SM3 算法讲解

一、SM3 简介

SM3 是我国自主研发的杂凑算法，其主要应用于：数字签名、身份认证、电子交易、区块链等等。SM3 于 SHA-256 较为相似，它们的安全性也不相上下。SM3 的输入可以为任意长度的数值，但输出是固定的 256bit 的哈希值。目前我国公开的商业加密算法有:SM2;SM3;SM4。详细可参考《GBT 32905-2016 信息安全技术 SM3 密码杂凑算法》和《GMT 0004-2012 SM3 密码杂凑算法》。

二、SM3 原理讲解

SM3 加密的是不可逆的，无法通过加密后的摘要恢复出原始数据，这也是非常重要的一个点。此文档意在为使用硬件描述语言实现数电逻辑的 SM3 算法，如大家需别的语言可根据原理讲解自行实现。

SM3 运算主要分为 4 个步骤：

1) 消息填充
2) 消息扩展
3) 迭代压缩
4) 结果输出

那么接下来让我深入理解一下每个步骤需要做的事情。首先我们要知道 SM3 可以输入任意长度的数值。那么我们在运算时需要将任意长度的输入数值扩展成一个有规律长度的数值，在 SM3 中的扩展规律为：（输入 bit 数量 + 1 + 64 + K）mod 512 = 0。K 为填充的 bit 数量。

我们来看一个例子：

输入一个 256bit 的数值，则(256 + 1 + 64 + K) mod512 = 0

则:K = 191;

那么这个过程就是我们第一步------消息填充。

消息填充需要将我们输入的信息，现在尾端补充 1bit 的数值 1。然后在补充 K 个 bit 的数值 0。最终使我们的长度满足（输入 bit 数量 + 1 + K）mod 512 = 448；然后再尾端补充 64bit 的数据长度的二进制。最终我们得到一个长度为 512bit 的正整数倍的数据。

我们将这个数据简称 B。我们将 B 以 512bit 为一组进行分组，分组成 B(0);B(1).....B(n)。n = （输入 bit 数量 + 1 + K + 64）/512。然后我们再将每一组 B（x）里的数据以 word 的单位进行划分（word 为字，一个字 = 4 个字节）。B（x） = W0,W1,.....W15;每组一共包含 16 个字。

第二步我们要进行消息扩展。在第一步时我们得到了 16 个字，但这 16 个字不足我们下面的计算过程，因此我们在此过程中需要将 16 个字扩展为 132 个字。扩展算法如下所示。

b) FOR j=16 TO 67

$$W_j \leftarrow P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6}$$

ENDFOR

c) FOR j=0 TO 63

$$W'_j = W_j \oplus W_{j+4}$$

ENDFOR

图 1

通过上面的算法我们将得到 132 个字。这 132 个字将全部参与第三步的迭代计算。值得注意的是，在扩展字时，每个字要转换为大端存储（将最低的字节放到最高字节位）。

第三步迭代计算是此算法的核心部分。我们需要在此过程中执行大量的循环迭代，那么我们都知道循环迭代在 CPU 中是非常消耗时间的。那么我们在 FPGA 的设计中需要充分的考虑并行化计算。

此部分的一些固定参数与公式：

初始值：

$$IV = 7380166f\ 4914b2b9\ 172442d7\ da8a0600\ a96f30bc\ 163138aa\ e38dee4d\ b0fb0e4e$$

常量：

$$T_j = \begin{cases} 79cc4519 & 0 \le j \le 15 \\ 7a879d8a & 16 \le j \le 63 \end{cases}$$

布尔函数：

$$FF_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) & 16 \le j \le 63 \end{cases}$$

$$GG_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \wedge Y) \vee (\neg X \wedge Z) & 16 \le j \le 63 \end{cases}$$

式中X, Y, Z为字。

置换函数：

$$P_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17)$$

$$P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$$

式中X为字。

图 2

我们必须要将上面的固定参数与公式理解清楚，不然后面的运算过程就会云里雾里，大家一定要掌握好上面的内容再向下看。

那么下面我们需要进行迭代运算。迭代运算需要给压缩函数迭代 64 次。压缩函数为：

$$V^{i+1} = CF(V^{(i)}, B^{(i)}), 0 \leq i \leq n-1$$

图 3

B（i）为我们在第一步时分的组

V（i）为一次 CF(压缩函数)算法的哈希值

从此公式中我们可以看出，我们有 n 个 B 的分组就需要执行 n 次压缩函数，即进行 64 次迭代。并将每一次的迭代结果参与到下一次运算中。然后将最后一次迭代的结果与上一次迭代的结果进行异或就是得到 SM3 的输出结果了。

那么接下来我们看一下 CF(压缩函数)运算的过程：

ABCDEFGH 为输入的哈希值的字。A 代表最高的字，以此类推。

SS1;SS2;TT1;TT2 即为中间变量

$$ABCDEFGH \leftarrow V^{(i)}$$

FOR j=0 TO 63

$$SS1 \leftarrow ((A \lll 12) + E + (T_j \lll j)) \lll 7$$

$$SS2 \leftarrow SS1 \oplus (A \lll 12)$$

$$TT1 \leftarrow FF_j(A,B,C) + D + SS2 + W_j'$$

$$TT2 \leftarrow (E,F,G) + H + SS1 + W_j$$

$$D \leftarrow C$$

$$C \leftarrow B \lll 9$$

$$B \leftarrow A$$

$$A \leftarrow TT1$$

$$H \leftarrow G$$

$$G \leftarrow F \lll 19$$

$$F \leftarrow E$$

$$E \leftarrow P_0(TT2)$$

ENDFOR

$$V^{i+1} \leftarrow ABCDEFGH \oplus V^{(i)}$$

图 4

根据以上过程即可完成对 SM3 的加密运行，并得到最终的摘要数值。

三、SM3 实现思路

以上过程我们最终要使用 verilog 实现。我们在设计时需充分考虑迭代运算的并行化，并要防止组合逻辑过长造成的时序违例。

四、SM3 代码

详细请见文件夹中的 verilog 代码。