Digital Design Lab Project

SHA-1 Hash Processor Design

Prof. Yongtae Kim

Computer Science and Engineering
Kyungpook National University

Introduction to SHA-1

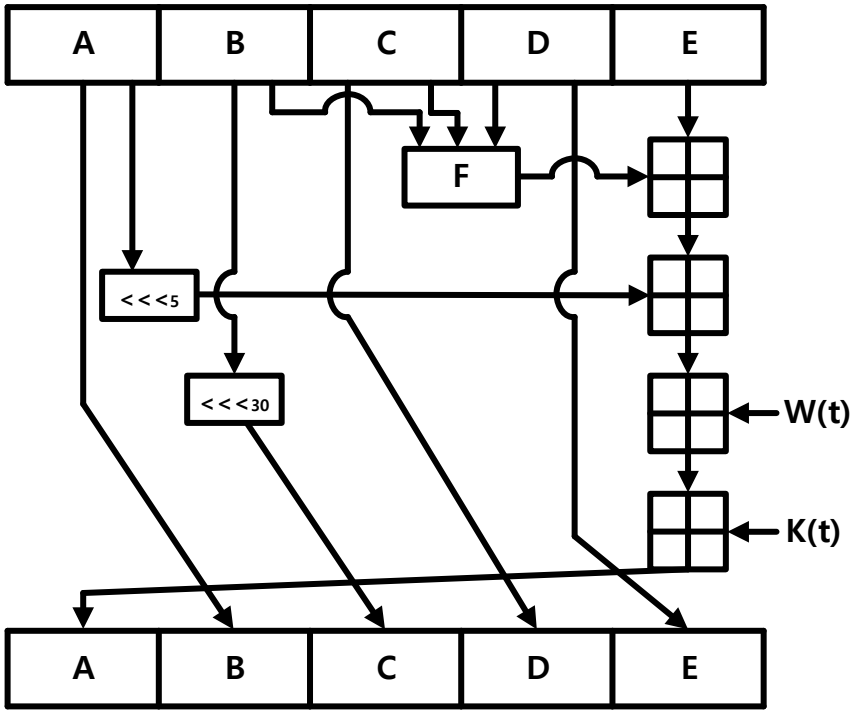
- **SHA-1 is a widely used hash function in the past producing a 160-bit hash value**

- **SHA-1 (Secure Hash Algorithm 1) history**
  - SHA-1 was developed as part of the US government's capstone project
  - SHA-1 was widely used in many places until 2010s
  - In 2017, Google announced a collision attack against SHA-1, publishing two dissimilar PDF files which produced the same SHA-1 hash.
  - All major web browser vendors ceased acceptance of SHA-1 SSL certificates in 2017, but it still secure for HMAC

- **SHA01 is designed by the US National Security Agency**
  - 512 bits blocks and 160 bits message digest value
  - Specified in 2001 as RFC 3174: https://www.rfc-editor.org/rfc/rfc3174.html
  - Note that the RFC includes algorithm details and source codes

Overall Structure

- **Input**
  - The input message is broken up into chunks of 512 bits blocks
  - W(t) is the expanded message word of round t
  - K(t) is the round constant of round t
- **Main algorithm**
  - A, B, C, D and E are 32 bits words of the state
  - F is a nonlinear function that varies
  - Each round has 20 operations
  - Total 80 operations
- **Output bits**
  - A fixed-length output of 160 bits



Main Algorithm (1)

- **Processing one message block (512 bits) consists of four rounds to produce 160 bits output**
  - Each round includes 20 operations

- **Works for a single 160 bits state of 32 bits word buffers named A, B, C, D and E**
  - A, B, C, D and E are initialized with a prescribed constant value

Block	Constant Value
A	0x67452301
B	0xEFCDAB89
C	0x98BADCFE
D	0x10325476
E	0xC3D2E1F0

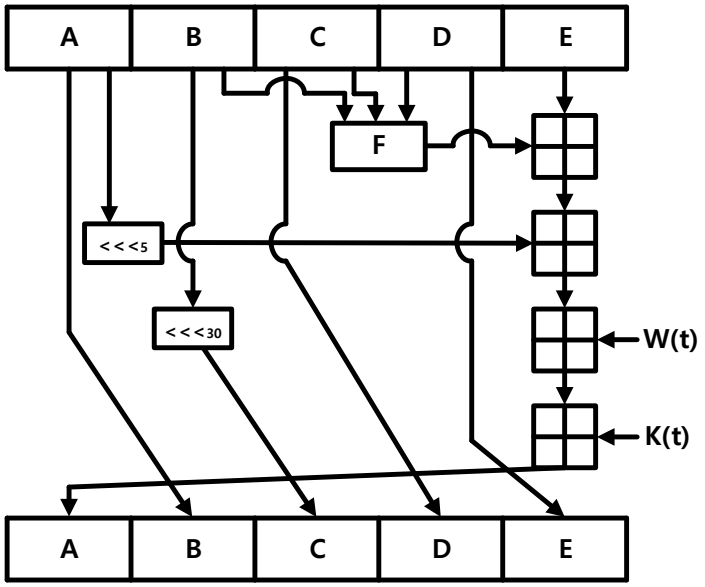
Main Algorithm (2)

- **A sequence of logical function F**
  - F function operates on three 32 bits words B, C, D and produces a 32 bits word as output

```
F(t, B, C, D) = (B & C) | ((~B) & D)          ( 0 ≤ t < 20)
F(t, B, C, D) = B ^ C ^ D                     (20 ≤ t < 40)
F(t, B, C, D) = ((B & C) | (B & D) | (C & D)) (40 ≤ t < 60)
F(t, B, C, D) = B ^ C ^ D                     (60 ≤ t < 80)
```

- **A sequence of constant word K(t)**

```
K(t) = 5A827999  (0  ≤ t < 20)
K(t) = 6ED9EBA1  (20 ≤ t < 40)
K(t) = 8F1BBCDC  (40 ≤ t < 60)
K(t) = CA62C1D6  (60 ≤ t < 80)
```



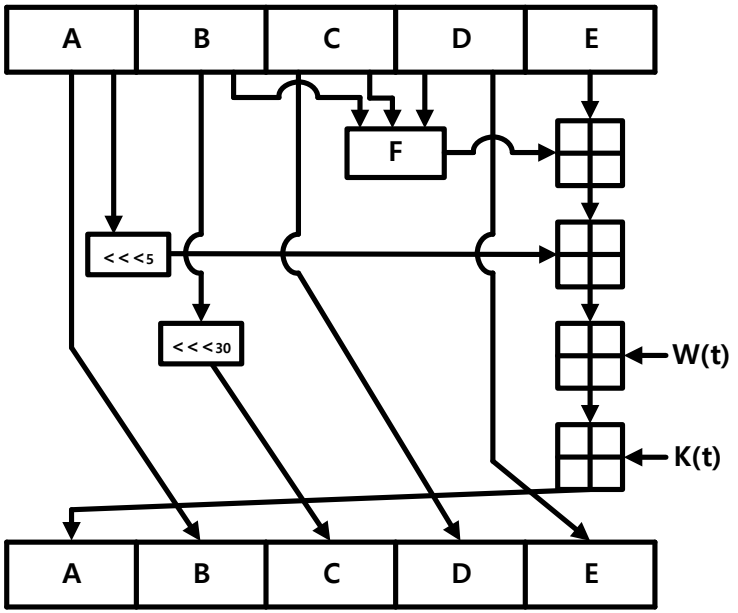
Main Algorithm (3)

- **W(t) is a 32 bits word**
  - The 512-bit input message is divided into 16 words W(0) ~ W(15)
  - W(0) is the left-most word
  - W(16) ~ W(79) are derived as

$$W(t) = W(t-3) \oplus W(t-8) \oplus W(t-14) \oplus W(t-16) \lll 1 \quad (16 \le t < 80)$$

- **<<< is a circular left shift operation**

$$\lll n = (word \ll n) \mid (word \gg 32-n)$$



Test Vector

- **Input Text: test**

```
Input 512-bits:
74657374800000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000020

Round Outputs:
Round1 → A: A594E7C5 B: D7E2BB59 C: 4A36181F D: FB609E49 E: 01A43078
Round2 → A: 80F549BF B: ED3E0D75 C: 6CCAC64C D: 462AE26F E: 0F93A146
Round3 → A: 1387E9E4 B: 2A106003 C: D91959BB D: 6CF9FCDF E: 84D25BDA
Round4 → A: 42056CE4 B: DCE3F01D C: 83912B75 D: C35F9511 E: D45CD9E3

Output 160-bits:
A94A8FE5CCB19BA61C4C0873D391E987982FBBD3
```

- **More test vectors are provided in a separate text file**

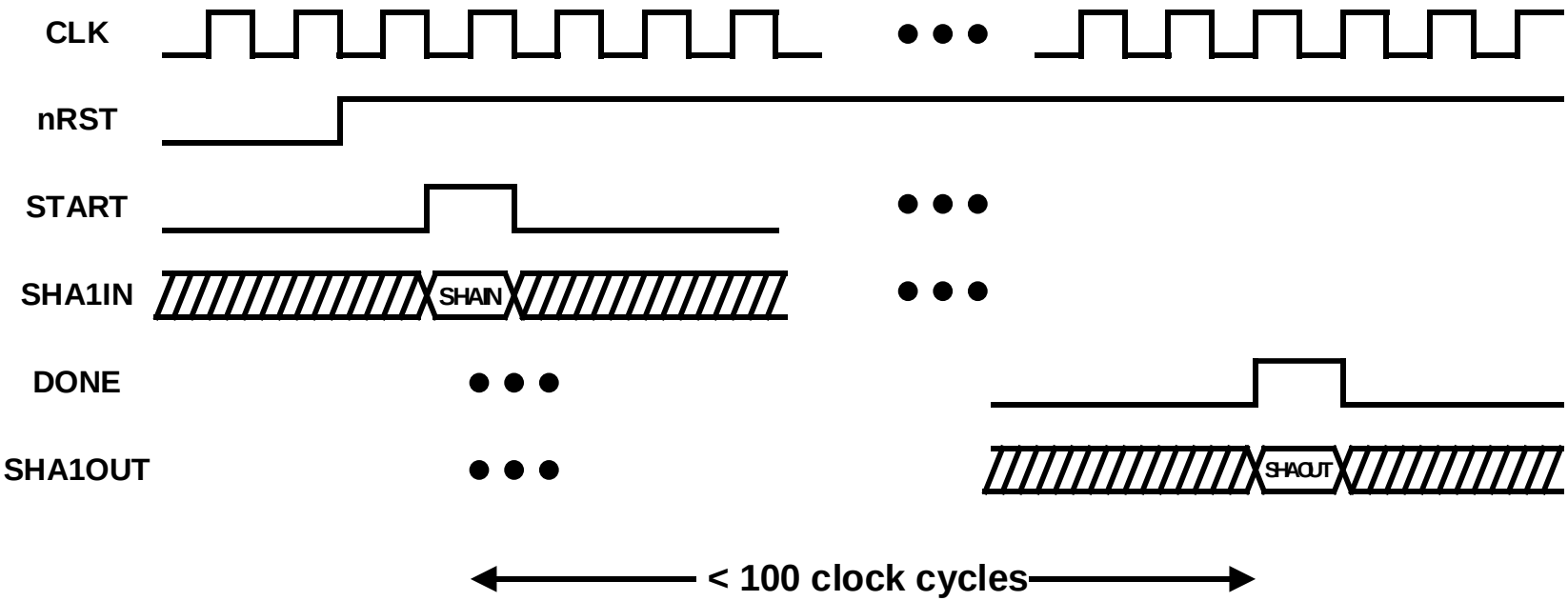
Goal (1)

- **Design a SHA-1 hash processor**
  - SHA-1 Hashing
  - Synthesizable design (RTL) and testbench (stimulus)

- **Design Specification**
  - Input: CLK, nRST, START, SHA1IN[511:0]
    - START: SHA-1 start signal (hold for only 1 cycle)
    - SHA1IN : 512 bits input message (hold for at least 1 cycle)
  - Output: DONE, SHA1OUT[159:0]
    - DONE: SHA-1 complete signal (hold for only 1 cycle)
    - SHA1OUT: 160 bits SHA-1 output (hold for at least 1 cycle)
  - 1 clock cycle per operation

Goal (2)

- **Timing**
  - Posedge clock (CLK) and asynchronous negedge reset (nRST)
  - SHA-1 needs to complete within 100 clock cycles



Grading

- **Total: 100 pts**
  - C/C++ implementation: 20 pts
  - Verilog RTL implementation: 80 pts
    - SHA-1 algorithm: 50 pts
    - Synthesis: 20 pts
    - Overall structure: 10 pts

- **Submission Due**
  - Due: 12/12, Sunday, 23:59pm
  - No late submission is allowed

- **What to submit**
  - C/C++, Verilog source code (design and testbench)
  - Design report (Datapath, FSM, Waveforms, Source description, Synthesis)
  - PDF only (No paper submission)