

Mapping the MD5 Hash Algorithm onto the NAPA Architecture

Jeffrey M. Arnold
Consultant
jmarnold@znet.com

1 Introduction

One-way hash functions are typically used to provide a “fingerprint” for a message or file, M , that is unique[3]. A one-way hash function, $H(M)$, computes a fixed-length hash value, h , from an arbitrary length message, M , such that:

- Given M , it is easy to compute h .
- Given h , it is hard to compute M .
- Given M , it is hard to find another message, M' , such that $H(M) = H(M')$.

“Hard” is taken to mean computationally infeasible, although not provably impossible.

The MD5 hash algorithm processes the input text in 512-bit blocks divided into sixteen 32-bit words. The output of the algorithm is a set of four 32-bit words concatenated to form a 128-bit hash value.

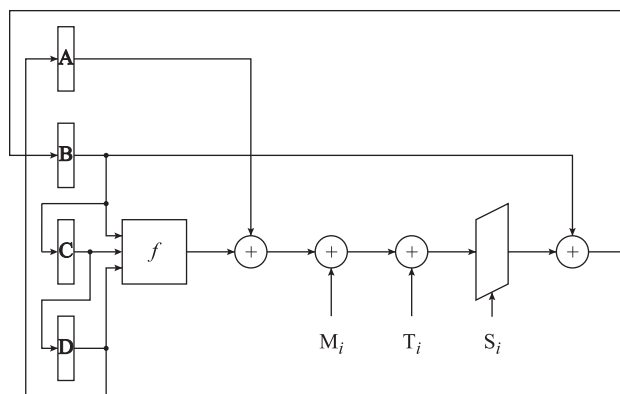


Figure 1: Basic MD5 operation

The outer loop processes each block of data in four “rounds”, with each round applying the kernel operation 16 times. The kernel operation is shown in Figure 1. At the beginning of the algorithm the four “chaining” variables $\{A, B, C, D\}$ are initialized to known values; at the end of each step these registers are updated according to the logic shown. The function f

is a non-linear bit-wise function of three inputs; four different functions are used, one for each round. The sums are 32-bit operations. The index i indicates the round number (1-4) and the step within the round (1-16). Input M_i represents one word of the message, selected from the current message block according to a fixed schedule. Input T_i is one of 64 32-bit constants from a fixed table. Input S_i is one of 16 rotate distances, again chosen from a fixed schedule[1].

National Semiconductor’s Adaptive Processing Architecture (NAPA) integrates a Fixed Instruction set Processor (FIP), an Adaptive Logic Processor (ALP), memory and other support circuitry into a single reconfigurable computing device. In the NAPA1000 the FIP is a small 32-bit RISC microprocessor and the ALP is a 64×96 array of fine grain reconfigurable logic cells. The NAPA1000 also contains two banks of 2048×32 Pipeline Memory Array (PMA), eight banks of 256×8 Scratchpad Memory Array (SMA), and one bank of 1024×32 Local Memory Array (LMA). External to the NAPA1000 are two banks of DRAM and an interface to a host computer. The Toggle Bus transceiver is the interface to a multi-stage interconnect network, and is capable of performing arbitrary reflections and rotations on 32-bit words. The Reconfiguration Pipeline Control unit (RPC) can also serve as a DMA engine[2].

2 Implementation

At first glance it would seem that MD5 is a poor match for reconfigurable computing. There is little parallelism and the feedback into B severely limits the opportunity for pipelining. However, the bit-wise non-linear functions, the variable distance rotate, the re-use of message data and the indirect addressing are well suited to implementation in the NAPA architecture.

One strategy for mapping an algorithm onto the NAPA architecture is to perform the inner loops on the ALP and use the FIP to control the outer loop. For the MD5 application we define the “inner loop” to be four rounds. Since each word of the message data

is visited once per round, we exploit locality by copying the current block into the PMA. While message block n is being processed by the ALP, the RPC will copy the block $n + 1$ into the other bank of the PMA. Once block n has been processed and $n + 1$ copied, the roles of the two PMA banks are reversed, with block $n + 1$ being processed while $n + 2$ is fetched. The FIP initiates the DMA and signals the ALP to begin processing the data.

The chaining variables are stored in registers within the ALP and are loaded and retrieved by the FIP using the ALP register access mechanism. The variable distance rotate operation is performed in one clock cycle using the Toggle Bus Transceiver. The shift distances for all 64 steps are stored as a table in one Scratchpad Memory (SMA). The SMA is addressed by a concatenation of the 2-bit round number and the 4-bit step number. The 32-bit wide “T” constant table is stored across four SMAs, and is indexed using the same address as the shift table. The message block is stored in the PMA, and is loaded by the RPC from external DRAM using DMA. The message data is read by the ALP using indirect addressing through an index table stored in another SMA.

The algorithm kernel is implemented on the ALP in a three clock cycle execution pipeline: 1) the non-linear function, f , and the first three additions; 2) the rotate operation; 3) the final summation and the load into the chaining registers. Because of the feedback the pipeline can only be initiated once every three clock cycles. Therefore, the time to process one 512-bit message block is

$$\begin{aligned}
 t &= (4 \text{ rounds}) \times (16 \text{ steps}) \times (3 \text{ cycles}) \\
 &= 192 \text{ clock cycles} \\
 &= 3.8 \mu\text{sec. @ } 50 \text{ MHz}
 \end{aligned}$$

for a maximum throughput of 16.8 MB/sec.

In steady state there are two concurrent tasks: 1) message block $n + 1$ is streaming from the external DRAM into the PMA; and 2) block n is being processed by the ALP. The FIP is responsible for initiating both tasks and synchronizing their completion. During the stream operation we maximize the memory bandwidth available to the RPC by minimizing the FIP’s bus traffic using the “wait for interrupt” mechanism, which puts the FIP into a suspended state until the appropriate interrupt arrives. The FIP will wait for an interrupt from the RPC indicating completion of the DMA operation. Once the DMA operation is complete, however, memory bandwidth is no longer an issue, and we can use a more efficient polling approach to determine when the ALP has completed processing

the current block of the message.

3 Results

Table 1 shows the performance results obtained using the NAPAsim architecture simulator. The “FIP” row shows a C language implementation running entirely on the FIP; “P150” shows the benchmark C implementation running on a 150MHz Pentium; “ALP-P” shows the results of moving the inner loop to the ALP but using FIP program control to transfer the message block; “ALP-S1” shows the use of RPC DMA for one message block (64 bytes) per transfer; “ALP-S8” transfers eight message blocks (512 bytes) per transfer. For each implementation we show the number of clock cycles required to process one 16 KByte message, the run time in μsecs and the total throughput in MB/sec.

	# Cycles	Time	Throughput
FIP	2814015	56280	0.29
P150	1840000	12160	1.35
ALP-P	266189	5324	3.08
ALP-S1	99107	1982	8.26
ALP-S8	51001	1020	16.06

Table 1: Performance results

For small DMA transfers (ALP-S1) the performance is dominated by the overhead of the DMA. As the transfer size is increased to multiple message blocks (ALP-S8), the performance quickly approaches the theoretical maximum of 16.8 MB/sec.

4 Acknowledgements

This work was supported by National Semiconductor Corp., DARPA through Contract DAB63-94-C-0085 to NSC. Special thanks to the NAPA1000 team, including Stephen Bade, Tim Garverick, Maya Gokhale, Edson Gomersall, Harry Holt, Jeff Hutchings, Mark Landguth and Charlé Rupp,

References

- [1] Rivest, R.L., “The MD5 Message-Digest Algorithm,” Network Working Group RFC1321, April 1992.
- [2] Rupp, C. et al., “The NAPA Adaptive Processing Architecture,” Submitted to FCCM’98.
- [3] Schneier, B., *Applied Cryptography*, John Wiley and Sons, New York, 1996.