RAM-based Reconfigurable Implementation of the MD6 Hash Function

Xianwei Gao¹, Jianxin Wang¹, Haiwen Ou² and Xiuying Li¹

¹Dept. of Electronic and Information Engineering
²Dept. of Information Security
Beijing Electronic Science and Technology Institute
Beijing, P.R. China
gao_xianwei@163.com

Abstract—Recent breakthroughs in cryptanalysis of standard hash functions like SHA-1 and MD5 raise the need for alternatives. The MD6 hash function is developed by a team led by Professor Ronald L. Rivest in response to the call for proposals for a SHA-3 cryptographic hash algorithm by the National Institute of Standards and Technology. The hardware performance evaluation of hash chip design mainly includes efficiency and flexibility. In this paper, a RAM-based reconfigurable FPGA implementation of the MD6-224/256/384/512 hash function is presented. The design achieves a throughput range from 118 to 227 Mbps at the maximum frequency of 104MHz on low-cost Cyclone III device. The implementation of MD6 core functionality uses mainly embedded Block RAMs and small resources of logic elements in Altera FPGA, which satisfies the needs of most embedded applications, including wireless communication. The implementation results also show that the MD6 hash function has good reconfigurability.

Keywords—hash; SHA-3; MD6; reconfigurable; iterative; RAM-based

I. INTRODUCTION

Hash functions are a fundamental primitive in modern cryptography, such functions process data of finite length and produce a small fixed size output referred to as a digest or hash. Those functions must meet four requirements in order to be employed in modern systems. First, the function must be preimage resistant. This implies that for a given fixed size generated output value, it must be computationally infeasible to find the input data to the function that generated that value. Second, function should be second preimage resistant. Given a fixed input value, second preimage resistance implies that it should be infeasible to find another input value that results in the same hash as the first input value. Third, the function must be strongly collision resistant, which means that it must be computationally infeasible to find two differing input values which hash to the same output value. Lastly, it is important for the cryptographic hash function to be practically efficient, meaning that computing a hash does not become a bottleneck in the system [1, 2].

In recent years there have been a series of serious and alarming cryptanalytic attacks on several commonly-used hash functions, such as MD5, SHA-0, and SHA-1 [3]. These culminated with the celebrated work of Wang, Yin, and Yu from 2005, which demonstrated relatively efficient methods for finding collisions in the SHA-1 hash function [4]. Although there are several cryptographic hash functions, such as the SHA-2 family [5], that have not yet succumbed to such attacks, the U.S. National Institute of Standards and Technology (NIST) put out a call in 2007 for candidate proposals for a new cryptographic hash function family, to be known as SHA-3 [6]. The candidate algorithm is required to be a drop-in replacement for SHA-2, meaning that it takes the same inputs as SHA-2, and produces message digests of 224, 256, 384, and 512 bits. Not only is the candidate algorithm expected to be at least as secure as SHA-2, it also should show good performance on various platforms.

Submissions were due October 31, 2008, with a list of candidates accepted for the first round published December 9, 2008. Of the 64 who submitted entries, 51 were accepted. The timeline for this competition calls for a peer/community review of the candidates for both performance and security properties to be performed until the second quarter of 2010, NIST will likely select one as the new SHA-3 hash algorithm sometime in the year 2012.

A team from MIT led by Ron L. Rivest (founder of RSA Data Security and author of the popular MD5 hash function) has put forth the MD6 algorithm [7]. It utilizes the design principle of that a larger number of simple rounds is more secure than fewer complex rounds.

The primary requirement for a hash function is the security. Unfortunately, evaluating the security of an algorithm is lots of hard work. As the final SHA-3 candidate has to have good engineering properties, perhaps we can first analyze the engineering aspects of each candidate, it’s more efficient to first select on the engineering properties and then perform the security analysis on the candidates that have good engineering properties [8].

The published NIST SHA-3 criteria about cost and implementation characteristics of the candidates include computational efficiency, RAM requirements, flexibility, and simplicity [6]. In addition, the submitted algorithm may include a tunable security parameter, such as the number of rounds, which would allow the selection of a range of possible security/performance tradeoffs.

In this paper, a RAM-based reconfigurable FPGA design and implementation of the MD6 hash function is presented. The basic iterative architecture target low-cost device was implemented, which can be configured into different modes with different throughput. It has to be noticed that, to the best of our knowledge, this is the first report about the RAM-based reconfigurable hardware implementations of the MD6 hash function design.
II. DESCRIPTION OF MD6 COMPRESSION FUNCTION

The MD6 hash function is comprised of two main components: compression function and the mode of operation. The MD6 compression function maps 89 64-bit words of input (64 words of data B, 8 words for the key K, 15 fixed words Q, and 2 auxiliary information words) down to 16 64-bit words of output. Therefore in practice it is a function $f : \{0, 1\}^k \rightarrow \{0, 1\}^c$ with $k = 8w$, $n = 74w$, and $c = 16w$ (where $w = 64$), as shown in Fig.1.

The compression function input.

Note that although it takes in 89 words of input, 15 words are fixed for the constant Q, hence the compression function takes the “feedback tap positions”, $t_0(17)$, $t_1(18)$, $t_2(21)$, $t_3(31)$, $t_4(67)$, each in the range 1 to $n-1 = 88$, as parameters. The compression function is computed as shown in Fig. 2.

The compression function input.

To create a fixed-length digest from an arbitrarily long input, the values $r_{i,a}$, $l_{i,a}$ are the initial right-shift amount and the subsequent left-shift amount within the step with index $i$; the index $i-n$ is taken modulo $c = 16$, as shown in Fig.3.

The MD6 mode of operation is thus optionally parameterized by the integer $L$, $0 \leq L \leq 64$, which allows a smooth transition from the default tree-based hierarchical

The compression function is computed as shown in Fig. 2.

The MD6 mode of operation describes how to apply the fixed-length compression function repeatedly in order to create a fixed-length digest from an arbitrarily long input.

The standard mode of operation is a hierarchical, tree-based mode of operation. MD6 provides a height-limiting parameter $L$. The MD6 mode of operation is thus optionally parameterized by the integer $L$, $0 \leq L \leq 64$, which allows a smooth transition from the default tree-based hierarchical
mode of operation (for L=64, as shown in Fig.4(a)) down to an iterative mode of operation (for L=0, as shown in Fig.4(b)).

III. FPGA IMPLEMENTATION

Cryptographic solutions using software methods can be used for those security applications where data traffic is not too large and low encryption rate is tolerable. On the other hand, hardware methods offer high-speed solutions making them highly suitable for applications where data traffic is fast and large data is required to be encrypted in real time. ASIC and FPGA are two alternatives for implementing cryptographic algorithms in hardware. FPGA offer several benefits for cryptographic algorithm implementations over VLSI as they offer high flexibility. Moreover, basic primitives in most cryptographic algorithms can efficiently be implemented in FPGA [9-11]. The choice of FPGA as a target platform for hash algorithm implementations appears to be a practical solution for embedded systems.

Reconfigurable architecture design is an interesting research topic of cryptosystem recently; among the existing reconfigurable designs, some focus on hash function, e.g., SHA-2 [12-15]; some focus on block cipher algorithm, e.g., AES [16,17]; some focus on public-key cryptography, e.g., elliptic curve cryptography [18,19]. Reconfigurable design can be configured for different mode up to the requirement of cryptosystem, provide different performance and flexibility, which is much more attractive than fixed architecture design.

A. Hash implementation categories

According to [12], FPGA implementation of hash function can be classified into three categories, as shown in Fig.5 (a) to (c).

- Fully autonomous implementation. Such hardware implementations include the complete functionality of a hash function. That means the input message can be loaded piecewise into the module and it delivers the message digest as output. All hash calculations happen exclusively within the hardware module. If integrated in a system, the achievable throughput of a fully autonomous implementation depends on the speed of the hardware module itself and the speed of the (system dependent) data interface delivering the input message.

- Implementation with external memory. These implementations use external memory to hold intermediate values during the hashing of a message. The implemented hardware itself normally consists of a similar hardware implementation of the compression function, some registers for short-lived temporary values, and a memory controller for access to the external memory. Such implementations can load the input message either over a dedicated interface (similar to a fully autonomous implementation) or from the external memory. In order to reach the maximal throughput of the hardware module, the external memory must be sufficiently fast.

- Implementation of core functionality. Such implementations comprise only important parts of the hash function (e.g., the compression function), which normally allows to get a first-order estimate of the performance figures of full implementations.

In this work, we will assume that the initial message has been properly padded and parsed, and only the MD6 core functionality of L = 0 operation is implemented.

B. RAM-based reconfigurable design of MD6 hash function

It is known that the final performance of cryptographic algorithms heavily depends on the efficiency of their underlying field arithmetic-MD6 is no exception. A feature of MD6 is that the input message block size to the compression function is very large - 512 bytes (not 512 bits). RAM-based design uses block RAM arrays to replace registers for the input data, output data and intermediate data; which allows significant reduction in the area usage.
Circuit, RAMs and Hash Computation Circuit, FSM is used for this purpose. The Mode signal firstly selects the desired message digest length, and then the Control Circuit coordinates all the system operations and processes. When mode = “00", MD6-224 needs to be performed; similarly, if mode = “01" MD6-256 will be performed, and so on, thus making the design tunable and reconfigurable. Six synchronous “256×64” bit dual port Ramps (Ram0 to Ram5) are used as data memory. In fact, one RAM is theoretically enough for MD6 hash computation; others are duplicated and used to speed up the hash computation. Each single Block RAM interface consists of a 64-bit wide data in and out interfaces, 8 bit read and write address, read signal, write enable, and a synchronous clock interface. Because of the hardware constraints and timing issues involved with accessing memory, only one element of a Block RAM may be accessed in any given clock cycle. Reading from it requires supplying the address one clock cycle before the data appears at the output. This feature can be viewed as a pipeline stage introducing a delay of one clock cycle. Fig.6 (b) describes the RAM address space, divided into several segments.

The Hash Computation Circuit includes all logical steps needed for accomplishing the md6 compression function computation, as shown in Fig. 7.

![Figure 7. Architecture of the MD6 iterative logic.](image)

The design implemented one-step computation of the MD6 hash function in one clock cycle; it utilizes less area but consumes more clock cycles resulting on a relatively low speed. On the rising edge of every clock cycle, the one-step logic compute the six input data(Ai-n, Ai-0, Ai+1, Ai+2, Ai+3, Ai+4, Ai+5)reading in parallel from the six synchronous RAMs [RAM0-RAM5] with Sa-n and write the result Ai to the RAMs immediately, such read-compute-write pattern can save a lot of registers.

Round constants S[n] are generated on-the-fly with the round operation goes, this approach is to combine the round constants arrangement and round computation into one single process; in each round the round computation process is performed right after the round constants is constructed, and the current round constants is to be replaced by the next round constants, and round computation is done again, cycling the same process until end of the required computation round.

C. Signals on top module

According to the above mentioned design method, the top module's input and output signals are:

1) Input signals and signals meaning:
   a) Clk: The main system clock.
   b) Clr: Low state resets the chip.
   c) Rd: Reads the result to the output port.
   d) Wen: RAM write enable. Active high write.
   e) Waddr: RAM write address port.
   f) Raddr: RAM read address port.
   g) Data: RAM write data port.
   h) Mode: Selects the desired message digest length.
   i) En: High state enables hash process.

2) Out signals and signals meaning:
   a) digest: Hash text.
   b) rdy: Generated by the chip. Low state indicates hash computation in progress. Change from low to high means end of one block hash progress.

The operation of the basic iterative module can be summed up as the following steps:

- Reset all the state and registers;
- Input the first 89w vector into the RAM at locations [0...88];
- The Mode signal selects the desired message digest length. When mode = “00", MD6-224 needs to be performed; similarly, if mode = “01" MD6-256 will be performed, and so on;
- The hash iterative logic read the six input words simultaneously at addresses: 0, 22, 58, 68, 71 and 72; execute one step computation; write the result to the RAM at address 89;
- The read addresses and write address added by one, repeat another step computation; it has to be noted that the write address must be restricted at 89 to 255 before the last round. At the last round, write the 16 chaining variables to the addresses from 25 to 40, and valid the rdy signal;
- Input the next 48w(3072 bits) block vector into the RAM at addresses [41...88], repeat step3 and step4;
- When the last block hash process ended, read out the hash value.

IV. IMPLEMENTATION RESULTS

We implemented the reconfigurable design that performs both MD6-224/256/384/512 using target device of Altera Stratix III EP3SE50F484C2 and Cyclone III EP3C5F256C6. ALM (adaptive logic module) is the basic building block of logic in the Stratix III architecture, which contains a variety of look-up table (LUT)-based resources that can be divided between two combinational adaptive LUTs (ALUTs) and two registers. Cyclone III devices are Altera's third-generation of low-cost FPGAs offering lower power with greater logic and memory. The described circuits have been implemented in VHDL and simulated, synthesized, placed,
and routed using Quartus II Web Edition version 8.1. The architecture was simulated for verification of the correct functionality, by using the test vectors provided by the MD6 report [7]. The reconfigurable implementation result is presented in Table I. Five performance metrics such as area, memory, registers, clocking frequency (MHz) and the throughput (Mbps) were computed.

When executing the iterative MD6 design, each block requires 1540/1668/2180/2692 clock cycles; the maximum throughput can be computed according to equation:

\[
\text{Throughput} = \frac{\text{maximum clock frequency}}{\text{number of clock cycles}} \times \text{Message block size}.
\]

Note that when \( L = 0 \), the input is divided into 48-word (three-chunk, 3072 bits) data-blocks.

On Cyclone III device, the reconfigurable design requires 1789 LEs, achieves a throughput ranges from 118 to 227 Mbps at the maximum frequency of 104MHz; on Stratix II device, the reconfigurable design requires 923 ALMs, achieves a throughput ranges from 225 to 394 Mbps at the maximum frequency of 198MHz.

### Table I. Performances of the MD6 Reconfigurable Implementation

<table>
<thead>
<tr>
<th>Device</th>
<th>Performance metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area</td>
</tr>
<tr>
<td>EP3SE50</td>
<td>923 ALMs</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>EP3C5</td>
<td>1789 LEs</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^a\) MD6-224; \(^b\) MD6-256; \(^c\) MD6-384; \(^d\) MD6-512.

V. Conclusion and Future Work

Most of the low-end applications do not require high encryption speeds. As a result, a low-cost RAM-based reconfigurable FPGA implementation for the MD6 hash family was developed and evaluated. Our implementation firstly focuses on embedded functions inside of the FPGA, such as large dual-ported block RAM with the goal of minimizing the use of registers and look-up tables that could otherwise be used for other functions. Secondly, we present a reconfigurable architecture which will be able to perform either a MD6-224/256/384/512 hash operators without the necessity to completely reconfigure the internal layout of the target device, compared with separate implementations, the reconfigurable scheme not only provides full hardware support for the MD6 implementing modes, but also reduces the required logic elements on equivalent frequency, and provides a flexible scheme for different requirements. Future works include loop-unrolled reconfigurable design and optimization of the MD6 hash function; In addition, secure implementation and low-power design of the MD6 algorithm is another important issue in addition to area-throughput tradeoffs.

ACKNOWLEDGMENT

This work was partially supported by the National 863 project (No. 2007AA014Z472) and funded by the Key Laboratory of Security and Privacy (No.YZD0809), the General Office of Central Committee, P.R. China.

REFERENCES