The Design of the SHA1 Co-Processor

ECE 111 Final Project

Winter Quarter 2015

Date: March 17th, 2015

Team Members:

Xiaoyu Liu A09819105

Wai Ho Leung A

**Introduction**

SHA-1, which stands for “secure hash algorithm - 1,” is designed by the NSA and published by the United States NIST in 1995. It is used to be the most widely used SHA hash functions. The purpose of this algorithm is to produce a 160 bits, or 20 bytes, hash value, given a message of length less than 2^64 bits.

SHA-1 forms some of the widely used security applications and protocols, including TLS, SSH, SSL, etc. It is also used in distributed version control systems such as git, to identify objects and detect data corruption in case of hardware failure.

One of the importance of SHA-1 is it sets industry standard in cryptography during the 90s because of its stronger resistance to attacks than its ancestor, SHA-0. However, until recent years, researchers suggest SHA-1 might not be secure enough for ongoing uses, thus major tech companies and government agencies stopped accepting SHA-1 related applications. All in all, SHA-1 is still said to be the foundation of modern computer security and cryptography.

**Description of the SHA-1 Algorithm**

Input: M with length < 2^64 bits

Output: 160 bits

Step 1: Pre-processing(M):

1.Add bit 1 to the end of M.

2.Keep adding 0 bit until message is 64 bits less than some multiple of 512.

Step 2: Pre-processing(M), cont:

1. Add 64 bits representation of the length of M at the end, forming a message of size = n\*512, where n is a positive integer.

Ex: M with length 900 bits.

M

1

0

0

0

…

900 bits

59 0’s

900

64 bits

2 x 512 = 1024 bits

Step 3: Initialization(H, K(t), F(t)):

4. H0 = 67452301 H1 = efcdab89 H2 = 98badcfe

H3 = 10325476 H4 = c3d2e1f0

5. 0 ≤ t ≤ 19, Kt = 5a827999 20 ≤ t ≤ 39, Kt = 6ed9eba1

40 ≤ t ≤ 59, Kt = 8f1bbcdc 60 ≤ t ≤ 79, Kt = ca62c1d6

6.0 ≤ t ≤ 19, F(X, Y, Z) = (X AND Y) XOR (~X AND Z)

20 ≤ t ≤ 39, F(X, Y, Z) = (X) XOR (Y) XOR (Z)

40 ≤ t ≤ 59, F(X, Y, Z) = (X AND Y) XOR (X AND Z) XOR (Y AND Z)

60 ≤ t ≤ 79, F(X, Y, Z) = (X) XOR (Y) XOR (Z)

Step 4: Main-processing(M, K(t), F(t), H):

7.Split M into chucks of 512 sub message, M0, M1, M2...

8.For each Mi start at i = 0,

Initialize (A, B, C, D, E) = (H0, H1, H2, H3, H4)

Initialize Wt: = t-th 32-bit word of Mi. If t < 16

XOR Wt-3, Wt-8, Wt-14, Wt-16, circular shift left by 1 bit. Else

For t = 0, t <= 79

T = (A circular shift left by 5 bits) + F(B,C,D) + Wt + Kt + E

Update (A, B, C, D, E) = (T, A, B circular shift left by 30 bits, C, D)

Update (H0, H1, H2, H3, H4) = (H0+A, H1+B, H2+C, H3+D, H4+E)

Step 5: Output

Output = (H0,H1,H2,H3,H4)

**Design Details**

* Describe the approach/strategy that you took for the design.
* Did you focus on getting the design to work and made the design simple by performing each SHA-1 round using combinational logic and by using separate states for reading the memory?
* Did you aim to reduce the clock period by breaking up the computation of each SHA-1 round into multiple states? e.g., computed F(B, C, D) and ( Wt-3 ^ Wt-8 ^ Wt-14 ^ Wt-16 ) <<< 1 in separate states?
* Did you aim to reduce the #cycles and clock period by breaking the SHA-1 round computations and memory reads into pipelined stages? Provide some explanation as to how you achieved it.
* If you did multiple versions of your design, what approach did you take initially? What did you learn from the earlier designs? What did you do differently in later versions of your design based on what you learned from earlier designs?
* Did you target your design for Delay, Area x Delay, or both?
* Provide state diagram(s) for your design(s).

In our design, we used

**Working in Teams** (only applies if you worked in a group)

* How were the work divided among the team members?
* How did you coordinate the work and communicate between the team members?
* In which ways do you think your team functioned well?
* What have you learned from this quarter’s experience about functioning in a team?
* What can the instructors do in future quarters to make this aspect of the course better?

There are only two people in the team. Wai-Ho is responsible for majority of the coding, and Xiaoyu is in charge of report and some debugging. Work and communication are handled by emails and meetings.

Some of the parts our team functioned well are to get things started quickly and the ability to fix bugs.

The experience we gained from this quarter working in a team helped us to see our strength and weakness in terms of technical abilities and nontechnical skills such as time management. For instance, Wai-Ho is an expert in Verilog programming, so he is able to get things going quickly. Xiaoyu is a slow starter who usually needs to spend some time to get himself into the mood before jumping right in, but he likes to see things in a high level. By utilizing our strength, we are able to successfully complete the project in a short period of time. So the biggest lesson we learned is to contribute based on what you are good at and at the same time, improve your weakness, in order to get the most out of the project.

In regard to future improvement for this course, we would recommend increase team size to maximum three people. This way, it will create a better, more competitive atmosphere..

**Summary of Results**

* If you are just turning in one design, summarize the results for this design in terms of #ALUTs, #Registers, Area = #ALUTs + #Registers, Clock period = 1/Fmax, #cycles for the SHA1\_hash\_testbench\_v6.v, Delay = Clock period x #cycles, Area\*Delay = Area x Delay.
* If you are turning in two designs, then say explicitly that you are turning in two designs and provide the same summary of results for:
  + Best Delay Design: …
  + Best Area x Delay Design: …

**References**

http://en.wikipedia.org/wiki/SHA-1