SM4 算法讲解

一、SM4 简介

SM4 是一种对称加密算法，与 AES 十分相似。我们之前写过 SM3 的加密算法，但 SM3 是一个单向的加密算法，只能加密不能解密，SM3 多用于生成数据摘要。SM4 与 SM3 有很大的区别，它们的关系并不是算法上的升级，而是分属于不同的应用场景。那么我们来看一下 SM4 的应用场景，加密算法在我们的常规思维里是对数据进行加密，然后去传输加密数据，然后接收端接收到加密数据进行解密得到原始数据。SM4 就是对数据进行加密的，它可以加密数据也可以解密数据，加密与解密的密钥是唯一的。

专业术语：

摘要：对一段不定长数据生成的一段固定的唯一校验码。

明文：原始数据，没有被加密的数据。

密文：加密后的数据。

对称加密：加密与解密使用相同的密码。

非对称加密：加密与解密使用不同的密码。

密钥：应用于对称加密的密码。

公钥：非对称加密中的可公开的密码。

私钥：非对称加密中不可公开的密码。

SM4 在加密过程中需要对数据进行分组，一般为 128bit 为一组。在加密的过程中，这 128bit 的数据要参与 32 次轮函数迭代运算，并且每一轮的密钥都不相同，需要使用密钥扩展函数生成每一轮加密的轮密钥。

二、SM4 原理讲解

如果大家把 SM3 搞清楚，并独立的使用代码实现出来，那么 SM4 对你来说就是小菜一碟。SM4 的运算主要涉及到：异或、S 盒变换、移位、反序，这些运算非常合适数字电路进行运算，大部分都是位运算，采用简单的门电路就可以实现，不会构成较长的组合逻辑。

在 SM4 运算的过程中也是以 Word（字）为单位的，我们 SM4 的输入是一个 128bit 的数据，可以分成 4 个字，分别为 X0;X1;X2;X3。然后我们利用这四个字进行 32 次迭代，迭代结果为 X4……X35。每做一次迭代都会生成一个字。最终我们取{X35,X34,X33,X32}作为我们输

入 128bit 的加密结果。

我们清楚了这个过程之后，就对算法整体有了一定的认识。那么接下来我们需要搞清楚，加密步骤、轮密钥生成步骤、解密步骤。

加密过程如图 1 所示。

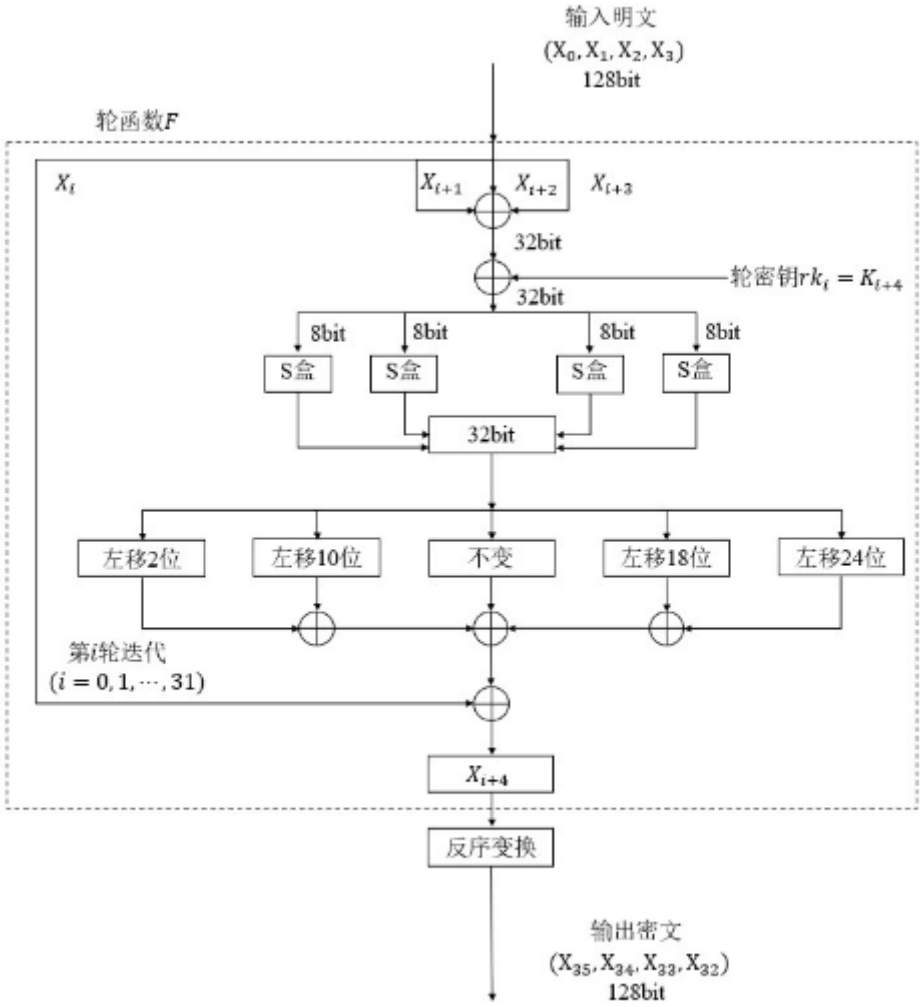

图 1

在加密的过程中的 S 盒如 X 所示。

表格 1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	**D6**	**90**	**E9**	**FE**	**CC**	**E1**	**3D**	**B7**	**16**	**B6**	**14**	**C2**	**28**	**FB**	**2C**	**05**
1	**2B**	**67**	**9A**	**76**	**2A**	**BE**	**04**	**C3**	**AA**	**44**	**13**	**26**	**49**	**86**	**06**	**99**
2	**9C**	**42**	**50**	**F4**	**91**	**EF**	**98**	**7A**	**33**	**54**	**0B**	**43**	**ED**	**CF**	**AC**	**62**
3	**E4**	**B3**	**1C**	**A9**	**C9**	**08**	**E8**	**95**	**80**	**DF**	**94**	**FA**	**75**	**8F**	**3F**	**A6**
4	**47**	**07**	**A7**	**FC**	**F3**	**73**	**17**	**BA**	**83**	**59**	**3C**	**19**	**E6**	**85**	**4F**	**A8**
5	**68**	**6B**	**81**	**B2**	**71**	**64**	**DA**	**8B**	**F8**	**EB**	**0F**	**4B**	**70**	**56**	**9D**	**35**
6	**1E**	**24**	**0E**	**5E**	**63**	**58**	**D1**	**A2**	**25**	**22**	**7C**	**3B**	**01**	**21**	**78**	**87**
7	**D4**	**00**	**46**	**57**	**9F**	**D3**	**27**	**52**	**4C**	**36**	**02**	**E7**	**A0**	**C4**	**C8**	**9E**
8	**EA**	**BF**	**8A**	**D2**	**40**	**C7**	**38**	**B5**	**A3**	**F7**	**F2**	**CE**	**F9**	**61**	**15**	**A1**
9	**E0**	**AE**	**5D**	**A4**	**9B**	**34**	**1A**	**55**	**AD**	**93**	**32**	**30**	**F5**	**8C**	**B1**	**E3**
A	**1D**	**F6**	**E2**	**2E**	**82**	**66**	**CA**	**60**	**C0**	**29**	**23**	**AB**	**0D**	**53**	**4E**	**6F**
B	**D5**	**DB**	**37**	**45**	**DE**	**FD**	**8E**	**2F**	**03**	**FF**	**6A**	**72**	**6D**	**6C**	**5B**	**51**
C	**8D**	**1B**	**AF**	**92**	**BB**	**DD**	**BC**	**7F**	**11**	**D9**	**5C**	**41**	**1F**	**10**	**5A**	**D8**
D	**0A**	**C1**	**31**	**88**	**A5**	**CD**	**7B**	**BD**	**2D**	**74**	**D0**	**12**	**B8**	**E5**	**B4**	**B0**
E	**89**	**69**	**97**	**4A**	**0C**	**96**	**77**	**7E**	**65**	**B9**	**F1**	**09**	**C5**	**6E**	**C6**	**84**
F	**18**	**F0**	**7D**	**EC**	**3A**	**DC**	**4D**	**20**	**79**	**EE**	**5F**	**3E**	**D7**	**CB**	**39**	**48**

S 盒变化其实就一个将输入的 8bit 数据拆分成高 4bit 与低 4bit，然后令高 4bit 为行数，令低 4bit 为列数去 S 盒的表格中查询到对应的数值，然后用这个数据替换原数值，此部分是算法非线性的体现。

那么我们只要将上面的流程图搞懂，我们就可以对输入的 128bit 进行加密了。但其中还有一个未知数，就是 rk（i）。我们在图 1 中可以看到 rk（i） = K（i + 4）。那么 K（i + 4）又等于什么呢？

这个就涉及到轮密钥扩展了。轮密钥扩展步骤如图 2 所示。在图 2 中，我们可以看到我们需要设置一个初始的密钥。然后将这个密钥输入进行迭代 32 次，每迭代一次产生的结果为 K（i + 4）。也就是说我们密钥扩展产生的结果要提供给加密的迭代函数使用。在密钥扩展中，我们有两个参数：系统参数 FK;固定参数 CK（i）。

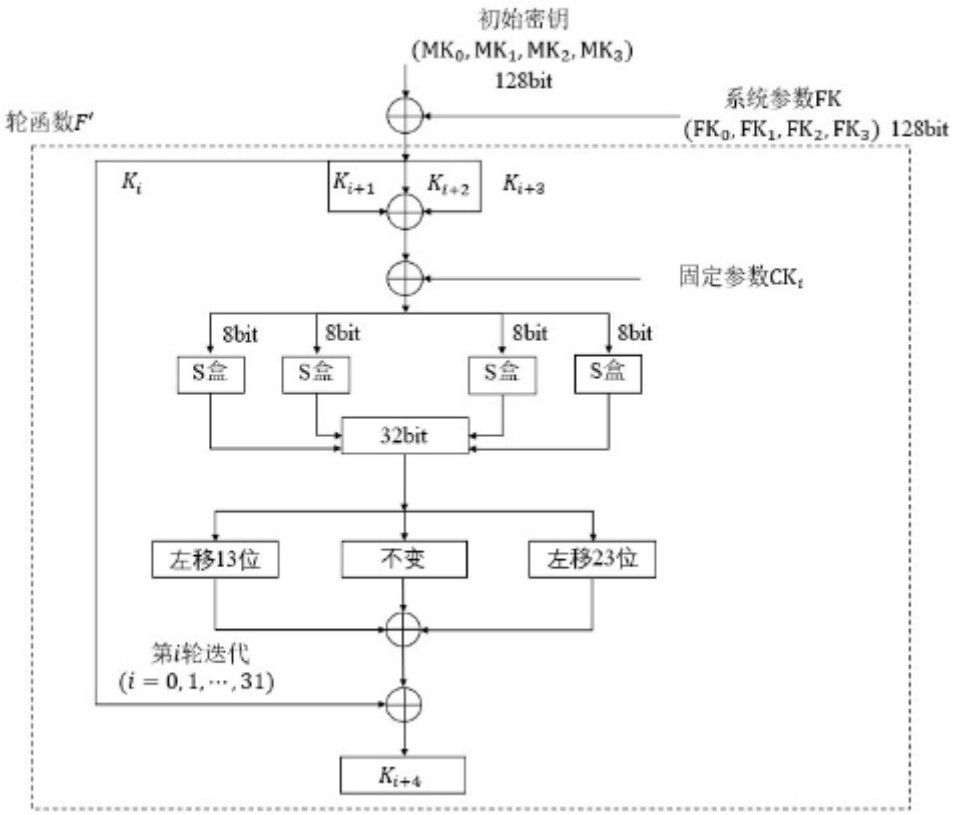

图 2

系统参数 FK 为 128bit，也以字为单位进行划分。FK = {FK0,FK1,FK2,FK3}。数值分别为：

A3B1BAC6;56AA3350;677D9197;B27022DC;

固定参数一共有 32 个字：

00070E15;1C232A31;383F464D;545B6269;70777E85;8C939AA1;A8AFB6BD;C4CBD2D9;

E0E7EEF5;DC030A11;181D262D;343B4249;50575E65;6C737A81;888F969D;A4ABB2B9;

C0C7CED5;DCE3EAF1;F8FF060D;141B2229;30373E45;4C535A61;686F767D;848B9299;

A0A7AEB5;BCC3CAD1;D8DFE6ED;F4FB0209;10171E25;2C333A41;484F565D;646B7279;

我们现在已经非常清楚加密过程与轮密钥生成过程。那么我们接下来看一下解密的过程。解密过程与加密过程大致相同。首先我们都需要产生轮密钥。但在轮密钥使用顺序上是不相同的。在解密过程中 rk（i） = K（35 - i）。最终将生成的数据以字为单位进行倒序即可得到明文。

与 SM3 相比，SM4 的算法迭代次数少，运算更加清晰。那么我们在 Verilog 代码设计时难度就会减少很多，但我们在做 SM4 算法时需使用流水线完成算法的设计。因为此算法常

用于对大量的数据流进行加密，这就要求我们的 SM4 算法可以 pipe-line 处理。

三、SM4 实现思路

略。

四、SM4 代码

详见代码工程。