FPGA implementation of AES for multi-gigabit protocols [C]// Proceedings of the 2012 10th International Conference on Frontiers of Information Technology. Piscataway, NJ: IEEE, 2012: 215 – 218.

[10] RODRIGUEZ-HENRIQUEZ F, SAQIB N A, DÍAZ-PÉREZ A. 4.2 Gbit/s single-chip FPGA implementation of AES algorithm[J]. Electronics Letters, 2003, 39(15): 1115 – 1116.

[11] MCLOONE M, MCCANNY J V. High performance single-chip FPGA Rijndael algorithm implementations[C]// Proceedings of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems. London: Springer-Verlag, 2001: 65 – 76.

[12] HENZEN L, FICHTNER W. FPGA parallel-pipelined AES-GCM core for 100 G Ethernet applications[C]// Proceedings of the 2010 European Solid-State Circuits Conference. Piscataway, NJ: IEEE, 2010: 202 – 205.

[13] ZHOU G, MICHALIK H, HINSENKAMP L. Efficient and high-throughput implementations of AES-GCM on FPGAs[C]// Proceedings of the 2007 International Conference on Field-Programmable Technology. Piscataway, NJ: IEEE, 2008: 185 – 192.

[14] ZHOU G, MICHALIK H, HINSENKAMP L. Improving throughput of AES-GCM with pipelined Karatsuba multipliers on FPGAs[C]// Proceedings of the 5th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications. Berlin: Springer-Verlag, 2009: 193 – 203.

[15] LIU Q, XU Z, YUAN Y. A 66.1 Gbps single-pipeline AES on FPGA[C]// Proceedings of the 2013 International Conference on Field-Programmable Technology. Piscataway, NJ: IEEE, 2013: 378 – 381.

[16] PUB N F. Announcing the Advanced Encryption Standard (AES) [S/OL]. [2016-05-20]. http://www.just.edu.jo/~tawalbeh/cpe542/slides/aes/fips-197.pdf.

[17] 杨海钢, 孙嘉斌, 王慰. FPGA 器件设计技术发展综述[J]. 电子与信息学报, 2010, 32(3): 714 – 727. (YANG H G, SUN J B, WANG W. An overview to FPGA device design technologies [J]. Journal of Electronics & Information Technology, 2010, 32(3): 714 – 727.)

[18] 田耘, 胡彬, 徐文波. Xilinx ISE Design Suite 10.x FPGA 开发指南: 逻辑设计篇[M]. 北京: 人民邮电出版社, 2008: 8 – 12. (TIAN Y, HU B, XU W B. Xilinx ISE Design Suite 10.x FPGA Development Guide: Logic Design[M]. Beijing: Posts and Telecom Press, 2008: 8 – 12.)

This work is partially suuported by the Youth Science Fund of National Natural Science Foundation of China (61402475), the Xinjiang Uygur Autonomous Region Science and Technology Project (201230123).

ZHANG Wang, born in 1992, Ph. D. candidate. His research interests include information security, computer architecture.

JIA Jia, born in 1981, Ph. D., engineer. His research interests include communication and information engineering.

MENG Yuan, born in 1983, M. S. candidate. His research interests include information security.

BAI Xu, born in 1990, Ph. D. candidate. His research interests include information security, computer architecture.