Skein算法的流水线结构设计与实现

闫丁丽 刘 航 郭达伟 李 杨 洪 亮
(西北工业大学自动化学院 西安 710129)

摘 要 硬件实现的速度和性能是SHA-3算法甄选的重要指标。针对SHA-3末轮5个候选算法之一的Skein算法，结合其4轮迭代结构的关键路径较短而8轮迭代结构实现所用的选择器较少的优点，采用FPGA实现了一个两级流水线结构的Skein算法IP核。仿真验证结果表明，该算法在Xilinx Virtex-5上数据吞吐量达到6.4Gbps，比之前的非流水线结构速度性能提高了82%以上，硬件资源利用率提高了21%，特别适用于Hash树计算。

关键词 SHA-3，Skein，流水线，FPGA
中图法分类号 TP309 **文献标识码** A

Pipeline Based Skein Algorithm Design and Implementation

YAN Ding-li LIU Hang GUO Da-wei LI Yang HONG Liang
(School of Automation, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract The evaluation of speed and performance in hardware is very important with SHA-3 competition. And for one of the final round candidates—Skein algorithm, its 4-unrolled structure has short critical path and 8-unrolled structure used fewer multiplexers. So combining the advantages of the two structures, we proposed a pipeline design with two stages and implemented on Xilinx Virtex-5. Finally the experimental simulation shows that this approach can greatly increase the throughput of Skein algorithm.

Keywords SHA-3, Skein, Pipeline, FPGA

1 引言

近年来，随着MD5、SHA-1等算法被相继破解[1-3]，安全Hash算法面临前所未有的危机。2007年美国国家标准技术研究所正式宣布在全球范围内甄选SHA-3算法[4]，截止到2010年12月，BLAKE、Grostl、JH、Keccak、Skein等5个算法进入最终角逐[5]，并将于2012年公布最终结果，这必将成为安全Hash算法发展历程中一个新的里程碑。

在SHA-3末轮的5个候选算法中，Skein算法在速度、安全性和资源消耗上都有较突出的性能，具有很强的实用性[6]。而FPGA(Field Programmable Gate Array)等可重构硬件能够在较短的开发周期内，以较低的设计、验证和测试成本，得到全功能的算法原型验证系统，并且在算法扩展/升级/修正、体系结构更改与计算性能之间获得非常理想的平衡点，基于FPGA的Skein算法硬件实现评估受到关注[7-12]。由于Skein算法中的72轮迭代运算具有规律性，文献[7,8]在硬件上实现了单轮迭代结构，通过循环调用此结构完成72轮循环，既缩短了关键路径的长度，提高了最高时钟频率，又减少了资源使用。文献[9-12]在此基础上分别实现了4轮和8轮迭代结构，不仅减少了迭代轮数，而且减少了迭代运算的选择器的使用，通过综合工具优化多轮组合逻辑，进一步提高了数据处理速度。其中，文献[12]在Xilinx Virtex-5上实现的Skein-512-512其吞吐量达到了3.5Gbit/s。

上述文献主要针对Skein算法的链式结构实现进行研究，虽然也可以实现抗重放攻击的树状Hash计算[13]，但由于并行性差，难以充分发挥密码计算单元的计算效能。本文通过分析Skein算法内部迭代规律——4轮迭代结构关键路径短而8轮迭代结构使用选择器少以及哈希树中非兄弟结点可并行处理的特点，提出了一种Skein算法流水线结构的FPGA实现方案，并用Verilog语言对其进行了描述。仿真结果表明，该算法可大幅度提高计算吞吐量。

2 Skein算法原理[14]

Skein算法是由Threefish分组密码延伸的单向函数，主要由3个部件组成：唯一分组迭代(Unique Block Iteration, UBI)、Threefish、可选参数系统(Optional Argument System, OAS)。

2.1 UBI

UBI是Matyas-Meyer-Oseas结构的变体，有3个输入，分别为M、G、T。其中M为待处理的消息块；G是上一轮

到稿日期：2011-03-06 返修日期：2011-06-09

闫丁丽(1986－)，女，硕士生，主要研究领域为FPGA可重构计算、FPGA远程安全更新技术，E-mail：yandingli@163.com；刘 航(1973－)，男，博士，副教授，主要研究领域为网络安全与可重构嵌入式系统、车载网络协议设计；郭达伟(1968－)，男，博士，副教授，主要研究领域为信息安全、网络化控制等；李 杨(1986－)，男，硕士生，主要研究领域为无线自组网路由协议、嵌入式linux；洪 亮(1979－)，男，副教授，主要研究领域为网络安全、无线自组网路由协议。

UBI 的结果；T 是 Tweak 值，其格式如表 1 所列。3 个参数输入 UBI 后，首先利用 G、T 计算子密钥；然后将所得子密钥与 M 输入 Threefish 进行计算(2.2 节将作详细说明)；最后将 Threefish 计算结果与输入消息 M 异或得到 UBI 输出结果。

表 1 Tweak 格式

名称	位	描述
Position	0-95	目前已处理的消息位数
Reserved	96-111	保留位
TreeLevel	112-118	树的相关信息
BitPad	119	填充标识
Type	120-125	UBI 类型
First	126	第一块 UBI 需设置
Final	127	最后一块 UBI 需设置

2.2 Threefish

Threefish 是 UBI 的主体计算模块。以 512 位为例，在 Threefish 中，输入消息块大小为 512 位，消息 M 输入后将其分为 8 个 64 位数据块，并进行 72 轮迭代运算，而在 72 轮迭代运算中，每隔 4 轮其运算结果需与子密钥进行一次 64 位模加。单轮迭代运算如下：4 组相邻的数据块同时进行 Mix 运算，如式(1)所示；然后对 Mix 输出结果进行 Permute 置换运算，如表 2 所列。子密钥计算公式如式(2)所示。

$$\begin{cases} y_0=(x_0+x_1) \bmod 2^{64} \\ y_1=(x_1<<<R_{(d \bmod 8),j})\oplus y_0 \end{cases} \tag{1}$$

式中，x_0，x_1 为 Mix 输入，y_0，y_1 为 Mix 输出，R 如表 3 所列，d 为迭代轮数。

表 2 Permute 运算置换表

Input	0	1	2	3	4	5	6	7
Output	2	1	4	7	6	5	0	3

$$\begin{cases} k_{s,j}=k_{(s+i) \bmod (N_w+1)}, & i=0,\cdots,N_w-4 \\ k_{s,j}=k_{(s+i) \bmod (N_w+1)}+t_{s \bmod 3}, & i=N_w-3 \\ k_{s,j}=k_{(s+i) \bmod (N_w+1)}+t_{(s+1) \bmod 3}, & i=N_w-2 \\ k_{s,j}=k_{(s+i) \bmod (N_w+1)}+s, & i=N_w-1 \end{cases} \tag{2}$$

式中，k_i 表示输入 G 的第 i 个 64 位分块，t_i 为输入 T 的第 i 个 64 位分块，$k_8=C_{240}\overset{N_w-1}{\underset{i=0}{\oplus}}k_i$，$t_2=t_1\oplus t_0$，$K$、$T$ 为密钥产生模块的输入值，C_{240} 为一常数，s 为加入子密钥的轮数。

表 3 Mix 循环左移位数查询表

j	d mod 80							
	0	1	2	3	4	5	6	7
0	46	33	17	44	39	13	25	8
1	36	27	49	9	30	50	29	35
2	19	14	36	54	34	10	39	56
3	37	42	39	56	24	17	43	22

2.3 OAS

为了增加 Skein 算法的灵活性与安全性，可以根据需要多次调用 UBI，如图 1 所示。

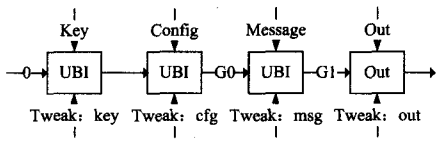

图 1 Skein 中 UBI 调用模式

当进行 Skein 计算时，先选择 UBI 模式，然后根据所选项依次进行 UBI 计算，所选模式对应的 Tweak 中的 Type 值如表 4 所列。

表 4 UBI 可选类型

UBI 类型	值	描述
Key	0	密钥
Cfg	4	配置块
Prs	8	个人自定义
PK	12	公钥(数字签名等)
Kdf	16	密钥标识
Non	20	随机数
Msg	48	消息
Out	63	输出

2.4 Skein 哈希树

哈希树是一种并行化的哈希结构。计算时首先将原始数据分组进行哈希计算；然后再将结果作为其双亲结点的输入分组计算[15]；以此类推，直到根结点，在根结点计算出最终哈希值。Skein 哈希树的结构由 Y_l、Y_f 和 Y_m 3 个参数决定[14]，Y_l 表示每个叶子结点的大小为 $N_b 2^{Y_l}$ 比特，N_b 表示消息块的大小；Y_f 表示结点扇出为 2^{Y_f}；Y_m 表示树的最大高度。例如 $Y_l=1$，$Y_f=2$，$Y_m=2$ 时，树的结构如图 2 所示。

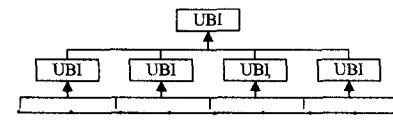

图 2 Skein 哈希树结构

3 Skein 算法的非流水线结构实现

Skein 算法非流水线结构主要由 3 大模块组成：控制模块、子密钥产生模块及 Threefish 模块，如图 3 所示。

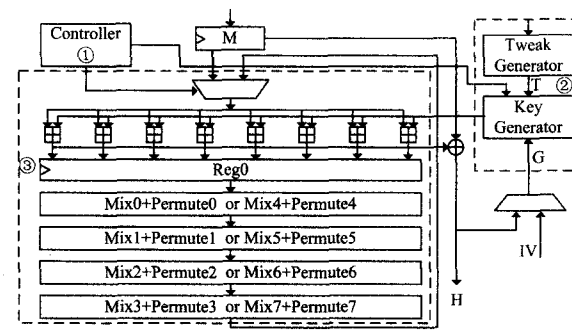

图 3 Skein 算法非流水线实现结构

3.1 控制模块

控制模块主要控制外部消息 M 的加载、UBI 模式选择、Tweak 的参数值、迭代轮数的控制、Threefish 输入数据的选择以及子密钥产生模块的启动。

在初始化阶段，控制模块选择 UBI 类型；然后根据外部输入及默认的初始信息填写 T 的初始参数；G 为默认初始值；当 M 有效时，加载 M 值，否则等待；M 加载完成后，启动子密钥产生模块。

初始化完成后，控制器对迭代轮数进行判断，若一次 Threefish 已经完成，则加载下一组数据，否则继续迭代。

输出阶段，当全部消息处理完成后，控制模块选择 UBI 输出模式，对结果进行输出处理。处理完成后，若输出位高，说明输出有效。

3.2 子密钥产生模块

由式(2)可知，子密钥可以通过移位寄存器计算[9]，如图

4 所示。

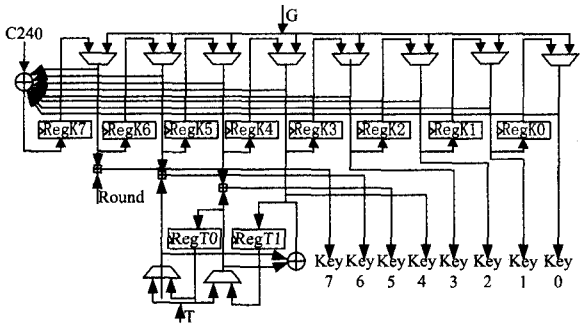

图 4 Skein 算法子密钥产生模块实现结构

3.3 Threefish 模块

Threefish 主要完成 72 轮迭代运算。由于算法中每 4 轮迭代运算后要与子密钥进行一次模加，且硬件结构可以复用，故将 4 轮迭代运算展开实现，并在 4 轮迭代运算前加入 8 个加法器进行子密钥模加；然后进行循环计算。其中 Mix 中的移位运算与 Permute 运算都通过线移实现。

4 Skein 算法的流水线结构实现

上述非流水线结构虽然具有较强的通用性，但是没有充分利用树状 Hash 计算过程可并行化的特点。通过对 Skein 算法的进一步分析可以看出：在 Skein 算法的主体计算模块 Threefish 中，72 轮迭代运算每 8 轮循环一次，即每 8 轮运算的 Mix 与 Permute 完全相同。所以若将 8 轮迭代运算展开实现，再对其分时复用进行循环迭代，可以有效地减少 Threefish运算中的选择器的使用，这既减少了资源消耗，又在一定程度上减少了选择的时间，提高了计算吞吐量；但是与 4 轮迭代结构相比，其关键路径增长，最高时钟频率明显下降。因此，本文结合 4 轮迭代结构与 8 轮迭代结构的优点，提出 1 个两级流水线结构：将 8 轮迭代运算中前 4 轮 Mix/Permute 运算与后 4 轮 Mix/Permute 运算划分到两个时钟周期计算。这种结构既保持了 4 轮迭代结构关键路径短的优点，又可以体现 8 轮迭代结构选择计算少的优点，还可对哈希树的两个非兄弟结点同时进行 Hash 计算，显著提高了 Skein 算法的计算并行度和性能。

与上述非流水线结构一样，该流水线结构也包括了控制模块、子密钥产生模块以及 Threefish 模块，如图 5 所示。

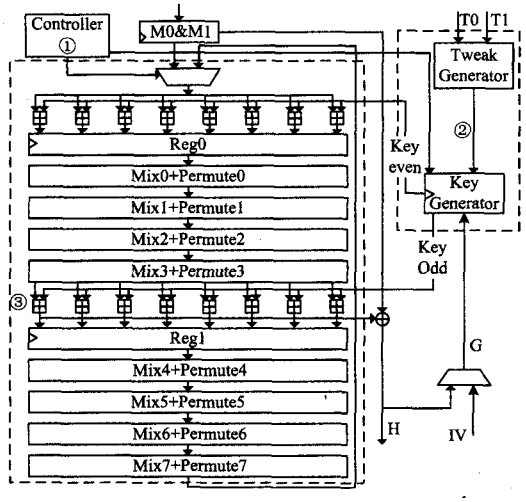

图 5 Skein 算法流水线实现结构

4.1 控制模块

在流水线结构中，控制模块同样需要控制外部数据的加载、UBI 模式选择、Tweak 的参数值、迭代轮数的控制、Threefish 输入数据选择以及子密钥产生模块的启动等。但由于需要计算两组数据的 Hash 值，故在控制模块中需要加入额外的选择器用于 UBI 模块中两组数据的调度。

4.2 子密钥产生模块

在流水线结构中，有两组数据同时进行计算，故必须同时计算两组子密钥。与非流水线结构中子密钥产生模块类似，流水线结构中的子密钥也是通过移位寄存器产生的，只需使子密钥与迭代运算中的数据正确对应即可。

4.3 Threefish 模块

如图 5 所示，将上述非流水线结构 Threefish 中 4 轮迭代结构展开为 8 轮，并在 Permute3 与 Mix4 之间增加一组寄存器 *Reg*1。其时序如下：当 $clk=1$ 时，*M*0 存入 *Reg*0，前 4 轮迭代结构对 *M*0 进行计算，此时后 4 轮迭代结构空闲；当 $clk=2$ 时，*M*1 存入 *Reg*0，前 4 轮迭代结构对 *M*1 进行计算，而 *M*0 的计算结果存入 *Reg*1，由后 4 轮迭代结构对其进行计算；以此类推，直到 $clk=19$，后 4 轮迭代结果与 *M*0 进行异或，得到 *M*0 的哈希值；同理，当 $clk=20$，计算出 *M*1 的哈希值，即仅需 20 个时钟就可以计算出两组数据的哈希值。

5 性能评估

本文设计采用 Verilog 语言描述，在 Xilinx ISE10.1 环境下编译，所选器件为 Xilinx xc5vlx110，经仿真验证，其功能正确。

综合结果如表 5 所列，本文的 4 轮非流水线结构与文献[9]的 4 轮非流水线结构相比，吞吐量提高了 3.9%，资源节省了 11.7%；但与文献[12]的 8 轮非流水线结构相比，吞吐量减少了 5.9%，资源节省了 6.25%，这是因为 8 轮非流水线结构不仅减少了运算中轮数的选择判断，而且综合工具将对更多的逻辑资源进行整体优化，故而吞吐量高于 4 轮非流水线结构。但由于实现了更多的迭代运算，故而增加了一定的资源消耗。而与上述 3 种非流水线结构相比，本文所实现的流水线结构在资源仅增加不多于 59%的情况下，数据吞吐量提高了 82%以上，平均每个 Slice 的吞吐量提高了至少 21%，即通过较少的资源占用换取了较大的性能提升。

表 5 Skein 算法实现性能比较

结构类型	最高时钟频率 (MHz)	吞吐量 (Mbps)	资源占用 (Slices)	吞吐量/资源占用比 (Mbps/slice)
4 轮非流水线[9]	119.1	3209	1716	1.87
8 轮非流水线[12]	69.04	3535	1632	2.166
本文 4 轮非流水线	123.788	3335.8	1536	2.172
本文流水线结构	126.081	6455.6	2448	2.637

结束语 本文给出了 Skein 算法流水线结构的 FPGA 实现方案，利用哈希树非兄弟结点可并行计算的特点所设计的流水线结构大幅度地提高了算法的计算性能。

参 考 文 献

[1] Wang Xiao-yun, Yin Y L, Yu Hong-bo. Finding collisions in the full SHA-1[C]// International Association for Cryptologic Research. 25th Annual International Cryptology Conference. Berlin: SpringerVerlag, 2005: 17-36

[2] Wang Xiao-yun, Yu Hong-bo. How to break MD 5 and other hash functions[C]// International Association for Cryptologic

Research. 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer Verlag, 2005: 19-35

[3] 刘建东，余有明，江慧娜. 单向 Hash 函数 SHA-1 的统计分析与算法改进[J]. 计算机科学，2009，36(10)：141-145

[4] National Institute of Standards and Technology (NIST). Cryptographic Hash Algorithm Competition[EB/OL]. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html, 2007-11-02

[5] National Institute of Standards and Technology (NIST). Final Round Candidates[EB/OL]. http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html, 2009-07-16

[6] 薛宇，吴文玲，王张宜. SHA-3 杂凑密码候选算法简评[J]. 中国科学院研究生院学报，2009，26(5)：577-586

[7] Long M. Implementing Skein Hash Function on Xilinx Virtex-5 FPGA Platform[EB/OL]. http://www.skein-hash.info/sites/default/files/skein_fpga.pdf, 2009

[8] Namin A H, Hasan M A. Implementation of the Compression Function for Selected SHA-3 Candidates on FPGA[EB/OL]. http://comsec.uwaterloo.ca/seminarfiles/ReviewSeminar2010/Implementation_SHA3_Candidates_on_FPGA.pdf, 2010

[9] Homsirikamol E, Rogawski M, Gaj K. Comparing. Hardware Performance of Fourteen Round Two SHA-3 Candidates Using FPGAs[EB/OL]. http://eprint.iacr.org/2010/445.pdf, 2010

[10] Baldwin B, Hanley N, Hamilton M, et al. FPGA Implementations of the Round Two SHA-3 Candidates[C]// Field Programmable Logic and Applications (FPL), 2010. Milano, Italy, 2010: 400-407

[11] Tillich S, Feldhofer M, Kirschbaum M, et al. High-Speed Hardware Implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Lua, Shabal, SHAvite-3, SIMD, and Skein[EB/OL]. http://eprint.iacr.org/2009/510.pdf, 2009

[12] Tillich S. Hardware Implementation of the SHA-3 Candidate Skein[DB/OL]. http://eprint.iacr.org/2009/159.pdf, 2009

[13] Hou Fang-yong, He Hong-jun, Xiao Nong. Hash Tree Based Integrity Protection Appropriate for Disk [C]// 2009 WASE International Conference on Information Engineering. Taiyuan, China, 2009: 242-245

[14] Ferguson N, Lucks S, Schneieret B, et al. The Skein Hash Function Family[EB/OL]. http://www.skein-hash.info/sites/default/files/skein1.3.pdf, 2010

[15] Schorr A, Lukowiak M. Skein Tree Hashing on FPGA[C]// International Conference on Reconfigurable Computing. 2010. Cancun, Mexico, 2010: 292-297

(上接第 31 页)

[27] Cernekova Z, Pitas I, Nikou C. Information theory-based shot cut/fade detection and video summarization [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2006, 16(1): 82-91

[28] Cao Yu, Tavanapong W, Kim K. Audio-Assisted Shot Clustering Techniques for Story Browsing. 2008

[29] 庄越挺，潘云鹤，吴飞. 网上多媒体信息分析与检索 [M]. 北京：清华大学出版社，2002

[30] Herranz L, Tiburzi F, Bescos J. Extraction of Motion Activity from Scalable-coded Video Sequences [J]. Semantic Multimedia, 2006: 148-158

[31] Wolf W. Key frame selection by motion analysis [C]// Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. 1996, 2: 1228-1231

[32] Liu Tian-ming, Zhang Hong-jiang, Qi Fei-hu. A novel video key-frame-extraction algorithm based on perceived motion energy model [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(10): 1006-1013

[33] Song Xiao-mu, Fan Guo-liang. Joint key-frame extraction and object segmentation for content-based video analysis [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2006, 16(7): 904-914

[34] 王方石，须德，吴伟鑫. 基于自适应阈值的自动提取关键帧的聚类算法 [J]. 计算机研究与发展，2005，42(10)：1752-1757

[35] Lo C C, Wang S J. Video segmentation using a histogram-based fuzzy c-means clustering algorithm [J]. Computer Standards & Interfaces, 2001, 23(5): 429-438

[36] Zeng Xianglin, Hu Weiming, Li Wanqing, et al. Key-frame extraction using dominant-set clustering [C]// Proceedings of the IEEE International Conference on Multimedia and Expo. 2008: 1285-1288

[37] Chasanis V T, Likas A C, Galatsanos N P. Scene detection in videos using shot clustering and sequence alignment [J]. IEEE Transactions on Multimedia, 2009, 11(1): 89-100

[38] Sze K W, Lam K M, Qiu Guoping. A new key frame representation for video segment retrieval [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2005, 15(9): 1148-1155

[39] Song Xiao-mu, Fan Guo-liang. Key-frame extraction for object-based video segmentation [C]// Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. 2005: 689-692

[40] Bhat D N, Nayar S K. Ordinal measures for image correspondence [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, 20(4): 415-423

[41] Mohan R. Video sequence matching [C]// Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. 1998, 6: 3697-3700

[42] Hua Xian-sheng, Chen Xian, Zhang Hong-jiang. Robust video signature based on ordinal measure [C]// Proceedings of the International Conference on Image Processing. 2005, 1: 685-688

[43] Kim C, Vasudev B. Spatiotemporal sequence matching for efficient video copy detection [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2005, 15(1): 127-132

[44] Paisitkriangkrai S, Mei Tao, Zhang Jian, et al. Scalable clip-based near-duplicate video detection with ordinal measure [C]// Proceedings of the ACM International Conference on Image and Video Retrieval. 2010: 121-128

[45] Shang Li-feng, Yang Lin-jun, Wang Fei, et al. Real-time large scale near-duplicate web video retrieval [C]// Proceedings of the International Conference on Multimedia. 2010: 531-540

[46] Yang Xian-feng, Tian Qi, Chang E C. A color fingerprint of video shot for content identification [C]// Proceedings of the ACM International Conference on Multimedia. 2004: 276-279

[47] 薄华，马缚龙，焦李成. 图像纹理的灰度共生矩阵计算问题的分析 [J]. 电子学报，2006，34(1)：155-158

[48] Manerba F, Benois-Pineau J, Leonardi R, et al. Multiple moving object detection for fast video content description in compressed domain [J]. EURASIP Journal on Advances in Signal Processing, 2008: 5

[49] Su C W, Liao H Y M, Tyan H R, et al. Motion flow-based video retrieval [J]. IEEE Transactions on Multimedia, 2007, 9(6): 1193-1201