LETTER

Design and Implementation of a Non-pipelined MD5 Hardware Architecture Using a New Functional Description

Ignacio ALGREDO-BADILLO[†a)], *Student Member*, Claudia FEREGRINO-URIBE[†b)], René CUMPLIDO[†c)], *Nonmembers, and* Miguel MORALES-SANDOVAL[†d)], *Member*

SUMMARY MD5 is a cryptographic algorithm used for authentication. When implemented in hardware, the performance is affected by the data dependency of the iterative compression function. In this paper, a new functional description is proposed with the aim of achieving higher throughput by mean of reducing the critical path and latency. This description can be used in similar structures of other hash algorithms, such as SHA-1, SHA-2 and RIPEMD-160, which have comparable data dependence. The proposed MD5 hardware architecture achieves a high throughput/area ratio, results of implementation in an FPGA are presented and discussed, as well as comparisons against related works.

key words: *MD5 algorithm, hardware design, FPGA implementation, hardware architectures*

1. Introduction

New communication technologies motivate the development of more demanding applications that require the exchange of secure information, such as electronic mail, electronic banking, medical databases, multimedia, secure wireless systems and electronic commerce. Cryptography provides several security services at different levels by using powerful mechanisms to protect data, but this comes at the expense of computing power. The more demanding applications require that the cryptographic applications have a high throughput and high flexibility [1].

Cryptographic algorithms are more efficiently implemented in custom hardware than in software running on general-purpose processors [2]. Authentication algorithms or hash functions, considered cryptographic algorithms, do not cipher the complete message; they are based on compression functions that generate blocks of length m from blocks of length n. The MD5 message digest algorithm is used to generate a 128-bit output from an input message of arbitrary length. This output is a compressed but irreversible representation of the entire input message, determining whether the message has been tampered within transit. Using MD5 in HMAC (keyed-Hash Message Authentication Code) processes, authentication of data can be provided in applications such as IPsec and SSH.

Manuscript received November 30, 2007.
Manuscript revised May 31, 2008.

[†]The authors are with the Department of Computer Science, National Institute for Astrophysics, Optics and Electronics, Luis Enrique Erro 1, CP 72840, Sta. Ma. Tonantzintla, Puebla, México.
a) E-mail: algredobadillo@ccc.inaoep.mx
b) E-mail: cferegrino@ccc.inaoep.mx
c) E-mail: rcumplido@ccc.inaoep.mx
d) E-mail: mmorales@ccc.inaoep.mx

DOI: 10.1093/ietisy/e91–d.10.2519

Although hardware implementations are even more efficient for a specific instance, they depend on the design efficiency [2]. For MD5 hardware architectures there is a data dependence, which limits the throughput that these architectures achieve. Similar structures can be found in other algorithms with data dependence, and so it is necessary to provide solutions for designing efficient hardware architectures for these algorithms. In this paper, a new functional description is proposed focusing on improving the performance in order to achieve a high throughput/area ratio.

2. MD5 Algorithm

Many security protocols like in IPSec or SSH, utilize MD5 algorithm for electronic funds transfer, authentication of electronic data transfers and encrypted data storage. This hash function is important and widely deployed [3]. For example, Secure Sockets Layer version 3 (SSL3) uses a concatenation of MD5 and SHA-1 algorithms as part of the client authentication process.

MD5 is a 128-bit hash function, which produces a 128-bit output, called message digest, from an arbitrary length input message. The first step is to split the input message into blocks of 512 bits. Next, each 512-bit block is padded into 32-bit data words. It requires a 128-bit state buffer consisting of four 32-bit words, see Fig. 1 (A). The compression function consists of 32-bit elementary operations, such as addition, XOR, AND, OR, and rotation. This compression function is used 64 times for each 512-bit block, where each round mixes the entire message block into the state buffer. After 64 rounds, initial and final state buffers are added to produce the output [4].

The computational structure of each round of this algorithm is depicted in Fig. 1 (B) [5]. In general, the value $X[k]$ is the 32-bit data word (where k takes values from 1 to 16), for the i round, and the $T[i]$ value represents the 32-bit constant that also depends on the round. The 32-bit values from A to D variables are updated in each round and the new values are used in the following round. A given function (F, G, H or I) with a defined shift is executed in each round. In this structure, data dependence is present in additions and 32-bit operations, where data are necessary to compute the next operation.

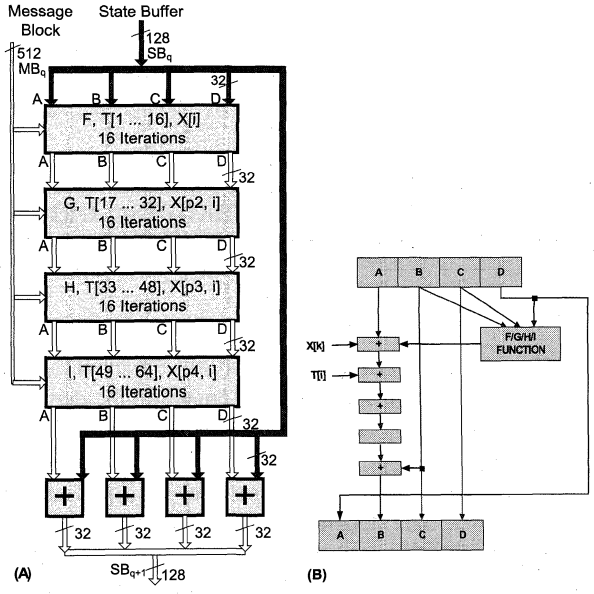

Fig. 1 (A) General block diagram of the MD5 algorithm. (B) Computational structure of each round.

3. Proposed Hardware Architecture and FPGA Implementation

Three hardware architectures are developed:

- Architecture 1 - the basic modular architecture with minimal FPGA gate requirements, which is a straightforward design,
- Architecture 2 - partially unrolled architectures, which uses unrolling technique to improve the throughput of the straightforward design, and
- Architecture 3 - final architecture that uses a new functional description.

Each one of these architectures was developed to obtain a high throughput, using minimal hardware resources and decreasing critical path.

The MD5 architectures are written in VHDL and simulated in FPGA Advantages 6, and implemented in Xilinx ISE 9 for the measurement of hardware parameters such as used logic and clock frequency. The proposed architectures are implemented on a Virtex-II Xilinx FPGA (XC2V1000-6) for the purposes of validation and comparison between the proposed architectures. In the Sect. 4, comparisons against related work are made.

Architecture 1: *Basic Modular Architecture.*

This hardware architecture is a straightforward implementation of MD5 algorithm that besides providing a basic work platform is optimized to achieve better performance, see Fig. 2. The throughput is improved by using modular parallelization, defining data buses and designing specialized functional modules. Also, paths are balanced and required latency is reduced to have the high performance and compact architecture. The critical path is on the MD5_Round_v1 module, which executes main operations

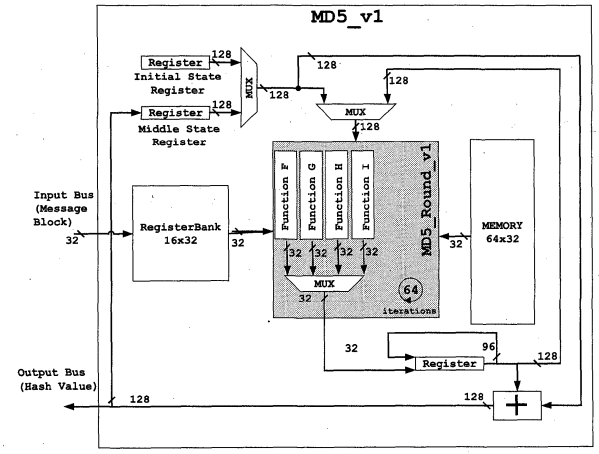

Fig. 2 Block diagram of the MD5_v1 architecture.

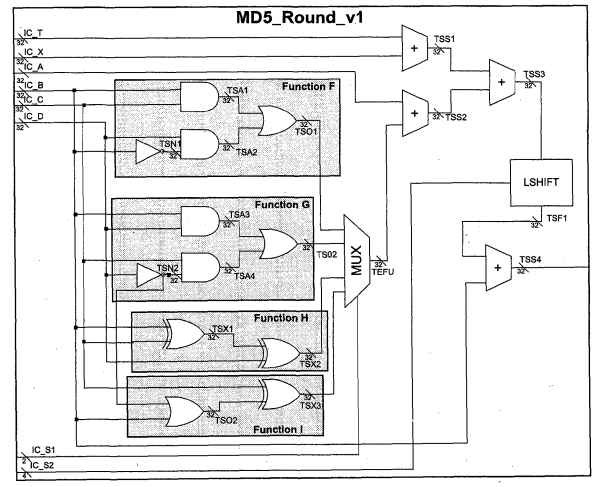

Fig. 3 Block diagram of the MD5_Round_v1 module.

Table 1 Implementation results of the MD5 algorithm for the three proposed architectures.

Design Parameter	MD5_v1	MD5_v2_4l	MD5_v3
Period (ns)	19.38	38.31	31.06
Frequency (MHz)	51.60	26.10	32.18
Slices	899	1387	2158
Clock cycles	65	16	16
Throughput (Gbps)	0.406	0.836	1.030
Throughput/Area (Gbps/Slices x10-3)	0.45	0.60	0.47

and updates state buffers, see Fig. 3.

A Post-Place & Route (P&R) simulation model was created to validate the operation of each design. The implementation data taken from the P&R report are shown in Table 1. By analyzing the MD5_v1 hardware architecture, we identified that using partial unrolling could improve the throughput of the hardware architecture.

Architecture 2: *Partially Unrolled Architecture.*

There are three versions: MD5_v2_2l, MD5_v2_4l, and MD5_v2_8l, which are 2-loop, 4-loop and 8-loop par-

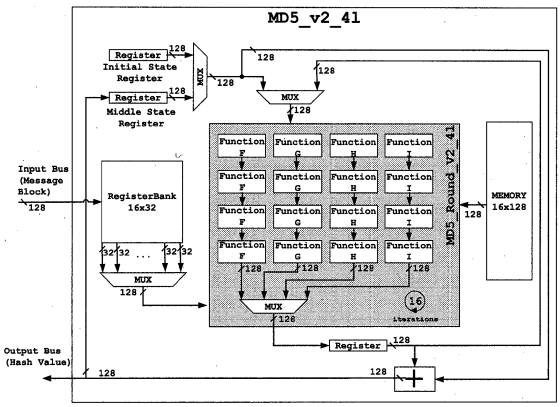

Fig. 4 Block diagram of the MD5_v2_4l architecture.

tially unrolled architectures that reached throughputs of 0.632, 0.836, and 0.789 Gbps, respectively. Figure 4 shows the MD5_v2_4l architecture, which is a 4-loop partially unrolled. The main module of this architecture is MD5_Round_v2_4l that was obtained by modifying MD5_Round_v1. Despite of the feedback and data dependence of the MD5 algorithm, the partially unrolled implementations improve the throughputs, by using more hardware resources. Results indicate that unrolling or using more hardware resources does not necessarily improve the overall performance when comparing among architectures with different levels of unrolling. For example, MD5_v2_4l improves the performance of MD5_v2_2l (four loops versus two loops), but MD5_v2_8l does not improve the performance of MD5_v2_4l (eight loops versus four loops). The 4-loop partially unrolled architecture, MD5_v2_4l, has the best throughput, reaching 0.836 Gbps. From these results it can be seen that, for this type of algorithms, fully unrolled architecture does not necessarily leads to higher throughput, being necessary to find a good trade-off between the number of unrolled loops and hardware resources. MD5_v2_4l is the partially unrolled architecture with the highest throughput. Next, a new architecture is proposed, which has a shorter critical path, thus a higher throughput.

Architecture 3: *Final Proposed Architecture.* In this part, the final MD5 architecture, MD5_v3, is developed by modifying the MD5_v2_4l architecture. In order to increase the throughput, it is needed to reduce the critical path. For reducing critical path, the general idea is to group logic components, by dividing its MD5_Round_v2_4l module into several new functional blocks (RFx). This division depends on the unrolled-loop number; therefore there are four new functional blocks. Figure 5 shows MD5_v3 architecture, which consists of four new RFx functional blocks. Each of these RFx modules is integrated by 4 loops, which are partially unrolled. Implementation results of the MD5_v3 are shown in Table 1, reporting a throughput of 1.030 Gbps.

The MD5_v3 critical path is shorter than the critical path of MD5_v2_4l architecture, due to the new functional description. The feedback in each RFx shows a reduced path time, each RFx is used four times, and after, the next

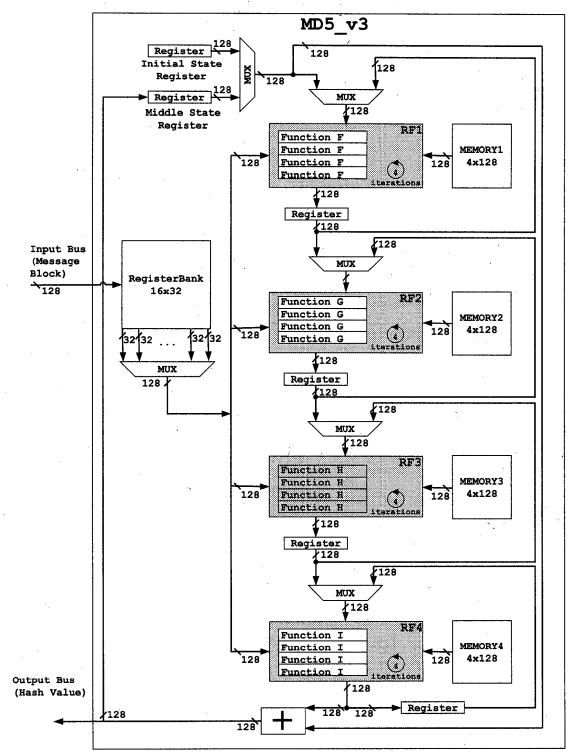

Fig. 5 Block diagram of the MD5_v3 architecture.

RFx is used. The main goal of this work is to achieve high throughput, for ciphering data at a rate of 1 Gbps. The unrolled architectures report limited throughput, because they connect more hardware resources with large paths. The architecture MD5_v3 reduces this path by proposing a new functional description, which is based on partially unrolled loops. In Table 1, there are two metrics than can be highlighted: hardware resources and implementation efficiency (throughput/area); MD5_v1 presents the most compact architecture, but with a low throughput, whereas MD5_v2_4l has the highest efficiency, using adequately fewer hardware resources to compute the hash result. As result, the 4-loop partially unrolled architecture (see Fig. 5) improves the throughput significantly at the cost of higher FPGA resource requirements. Although MD5_v3 utilizes the same number of unrolled loops that MD5_v2_4l, the specialization of functional blocks results in a reduced critical path and a performance improvement. This new functional description is the main contribution of this work, and it can be used for designing hardware architectures with high throughput for other hash algorithms or processes, such as SHA-1, SHA-2 or RIPEMD-160 algorithms, which present compression functions with similar structure to the one of the MD5, see Fig. 1 (B). For example, the compression function for the SHA-2 algorithm (Fig. 6) has a basic structure similar to MD5. It updates 32-bit words by executing operations such as additions module 2^{32}, rotations and nonlinear functions.

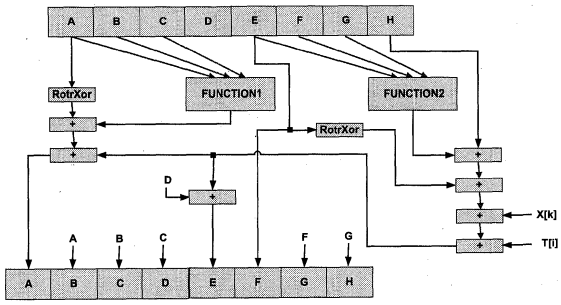

Fig. 6 Compression function of the SHA-2 algorithm for 512 bits.

Table 2 Implementation results of the MD5_v3 architecture. Note: Freq = Frequency, Thrput = Throughput.

Design Parameter	MD5_v3 Virtex	MD5_v3 Virtex-II	MD5_v3 Virtex-4	MD5_v3 Virtex-5
Period (ns)	46.06	31.06	23.41	18.69
Freq. (MHz)	21.70	32.18	42.71	53.49
LUTs	3863	3879	3872	1034 SRs
Slices	2201	2158	2157	3466 SLs
Clock cycles	16	16	16	16
Thrput. (Gbps)	0.694	1.030	1.366	1.711
Thrput/Area	0.31	0.47	0.63	1.65 /SR
(Gbps x 10-3)	/Slice	/Slice	/Slice	0.49 /SL

4. Comparisons

In Sect. 3, the MD5 hardware architectures were implemented on a Virtex-II. In this section, the final proposed MD5 hardware architecture is synthesized, mapped, placed and routed on Virtex, Virtex-II, Virtex-4 and Virtex-5 Xilinx FPGAs (XCV1000-6, XC2V1000-6, XC4VLX80-11 and XC5VLX50-3) to validate the hardware architecture and to have a fair comparison against other works, see Table 2. These implementation results show an improved throughput only by changing FPGA technology. Hardware implementations for Virtex-II and Virtex-4 use a similar amount of hardware resources, but the implementation on Virtex-4 reports a higher throughput. The implementation on Virtex-5 uses a different technology, it uses slice registers (SRs) and slice LUTs (SLs), achieving the highest throughput.

In Table 3, measurements for different MD5 implementations of commercial and research works are illustrated, and although they use different techniques and hardware devices, they have been considered for comparison purposes. These MD5 architectures have different clock frequencies, which depend on the particular design and technology. In general, most of the hardware and software implementations of the hash functions do not have greater throughputs than 1 Gbps, except the commercial implementation in [13]. However, comparing implementations in the same technology, MD5_v3 reports an excellent performance, showing the benefits of using the new functional description.

Considering unrolled architectures, [5] reports a fully unrolled architecture on Virtex, which uses 4763 slices and has a throughput of 0.354 Gbps. This fully unrolled architecture and the partially unrolled architectures proposed in this work (MD5_v1 and MD5_v2_12/4/8) report a low throughput, which is affected by large critical paths, due to the connecting of more hardware resources for placing and routing. MD5_v3 with the new functional description helps unrolled architectures to improve the throughput. Comparing against the completely unrolled architecture described in [5], MD5_v3 uses less than half of hardware resources, yet achieves almost twice the throughput using the same technology (Table 3).

Table 3 Results comparison of the MD5 implementations on FPGA devices.
Note: Freq. = Frequency, Thrput = Throughput, I = Iterative, U = Unrolled, uP = Processor, P = Pipelined.

Work-Device	Type	Freq. (MHz)	Clock cycles	Thrput. (Gbps)
[1]-Virtex-II 3000	I	60.20	66	0.467
[6]-Altera 1000	I	18.00	65	0.142
[7]-Virtex-II Pro 100	I	106.60	197	0.277
[8]-Virtex-II 250-5	I	81.00	66	0.627
[9]-Virtex-II-6	I	96.00	66	0.744
[10]-N.A.	I	100.0	64	0.780
[5]-Virtex 1000	U	71.40	1	0.354
[11]-Virtex-II 4000	P	78.30	66	0.607
[12]-Virtex-II Pro	uP	97.64	N.A.	0.331
[13]-Virtex-4-11	I	N.A.	66	0.945
[13]-Virtex-5-3	I	N.A.	66	1.345
This work - Virtex-II 1000-6	U	32.18	16	1.030
Virtex-4 VLX80-11	U	42.71	16	1.366
Virtex-5 VLX50-3	U	53.49	16	1.711

5. Conclusions

An efficient hardware design of the MD5 algorithm was presented, which is based on a new functional description that helps to improve the performance in structures with high dependence of data. This work shows that logic replication or unrolling is not sufficient to increase the throughput of the architecture, but it is necessary to group the logic blocks, to make new functional description and to decrease the critical path time. Implementation results showed a significant gain compared to the existing commercial cores and related academic works.

References

[1] J.M. Diez, S. Bojani, L. Stanimirovi, C. Carreras, and O. Nieto-Taladriz, "Hash algorithms for cryptographic protocols: FPGA implementations," 10th Telecommunications Forum TELFOR'2002, Nov. 2002.

[2] R.R. Taylor and S.C. Goldstein, "A high-performance flexible architecture for cryptography," Proc. Workshop on Cryptographic Hardware and Embedded Systems 1999 (CHES), 1999.

[3] M. Szydlo and Y.L. Yin, "Collision-resistant usage of MD5 and SHA-1 via message prepro-cessing," IACR Eprint Archive, 2005. Available at: http://eprint.iacr.org/2005/248.pdf

[4] R. Rivest, "The MD5 message-digest algorithm," RFC 1321, MIT and RSA Data Security, April 1992.

[5] J. Deepakumara, H.M. Heys, and R. Venkatesan, "FPGA implementation of MD5 hash algorithm," Proc. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2001), May 2001.

[6] K.K. Yong, W.K. Dae, W.K. Taek, and R.C. Jun, "An efficient implementation of hash function processor for IPSec," Proc. Third IEEE Asia-Pacific Conference on ASICs, Aug. 2002.
[7] J. Lu and J. Lockwood, "IPSec implementation on xilinx virtex-II pro FPGA and its application," Reconfigurable Architectures Workshop (RAW), April 2005.
[8] Alma Technologies, MD5 high performance hash function core, Datasheet, 2002. Available at: http://www.alma-tech.com/Data-Sheets/MD5_pre_sales.pdf
[9] Helion Technology, High performance MD5 hash core for xilinx FPGA, Datasheet, 2003. Available at: http://www.heliontech.com
[10] Jetstream Media Technologies, JetHash family: Secure hashing and HMAC cores, Datasheet, 2006. Available at: www.security-cores.com
[11] K. Jrvinen, M. Tommiska, and J. Skytt, "Hardware implementation analysis of the MD5 hash algorithm," Proc. 38th Hawaii International Conference on System Sciences, 2005.
[12] T.S. Ganesh, M.T. Frederick, A.K. Somania, and T.S.B. Sudarshanb, "HashChip: A shared-resource multi-hash function processor architecture on FPGA," Integration: The VLSI Journal, vol.40, no.1, pp.11–19, 2007.
[13] Helion Technology, MD5 hashing cores, Datasheet, 2007. Available at: http://heliontech.com/md5.htm