FPGA上SHA-1算法的流水线结构实现

李 磊 韩文报

(解放军信息工程大学信息研究系 郑州 450002)

摘 要 哈希算法SHA-1算法广泛地应用于电子商务、商用加密软件等信息安全领域。通过对SHA-1算法的深入分析,提出了流水线结构的硬件实现方案。通过缩短关键路径,使用片内RAM代替LE寄存器实现流水线中间变量的数据传递,有效地提高了工作频率和单位SHA-1算法的计算速度。这种硬件结构在Altera系列芯片上的实现性能是Altera商用SHA-1算法IP核的3倍以上。

关键词 哈希算法(SHA-1),关键路径,流水线结构,单位时空吞吐率(TPPAT),CSA

中图法分类号 TN402 **文献标识码** A

Implementation of Pipeline Structure on FPGA for SHA-1

LI Lei HAN Wen-bao

(Department of Information Researching, PLA Information Engineering University, Zhengzhou 450002, China)

Abstract Hash algorithm SHA-1 is used widely in cryptographic applications such as E-commerce and commercial encryption softwares. By making an overall analysis, implementation of pipeline structure was proposed. By shorting the critical path, using inside RAMs to realize the data translation instead of LE, we improved the frequency and the speed of per SHA-1 unit. The performance on FPGAs of Altera is three times faster of commercial IP cores for SHA-1.

Keywords Hash algorithm(SHA-1), Critical path, Pipeline structure, Throughput of per AT production unit(TPPAT), CSA

1 概述

哈希函数是密码学中一种重要的工具,其作用是对任何不定长的消息计算出一个定长的消息摘要即哈希值。哈希函数的两个重要特性是:单向性和抗碰撞性。目前最常用的散列函数是NIST于1995年颁布的安全散列算法SHA-1[1]。

由于SHA-1算法的良好特性,它被广泛应用于诸如电子商务这样的现代安全领域和WORD,RAR,WinZip等带有加密算法的商用软件中。目前在很多相关密码协议、标准或系统中,都包括了SHA-1算法,包括SSL,IPSec和PKCS。但是由于SHA-1算法本身的复杂性和在应用中的多次迭代使用,因此需要快速地实现SHA-1的计算。

本文提出流水线的硬件实现方法,该方法通过流水线技术,达到增加工作频率和并行化的目的,从而提高SHA-1的计算速度。

2 SHA-1算法

2.1 算法描述

SHA-1算法是由美国国家标准技术研究院(NIST)和美国国家安全局(NSA)共同设计的哈希算法。它可以对长度不超过2^{64}bit的消息进行计算,产生160bit的消息摘要,作为输出。为了处理长度超过512 bit的数据,首先需要将输入的数据每512 bit分成一块,然后将最后一块不足512 bit的数据按一定规则补齐。

分块之后就可以对每块数据按下述方法依次进行处理。

1)将512bit分成16个32bit字$W_0,W_1,\cdots,W_{15}$;

2)当$15<t<80$时,令$W_t=S^1(W_{t-3}\oplus W_{t-8}\oplus W_{t-14}\oplus W_{t-16})$;

3)令$A=H_0,B=H_1,C=H_2,D=H_3,E=H_4$(当处理第一分块时,$H_0$到$H_4$为固定常数,否则为上一分块输出结果);

4)从$t=0$至79做如下运算:

$A\_temp=S^5(A)+f_t(B,C,D)+E+K_t+W_t;E=D;D=C;C=S^{30}(B);B=A;A=A\_temp;$

5)令$H_0=A+H_0,H_1=B+H_1,H_2=C+H_2,H_3=D+H_3,H_4=E+H_4$。

所有消息块处理完后得到的5个32bit构成了160bit的消息摘要数据。

算法中使用的逻辑函数f_t和常数K_t分别为:

$$f_t=\begin{cases}(B\wedge C)\vee(\sim B\wedge D), & 0\leqslant t\leqslant 19\\ B\oplus C\oplus D, & 20\leqslant t\leqslant 39\\ (B\wedge C)\vee(B\wedge D)\vee(C\wedge D), & 40\leqslant t\leqslant 59\\ B\oplus C\oplus D, & 60\leqslant t\leqslant 79\end{cases}$$

到稿日期:2010-08-22 返修日期:2010-11-22 本文受国家"863"计划重点项目(2009AA012201),上海市科委重大科技攻关项目(08dz501600)资助。

李 磊(1978—),男,博士生,主要研究方向为密码学、信息安全,E-mail:leilimoon@hotmail.com;韩文报(1963—),男,教授,博士生导师,主要研究方向为密码学、信息安全。

$$K_t=\begin{cases}0x5827999, & 0\leqslant t\leqslant 19\\ 0x6ED9EBA1, & 20\leqslant t\leqslant 39\\ 0x8F1BBCDC, & 40\leqslant t\leqslant 59\\ 0xCA62C1D6, & 60\leqslant t\leqslant 79\end{cases}$$

算法中 S^1，S^5，S^{30} 分别表示循环左移 1 位，5 位和 30 位。

2.2 算法分析

从算法描述可以看出，SHA-1 最核心的计算是第 4 步的迭代运算。在标准的 SHA-1 算法中计算 A_temp 需要在一个周期内完成 4 次 32bit 加法，并且经过 80 个时钟周期才能完成一个 512bit 分块的处理。因此要提高 SHA-1 的硬件实现速度可以从两个方面考虑：一个是提高工作频率，这就需要对关键路径进行分解，减少一个周期内的串行加法个数；第二是增加并行程度，增加同时处理数据的能力[2]。

文献[3]中提出增加中间变量进行预计算来减少关键路径延时，但是这样做的代价是增加寄存器的开销。对这种方法的一种改进是在一个时钟周期内处理 64bit 的信息，但这又会带来降低时钟频率的问题。

为了同时解决频率和处理数据宽度的问题，本文采用了工作流水线技术。流水线结构是在实现整个工作流程中加入中间寄存器，将整个过程划分成为工作步骤相连的多级实体，每一个环节只完成数据处理的一个步骤，一个时钟周期完成一级数据处理，然后在下一个时钟沿到来时再将处理后的数据传递给下一级。不同于文献[4,5]中采用的对 SHA-1 算法进行多级流水线实现，本文进行了 80 级全展开的方法。这样做的好处在于增加了对公共寄存器的复用，减少了整个系统的开销。在本文中第一组 32bit 数据进入流水线后，经过一个时钟周期传递到第二级，同时第二组 32bit 数据进入第一级，数据队列依次前进。由于在 80 个周期中只是前 16 个是需要输入的，因此数据宽度是 512bit。这样就使得一个时钟内有 80 个数据块同时在各级中处理。虽然每组数据都要经过 80 个周期的整个流水线后才能得到最后的哈希值，但是作为整个流水线，每个时钟周期都能计算出一组哈希值，所以平均计算一组数据只需要一个时钟周期，大大提高了数据处理速度，保证了整个系统以较高的频率工作。

3 硬件实现 SHA-1 的流水线结构

3.1 关键路径分解

观察算法可以发现，第 4 步计算 $A\_temp=S^5(A)+f_t(B,C,D)+E+K_t+W_t$ 的迭代是延时最长的。原因就在于 A_temp 的计算需要在一个周期内完成 4 次 32bit 加法。而加法在 FPGA 的实现中要比位操作的延时大得多。为了降低关键路径的延时，本设计采用 CSA 方式对加法进行计算。CSA(Carry Save Addition)方式是根据如下特性做出的：如果 $Result=A+B+C$，则有 $S=A$ xor B xor C；$Ca=((A$ and $B)$ or$(B$ and $C)$ or$(A$ and $C))<<1$；最后结果为 $Result=Ca+S$。

采取这种方式的好处是在处理多个加法连续运算时，只需在最后一步进行加法运算，而其他的可通过位运算来实现，因此延时就近似于最后的一次加法延时。本文采取 5 进 2 出的 CSA 算法来实现这 5 个 32-bit 数的相加：$(Ca,S)=CSA52(S^5(A)+f_t(B,C,D)+E+K_t+W_t)$。而 $A\_temp=Ca+S$，即为所求。下面用 ADDCSA 来表示这一计算过程。

3.2 流水线结构

3.2.1 I/O 接口

对一个 512bit 分块进行计算时，只能同时处理一个 32bit 分组，在 80 个时钟周期需要 16 次输入。而在流水线实现时，可以同时处理这 16 个 32bit 输入，因此输入要求 512bit 的 16 个不同的分组。

输出即为要处理分组的 160bit 的散列值。

3.2.2 流水线结构的 SHA-1 算法

流水线结构的 SHA-1 算法可以通过如下方法来计算：

1)将需要处理的不同 512bit 分成相应的 16 个 32bit 字 $W_{n,0}$，$W_{n+1,1}$，…，$W_{n+15,15}$；

2) 令 $A_0=H_0$，$B_0=H_1$，$C_0=H_2$，$D_0=H_3$，$E_0=H_4$（当处理第一分块时，H_0 到 H_4 为固定常数，否则为上一分块输出结果）；

3)在一个周期内对 n 从 0 到 79 同时做下列运算：

$W_{n,t}=S^1(W_{n,t-3}\oplus W_{n,t-8}\oplus W_{n,t-14}\oplus W_{n,t-16})$，$t\geqslant 16$；

$A_{n+1}=ADDCSA(S^5(A)+f_t(B,C,D)+E+K_t+W_t)$；

$E_{n+1}=D_n$；$D_{n+1}=C_{n+1}$；$C_{n+1}=S^{30}(B_n)$；$B_{n+1}=A_n$；

4)令 $H_0=A_{80}+H_0$，$H_1=B_{80}+H_1$，$H_2=C_{80}+H_2$，$H_3=D_{80}+H_3$，$H_4=E_{80}+H_4$。

最后的 H_0 到 H_4 就是当前的散列值输出。

4 FPGA 实验结果

流水线结构的实现需要增加中间的寄存器来实现上一级与下一级流水的数据传递，但是这样就会带来资源的大大增加。本设计中使用 RAM 代替 LE 来实现流水线中间寄存器的方法。这样既可以减少寄存器资源的占用又可以平衡 FPGA 芯片内部资源的使用，增加布线资源并且有效地增加工作频率。

本设计使用的软件是 Quartus II 8.1，在 Altera 的 Cyclone III C6 芯片上验证实现，并在 Altera 的 Cyclone II，Stratix II 和 Stratix III 系列芯片上进行了综合和布线。表 1 是在上述芯片上完成一条完整流水线的资源占用情况。需要指出的是一条流水线的完成相当于同时进行 81 个标准的 SHA-1 算法。因此在表 1 中列出了相当于 1 个标准的 SHA-1 算法的资源使用情况。

表 1 在 Altera 系列芯片上实现性能指标

	Cyclone II C6	Cyclone III C6	Stratix II C3	Stratix III C2
逻辑资源	316 11x M4K	317 10x M9K	138 11x M4K	140 12x M9K
频率	144M	157M	213M	272M
吞吐率(Mbps)	910	992	1346	1719
时空积	2.19	2.02	0.64	0.51

表 2 是 Helion 提供的 Altera 的商用 SHA-1 算法 IP 核的性能指标[6]。

表 2 Altera 商用 SHA-1 算法 IP 核的性能指标

	Cyclone II C6	Cyclone III C6	Stratix II C3	Stratix III C2
逻辑资源	1264 3x M4K	1323 3x M9K	766 3x M4K	781 3x M9K
频率	153M	171M	247M	298M
吞吐率(Mbps)	955	1067	1542	1860
时空积	8.26	7.73	3.10	2.62

为了能作为衡量设计的性能标准，本文引入单位时空吞吐率的概念。

定义 1 单位时空吞吐率(TPPAT)是吞吐率与时空积的比值：

TPPAT=(max throughput)/(AT product)。

表 3 将本文结果与 Helion 提供的 Altera 的商用 SHA-1 算法 IP 核进行了比较。

表 3 两种实现方式性能指标的比较

	Cyclone II C6	Cyclone III C6	Stratix II C3	Stratix III C2
本文	415	491	2103	3370
IP 核	120	138	497	710
比率(%)	345.83	355.79	423.13	474.64

从表 3 可以看出，利用本文提出的流水线结构实现 SHA-1 算法的单位时空吞吐率是商用 SHA-1 算法 IP 核的 3 倍以上。

结束语 本文提出的流水线结构方法在 FPGA 上实现 SHA-1 算法，缩短了关键路径，使用片内 RAM 代替 LE 寄存器实现流水线中间变量的数据传递，改善了布线资源，有效地提高了资源使用率，提高了单位 SHA-1 算法的计算速度。

本文结果对并行处理多个 SHA-1 算法的使用环境有着显著的加速效果。与 Helion 提供的 Altera 的商用 SHA-1 算法 IP 核相比较，单位时空吞吐率(TPPAT)提高了 3 倍以上。

参 考 文 献

[1] FIPS PUB 180-1, Secure Hash Standard(SHA-1)[S]. National Institute of Standards and Technology(NIST), 1995

[2] Bosselaers A, Dovaerts R, Vandewalle J. SHA: A design for parallel architectures [C]// Fumy W. Adances in Cryptology-EUROCYPT'97. Heidelberg: Springer-Verlag, 1997: 348-362

[3] 黄谆，白国强，陈弘毅. 快速实现 SHA-1 算法的硬件结构[J]. 清华大学学报：自然科学版，2005，45(1)

[4] Sklavos N, Kitsos P, Alexopoulos E, et al. Open Mobile Alliance (OMA) Security Layer: Architecyure, Implementation and Performance Evaluation of the Integrity Unit[C]// New Generatation Computing: Computing Paradigms and Computatioal Intelligence. Springer-Verlag, 2004

[5] Sklavos N, Alexopoulos E, Koufopavlou O. Networking Data Integrity: High Speed Architectures and Hardware Implementations[J]. IAJIT Journal, 2003, 1: 54-59

[6] Helion Inc[OL]. http://www.heliontech.com/downloads/sha1_altera_datasheet.pdf, 2010

(上接第 34 页)

[4] Majercik S M, Littman M L. MAXPLAN: A new approach to *probabilistic planning*[J]. Artificial Intelligence Planning Systems, 1998: 86-93

[5] 徐丽，赵成丽，等. 图规划框架下的概率规划[C]// 第三届不确定系统年会. 2005: 105-111

[6] Yoon S, Fern A, Givan B. FF-replan: a baseline for probabilistic planning[C] // 17th International Conferenceon Automated Planning and Scheduling(ICAPS-07). 2007

[7] Draper D, Hanks S, Weld D. Probabilistic Planning with Information Gathering and Ontingent Execution[C] // Proceedings AIPS-94. 1994: 31-36

[8] Blum A, Langford J. Probabilistic planning in the Graphplan framework[C]// Proceedings of the Fifth European Conference on Planning. Durham, United Kingdom, 1999

[9] Gu Wen-xiang, Ou Hua-jie, Liu Ri-xian, et al. An Improved Probabilistic Planning Algorithm Based on Pgraphplan[C]// Proceedings of the Third International Conference on Machine Learning and Cybernetics. 2004: 2374-2377

[10] Mausam, Weld D. Solving concurrent Markov decision processes [C]// AAAI'04. 2004

[11] Mausam, Weld D. Cuncurrent probabilistic temporal planning [C]// ICAPS'05. 2005: 120-129

[12] Mausam, Weld D. Probabilistic Temporal Planing with Uncertain Durations[C]// AAAI'06. 2006

[13] Little I, Thiébaux S. Concurrent probabilistic planning in the Graphplan framework[C]// The 16th International Conference on Automated Planning and Scheduling(ICAPS). The English Lake District, Cumbria, U. K, 2006

[14] Bryce D. Probabilistic Planning is Multi-objective! [C]// ASU CSE-07-006. 2007

[15] Florent T K, Infantes G, Kuter U. RFF: A Robust, FF-based MDP Planning Algorithm for Generating Policies with Low Probability of Failure[C]// AAAI' 08. 2008

[16] Florent T K, Infantes G, Kuter U. FSP*: Optimal Forward Stochastic Planning Using Relaxed PPDDL Operators[C]// AAAI' 08. 2008

[17] http://andorfer.cs.uni-dortmund.de/~edelkamp/ipc-4/

[18] Bonet B, Geffner H. mGPT: Aprobagilistic planner based on heuristic search[J]. Journal of Artificial Intelligence Research, 2005, 24: 933-944

[19] Hansen E, Zilberstein S. LAO*: A heuristic search algorithm that finds solutions with loops[J]. Artifical Intelligence, 2001, 129: 35-62

[20] http://zeus.ing.unibs.it/ipc-5/

[21] http://ippc-2008.loria.fr/wiki/index.php/Main_Page

[22] Tran D-V, Nguyen H-K, Pontelli E, et al. CPA(C)/(H): Two Approximation-based Conformant Planners[C]// AAAI' 08. 2008

[23] Kissmann P, Edelkamp S. Gamer: Fully-observable Non-deterministic Planning via PKKL-Translation into a Game[C]// AAAI' 08. 2008

[24] Younes H L S, Littman M L. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects [R]. CMU-CS-04-167. Carnegie Mellon University, 2004

[25] 闫书亚，殷明浩，谷文祥，等. 概率规划的研究与发展[J]. 智能系统学报，2008

[26] Mausam, Weld D. Solving concurrent Markov decision processes [C]// AAAI'04. 2004: 716

[27] Bonet B, Geffner H. Labeled RTDP: Improving the convergence of realtime dynamic programming[A]// Proceedings of 13th International Conference on Automated Planning and Scheduling (ICAPS)[C]. Trento, Italy, 2003

[28] 刘小飞. 多目标概率规划算法的研究与实现[D]. 长春：东北师范大学，2009

[29] Little I, Thiebaux S. Probabilistic palnning vs replanning[C]// ICAPS Workshop on Planning Competitions: Past, Present, and Future. 2007

[30] Hoffmann J, Nebel B. The FF planning system: Fast plan generation through heuristic search[J]. Journal of Artificial Intelligence Research, 2001, 14: 263-302