Keywords: authentication, SHA-1 coprocessor, verilog, ASIC, FPGA

APPLICATION NOTE 5485

Understanding the DSSHA1 Synthesizable SHA-1 Coprocessor

By: Bernhard Linke, Principal Member Technical Staff
Stewart Merkel, Senior Member Technical Staff
Sep 21, 2012

Abstract: Challenge-and-response authentication requires a MAC originator and a MAC recipient to compute a message authentication code based on a hidden secret and public data. The originator is typically a SHA-1 authenticator or a protected memory with SHA-1 engine. The MAC recipient is the application's host processor. This application note describes the DSSHA1 synthesizable SHA-1 coprocessor, which can be implemented in an application-specific integrated circuit (ASIC) or field-programmable gate array (FPGA) as alternative to the DS2460 SHA-1 coprocessor or a microprocessor-based implementation.

Introduction

Challenge-and-response authentication is based on the computation of message authentication codes (MACs). The method involves two entities, the MAC originator and MAC recipient, which share a hidden secret. To prove the authenticity of the MAC originator, the MAC recipient generates a random number and sends it as a challenge to the originator. The MAC originator must then compute a new MAC based on the secret, message, and challenge and send it back to the recipient. If the originator proves capable of generating a valid MAC for any challenge, it is very certain that it knows the secret and therefore can be considered authentic. A thoroughly scrutinized and internationally certified algorithm to compute message authentication codes is SHA-1, which was developed by the National Institute of Standards and Technology (NIST).

Maxim manufactures a series of authentication devices that employ the SHA-1 algorithm. Tutorial 3675, "Protecting the R&D Investment with Secure Authentication," explains the Maxim authentication solution in the form of secure memories and the DS2460 SHA-1 coprocessor. The DSSHA1 memory-mapped SHA-1 coprocessor allows the computational capabilities of the DS2460 to be implemented in an application-specific integrated circuit (ASIC) or field-programmable gate array (FPGA), eliminating the need to develop software to perform the complex SHA-1 computation. The MAC computed by the DSSHA1 or DS2460 applies only to Maxim SHA-1 devices.

Description

The DSSHA1 is a synthesizable, memory-mapped SHA-1 coprocessor that includes a 64-byte general-purpose RAM that stores the 64-byte message. The input message is used to compute the SHA-1 MAC. The DSSHA1 input and output port signals are designed to internally connect to a 32-bit bus. By a positive comparison result, authentication security is achieved between a host system and slave accessories.

Figure 1 shows the DSSHA1 block diagram. **Table 1** describes the signals that connect the DSSHA1 to the host system. Using the data bus input, address, and control signals, the 64-byte SHA-1 message is inserted into the RAM. Triggering the input signal RUN_SHA to logic-high starts the SHA-1 computation. The output BUSY signal indicates an occurring computation. Upon completion of the BUSY signal, the result registers contain the 20-byte message digest for reading.

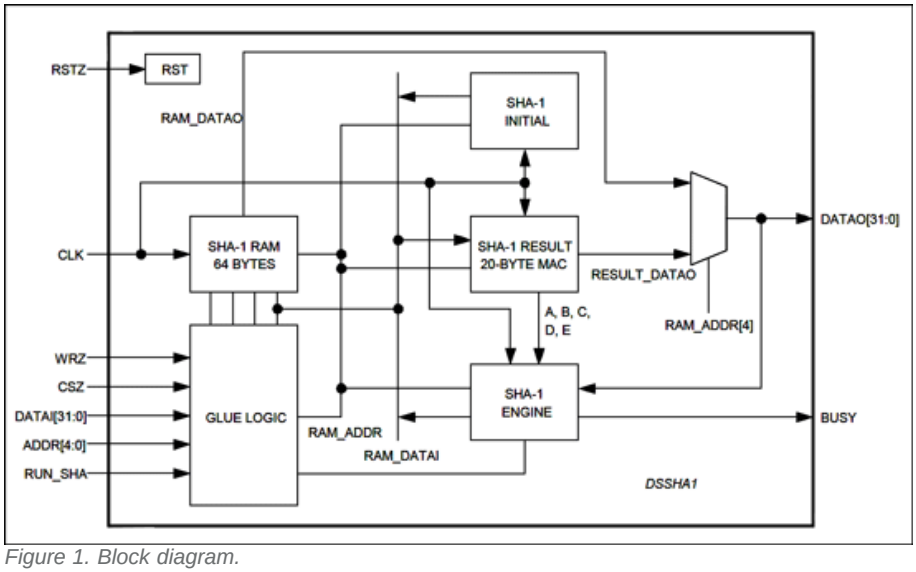

Figure 1. Block diagram.

Table 1. Signal Description

Name	Type*	Function
CLK	I	Clock. On the positive edge, data on signals DATAI[31:0] and DATAO[31:0] are clocked in and out.
RSTZ	I	Active-low reset. The RSTZ signal is evaluated at each interval of the positive edge of the CLK signal. It is necessary to do a reset before every load of a 512-bit message and MAC computation.
CSZ	I	Active-low chip select. This signal must be low for all accesses to registers and memory.
WRZ	I	Active-low write enable. This signal must be low during all write operations.
ADDR[4:0]	I	Address[4:0]. These five signals are the address signals.
DATAI[31:0]	I	Data bus input. These 32 signals are the input data bus.
DATAO[31:0]	O	Data bus output. These 32 signals are the output data bus.
BUSY	O	Busy. When high, this signal indicates that the SHA-1 coprocessor is busy performing a computation. There should be no data accesses while this signal is high.
RUN_SHA	I	Run SHA-1. This signal must only be one clock period wide and initiates a SHA-1 computation upon the positive edge of the CLK signal.

*I = input, O = output.

Detailed Register Description

The DSSHA1 memory consists of twenty-one 32-bit words, beginning with the input buffer and ending with the registers to read the MAC result (**Table 2**).

Table 2. Memory Map

Address (Hex)	Type	Access	Function
00h to 0Fh	RAM	Read/write	64-byte buffer input. This is the 512-bit input block that usually includes the 64-bit slave device secret and a 448-bit input message consisting of a random challenge and various data.
10h to 14h	Registers	Read	20-byte result. This is the MAC for comparison to the received MAC of the SHA-1 slave device.

Input Buffer (00h to 0Fh)

The SHA-1 engine receives the data to be processed through the 64-byte input buffer. This buffer holds the 512-bit message that the SHA-1 engine processes to generate a MAC. Secret and other message data are contained in the input buffer. Security of the secret is a task left for the designer. The format of the data is defined by each Maxim SHA-1 slave device.

MAC Result (10h to 14h)

A 20-byte MAC of a SHA-1 computation resides in the MAC result address space.

Device Operation

The typical use of the DSSHA1 in an application involves writing, reading, and running the SHA-1 engine, and using the MAC result to externally compare this block to the MAC of a 1-Wire SHA-1 device. All these activities are controlled through the 32-bit interface with separate data input and output lines to easily connect to the internal bus inside an ASIC or FPGA. The SHA-1 Engine Control section below explains the data input and output format and how to instruct the SHA-1 engine to perform a MAC computation.

SHA-1 Engine Control

The DSSHA1 performs the job of a SHA-1 engine. The input buffer accepts the message. The MAC output buffer receives the resultant SHA-1 computation. **Figure 2** illustrates data flow into and out of the SHA-1 engine.

Applying a power reset initiates the first step of using the SHA-1 engine. Next, a message is loaded into the input buffer in the format of **Table 3**. Upon completion of a message load, the user pulses the RUN_SHA input signal. For the duration of the SHA-1 computation, the BUSY signal goes and remains logic-high. A BUSY signal goes logic-low again when the SHA-1 computation completes. All five of the MRR registers (see **Table 4**) contain the MAC result for reading.

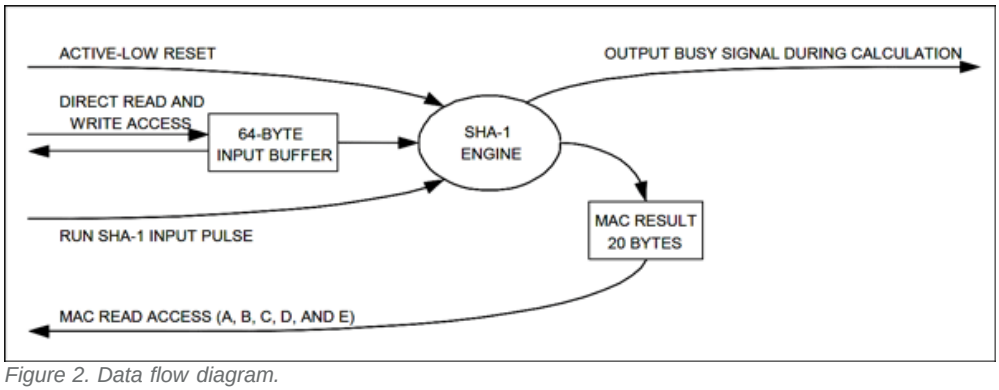

Figure 2. Data flow diagram.

Table 3. Input Message Format			
M0[31:24] = (IB + 0)	M0[23:16] = (IB + 1)	M0[15:8] = (IB + 2)	M0[7:0] = (IB + 3)
M1[31:24] = (IB + 4)	M1[23:16] = (IB + 5)	M1[15:8] = (IB + 6)	M1[7:0] = (IB + 7)
M2[31:24] = (IB + 8)	M2[23:16] = (IB + 9)	M2[15:8] = (IB + 10)	M2[7:0] = (IB + 11)
M3[31:24] = (IB + 12)	M3[23:16] = (IB + 13)	M3[15:8] = (IB + 14)	M3[7:0] = (IB + 15)
M4[31:24] = (IB + 16)	M4[23:16] = (IB + 17)	M4[15:8] = (IB + 18)	M4[7:0] = (IB + 19)
M5[31:24] = (IB + 20)	M5[23:16] = (IB + 21)	M5[15:8] = (IB + 22)	M5[7:0] = (IB + 23)
M6[31:24] = (IB + 24)	M6[23:16] = (IB + 25)	M6[15:8] = (IB + 26)	M6[7:0] = (IB + 27)
M7[31:24] = (IB + 28)	M7[23:16] = (IB + 29)	M7[15:8] = (IB + 30)	M7[7:0] = (IB + 31)
M8[31:24] = (IB + 32)	M8[23:16] = (IB + 33)	M8[15:8] = (IB + 34)	M8[7:0] = (IB + 35)
M9[31:24] = (IB + 36)	M9[23:16] = (IB + 37)	M9[15:8] = (IB + 38)	M9[7:0] = (IB + 39)
M10[31:24] = (IB + 40)	M10[23:16] = (IB + 41)	M10[15:8] = (IB + 42)	M10[7:0] = (IB + 43)
M11[31:24] = (IB + 44)	M11[23:16] = (IB + 45)	M11[15:8] = (IB + 46)	M11[7:0] = (IB + 47)
M12[31:24] = (IB + 48)	M12[23:16] = (IB + 49)	M12[15:8] = (IB + 50)	M12[7:0] = (IB + 51)
M13[31:24] = (IB + 52)	M13[23:16] = (IB + 53)	M13[15:8] = (IB + 54)	M13[7:0] = (IB + 55)
M14[31:24] = (IB + 56)	M14[23:16] = (IB + 57)	M14[15:8] = (IB + 58)	M14[7:0] = (IB + 59)
M15[31:24] = (IB + 60)	M15[23:16] = (IB + 61)	M15[15:8] = (IB + 62)	M15[7:0] = (IB + 63)

Mt = input buffer of SHA-1 engine; 0 ≤ t ≤ 15; 32-bit words with a start address at 00h and ending address at 0Fh. IB = input buffer.

Table 4 shows how the five 32-bit variables A to E that hold the MAC are mapped to the respective locations.

Table 4. Output Message Format	
Address (Hex)	**MAC Result Registers (MRR)**
10h	MRR[31:0] = A[31:0] (least significant)
11h	MRR[31:0] = B[31:0]
12h	MRR[31:0] = C[31:0]
13h	MRR[31:0] = D[31:0]
14h	MRR[31:0] = E[31:0] (most significant)

MAC Comparison

The master has the requirement to test the slave MAC against the DSSHA1 MAC. Authenticity is verified if the slave MAC and the DSSHA1 MAC are equal in value. A fraud is verified if the slave MAC and the DSSHA1 MAC are different.

Functional Verification

To test the DSSHA1, the test message "abc" can verify functionality. This test message with proper padding can be translated into an input block of:

W[0] = 61626380	W[8] = 00000000
W[1] = 00000000	W[9] = 00000000
W[2] = 00000000	W[10] = 00000000
W[3] = 00000000	W[11] = 00000000

W[4] = 00000000	W[12] = 00000000
W[5] = 00000000	W[13] = 00000000
W[6] = 00000000	W[14] = 00000000
W[7] = 00000000	W[15] = 00000018

Using the format of Table 3, the input block of this test message will be the values in **Table 5**.

Table 5. SHA-1 Input for "abc" Test Packet

M0[31:24] = 61h	M0[23:16] = 62h	M0[15:8] = 63h	M0[7:0] = 80h
M1[31:24] = 00h	M1[23:16] = 00h	M1[15:8] = 00h	M1[7:0] = 00h
M2[31:24] = 00h	M2[23:16] = 00h	M2[15:8] = 00h	M2[7:0] = 00h
M3[31:24] = 00h	M3[23:16] = 00h	M3[15:8] = 00h	M3[7:0] = 00h
M4[31:24] = 00h	M4[23:16] = 00h	M4[15:8] = 00h	M4[7:0] = 00h
M5[31:24] = 00h	M5[23:16] = 00h	M5[15:8] = 00h	M5[7:0] = 00h
M6[31:24] = 00h	M6[23:16] = 00h	M6[15:8] = 00h	M6[7:0] = 00h
M7[31:24] = 00h	M7[23:16] = 00h	M7[15:8] = 00h	M7[7:0] = 00h
M8[31:24] = 00h	M8[23:16] = 00h	M8[15:8] = 00h	M8[7:0] = 00h
M9[31:24] = 00h	M9[23:16] = 00h	M9[15:8] = 00h	M9[7:0] = 00h
M10[31:24] = 00h	M10[23:16] = 00h	M10[15:8] = 00h	M10[7:0] = 00h
M11[31:24] = 00h	M11[23:16] = 00h	M11[15:8] = 00h	M11[7:0] = 00h
M12[31:24] = 00h	M12[23:16] = 00h	M12[15:8] = 00h	M12[7:0] = 00h
M13[31:24] = 00h	M13[23:16] = 00h	M13[15:8] = 00h	M13[7:0] = 00h
M14[31:24] = 00h	M14[23:16] = 00h	M14[15:8] = 00h	M14[7:0] = 00h
M15[31:24] = 00h	M15[23:16] = 00h	M15[15:8] = 00h	M15[7:0] = 18h

Mt = input buffer of SHA-1 engine; 0 ≤ t ≤ 15; 32-bit words with a start address at 00h and ending address at 0Fh.

The output of the computation of this block is:

A[31:0] = 42541B35
B[31:0] = 5738D5E1
C[31:0] = 21834873
D[31:0] = 681E6DF6
E[31:0] = D8FDF6AD

The Maxim devices take these words as most significant word first and the individual bytes as least significant byte (LSB) first. So the byte level transmission sequence of the MAC would be:

AD F6 FD D8	F6 6D 1E 68	73 48 83 21	E1 D5 38 57	35 1B 54 42
(E)	(D)	(C)	(B)	(A)

Timing Specification

Figure 3 and **Figure 4** show the timing diagram for writing to and reading from the DSSHA1. **Table 6** shows delay values measured from 50% of supply to 50% of supply using an ARM TSMC CL018G (0.18µm generic process) 1.8V SAGE-X standard cells library, version 2004q3v1, at +25°C. The output signals are not loaded. Input signals are driven with a standard slew of 0.200ns from 10% to 90% of supply.

Table 6. Data Bus Interface Timing

Parameter	Symbol	Min	Max	Units
CLK cycle (Note 1)	t_{CYC}	12.500		ns
Chip select setup before rising edge of CLK (Note 1)	t_{CSS}	0.229		ns
Chip select hold after rising edge of CLK (Note 1)	t_{CSH}	0.000		ns
Address and data setup before rising edge of CLK (Note 1)	t_{AS}	0.229		ns
Address and data hold after rising edge of CLK (Note 1)	t_{AH}	0.000		ns
Active output time to DATAO valid (Notes 1, 2)	t_{AO}		0.984	ns
Deactivate DATAO[31:0] (Note 1)	t_{D}		0.984	ns

Note 1: These values depend upon the process used to realize the circuit. Values shown are for example purposes only and modeled using the ARM TSMC CL018G (0.18µm generic process) 1.8V SAGE-X standard cells library, version 2004q3v1. The ARM part number is A0082.

Note 2: This time is defined as the longest possible delay to valid output for the typical corners.

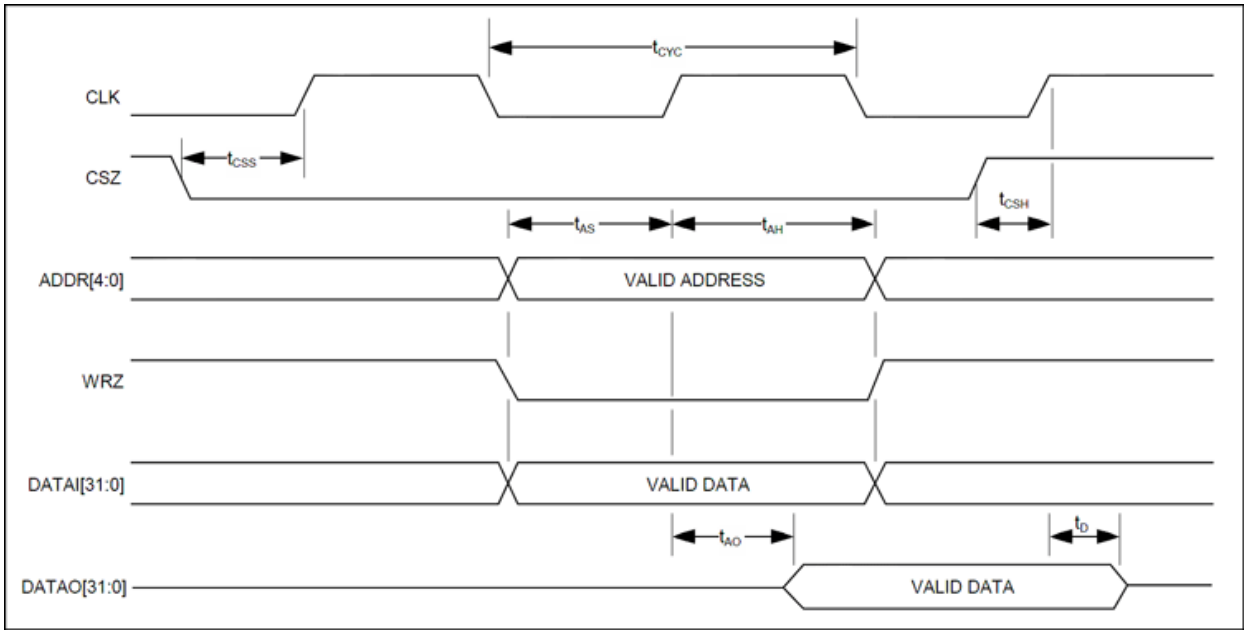

Figure 3. Write cycle.

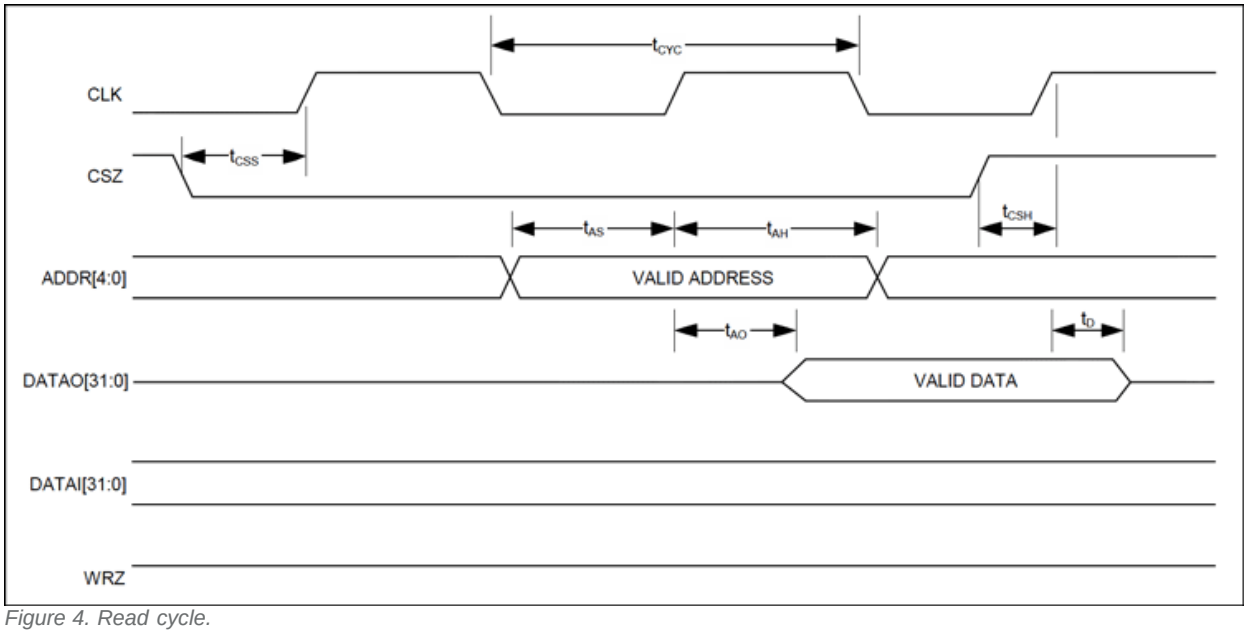

Figure 4. Read cycle.

Applications Information

FPGAs or ASICs integrate the designed DSSHA1. Using several modules, an achievable authentication method makes a design secure. In **Figure 5**, the design module with a microprocessor can offload the SHA-1 computation to the DSSHA1. In Figure 5, the designer first crafts a randomly generated challenge and compares the result from DSSHA1 to the response received from the DS28E01-100. If the result and response match, then the design has been authenticated and can enable the product's functionality. It is often desirable to make variations in the authentication process in software and hardware. This makes successful attacks less likely. Refer to application note 1098, "White Paper 3: Why are 1-Wire SHA-1 Devices Secure?" for more information.

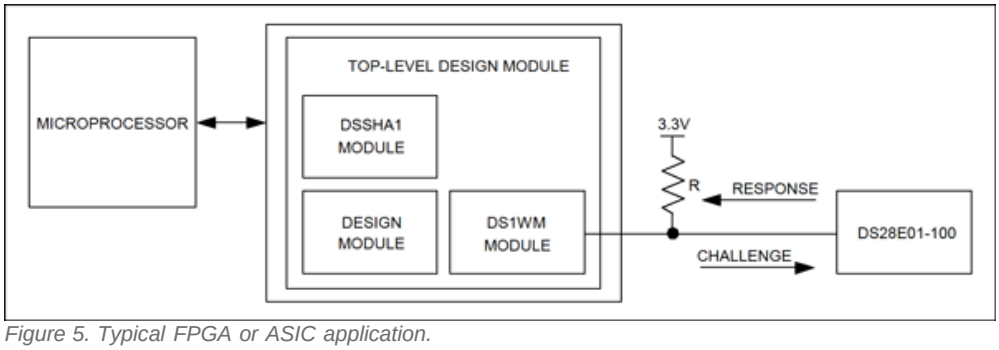

Figure 5. Typical FPGA or ASIC application.

Physical Estimates

- Gate count 6,423 (NAND 2x1 used for calculation).
- Area is 85,470µm² without routing.
- Area is 102,256µm² with routing estimate.

Library used for estimate:

ARM TSMC CL018G (0.18µm generic process) 1.8V SAGE-X standard cells library, version 2004q3v1. The ARM part number is A0082.

Verification

The industry typically denotes the level of verification of an IP block with the following conventions:

- Gold IP has been to target silicon.
- Silver IP has been to target silicon in FPGA.
- Bronze IP has been verified in silicon models with logical timing closure.
- In-development IP has not yet been verified.

Note: The DSSHA1 has achieved silver status.

Deliverables

The DSSHA1 package comes complete with:

- Verilog HDL
- Verilog Test Bench
- Readme information on setup and scripts

The free DSSHA1 IP is available by request at https://support.maximintegrated.com/1-Wire.

Summary

The DSSHA1 synthesizable SHA-1 coprocessor is an alternative to the DS2460 or a microprocessor-based implementation. It can be embedded in an FPGA or ASIC where it appears as a memory-mapped device. For operation, the 16-word input buffer is first filled with data for the MAC computation. Activating the RUN_SHA signal starts the computation process, which changes the BUSY signal from low to high. After the BUSY signal has returned to low, the MAC is ready and can be read from the 5-word result register. The host processor compares the MAC computed by the DSSHA1 to the MAC delivered by the secure memory. Authenticity is confirmed if both MAC values are identical.

Related Parts		
DS1961S	1Kb Protected EEPROM ¡Button with SHA-1 Engine	
DS1963S	SHA ¡Button	
DS2432	1Kb Protected 1-Wire EEPROM with SHA-1 Engine	Free Samples
DS2460	SHA-1 Coprocessor with EEPROM	Free Samples
DS28CN01	1Kbit I²C/SMBus EEPROM with SHA-1 Engine	Free Samples
DS28E01-100	1Kb Protected 1-Wire EEPROM with SHA-1 Engine	Free Samples
DS28E02	1-Wire SHA-1 Authenticated 1Kb EEPROM with 1.8V Operation	Free Samples
DS28E10	1-Wire SHA-1 Authenticator	Free Samples

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5485: http://www.maximintegrated.com/an5485
APPLICATION NOTE 5485, AN5485, AN 5485, APP5485, Appnote5485, Appnote 5485, 19-5870