IEEE802.15.4 中 AES-CCM 协议的扩展指令集实现

封 斌* 齐德昱 韩海雯
(华南理工大学计算机系统研究所 广州 510640)

摘 要：该文在高级加密标准(AES)快速算法的基础上，设计了一组基于可配置处理器 NiosII 上的扩展指令，用于 IEEE802.15.4 标准媒体访问控制层中基于 AES 算法的计数器模式和密码分组链接消息验证码(AES-CCM)协议的硬件加速。该文首先推导出快速算法中用于轮变换的查找表与 S 盒的逻辑关系，然后通过复合域变换方法用硬件电路实现 S 盒的计算，从而消除了支撑扩展指令集的硬件逻辑对片上存储空间的消耗。同时给出该协议基于查表法的扩展指令集和协处理器的设计方案，并在 EP2C35 芯片上进行实现和对比。该方案仅消耗 223 个逻辑单元(LE)，吞吐量为 668.7 kbps，时钟周期数比软件算法加速 174.6 倍，芯片面积仅为协处理器方案的 9.5%，显著降低了无线传感网节点设备的成本和功耗。

关键词：无线传感网；扩展指令集；IEEE802.15.4；高级加密标准的计数器模式和密码块链信息认证码(AES- CCM)协议；S 盒；复合域

中图分类号：TP393; TP302.1 **文献标识码**：A **文章编号**：1009-5896(2013)02-0335-06

DOI: 10.3724/SP.J.1146.2012.00854

Instruction Set Extension Implementation for AES-CCM in IEEE802.15.4

Feng Bin Qi De-yu Han Hai-wen
(*Research Institute of Computer Systems*, *South China University of Technology*, *Guangzhou* 510640, *China*)

Abstract: An instruction set extension for Counter mode with Cipher block chaining Message authentication code protocol using Advanced Encryption Standard algorithm (AES-CCM) protocol in IEEE802.15.4 is presented based on AES fast algorithms and NiosII processor. The logical relationship between the lookup table used for round transformation and S-box is derived, then the S-box value is calculated with composite field transform method in hardware circuit, thereby eliminated the consumption of on-chip memory. The scheme is verified on EP2C35 chip, and the design and experimental data of look-up table method of the instruction set extension design and co-processor are also proposed for compare. This schemes increases the speedup by 174.6 times than software implementation, only uses 223 logic elements as 9.5% of coprocessor, throughput achieves 668.7 kbps, and reduces significantly the cost and power consumption of wireless sensor network node equipments.

Key words: Wireless Sensor Network (WSN); Instruction set extensions; IEEE802.15.4; AES-CCM protocol; S-box; Composite field

1 引言

IEEE802.15.4 标准[1]已成为无线传感器网络(WSN)底层设施事实上的工业标准，其定义了低速无线个域网(LR-WPAN)的物理层(PHY)和介质访问控制层(MAC)，有效满足了无线通信市场对低成本、低功耗、低速率设备的需求。WSN 的 MAC 层使用基于高级加密标准(AES)[2]的计数器模式和密码块链信息认证码(CCM)协议[3]作为通信链路层的安全机制，用以实现无线通信的机密性、完整性和可认证性。无线通信终端设备对成本、功耗、体积等指标都有严格的要求，由于计算量大及为保证自身安全性，其安全协议普遍采用专用协处理器的方法实现硬件加速。

协处理器适用于对粗粒度的任务进行硬件加速，运行中不需要主处理器的干预或支持，通过片内流水线等的设计，可达到较高的吞吐量。文献[4]通过将 802.15.4 标准中 AES-CCM 协议内部的两种工作模式串行交错运行，设计了时分复用的 AES 协处理器；文献[5]为 802.11i 的 AES-CCM 协议设计了协处理器；文献[6]为 802.16e 和 802.11i 设计了可重构的 AES-CCM 协处理器；文献[7]为 802.1ae 的 AES-GCM 协议设计了专用协处理器。由于 WSN

2012-07-04 收到，2012-09-13 改回
国家自然科学基金(61070015)和广东省自然科学基金团队项目(10351806001000000)资助课题
*通信作者：封斌 billfeng126@126.com

在各频段的应用对吞吐量要求最高仅为 250 kbps，重点在于降低 WSN 节点设备的成本和功耗，协处理器方法不能很好地满足 WSN 设备的相关要求。本文针对 WSN 节点设备的特点，采用适用于细粒度任务硬件加速的扩展指令集方法实现了 AES-CCM 协议。已有多种运行在不同处理器上的 AES 算法扩展指令集[8–10]，此类方案通过分析 AES 算法的运行数据，将耗时最多的轮函数用硬件电路实现，并提供相应的指令接口，但都没有考虑到 AES 算法自身的结构特点，专用指令所对应的硬件电路有进一步优化的空间。

AES 算法在 32 位处理器上的快速软件实现方法[11]，将 AES 加密算法中计算密集的轮变换中的 4 个轮函数合并为对 4 张前向查找表(FT)的查表操作，大幅降低了算法本身的运算复杂度。本文在此基础上，进一步推导出 FT 和 S 盒的逻辑关系，通过复合域变换的方法用硬件电路实现了 S 盒的计算，完全消除了片上内存的占用，在可配置处理器 NiosII 系统硬件环境下，用 12 条扩展指令完成了 AES-CCM 协议的硬件加速。为与用复合域变换实现 S 盒的扩展指令集方法进行分析对比，本研究还同时实现了基于查表法的扩展指令集及协处理器方案，并在基于 Altera 公司的 CycloneII EPS2C35 芯片的硬件环境下进行了验证和数据分析对比。

2 AES-CCM 协议分析

分组加密算法解决了一组明文的加密问题，而工作模式定义了各个分组数据间的相互关系，其将加密算法与一些反馈及简单的运算组合到一起，用以防止数据在传输过程中被篡改，并定义了在不同应用要求和场合下正确使用分组加密算法的规范。计数器(CTR)模式在 1979 年提出，用于对消息进行加密以保证数据的机密性，可并行处理加解密操作。密码分组链接消息验证码(CBC-MAC)模式最早在数据加密算法(DES)算法中用于计算消息的认证码，以保证数据的完整性鉴权。CCM 协议结合了 CTR 和 CBC-MAC 模式，同时实现了数据传输的机密性、完整性和可认证性。

CCM 协议消息加密流程包括数据的加密和生成认证码，其过程如下：CCM 协议消息加密流程包括数据的加密和生成认证码，其流程为：(1)将消息 $\boldsymbol{P}$ 按 128 bit 长度分组为 $\boldsymbol{B}(1),\boldsymbol{B}(2),\cdots,\boldsymbol{B}(n)$; (2)构建 $\boldsymbol{B}_0$，即 CBC 模式初始化向量 **IV**，并对其进行加密：$\boldsymbol{Y}_0=E(\boldsymbol{K},\boldsymbol{B}_0)$，其中 $\boldsymbol{K}$ 为密钥，E 为加密算法；(3)按 CBC 模式对消息分组进行加密：for $i=1$ to n, do $\boldsymbol{Y}_i=E(\boldsymbol{K},\boldsymbol{B}(i)\oplus\boldsymbol{Y}_{i-1})$，最后一组的 $\boldsymbol{B}(n)$如不足 128 bit 长则需填零补齐；(4)提取 $\boldsymbol{Y}_n$ 的高有效位 8 个字节作为产生认证码的数据 $\boldsymbol{T}$: $\boldsymbol{T}=MSB_8(\boldsymbol{Y}_n)$; (5)构建 CTR 模式计数器生成函数并产生计数器 $\boldsymbol{A}_i$; (6)为 CTR 模式产生密钥流 $\boldsymbol{S}_i$: for $i=1$ to n, do $\boldsymbol{S}_i=E(\boldsymbol{K},\boldsymbol{A}_i)$; (7)按 CTR 模式产生密文 $\boldsymbol{E}_p$，即将消息分组与密钥流异或：$\boldsymbol{E}_p=\boldsymbol{P}\oplus(\boldsymbol{S}_1,\boldsymbol{S}_2,\cdots,\boldsymbol{S}_n)$; (8)产生认证码 $\boldsymbol{U}$:$\boldsymbol{U}=\boldsymbol{T}\oplus E(\boldsymbol{K},\boldsymbol{A}_0)$; (9)将密文 $\boldsymbol{E}_p$ 与认证码 $\boldsymbol{U}$ 串接，形成密文 $\boldsymbol{C}$ 输出：$\boldsymbol{C}=\boldsymbol{E}_p||\boldsymbol{U}$。

CCM 协议的消息解密流程包括数据的解密和认证码校验。接收方收到密文 $\boldsymbol{C}$ 后，将其拆分为 $\boldsymbol{E}_p$ 和 $\boldsymbol{U}$，首先利用共享密钥对密文 $\boldsymbol{C}$ 解密，得到有效负荷 $\boldsymbol{P}_{\rm rec}$，再通过 $\boldsymbol{T}_{\rm rec}=\boldsymbol{U}\oplus MSB_8(\boldsymbol{S}_0)$，计算出 $\boldsymbol{T}_{\rm rec}$；利用 CBC-MAC 模式对有效负荷 $\boldsymbol{P}_{\rm rec}$ 进行重新计算产生 $\boldsymbol{T}_{\rm calc}$，当 $\boldsymbol{T}_{\rm calc}=\boldsymbol{T}_{\rm rec}$ 时，表明该消息正确接收，否则丢弃该报文。CCM 协议的流程如图 1 所示。

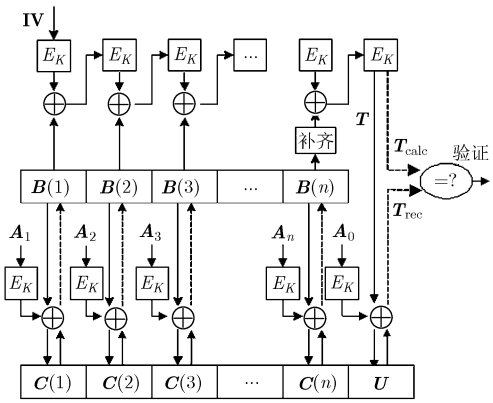

图 1 CCM 协议流程示意图

采用 AES 算法作为加密算法的 CCM 协议称为 AES-CCM 协议。AES-CCM 协议中的 CBC-MAC 和 CTR 两个模式共用了相同的加密算法、同一个密钥以及相同的子密钥流，本文在用硬件实现加速时，从节省面积消耗的角度出发，首先在两个工作模式间共享加密算法模块和扩展后的子密钥流，将并行的两个工作模式改为串行交叉进行；其次，可将 AES-CCM 协议分解为控制部分和计算密集部分，将控制部分交由软件实现，而采用扩展指令集的方法实现 AES-CCM 协议中计算密集的部分，可很好地解决效率与成本、功耗之间的矛盾。图 2 给出了 CCM 协议复用了 AES 加密模块后的结构示意图。

3 基于 S 盒复合域分解的扩展指令集实现

本研究在 x86 平台的 VC 环境下，用 C 语言实现 AES-CCM 协议并进行性能分析，可得 AES 算法占 AES-CCM 协议 92.1%的运算量。因此，用硬

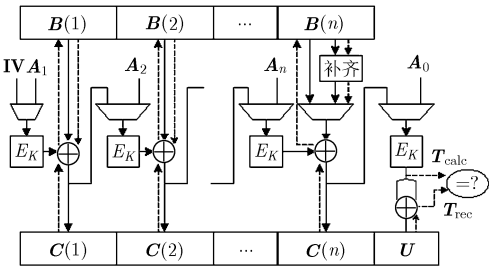

图 2 AES 模块复用示意图

件加速机制实现 AES-CCM 协议的关键是用硬件电路实现 AES 算法，而 AES 算法硬件实现的关键是 S 盒的实现方法[12,13]。

3.1 快速算法中 $\mathbf{FT_0}$ 的推导过程

将 AES 算法加密过程中轮操作的 128 bit 输入数据，以字节为单位排列成 4×4 的矩阵 $\boldsymbol{a}, \boldsymbol{a}_j$ 表示第 j 列元素，$\boldsymbol{a}_{i,j}$ 表示 $\boldsymbol{a}$ 中第 i 行第 j 列的字节，$S(\boldsymbol{a}_{i,j})$ 为输入字节 $\boldsymbol{a}_{i,j}$ 在 S 盒中得到的替换字节。AES 的快速算法将针对输入矩阵列元素 $\boldsymbol{a}_j$ 的 4 个轮函数操作，合并成对 FT 的查表操作，FT 由 256 个 32 bit 的表项组成，输入矩阵的 4 列数据对应 4 张不同的 FT。以 $\mathrm{FT}_j(\boldsymbol{a}_j)$ 表示第 j 列输入元素在 FT_j 中得到的替换值，则有

$$\left.\begin{aligned}
\mathrm{FT}_0(\boldsymbol{a}_0) &= [02\cdot S(\boldsymbol{a}_0), S(\boldsymbol{a}_0), S(\boldsymbol{a}_0), 03\cdot S(\boldsymbol{a}_0)]^{\mathrm{T}}\\
\mathrm{FT}_1(\boldsymbol{a}_1) &= [03\cdot S(\boldsymbol{a}_1), 02\cdot S(\boldsymbol{a}_1), S(\boldsymbol{a}_1), S(\boldsymbol{a}_1)]^{\mathrm{T}}\\
\mathrm{FT}_2(\boldsymbol{a}_2) &= [S(\boldsymbol{a}_2), 03\cdot S(\boldsymbol{a}_2), 02\cdot S(\boldsymbol{a}_2), S(\boldsymbol{a}_2)]^{\mathrm{T}}\\
\mathrm{FT}_3(\boldsymbol{a}_3) &= [S(\boldsymbol{a}_3), S(\boldsymbol{a}_3), 03\cdot S(\boldsymbol{a}_3), 02\cdot S(\boldsymbol{a}_3)]^{\mathrm{T}}
\end{aligned}\right\} \tag{1}$$

令 $\boldsymbol{e}$ 为128 bit以字节为单位排列为4×4的输出矩阵，$\boldsymbol{e}_j$ 为输出矩阵中的第 j 列，$\boldsymbol{k}_j$ 为本轮扩展密钥的第 j 列，则 $\boldsymbol{e}_j$ 与 $\boldsymbol{a}, \boldsymbol{k}_j$ 的关系为

$$\begin{aligned}
\boldsymbol{e}_j = \boldsymbol{k}_j &\oplus \mathrm{FT}_0(\boldsymbol{a}_{0,j\%4}) \oplus \mathrm{FT}_1(\boldsymbol{a}_{1,(j+1)\%4})\\
&\oplus \mathrm{FT}_2(\boldsymbol{a}_{2,(j+2)\%4}) \oplus \mathrm{FT}_3(\boldsymbol{a}_{3,(j+3)\%4})
\end{aligned} \tag{2}$$

图3给出了式(2)所对应的C语言代码。

```
#define AES_FROUND(e0,e1,e2,e3,a0,a1,a2,a3) {\
e0 = RK[0] \
    ^FT0[a0 >> 24 ] ^ FT1[a1 >> 16 ] ^ FT2[a2 >> 8]
    ^ FT3[a3];\
e1 = RK[1] \
    ^FT0[a1 >> 24 ] ^ FT1[a2 >> 16 ] ^FT2[a3>> 8]
    ^FT3[a0 ];\
e2 = RK[2] \
    ^FT0[a2 >> 24 ] ^FT1[a3 >> 16 ] ^ FT2[a0 >> 8]
    ^ FT3[a1 ];\
e3 = RK[3] \
    ^FT0[a3 >> 24 ] ^FT1[a0 >> 16 ] ^FT2[a1 >> 8]
    ^FT3[a2]; \
}
```

图3 AES快速算法中轮操作的C语言描述

由式(1)可见，4张FT的组成条项是相同的，其他3张FT可以通过对FT_0加上循环移位逻辑电路来产生。在末轮操作中，没有列混合函数，关键步骤是查询S盒，密钥扩展模块中的字节替换函数也使用了S盒，而S盒可以通过对FT_0的掩膜得到。因此，采用查表法的AES快速软件算法中，只需要存储1张容量为8 kbit的FT_0表。

3.2 $\mathbf{FT_0}$ 与 S 盒的关系

由于 WSN 的节点设备资源有限，8 kbit 的 FT_0 仍有必要进一步简化。从式(1)可得，FT 与 S 盒之间是基于有限域 $\mathrm{GF}(2^8)$的常量乘法关系。$\mathrm{GF}(2^8)$中的每一个元素都能够转化为 02 的不同幂次的和，用函数 xtime(b)来表示 $\mathrm{GF}(2^8)$上乘 02 的乘法，则式(1)可以转化为

$$\begin{aligned}
\mathrm{FT}_0(\boldsymbol{a}_0) &= [02\cdot S(\boldsymbol{a}_0), S(\boldsymbol{a}_0), S(\boldsymbol{a}_0), 03\cdot S(\boldsymbol{a}_0)]^{\mathrm{T}}\\
&= [\mathrm{xtime}(S(\boldsymbol{a}_0)), S(\boldsymbol{a}_0), S(\boldsymbol{a}_0),\\
&\quad (\mathrm{xtime}(S(\boldsymbol{a}_0)) \text{ xor } S(\boldsymbol{a}_0))]^{\mathrm{T}}
\end{aligned} \tag{3}$$

通过式(3)，列元素 $\boldsymbol{a}_0$ 所对应的 FT_0 值可从 S 盒中计算得到，对应输入矩阵列元素 $\boldsymbol{a}_1/\boldsymbol{a}_2/\boldsymbol{a}_3$ 的 $\mathrm{FT}_{1/2/3}$ 可通过对 FT_0 中元素的循环移位操作得到。xtime(b)函数可用简单的移位和异或运算来实现，其 VHDL 硬件描述语言代码如图 4 所示。

```
function xtime (b : std_logic_vector )
    return std_logic_vector is
  variable a : std_logic_vector (7 downto 0);
begin
  a := ( b(6 downto 4)                --bit(7 downto 5)
     & (b(3 downto 2) xor b(7))       --bit(4 downto 3)
     & b(1)                           --bit(2)
     & (b(0) xor b(7))                --bit(1)
     & b(7));                         --bit(0)
     return a;
end xtime;
```

图 4 xtime 函数的 VHDL 实现代码

3.3 S 盒的复合域计算

S 盒所需的存储空间仅为 2 kbit，但访问片上存储器内的 S 盒与用硬件电路直接计算 S 盒相比，速度上并不具备量优势，且消耗大量的芯片面积，因此，如果能用简单的硬件电路实现 S 盒的计算，可进一步降低节点设备的成本。S 盒是对输入数据在有限域$\mathrm{GF}(2^8)$上的乘法逆运算和对乘法逆运算的结果进行基于有限域 GF(2)的仿射变换两者构造而成，即 $S[i]=f(g(i))$，其中，$g(i)$ 表示元素 i 在有限域 $\mathrm{GF}(2^8)$上的求逆：$i \to i^{-1}$ in $\mathrm{GF}(2^8)$。而仿射变

化 f 只涉及到有限域加法，可以采用异或操作实现。对解密流程，逆 S 盒的运算步骤是，先对输入数据进行仿射变换，然后再对仿射变换的结果进行 $GF(2^8)$ 上的乘法逆运算，即：$S^{-1}[i]=g^{-1}(f^{-1}(i))=g(f^{-1}(i))$，其中，$f^{-1}$ 是 f 的逆变换。只要实现了 f，f^{-1} 和 g，即，只要实现 $GF(2^8)$ 上的乘法逆运算 $g(i)$，通过不同的线性运算则可分别得到 S 盒与逆 S 盒。

硬件实现 S 盒运算的方法主要有 4 种，即查表法，复合域变换，扩展的欧几里德算法及其变种和费马定理计算法的硬件实现方法。查表法易于实现，但芯片面积消耗大，扩展欧几里德法硬件实现复杂度较大，费马定理法中有平方乘方运算，效率较低，而复合域转换法相对简单。本文采取了基于复合域分解的方法实现 $GF(2^8)$ 的乘法求逆运算。当 $k=n\times m$ 时，称复合域 $GF((2^n)^m)$ 与 $GF(2^k)$ 是同构的，复合域可由低阶子域迭代构造而成，即将有限域 $GF(2^k)$ 求逆运算转换成 $GF((2^n)^m)$ 下的求逆操作。利用同构的复合域方法，可用结构简单的逻辑电路实现复合域算法。文献[12,13]阐述了从 $GF(2^8)$ 转换为 $GF((2^2)^2)^2$ 的方法。从电路复杂度和计算效率出发，本文将 $GF(2^8)$ 转换为 $GF(2^4)^2$，而把 $GF(2^4)$ 域上乘法求逆，直接用内存数组对应完成，降低了用硬件逻辑实现 $GF(2^4)$ 域上乘法求逆的电路复杂度。

3.4 NiosII 用户定制指令接口的实现

本研究以式(2)，式(3)为扩展指令的实现对象，式(2)表明了 AES 算法加密流程中的每一个 32 bit 的输出列元素都和 128 bit 的明文及密钥相关，而 NiosII 处理器的用户定制指令只提供 2 个 32 bit 的输入和 1 个 32 bit 的输出端口，因此，需要 4 条扩展指令来完成式(2)。

AES 算法中的末轮操作与前 9 轮操作类似，但其所需的 S 盒值需要从 FT 中掩模得到，并进行了额外的 24 个移位操作，因此需要单独定义 4 条针对末轮操作的扩展指令。密钥扩展中的字节替换与轮操作中的字节替换函数类似，但具体操作不同，也需要创建 4 条用于密钥扩展的扩展指令。所以，本文共用 12 条扩展指令来完成 AES-CCM 协议中的计算密集部分，这 12 条扩展指令的核心都是用复合域方法实现的 S 盒计算，在不考虑指令并行执行的前提下，这些扩展指令可以共用一个 S 盒复合域计算的硬件逻辑，从而将 12 条扩展指令集成到 1 个扩展指令硬件逻辑中，由 NiosII 处理器定制指令中的输入信号 n 来区别具体的指令。NiosII 处理器用户定制指令的详细定义方法请参见文献[14]。

在设计实现扩展指令的硬件逻辑电路并完成编译、仿真后，需要在 NiosII 处理器上层应用程序的头文件中按文献[14]中的规范要求定义扩展指令的调用接口，包括参数的数据类型、个数以及区分扩展指令的操作码等。图 5 为本方案中用于加密流程轮变换的 FT 查表扩展指令的操作码的定义及轮变换的宏定义，每条指令具有 2 个整型输入和 1 个整型输出。

```
#define FT0(A,B) __builtin_custom_inii(0,(A),(B))
#define FT1(A,B) __builtin_custom_inii(1,(A),(B))
#define FT2(A,B) __builtin_custom_inii(2,(A),(B))
#define FT3(A,B) __builtin_custom_inii(3,(A),(B))

#define AES_ROUND(X0,X1,X2,X3,Y0,Y1,Y2,Y3){\
 X0 = FT3(Y3,FT2(Y2,FT1(Y1,FT0(Y0,RK[0]))));\
 X1 = FT3(Y0,FT2(Y3,FT1(Y2,FT0(Y1,RK[1]))));\
 X2 = FT3(Y1,FT2(Y0,FT1(Y3,FT0(Y2,RK[2]))));\
 X3 = FT3(Y2,FT2(Y1,FT1(Y0,FT0(Y3,RK[3]))));}
```

图 5 轮操作扩展指令调用接口及宏函数定义

4 实验结果分析

本研究采用 VHDL 硬件描述语言进行了设计实现，硬件验证平台为基于 Altera CycloneII EP2C35F672C8 芯片的 FPGA 开发板，开发环境为 QuartusII 和 NiosII EDS 开发套件 9.1 版，所有的硬件方案统一采用 NiosII/s 标准型内核，测试程序采用 NIST 提供的用于 CCM 模式的标准测试向量[15]。

为与基于复合域计算 S 盒的扩展指令集的性能指标进行量化对比，本研究在同一硬件平台上分别实现了另外 4 种 AES-CCM 协议的运行方案。首先，用 C 语言以 NiosII 处理器上应用程序的形式在硬件平台上实现了以标准 AES 算法为核心的 AES-CCM 协议，以其在硬件平台上的运行性能数据为对比基准点。其次，用 AES 快速算法替换了上述代码中的 AES 标准算法，也在硬件平台上进行了测试运行，用以表明快速算法的结构优化对软件执行效率带来的影响。第 3 种方案为基于查表法实现 S 盒的协处理器方案，其将 AES-CCM 协议作为一个整体用硬件实现，采用图 1 描述 CTR 与 CBC-MAC 模式并行架构，通过 Avalon 总线接口挂载到 NiosII 处理器上。第 4 种方案为基于查找表实现 S 盒的扩展指令方案，该方案与基于复合域计算 S 盒的扩展指令方案思路相同，区别仅在于查表法实现的扩展指令中，每条指令是直接从 rom 中读取 FT_0 以获取输入数据的对应值。其顶层设计图如图 6 所示。

将加速比定义为：加速比=基准算法的执行时

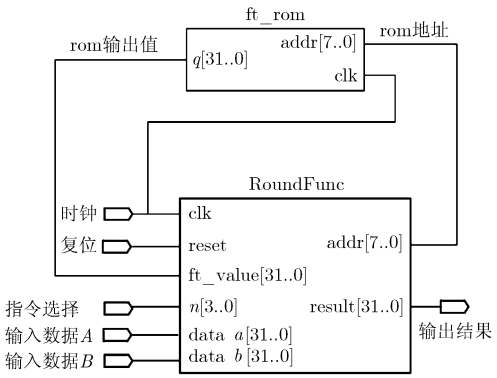

图 6 基于查表法的扩展指令顶层设计图

间/改进后算法的执行时间，表 1 给出了通过开发套件提供的性能分析工具，得到的各方案硬件逻辑所占用的 LE 资源数、片上存储器占用量、时钟周期数、加速比、吞吐量等数据，同时并给出了文献[5,6]中实现的 AES-CCM 协议的协处理器相关数据。

由表 1 所示数据可见，本研究实现的基于复合域实现 S 盒的 AES-CCM 扩展指令集方案，用结构简单的逻辑电路实现复合域算法的方法计算实现了 S 盒，可取消查表法中存储 S 盒所占用的片上存储器，且比基于查表法扩展指令集方案的加速比略有提高，LE 资源数略有下降。该方案比在 NiosII 上运行的纯软件 AES-CCM 协议可加速 174.6 倍，比经过结构优化的纯软件快速算法可加速 23.9 倍，在同样满足 WSN 节点设备吞吐量要求的情况下，本方案所消耗的硬件面积仅为协处理器方案的 9.5%，且未使用片上存储器，显著降低了产品成本和功耗。

5　结束语

IEEE802.15.4 标准的 MAC 层采用了 AES-CCM 协议作为其链路安全性保证，而 AES 算法计算量较大，在资源有限的 WSN 节点设备上，如用纯软件或硬件协处理器方法实现 AES-CCM 协议，存在安全性低、成本高、功耗高等的问题。本文用基于 NiosII 处理器的 12 条扩展指令实现了 AES-CCM 协议，在满足 WSN 应用所要求的吞吐量的条件下，显著降低了面积消耗，并达到了较高的加速比。同时，在无线通信领域的无线城域网(WMAN)标准 IEEE802.16e 和无线局域网(WLAN)标准 IEEE802.11i 中，其通信链路层也采用了 AES- CCM 协议作为通信安全保证，而 AES 算法及 S 盒更有着广阔的应用范围。本研究采用的扩展指令集方法，适用于对资源受限条件下细粒度任务的硬件加速，同时，对安全处理器、可信计算模块等设备中同类技术的实现都有积极的参考意义。

表 1 AES-CCM 各方案性能相关数据

方案	芯片	面积	片上存储器(kbit)	时钟周期数	加速比	吞吐量(bps)
基准算法	EP2C35	–	–	11737325	1.0	3.8 k
快速算法	EP2C35	–	–	1611204	7.3	27.8 k
协处理器法	EP2C35	2353LE	10	588	19961.4	1.5 G
查表法指令	EP2C35	233LE	8	68298	171.9	658.8 k
复合域法指令	EP2C35	223LE	0	67241	174.6	668.7 k
协处理器[5]	EP2C35	8438LE	64	–	–	688 M
协处理器[6]	xc5vlx50	2378LUT	360	–	–	1.63 G

参 考 文 献

[1] IEEE Computer Society. Standard for part 15.4: wireless medium access control (MAC) and physical layer (PHY) specification for low-rate wireless personal area networks (LRWPANs)[S]. IEEE Std802.15.4, 2003.

[2] National Institute of Standards and Technology (NIST). Federal information processing standards publication 197 (FIPS PUB 197): specification for the Advanced Encryption Standard(AES)[S]. NIST, 2001.

[3] Jonsson J. On the security of CTR+CBC_MAC[C]. Cryptography: 9th Annual International Workshop, SAC2002. Berlin, 2003: 76–93.

[4] Hamalainen P, Hannikainen M, and Hamalainen T D. Efficient hardware implementation of security processing for IEEE802.15.4 wireless networks[C]. 48th IEEE International Midwest Symposium on Circuits and Systems, Cincinnati, 2005: 484–487.

[5] Chakib A. New experimental results for AES-CCMP acceleration on Cyclone-II FPGA[J]. *International Journal of Computer Science and Network Security*, 2010, 10(4): 1–6.

[6] Algredo-Badillo I, Feregrino-Uribe C, Cumplido R, *et al.*. FPGA implementation cost and performance evaluation of the IEEE802.16e and IEEE802.11i security architectures based on AES-CCM[C]. 5th International Conference on

Electrical Engineering, Computing Science and Automatic Control, Mexico City, 2008: 304–309.

[7] 赵晶晶, 李丽, 潘红兵, 等. IEEE802.1AE 中 GCM 的高速硬件实现[J]. 电子与信息学报, 2010, 32(6): 1515–1519.
Zhao Jing-jing, Li Li, Pan Hong-bing, *et al.*. High-speed hardware implementation for GCM in IEEE802.1AE[J]. *Journal of Electronics & Information Technology*, 2010, 32(6): 1515–1519.

[8] Kumar M and Singhal A. Efficient implementation of Advanced Encryption Standard (AES) for ARM based platforms[C]. 1st International Conference on Recent Advances in Information Technology (RAIT), Dhanbad, 2012: 23–27.

[9] 夏辉, 贾智平, 张峰, 等. AES 专用指令处理器的研究与实现[J]. 计算机研究与发展, 2011, 48(8): 1554–1563.
Xia Hui, Jia Zhi-ping, Zhang Feng, *et al.*. The research and application of a specific instruction processor for AES [J]. *Journal of Computer Research and Development*, 2011, 48(8): 1554–1563.

[10] Bos J W, Özen O, and Stam M. Efficient hashing using the AES instruction set[C]. Lecture Notes in Computer Science, 2011, Volume 6917, Cryptographic Hardware and Embedded Systems-CHES, Springer, 2011: 507–522.

[11] Daemen J and Rijmen V. The Design of Rijndael: AES-The Advanced Encryption Standard [M]. Berlin: Springer, 2002: 58–62.

[12] 程桂花, 罗永龙, 齐学梅, 等. AES 算法中基于流水线的可逆 S 盒设计与实现[J]. 小型微型计算机系统, 2012, 33(3): 576–581.
Cheng Gui-hua, Luo Yong-long, Qi Xue-mei, *et al.*. Design and implementation of pipelining reversible S-BOX in AES algorithm[J]. *Journal of Chinese Computer Systems*, 2012, 33(3): 576–581.

[13] Wong M M, Wong M L D, Hijazin I, *et al.*. Composite field $GF(((2^2)^2)^2)$ AES S-box with direct computation in $GF(2^4)$ inversion[C]. 7th International Conference on Information Technology in Asia, Kuching Sarawak, 2011: 1–6.

[14] Altera Corporation. Nios II Custom Instruction User Guide[Z]. San Jose: Altera Corporation, 2008.

[15] National Institute of Standards and Technology(NIST). CCM Test Vectors [EB/OL]. http://csrc.nist.gov/groups/STM/cavp/index.html, 2011, 5.

封 斌： 男，1974 年生，博士生，研究方向为嵌入式系统、高性能计算.

齐德昱： 男，1959 年生，教授，博士生导师，研究方向为计算机系统结构、软件体系结构等.

韩海雯： 女，1973 年生，博士生，研究方向为计算机体系结构、云计算.