基于FPGA的AES加密算法可重构设计

刘建钊
(盐城工学院 优集学院,江苏 盐城 224051)

摘要:在信息的传输过程中,对信息进行加密处理是保证信息安全的关键。用FPGA技术实现了AES加密算法并针对算法的反馈工作模式,通过采取等效解密过程、就算法中的关键函数进行优化、实现密钥扩展与加/解密的并行进行等措施降低了算法实现的硬件复杂度,提高了数据处理速度,仿真过程验证了设计的正确性。

关键词:信息安全;FPGA;UART高级加密标准;优化
中图分类号:TP391.75　　**文献标识码**:A　　**文章编号**:1671-5322(2009)01-0057-05

信息在传输过程中面临着被非法截获、窃取等安全威胁,用高强度加密算法对重要数据进行加密处理使第三者即使获得了数据也无法明了其真实含意是保证信息安全的关键环节之一。本文用FPGA技术实现了AES,用硬件方式对传输数据进行加密,去除了以往软件加密方式存在的占用CPU资源、易被非法攻击篡改等缺点。

1　AES算法概述及其逻辑实现

AES是美国新一代数据加密算法,能够有效抵抗各种攻击手段。AES为一个区块加密法,加密与解密使用相同的密钥。数据分组(明文/密文)长度为128位,密钥长度可以为128、192及256位的选择,而运算回合数对3种密钥长度分别为10、12、14次,课题选取了128位的密钥长度。在加/解密过程中,分组要经过10个回合的转换操作,分组及每次回合操作的结果均被按字节排列成4×4的矩阵,除初始密钥外,每回合子密钥由密钥扩展算法获得[1]。加、解密过程如图1、图2所示。

1.1　AES算法的实现架构

在实际工作中,多用AES算法的反馈工作模式来加/解密数据,即要先处理完前一个分组才能处理后一个分组,每一个分组都对以后的分组产生影响。本文把研究重点放在反馈工作模式上。

由于在反馈工作模式下,必须等待前一个分组处理完毕后才能继续下一个分组,所以流水线结构的长处无法发挥,却会额外增加电路面积;循环展开结构则须付出相当大的电路面积来换取些微运算效能的提升,不仅不经济也无法满足某些应用需要较小电路面积的要求[2]。因此反馈工作模式下AES优化必须针对AES算法合理架构以提高处理速度和优化运算函数的电路面积上。我们选择了循环结构,在完成一次回合操作后,将计算结果反馈到寄存器进行下一回合,密钥扩展

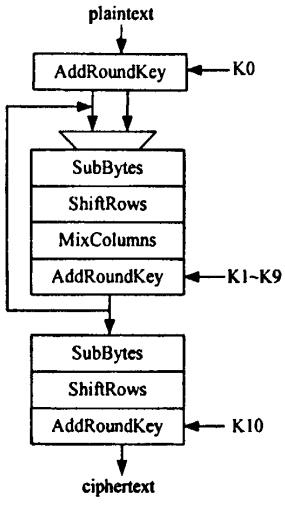

图1　加密流程
Fig. 1　Encryption process

收稿日期:2008-10-20
作者简介:刘建钊(1978-),男,河北石家庄人,助教,硕士研究生,主要研究方向为可编程逻辑设计。

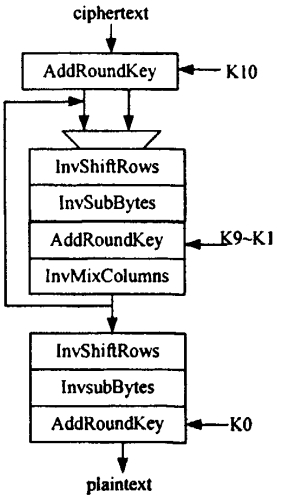

图 2 解密流程

Fig. 2 Decryption process

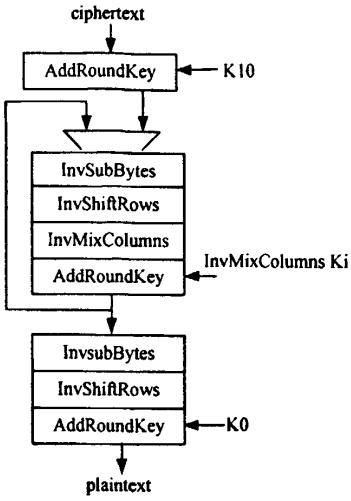

图 3 等效解密流程

Fig. 3 Equal Decryption Process

与回合操作并行进行,经过 10 轮迭代后完成一个分组的加/解密处理。

1.2 对运算函数的改进

1.2.1 解密过程的等效变换

AES 算法具有以下两个特性。

(1). InvShiftRows 不改变字节的值,只是进行移位;InvSubBytes 只进行字节替换,与字节的位置无关,因此两者的施加顺序可以颠倒而结果一致。

(2). InvMixColumns 变换是种线性变换。

解密过程中 AddRoundKey (State, RoundKey);InvMixColumns(State) 运算函数可表示为:

InvMixColumns(State ⊕ RoundKey) = InvMixColumns(State)⊕InvMixColumns(RoundKey)

即,只要子密钥在应用前经过了 InvMixColumns 变换,AddRoundKey 与 InvMixColumns 两种操作就可以颠倒顺序而结果不变。这样,基于上述两个特性,解密过程的等效形式如图 3 所示。

1.2.2 Subbytes/InvSubbytes 函数的整合

SubBytes/InvSubBytes 运算函数的实现,一种方式是采用复合域来实现求逆运算;另一种方法是以查表的方式来分别进行,而每张表需要 256 字节大小的存储空间。若采用并行执行 16 字节的替换,则在一轮运算中共需要 20 个 Subbytes 表(其中 16 个用于 SubBytes 操作,4 个用于密钥扩展操作)和 16 个 InvSubbytes 表,总共占用 9k 字节的空间,会占用不少存储空间。本文中实现了使两个函数共用一张 256 字节查找表的方式,并将其存储到 FPGA 内部的 ROM 中,根据输入字节的数值作为地址,对应地址空间里存放的就是被替换结果[3,4]。

SubBytes /InvSubBytes 操作可通过下式表示:

$$S'_{i,j} = M * \text{multiplicative\_inverse}(S_{i,j} + C)$$

$$S'_{i,j} = \text{multiplicative\_inverse}(M' * S'_{i,j} + C),$$

其中 $C = [0\,1\,1\,0\,0\,0\,1\,1]$,

$$M = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{vmatrix}$$

$$M' = \begin{vmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{vmatrix}$$

multiplicative_inverse( )为查表操作,如表 1 所示:

表 1　双向字节替换查询表

Table 1　multiplicative_inverse table

		Y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
X	0	00	01.	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
	1	74	BB	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
	2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
	3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
	4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
	5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
	6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
	7	79	B7	95	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
	8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
	9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
	A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
	B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
	C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
	D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
	E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
	F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

1.2.3　Mixcolumns/InvMixcolumns 函数

在这两种函数中，主要有以下两种计算过程：

MixColumns_out =

$$[2\ \ 3\ \ 1\ \ 1]*[b_0\ \ b_1\ \ b_2\ \ b_3]^T =$$

$$2(b_0+b_1)+b_1+(b_2+b_3)$$

InvMixColumns_out =

$$[E\ \ B\ \ D\ \ 9]*[b_0\ \ b_1\ \ b_2\ \ b_3]^T =$$

$$4(2(b_0+b_1)+2(b_2+b_3)+(b_0+b_2))+$$

$$2(b_0+b_1)+b_1+(b_2+b_3) =$$

$$4(2(b_0+b_1)+2(b_2+b_3)+(b_0+b_2))+$$

MixColumns_out

从上述公式可以看出，两种运算函数中存在可共享资源的部分。为使两种操作可整合在一个函数中实现，方案中使用了如下方式。

单字节的 byte - MixColumns 操作整合方式如图 4。

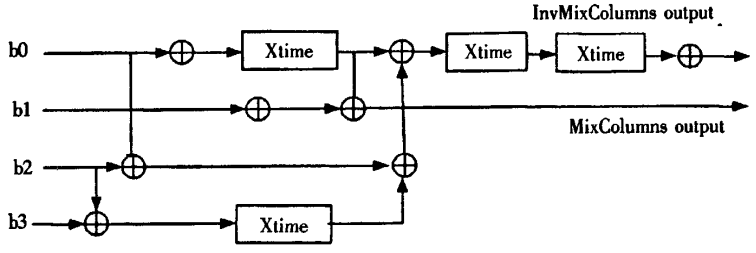

图 4　单字节的列混合

Fig. 4　byte - MixColumns

其中，函数 Xtime(in[7:0] = {in[6:0], in[3:0] ∧ {in[7], in[7], 1'b0, in[7]}, in[7]}。

4 字节的 word - MixColumns 操作如下，需要四个 byte - MixColumns：

其中 En/De 用来指示当前是加密亦或解密过程，选择 MixColumns_out 或 InvMixColumns_out 作为输出结果。

对于 16 字节的 Integrated - MixColumns 操作如图 5 所示，这样经过了解密过程的等效变换和函数整合后，一回合操作过程变成了如图 6 所示结构。

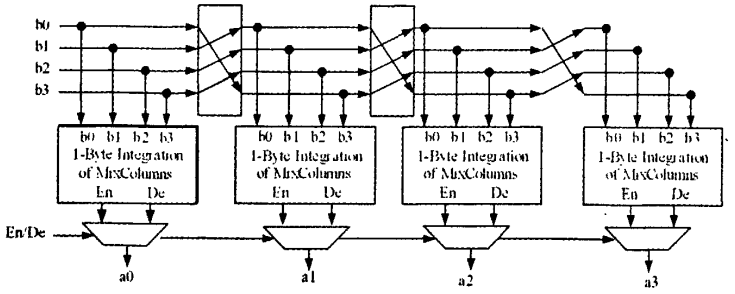

图5 4字节的列混合

Fig. 5 word - MixColumns

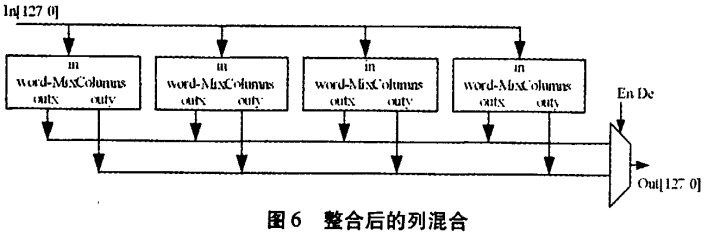

图6 整合后的列混合

Fig. 6 Integrated - MixColumns

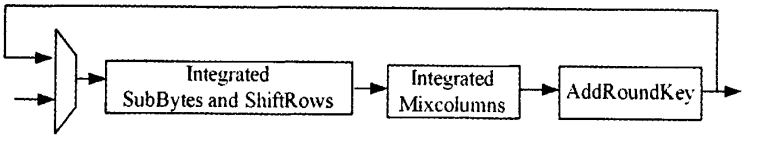

图7 加/解密流程

Fig. 7 Equal Decryption Process

这样就可以将加密器与解密器整合在一个模块中,实现了加/解密过程的资源共享,比之于加密器与解密器分开设计的方式大大减少了所占用的电路面积。

2 子密钥的扩展与调度

密钥扩展模块的功能是依据输入的初始密钥,产生AES算法每轮迭代所需要的子密钥。在本设计中,为了使所产生的各轮子密钥不占用存储空间,模块采用了密钥扩展与加/解密运算同时进行的on_the_fly方式。

设计中,我们进一步把加密和解密运算的子密钥生成电路通过如下方式整合到一起,节省了电路面积[5]。结果如图8所示。

3 仿真与综合结果

在设计中,以Altera公司的Stratix Ⅱ系列高性能FPGA为目标器件,仿真和综合工具分别选用了ModelSim SE 5.8c和Quartus II5.0来验证所设计的AES加密模块的逻辑功能的正确性和可综合性。此外,经过FPGA综合和布局布线进行后仿真,以保证时序的正确性。

表2列出了在Quartus II下的综合结果,包括时钟频率、占用的逻辑单元等。

表2 AES模块改进对比

Table 2 Comparison of optimized AE

	Clock (MHz)	LEs	Throughput (Mb/s)	Throughput /LEs
优化前	107.12	3.41k	273.9	81
优化后	102.9	3.17k	878	277

4 结束语

本文以大规模可编程逻辑FPGA为实现载体,用VerilogHDL硬件描述语言对AES进行了建模并对其实现过程做了优化,降低了硬件复杂度,

提高了数据吞吐率,优化结果对于一些电路面积有限的应用非常重要(如智能 IC 卡、网络通信设备等),同时这些优化措施也可应用于循环展开等实现方式。

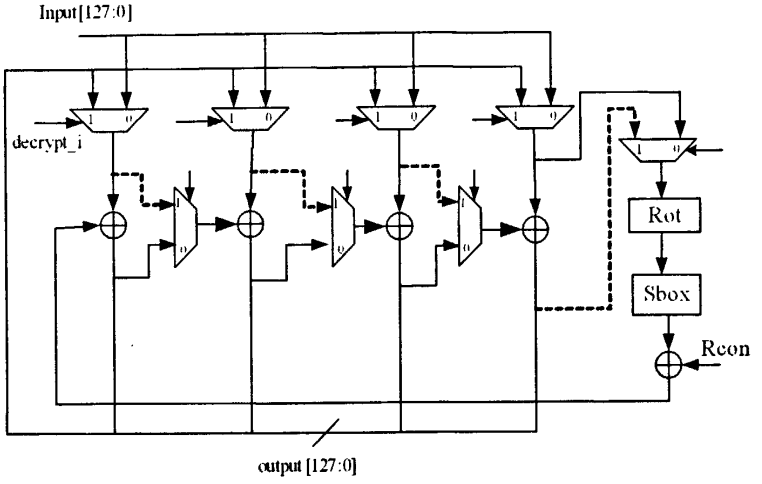

图 8　整合后的密钥扩展算法

Fig. 8　Integrated Key_generation Algorithm

参考文献:

[1] Daemon J, Rijmen V. AES Proposal: Rijndael, Version2[EB/OL]. 1999. http://www.esat.kuleuvan.ac.be/~rijndael.
[2] 林茂琼,熊凯,李敏强,等. 基于 AES 的数据加密方案[J]. 计算机工程,2002,28(4):140-142.
[3] 褚振勇,翁木云. FPGA 设计及应用[M]. 西安:西安电子科技大学出版社,2002.
[4] 夏宇闻. 复杂数字电路与系统的 VerilogHDL 设计技术[M]. 北京:航天航空大学出版社,2002.
[5] 杨晖. 大规模可编程逻辑器件与数字系统设计[M]. 北京:航天航空大学出版社,2002.

Reconfigurable Design for AES Algorithm Based on FPGA

LIU Jian-zhao
(UGS School of Yancheng Institute of Technology, Jiangsu Yancheng 224051, China)

Abstract: Enciphering information is the way to ensure information safety. This paper realizes the AES algorithm in FPGA and decreases the hardware complexity and speeds up data processing via adopting equivalent decryption process, optimizing functions and realizing the key expansion working concurrently. The following simulation proves its correctness.

Keywords: information safe; FPGA AES; optimization

(责任编辑:张英健;校对:沈建新)