中国集成电路 CIC
China Integrated Circuit

SHA-256哈希运算单元的硬件优化实现

汤煜,翁秀玲,王云峰
(厦门大学信息科学与技术学院,福建 厦门,361005)

摘要 在信息安全领域,SHA-256 算法被广泛应用于数字签名、身份认证、消息完整性验证等方面。为满足应用对 SHA-256 算法运算速度上的要求,本文利用流水线的设计思想,提出了一种四级流水的哈希计算结构,提高了运算速度,增大了吞吐率。最终,本文在 Virtex-7 系列 FPGA 开发板上实现了 SHA-256 哈希运算单元,最高工作频率可达到 319MHz,吞吐率达到 2405Gbps,吞吐率面积比达到 2.529Mbps/Slice。

关键词 哈希算法 流水线

Hardware optimization of SHA-256 hash operation unit

TANG Yu,WENG Xiu-ling, WANG Yun-feng
(School of information science and engineering ,Xiamen University,361005 Xiamen)

Abstract: SHA-256 algorithm is widely used in digital signature, identity authentication, message integrity verification etc. In order to speed up SHA-256 algorithm computing, a four-stage pipeline processing for SHA-256 is proposed. Implemented on virtex-7 FPGA series, the proposed SHA-256 can work at 319MHz, and archive a throughput of 2405Gbps.The ratio of throughput/area is 2.529Mbps/Slice.

Key words: HASH algorithm; pipeline

随着社会信息化的高速发展，信息安全成为人们所关注的焦点[1]。鉴别信息的来源及维护信息的完整、保证信息不被篡改至关重要。SHA-256 算法属于哈希算法的一种,用来生成信息摘要,具有高抗碰撞能力、不可逆性等特点[2]。在信息传递中,结合其他密码技术,能够保证信息完整性及不可否认性,被广泛地应用在数字签名、身份认证、消息完整性验证以及互联网安全协议等方面[3]。由于 SHA-256 算法

应用广，硬件实现 SHA-256 算法速度快且抗攻击性能力强，因此硬件实现 SHA-256 算法已变得越来越重要[4]。

文献[5]采用预计算与流水线相结合的方式实现 SHA-256 算法的哈希运算，利用三级流水结构，虽然在一定程度上减少了关键路径的长度，但是工作频率并没有明显提高，硬件资源使用也较多；文献[6] 采用两级流水结构实现 SHA-256 算法的哈希运算，并对 FPGA 内部编程资源优化以减少资源占用，虽然资源占用相对较少，但是工作频率和数据吞吐率没有提高；文献[7]采用四级流水的方式，通过减少关键路径的计算长度以达到提高工作频率的目的，虽然工作频率提高很多，但是面积占用相对也大；文献[8]采用流水和展开的方式，其中利用三级流水结构，使得关键路径延迟减少，提高了工作频率，但占用资源多；而利用展开的方式并没有使关键路径计算延时减少，虽然资源较少，但工作频率很低；文献[9]通过划分关键路径的计算长度，提出了一种三级流水线结构，虽然提高了工作频率，但是面积占用多。本文在文献[5]提出的三级流水结构上增加一级流水，以进一步减少关键路径的计算长度，获得高频率和高吞吐率，实现良好的吞吐率面积比。

1 SHA-256 算法

SHA-256 算法是指将长度不大于 2^{64} 比特的信息通过循环计算出其 256 位的二进制值哈希值[10]。哈希运算单元每次处理 512 位信息。该算法处理过程分为两个阶段：预处理和哈希计算阶段。其中预处理包括：

（1）对信息进行分割填充，生成 N 个 512 位的信息块，此过程一般由系统软件实现；

（2）初始化算法所制定的 256 位哈希值 H^0。

哈希计算阶段是 N 个 512 位的信息块 $M^{(i)}$ 进行循环计算处理，最终生成该消息的哈希值。具体过程说明如下：

For i=1 to N:

{

1. 将信息块 $M^{(i)}$ 分成 16 个 32 位的数据 $M_t^{(i)}$，（0≤t≤15）准备序列{W_t}：

$$W_t=\begin{cases} M_1^{(j)} & 0\leqslant t\leqslant 15 \\ \sigma_1^{\{256\}}(W_{t-2})+W_{t-7}+\sigma_0^{\{256\}}(W_{t-15})+W_{t-16} & 16\leqslant t\leqslant 63 \end{cases}$$

2. 将第（i-1）轮的哈希值 H^{i-1} 分成 8 个 32 位的数据 H_j^{i-1}（0≤j≤7），初始化 8 个 32 位的变量，a_0,b_0, c_0, d_0, e_0, f_0, g_0, h_0：

$a_0 = H_0^{(i-1)}$

$b_0 = H_1^{(i-1)}$

$c_0 = H_2^{(i-1)}$

$d_0 = H_3^{(i-1)}$

$e_0 = H_4^{(i-1)}$

$f_0 = H_5^{(i-1)}$

$g_0 = H_6^{(i-1)}$

$h_0 = H_7^{(i-1)}$

3. For t=0 to 63：

{

执行如图 1 所示循环，其中 K_t 为算法所制定的固定值。

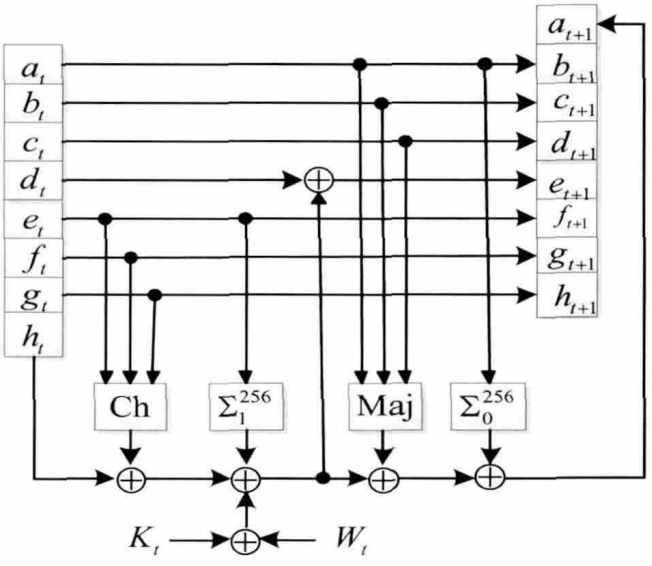

图 1 一次 SHA-256 计算内部循环

$T_t^{'} = h_t + \sum_1^{\{256\}}(e_t) + Ch(e_t, f_t, g_t) + K_t + W_t$

$T_t^{''} = \sum_0^{\{256\}}(a_t) + Maj(a_t, b_t, c_t)$

$h_{t+1} = g_t$

$g_{t+1} = f_t$

$f_{t+1} = e_t$

$e_{t+1} = d_t + T_t^{'}$

$d_{t+1} = c_t$

$c_{t+1} = b_t$

$b_{t+1} = a_t$

$a_{t+1} = T_t^{'} + T_t^{''}$

}

4. 完成 64 轮循环迭代计算后，将 a_{64}~h_{64} 与 H_j^{i-1} 做模 2^{32} 的加法运算后，得到第 i 轮的哈希值 H_j^i，如下所示：

$H_0^{(i)} = a_{64} + H_0^{i-1}$

$H_1^{(i)} = b_{64} + H_1^{i-1}$

$H_2^{(i)} = c_{64} + H_2^{i-1}$

$H_3^{(i)} = d_{64} + H_3^{i-1}$

$H_4^{(i)} = e_{64} + H_4^{i-1}$

$H_5^{(i)} = f_{64} + H_5^{i-1}$

$H_6^{(i)} = g_{64} + H_6^{i-1}$

$H_7^{(i)} = h_{64} + H_7^{i-1}$

}

外层循环迭代重复进行 N 次后，所得到的结果就是输入消息 M 的 256 位的哈希摘要。其中，‖ 是连接符号。

$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$

算法中，

$Ch(x,y,z) = (x \wedge y) \oplus (\sim x \wedge z)$

$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$

$\sum_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$

$\sum_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$

$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$

$\sigma_1^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$

其中，$\oplus$ 比特异或，$\wedge$ 是比特与，~ 是比特取反，$ROTR^n(x)$ 是循环右移 n 位，$SHR^n(x)$ 是逻辑右移 n 位。

2 SHA-256 算法优化设计

2.1 算法分解

由 SHA-256 算法的计算可知，哈希运算硬件实现的关键路径在于计算 a_t，根据内循环过程可得：

$$e_t = \sum_1^{\{256\}}(e_{t-1}) + Ch(e_{t-1}, f_{t-1}, g_{t-1}) + h_{t-1} + W_{t-1} + K_{t-1} + d_{t-1} \quad (1)$$

$$a_t = \sum_1^{\{256\}}(e_{t-1}) + Ch(e_{t-1}, f_{t-1}, g_{t-1}) + h_{t-1} + W_{t-1} + K_{t-1} + \sum_1^{\{256\}}(a_{t-1}) + Maj(a_{t-1}, b_{t-1}, c_{t-1}) \quad (2)$$

由于 $d_{t-1}=c_{t-2}$，因此式 (1) 可变形为：

$$e_t = \sum_1^{\{256\}}(e_{t-1}) + Ch(e_{t-1}, f_{t-1}, g_{t-1}) + h_{t-1} + W_{t-1} + K_{t-1} + c_{t-2} \quad (3)$$

根据上述式子，本文设计了 a_t 的四级流水计算过程，如图 2 所示。

具体如下：

在第一级流水中，令

$$\sigma_{t-2} = W_{t-1} + K_{t-1} + h_{t-1} \quad (4)$$

在第二级流水中，令

$$U_{t-1} = \sigma_{t-2} \quad (5)$$

$$\tau_{t-1} = \sigma_{t-2} + c_{t-2} \quad (6)$$

在第三级流水中，令

$$e_t = \sum_1^{\{256\}}(e_{t-1}) + Ch(e_{t-1}, f_{t-1}, g_{t-1}) + \tau_{t-1} \quad (7)$$

$$\lambda_{t-1} = \sum_1^{\{256\}}(e_{t-1}) + Ch(e_{t-1}, f_{t-1}, g_{t-1}) + U_{t-1} \quad (8)$$

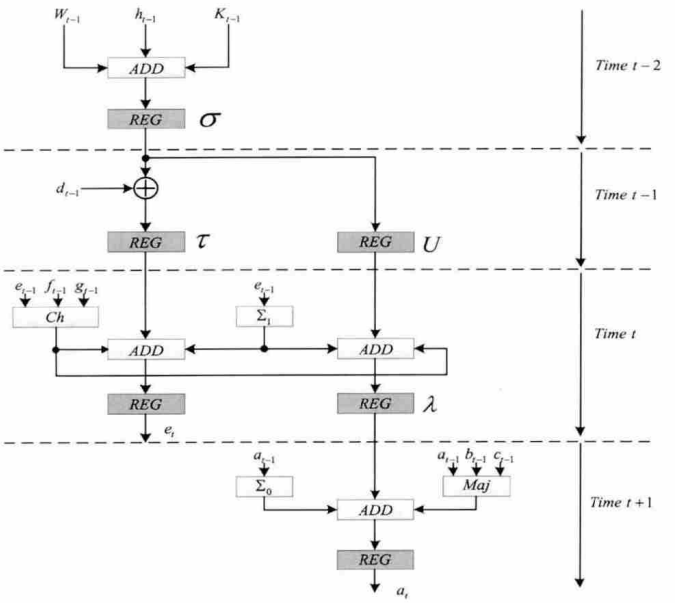

图 2 四级流水结构图

在第四级流水中，令

$$a_t=\sum_1^{\{256\}}(a_{t-1})+Maj(a_{t-1},b_{t-1},c_{t-1})+\lambda_{t-1} \quad (9)$$

对于 a_t 和 e_t 的计算，每级流水最多包含 2 次加法，与文献[5]采用的三级流水的处理方法和文献[6]采用流水线操作及数据并行处理来实现 SHA-256 算法相比，关键路径的计算长度由 4 次加法减少到 2 次加法，计算长度接近减少一半；与同样采用四级流水设计的文献[7]相比，在资源占用上，本文的四级流水设计比其四级流水设计少用一组 32 位寄存器和两组 32 位加法器。

由于流水线的引入，需要 4 个附加的时钟周期来处理，因此每次循环计算操作由 64 轮变成 68 轮。具体的运算步骤如表 1 所示。

表 1 具体计算过程

No.	步骤
1	$\sigma_1=W_0+K_0+h_0$ U_1 保持 τ_1 保持 λ_1 保持
2	$\sigma_2=W_1+K_1+g_0$ $U_2=\sigma_1$ $\tau_2=\sigma_1+d_0$ λ_2 保持
3	$\sigma_3=W_2+K_2+f_0$ $U_3=\sigma_2$ $\tau_3=\sigma_2+c_0$ $\lambda_3=\sum_1(e_0)+Ch(e_0,f_0,g_0)+U_2$ $e_1=\sum_1(e_0)+Ch(e_0,f_0,g_0)+\tau_2$ $f_1=e_0$ $g_1=f_0$ $h_1=g_0$
4	$\sigma_4=W_3+K_3+f_1$ $U_4=\sigma_3$ $\tau_4=\sigma_3+b_0$ $\lambda_4=\sum_1(e_1)+Ch(e_1,f_1,g_1)+U_3$ $e_2=\sum_1(e_1)+Ch(e_1,f_1,g_1)+\tau_3$ $f_2=e_1$ $g_2=f_1$ $h_2=g_1$ $a_1=\sum_0(a_0)+Ch(a_0,b_0,c_0)+\lambda_3$ $b_1=a_0$ $c_1=b_0$ $d_1=c_0$
5	$\sigma_5=W_4+K_4+f_2$ $U_5=\sigma_4$ $\tau_5=\sigma_4+b_1$ $\lambda_5=\sum_1(e_2)+Ch(e_2,f_2,g_2)+U_4$ $e_3=\sum_1(e_2)+Ch(e_2,f_2,g_2)+\tau_4$ $f_3=e_2$ $g_3=f_2$ $h_3=g_2$ $a_2=\sum_0(a_1)+Ch(a_1,b_1,c_1)+\lambda_4$ $b_2=a_1$ $c_2=b_1$ $d_2=c_1$
…	……
64	$\sigma_{64}=W_{63}+K_{63}+f_{61}$ $U_{64}=\sigma_{63}$ $\tau_{64}=\sigma_{63}+b_{60}$ $\lambda_{64}=\sum_1(e_{61})+Ch(e_{61},f_{61},g_{61})+U_{63}$ $e_{62}=\sum_1(e_{61})+Ch(e_{61},f_{61},g_{61})+\tau_{63}$ $f_{62}=e_{61}$ $g_{62}=f_{61}$ $h_{62}=g_{61}$ $a_{61}=\sum_0(a_{60})+Ch(a_{60},b_{60},c_{60})+\lambda_{63}$ $b_{61}=a_{60}$ $c_{61}=b_{60}$ $d_{61}=c_{60}$
65	σ_{65} 保持 $U_{65}=\sigma_{64}$ $\tau_{65}=\sigma_{64}+b_{61}$ $\lambda_{65}=\sum_1(e_{62})+Ch(e_{62},f_{62},g_{62})+U_{64}$ $e_{63}=\sum_1(e_{62})+Ch(e_{62},f_{62},g_{62})+\tau_{64}$ $f_{63}=e_{62}$ $g_{63}=f_{62}$ $h_{63}=g_{62}$ $a_{62}=\sum_0(a_{61})+Ch(a_{61},b_{61},c_{61})+\lambda_{64}$ $b_{62}=a_{61}$ $c_{62}=b_{61}$ $d_{62}=c_{61}$
66	σ_{66} 保持 U_{66} 保持 τ_{66} $\lambda_{66}=\sum_1(e_{63})+Ch(e_{63},f_{63},g_{63})+U_{65}$ $e_{64}=\sum_1(e_{63})+Ch(e_{63},f_{63},g_{63})+\tau_{65}$ $f_{64}=e_{63}$ $g_{64}=f_{63}$ $h_{64}=g_{63}$ $a_{63}=\sum_0(a_{62})+Ch(a_{62},b_{62},c_{62})+\lambda_{65}$ $b_{63}=a_{62}$ $c_{63}=b_{62}$ $d_{63}=c_{62}$
67	σ_{67} 保持 U_{67} 保持 τ_{67} 保持 λ_{67} 保持 $a_{64}=\sum_0(a_{63})+Ch(a_{63},b_{63},c_{63})+\lambda_{66}$ $b_{64}=a_{63}$ $c_{64}=b_{63}$ $d_{64}=c_{63}$
68	σ_{68} 保持 U_{68} 保持 τ_{68} 保持 λ_{68} 保持 $H_0^i=a_{64}+H_0^{i-1}$ $H_1^i=b_{64}+H_1^{i-1}$ $H_2^i=c_{64}+H_2^{i-1}$ $H_3^i=d_{64}+H_3^{i-1}$ $H_4^i=e_{64}+H_4^{i-1}$ $H_5^i=f_{64}+H_5^{i-1}$ $H_6^i=g_{64}+H_6^{i-1}$ $H_7^i=h_{64}+H_7^{i-1}$

2.2 SHA-256 哈希运算四级流水硬件实现

根据哈希运算四级流水的划分，引入寄存器 σ、U、τ、λ 四个 32 位寄存器，用来保存中间计算结果，则循环计算单元硬件实现框图如图 3 所示。

图 3 中，a、b、c、d、e、f、g、h 是 8 个 32 位寄存器，是进行循环计算的工作变量；Sel_hgf 和 Sel_bcd 是三选一选择器；ADD1、ADD2、ADD3 及 ADD5 代表进行两级加法操作，ADD4 则表示一级加法操作。整个计算过程共需要进行 68 轮。

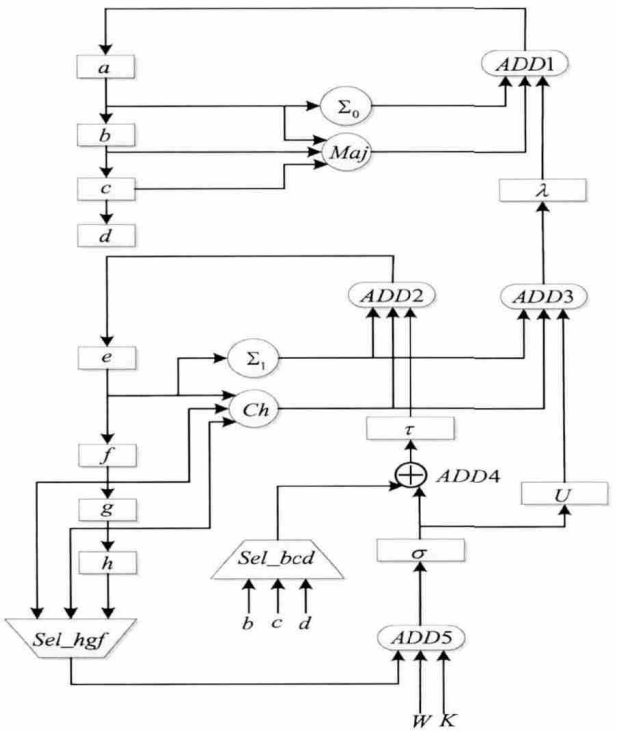

图 3 循环计算硬件结构图

3 综合与性能分析

本文所设计的 SHA-256 算法在不同的 FPGA 器件上的综合结果及相应的性能参数分别如表 2、表 3 和表 4 所示。其中归一化比较给出了单位面积所贡献的数据吞吐量，使各设计的比较更为直观合理。

通过对比，在相同的器件上，与文献[7]相比，本文设计的实现方式无论是在工作频率还是资源占用上都要更优；与文献[6]相比，本文在 Spartan3A 和 Virtex-4 器件上面积占用相对较多，但是工作频率和吞吐率都比文献[6]高很多；而在 Virtex-5 器件上，本文设计的实现方式无论是在工作频率还是资源占用上都要比文献[6]更优，所实现的吞吐率面积比文献[6]高 28.7%；与文献[5]相比，本文实现频率的更高，同时相应的吞吐率也比其高 28.6%；本文的设计

表 2 SHA-256 在 Virtex-5 等器件上的综合结果与对比

文献	FPGA 型号	频率 MHz	面积 slices	吞吐率 Mbps	归一化 Mbps/slice
[7]	Spartan3A	81	1214	610	0.503
	Virtex-4	128	1123	9601	0.856
	Virtex-5	224	993	1683	1.695
	Virtex-6	269	993	2009	2.023
	Virtex-7	316	993	2378	2.395
[6]	Spartan3A	82	795	615	0.773
	Virtex4	157	842	1178	1.401
	Virtex-5	184	973	1388	1.427
本文	Spartan3A	88	1131	665	0.588
	Virtex-4	181	1076	1363	1.266
	Virtex-5	232	951	1747	1.837
	Virtex-6	272	951	2080	2.187
	Virtex-7	319	951	2405	2.529

表 3 SHA-256 算法在 Virtex-2P 上的综合结果与对比

文献	FPGA 型号		频率 MHz	面积 slices	吞吐率 Mbps	归一化 Mbps/slice
[5]	SHA-256a	Xc2vp-7	104	1125	819	0.728
	SHA-256b		115	1149	902	0.785
	SHA-256c		110	1187	868	0.731
	SHA-256		115	1274	909	0.713.
本文	Xc2vp-7		155	1125	1170	1.04

表 4 SHA-256 算法在 Virtex-2 上的综合结果与对比

文献	FPGA 型号		频率 MHz	面积 slices	吞吐率 Mbps	归一化 Mbps/slice
[8]	Basic	Xc2v 2000	133	1373	1009	0.735
	unrolled		74	2032	997	0.491
本文	Xc2v2000		140	1128	1056	0.936

在 Virtex-2 器件上实现最高工作频率为 140MHz，吞吐率达到 1056Mbps，比文献[3]最优的实现要快 5.42%，面积也相对减少 17.8%，吞吐率面积比也要高 27.3%。

4 总结

本文通过对 SHA-256 算法进行硬件优化设计，一定程度上减少了面积占用，提高了工作频率，增大了吞吐率，进而提升了 SHA-256 算法性能。其中，本文实现 SHA-256 算法在 FPGA 开发板的 Virtex-7 系列上进行综合，最高工作频率可达到 319MHz，吞吐率达到 2405Gbps，吞吐率面积比达到 2.529Mbps/Slice。

参考文献

[1] Liu Y, Wu L, Niu Y, et al. A High-Speed SHA-1 IP Core for 10 Gbps Ethernet Security Processor; proceedings of the Computational Intelligence and Security (CIS), 2012 Eighth International Conference on, F, 2012 [C].

[2] Hua Z, Qiao L. Hardware design for SHA-1 based on FPGA; proceedings of the Electronics, Communications and Control (ICECC), 2011 International Conference on, F, 2011 [C].

[3] 杜江, 郭圣彬. 基于 SHA-256 的磁盘复制审计系统设计与实现 [J]. 计算机测量与控制, 2010, (2): 411-3.

[4] Juliato m, Gebotys C. A Quantitative Analysis of a Novel SEU-Resistant SHA-2 and HMAC Architecture for Space Missions Security [J]. Aerospace and Electronic Systems, IEEE Transactions on, 2013, 49 (3): 1536-54.

[5] Algredo-Badillo I, Feregrino-Uribe C, Cumplido R, et al. FPGA-based implementation alternatives for the inner loop of the Secure Hash Algorithm SHA-256 [J]. Microprocessors and Microsystems, 2013, 37 (6): 750-7.

[6] 刘钰力. 基于 FPGA 的 SHA256 高效数字加密系统 [D]. 兰州大学, 2011.

[7] 陈华锋. 高速 SHA-256 算法硬件实现 [J]. 浙江大学学报 (理学版), 2009, (06): 675-8.

[8] Mcevoy R P, Crowe F M, Murphy C C, et al. Optimisation of the SHA-2 family of hash functions on FPGAs; proceedings of the Emerging VLSI Technologies and Architectures, 2006 IEEE Computer Society Annual Symposium on, F, 2006 [C].

[9] Glabb R, Imbert L, Jullien G, et al. Multi-mode operator for SHA-2 hash functions [J]. journal of systems architecture, 2007, 53 (2): 127-38.

[10] 奎伟. 基于哈希引擎的安全器件的设计研究 [D]. 厦门大学, 2013.

作者简介

汤煜，硕士研究生，厦门大学信息科学与技术学院，主要研究方向为数字集成电路设计及 SLAM 系统软硬件设计。

翁秀玲，硕士研究生，厦门大学信息科学与技术学院，主要研究方向为数字集成电路设计及 SLAM 系统软硬件设计。

王云峰，博士，副教授，厦门大学信息科学与技术学院，主要研究方向为数字集成电路设计及 SLAM 系统软硬件设计。