1L-5

SHA Implementation on Soft Core Processor for Hardware/Software Optimization

Hoang Anh Tuan†, Katsuhiro Yamazaki † and Shigeru Oyanagi †

Abstract This paper describes a research on hardware/software division optimization for the SHA algorithm using MicroBlaze soft core processor. Nine designed patterns are given with different hardware/software rates, which affect the hardware size, used memory size and the computation time of the SHA. The tradeoff among those factors and the suitable hardware modules are analyzed and given. The results show that the speedup of the algorithm significantly depends on speed of computational modules with small memory. The results show that the effect of hardware implementation increases correspondingly to the hardware size and decrease correspondingly to the number of data transfer between hardware and software.

1. Introduction

The FPGA technology allows an application can be implemented into both in Hardware or Software. It helps to increase speeds of the applications, aims to the real time applications. In that scene, the optimization and tradeoff among hardware, software, speedup and memory are important. This research aims to optimize hardware and software for the co-design system, which helps the applications effectively implemented onto the FPGA. The research utilizes the MicroBlaze soft core processor and the user designed hardware, mounted into many patterns for the SHA-1 algorithm.

2. Hardware/Software Optimization Overview

The Hardware/Software optimization is based on the implementation and analysis of variable patterns of applications. An application will be implemented in both hardware and software in several modules, mounted on the MicroBlaze soft CPU to form some patterns. Hardware costs, memory size and speedup are analyzed to give the tradeoff among those factors. The MicroBlaze soft core processor works with software while variable hardware modules are implemented in hardware.

3. System Hardware Architecture

Figure 1 shows the hardware architecture used in the research. The MicroBlaze is configured as a 32 bit Harvard soft core processor architecture with pipeline. It contains 32 general purpose registers, instruction buffer, PC, decoder, an optimal ALU and 3 buses interfaces. The local memory bus is used for the data transfer between the CPU and RAM, the On-chip Peripheral Bus (OPB) is used for standard in/out ports such as RS232. Besides, there are 8 Fast Simplex Link (FSL) interface, which used for the data transfer among the general purpose registers and the user peripherals using a FIFO memory.

4. SHA-1 algorithm

Figure 2 shows the SHA-1 algorithm which contains preprocessing and hash computation. There are 2 input data for the algorithm: the original data

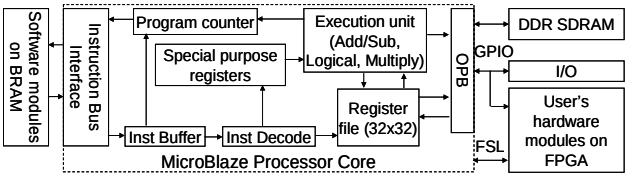

Figure 1. MicroBlaze Soft Core Processor Architecture

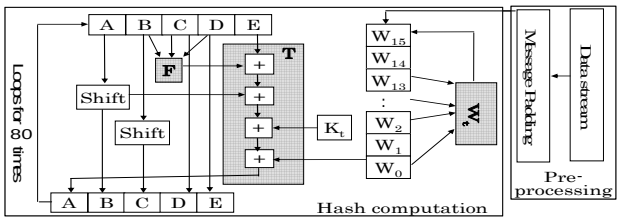

Figure 2. SHA-1 algorithm overview

stream that can be understood as original keys and the digest numbers A~E that constantly started. The preprocessing assure the length of the data stream be multiple of 512 by adding 1, 0s and the original length into it. The hash computation is operated based on the logic operation, plus and shift, divided into four main modules. The lowest level of the calling tree, the function of previous digest values and loop number (F), is called by the intermediate value calculation (T). The constant generation module K_t generates a constant depends on the loop number. The keys can be separately generated from the input data by the key generation (W). Hash computation stays at the top of the tree reuses the calculated values of T and W. It also controls the 80 times of loop for 512 bit data hashing computation. The implemented SHA-1 is separated into modules based on their functions above.

5. User Hardware Implementation Modules

5.1 Computational modules

The computational modules are F and T, which require a very small amount of memory for input, output and intermediate data. Figure 3 shows the design of those computational modules. The state controller controls hardware to receive data from the master FSL interface for several times before writing result back into slave FSL interface. The computation controller is added into the design, making the calculation process became pipelined and parallel with the reading/writing process in order to reduce the circuit delay.

† Ritsumeikan University, Graduate School of Science and Engineering

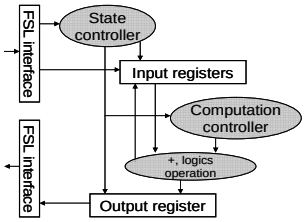

Figure 3. Computational modules (F/T) architecture

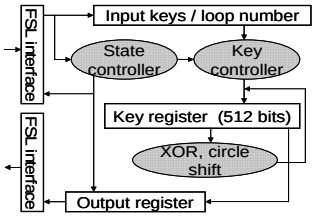

Figure 4. Key generation module (W) architecture

5.2 Key generation module

The key generation module (W) is shown in Figure 4, used to generate the 32-bit child keys from the 4 others. It contains 512-bit memory for the original and child keys. The state controller controls the 16 times original 32-bit keys receiving from the FSL interface and 64 times child key generation. The oldest key (W_0) is replaced after usage using FIFO manner by the child key generated by W_t computation. The key controller makes the memory access (read, computation and write processes) became pipelined.

5.3 Block hashing controller architecture

The hashing module is the combination of the computational and the key generation modules, controlled by a state controller. It controls not only the accessing process of hashed data for the temporary computation module (T) but also the key accessing in the key generation module. The design of T and W allows them to work in parallel with one clock delay overall. In this module, the data transfer is improved using buses or register, making its operation completely independent with the software.

6. Experiment Results and Discussion

The changes from Software (-) into Hardware (H) of the 4 modules of F, T, key generation and hashing controller forms 9 patterns as shown in Table 1. The patterns are ranged from software only in pattern 1 to almost hardware implementation in pattern 9.

Table 2 shows the implementation results of those patterns in terms of hardware size, memory size, execution time and hardware increase effectiveness. Figure 5 shows the relationship among hardware size, memory size and speedup of the application. The best speedup as a whole can be recognized in pattern 9 with the hardware hash computation, in which, 1% of hardware increase results into 0.12 times of speedup, makes the final speedup achieves 16.8 times. Pattern 8 with parallel computation using 2 communication ports also achieves 3 times of speedup. Other patterns from 3 to 7 also have good speedup correspondingly to the hardware size increase.

Table 1. Hardware/Software division patterns

Patterns	Function (F)		Temporary (T)		Key	Hashing
	F	Loop checked	T	Loop checked	generation	controller
(1)	-	-	-	-	-	-
(2)	H	-	-	-	-	-
(3)	H	H	-	-	-	-
(4)	H	-	H	-	-	-
(5)	H	H	H	H	-	-
(6)	-	-	-	-	H	-
(7)	H	H	-	-	H	-
(8)	H	H	H	H	H	-
(9)	H	H	H	H	H	H

Table 2. Patterns implementation results

Patterns	Number of gates		Memory (Flip-Flops)		Execution time (clocks)		Hardware increase effectiveness
	Hardware	Increase	Slices	Increase	Time	Speedup	
(1)	92,623	0 %	557	0 %	23,409	1	-
(2)	108,623	17.3 %	717	28.7 %	23,489	1	1 : 0
(3)	110,989	19.8 %	718	28.9 %	21,109	1.1	1 : 0.06
(4)	122,596	32.4 %	828	48.7 %	18,769	1.25	1 : 0.04
(5)	123,384	33.2 %	824	47.9 %	16,957	1.38	1 : 0.04
(6)	140,061	51.2 %	1,224	119.7 %	15,393	1.52	1 : 0.03
(7)	169,133	82.6 %	1,344	141.3 %	12,927	1.81	1 : 0.02
(8)	172,063	85.8 %	1,450	160.3 %	7,967	2.94	1 : 0.03
(9)	220,176	137.7 %	1,469	163.7 %	1,393	16.8	1 : 0.12

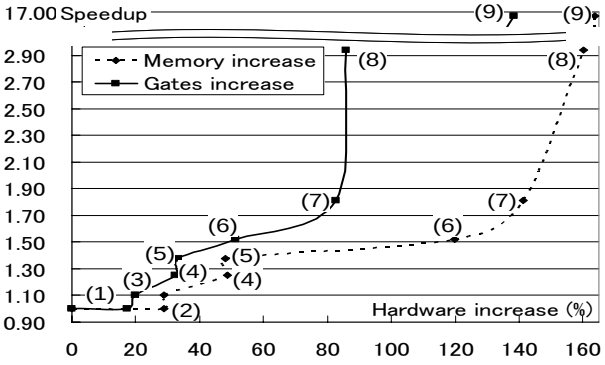

Figure 5. Relationship between hardware increase and speedup

In general, the small computation modules with many data movement between hardware and software have small effect to the speedup of the system due to the latency of huge amount of data transfer through port. However, when those modules are combined together into a united hardware module or working in parallel using several communication ports, the hardware effectiveness will be increased significantly due to the changes in the data transmission methodology, which allows data transfer through wide buses or registers and the parallel execution methodology. Besides, the number of communication ports between soft core processor and user hardware can be reduced in combined hardware module.

7. Conclusion and Future work

This paper described 9 implementation patterns of SHA algorithm for the Hardware/Software division optimization on MicroBlaze. The results show that the effect of hardware implementation increases correspondingly to the hardware size and decreases correspondingly to the number of data transfer between hardware and software. Other algorithms such as MD5, matrix inversion should be implemented into MicroBlaze board in the same manner.

References

[1] Federal Information Processing Standards (FIPS) Publication 108-2, Secure Hash Standard (SHS), U.S. DoC/NIST, Aug, 2002.

[2] Microblaze Processor Reference Guide from Xilinx UG081 (v6.0) June 1, 2006.