Using SHA-256 System Services in SmartFusion2 and IGLOO2 Devices - Libero SoC v11.6

Table of Contents

Purpose

This application note provides the design example for using the secure hash algorithm (SHA)-256 accelerator block in the SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) devices and IGLOO®2 FPGA devices.

© 2015 Microsemi Corporation

Introduction

SHA is a cryptographic hash function that can process a message to produce a condensed representation called a message digest. The SHA-256 algorithm is part of the standard SHA algorithm, defined in the national institute of standards and technology (NIST) as a U.S. federal information processing standard (FIPS) 180-3. The built-in SHA-256 accelerator block can be accessed and performed the SHA-256 operation in the selected devices of the SmartFusion2 and IGLOO2 families. These devices are marked as S (Data and Design Security) in the device part number. The SHA-256 accelerator block, also known as SHA-256 engine, resides inside the system controller of the SmartFusion2 and IGLOO2 devices. The SHA-256 accelerator block is accessible through the SHA-256 system services. The system services are system controller actions initiated by asynchronous events from the ARM® Cortex®-M3 processor in the SmartFusion2 device or a fabric master in the SmartFusion2 and IGLOO2 devices. In the SmartFusion2 and IGLOO2 devices, the SHA-256 engine can take message inputs of any size up to 2^{32} bits in length and digest them down to a 256-bit result, as per the SHA-256 standard. The SHA-256 cryptographic services can be used for data security applications and can be disabled using the factory or user security settings. This application note describes how to use the SHA-256 accelerator block using the SHA-256 system service in the SmartFusion2 and IGLOO2 devices.

References

The following are the references:

- *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*
- *UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide*
- *SmartFusion2 SoC Security FPGAs*
- *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide*
- *UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide*
- *IGLOO2 Security FPGAs*

Design Requirements

Table 1 lists the SmartFusion2 design requirements.

Table 1 • **SmartFusion2 Design Requirements**

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Security Evaluation Kit (M2S-EVAL-KIT): 12 V adapter (provided along with the kit) FlashPro4 programmer (provided along with the kit) M2S090TS-1FGG484	–
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero® System-on-Chip (SoC)	v11.6
SoftConsole	v3.4 SP1
ModelSim®	–

Table 2 lists the IGLOO2 design requirements.

Table 2 • **IGLOO2 Design Requirements**

| Design Requirements and Details	Description
Hardware Requirements	
IGLOO2 Evaluation Kit (M2GL-EVAL-KIT): 12 V adapter (provided along with the kit) FlashPro4 programmer (provided along with the kit) M2GL090TS-1FGG484	Rev D
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.6
Note: *The IGLOO2 design uses M2GL090TS-1FGG484 device in the IGLOO2 Evaluation Kit. However, the official IGLOO2 Evaluation Kit uses the M2GL010T-1FGG484 device. If you want to run the application note design in M2GL010T-1FGG484, refer to the KB5659 for migrating M2GL090TS-1FGG484 to M2GL010T-1FGG484.*	

Overview

SHA is a cryptographic hash function designed by the U.S. national security agency (NSA) and published in 2001 by the NIST - FIPS. This Standard specifies several secure hash algorithms, including SHA-256. SHA-256 is a one-way hash function that can process a message to produce a message digest. In this application note, the term message digest and hashed output are used interchangeably. The input data is a message and hash value is a message digest.

In the SmartFusion2 and IGLOO2 devices, the SHA-256 accelerator block is part of the Cryptographic Services block that resides in the system controller. The SHA-256 accelerator block implements the SHA-256 function.

The SHA-256 algorithm determines the integrity of the message—any change to the message, with a very high probability, results in a different message digest. This property is widely used in security applications and protocol, generation and verification of digital signatures and message authentication codes, and so on. Following are the basic properties of the SHA-256 algorithm:

- Message size: $< 2^{64}$ bits
- Block size: 512 bits
- Word size: 32 bits
- Message digest size: 256 bits

Figure 1 shows the basic SHA-256 operation.
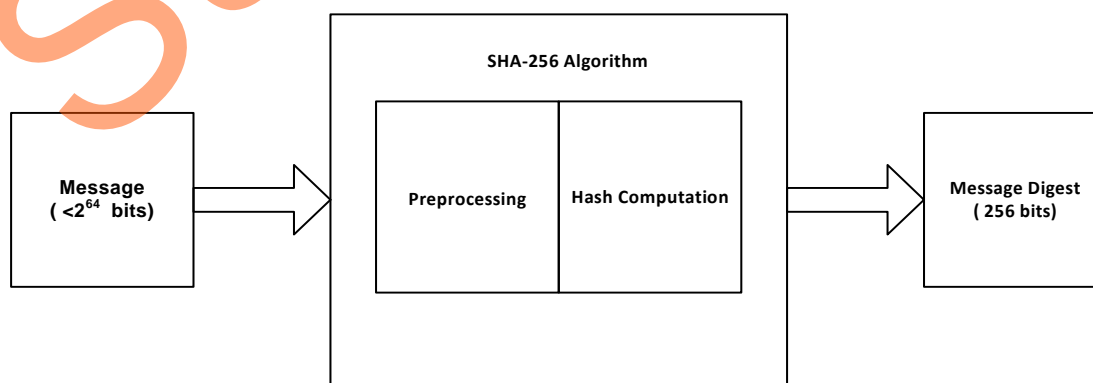
Figure 1 • **SHA-256 Operation**

The SHA-256 algorithm includes the following two stages:

- Preprocessing
- Hash computation

Preprocessing involves padding a message, parsing the padded message into 512-bit blocks, and setting initialization values to be used in the hash computation. The hash computation generates a message scheduled from the padded message and uses that schedule along with functions, constants, and word operations to iteratively generate a series of hash values. The final hash value generated by the hash computation is used to determine the message digest. Refer to the *FIPS PUB 180-3* publication for more information about preprocessing and hash computation.

Note: The hash algorithms is secured. For a given algorithm, it is computationally infeasible to:

- Find a message that corresponds to a given message digest
- Find two different messages that produce the same message digest
  Any change to a message, with a very high probability, results in a different message digest.

In the SmartFusion2 and IGLOO2 devices, the SHA-256 accelerator block can take message inputs of any size up to 2^{32} bits in length and process to a 256-bit result, as per the standard. If the message ends with a partial byte, the significant bits are assumed to be at the least significant bit (LSB) end of the byte. The unused most significant bits (MSBs) of the final byte are ignored. The input and output data format of the SHA-256 system service is little-endian type.

In SmartFusion2 and IGLOO2, the SHA accelerator block does not have the built-in algorithm-level distributed power architecture (DPA) countermeasures. The built-in design security applications are only used in protocols that effectively prevent DPA attacks from succeeding. When hashing is used with a secret value such as a key, the SmartFusion2 and IGLOO2 device built-in design security protocols strictly limit the number of uses of the secret to prevent its leaking through side-channels. Hashing is used with public data, where no secrets are processed, a low DPA resistance may not be a concern. However, when hashing is used in user data security applications involving secret data (such as an HMAC key), the user is responsible for using protocols that ensure that the same secret data is not used repeatedly, or it may become vulnerable to extract using the DPA techniques.

Cryptographic Services Block

This following sub-sections describe the Cryptographic Services block along with the system controller:

- Cryptographic Services Block in SmartFusion2 Device
- Cryptographic Services Block in IGLOO2 Device

Cryptographic Services Block in SmartFusion2 Device

The Cryptographic Services block can be accessed through the communication block, COMM_BLK. There are two communication block (COMM_BLK) instances—one is in the microcontroller subsystem (MSS) and the other one is in the system controller, which communicate with each other. The COMM_BLK consists an advanced peripheral bus (APB) interface, eight-byte transmit FIFO, and eight-byte receive FIFO. Refer to the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide* and *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide* for more information on communication block in the COMM_BLK chapter. The COMM_BLK provides a bi-directional message passing facility between the MSS and system controller.

Figure 2 shows the SmartFusion2 blocks used for SHA-256 system service.
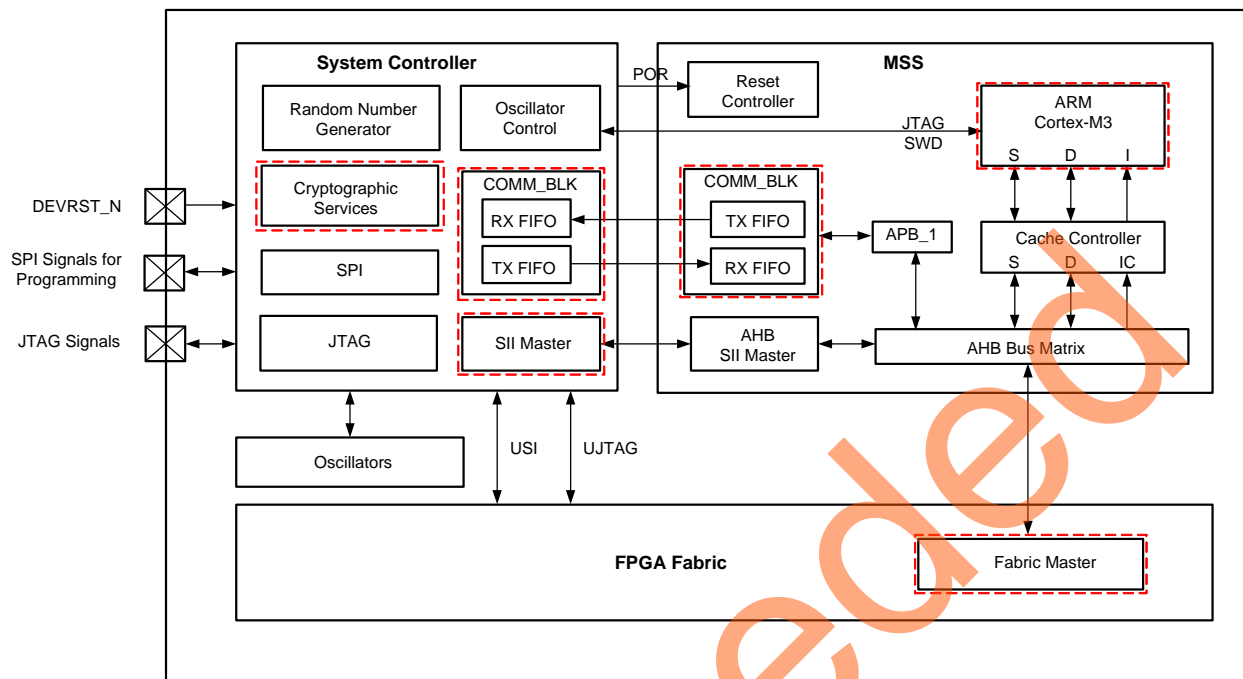
Figure 2 • **System Controller Block in SmartFusion2 Device**

The SHA-256 system services are initiated using the COMM_BLK block in the MSS. The SHA-256 system services can be read or written by any master on the advanced microcontroller bus architecture (AMBA®) high-performance bus (AHB) matrix; either by the Cortex-M3 processor or a design in the FPGA fabric (also known as a Fabric Master). The system controller receives the command through the COMM_BLK block in the system controller. The system controller uses the system IP interface (SII) master to transfer data to and from the MSS memory space for system services. When the SHA-256 service is requested, the data structure is written to the COMM_BLK block that contains the address in the memory of the input and resultant data, length of the data, and status. For more information about the data structure, refer to Table 3 on page 6. The message digest (SHA-256 hashed output) from SHA-256 operation returned by the system controller is written to a memory address specified in this data structure. On completion of the requested service, the system controller returns a status message through the COMM_BLK block.

Cryptographic Services Block in IGLOO2 Device

The architecture and uses of the SHA-256 accelerator block in IGLOO2 is similar to the SmartFusion2 device. The exceptions are:

- The COMM_BLK in the system controller communicates with the COMM_BLK in the high performance memory subsystem (HPMS).
- A fabric master can initiate the SHA-256 system services.

Microsemi® provides the CoreSysServices DirectCore IP that acts as a fabric master to use the SHA-256 system services. The CoreSysServices DirectCore IP communicates with the COMM_BLK block through one of the fabric interface controllers (FICs) to send the SHA-256 system service request, retrieve the message digest, and send it back to the user interface.

Figure 3 shows the IGLOO2 blocks used in SHA-256 system service.
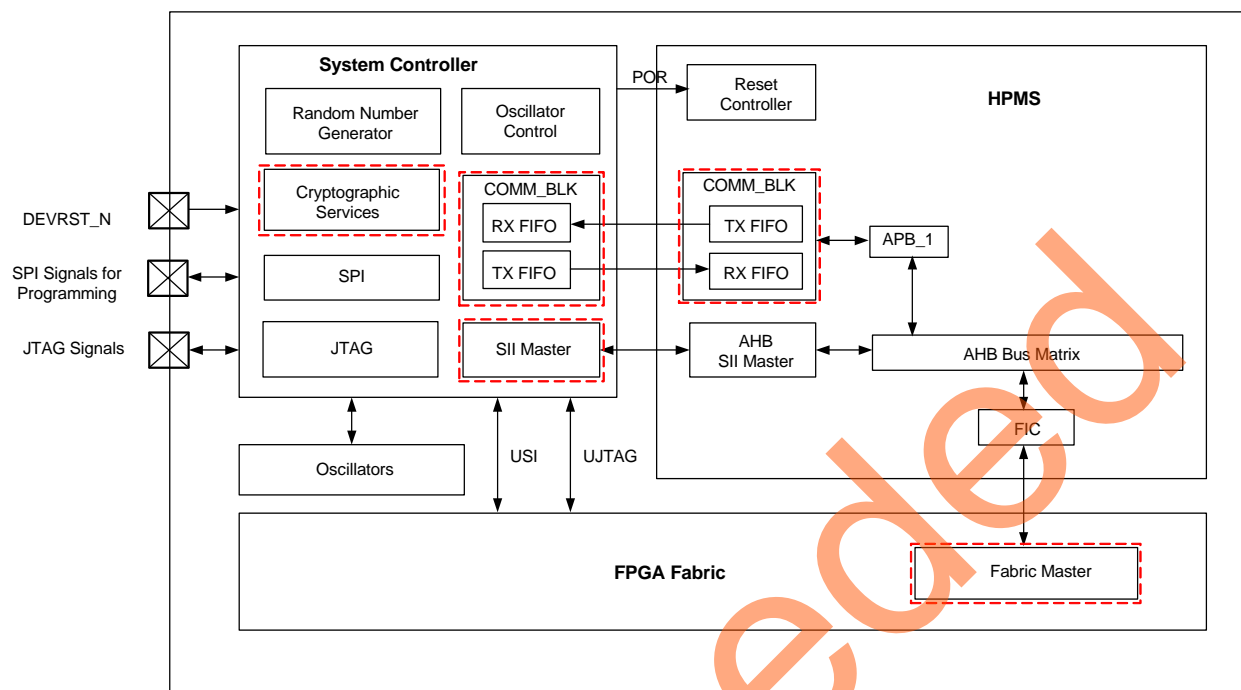
Figure 3 • **System Controller Block in IGLOO2 Device**

Refer to the *UG0450: SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide* for more information about system controller. Refer to the Communication Block chapter in the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide* and *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide* for more information about the COMM_BLK block.

Using SHA-256 System Services in SmartFusion2 and IGLOO2 Devices

This section describes the steps for accessing the SHA-256 system service in SmartFusion2 or IGLOO2 and its use model. Table 3 and Table 4 on page 7 list the data, and response data structures associated with this system service.

Following are the steps to use the SHA-256 system service in the SmartFusion2 or IGLOO2 device:

1. Set up the SHA256DATA descriptor data in the user memory space as listed in Table 3, containing the following 12 bytes:

Table 3 • **SHA256DATA Structure**

Offset	Length (bytes)	Field	Description
0	4	LENGTH	Length of data pointed to by DATAINPTR field in bits (up to 2^{32} bits).
4	4	HASHRESULTPTR	Pointer to 32-byte buffer to receive 256-bit hash result.
8	4	DATAINPTR	Pointer to data to be hashed.

2. Set up the input data at DATAINPTR location.

3. For embedded design in SmartFusion2, enable COMM_BLK_INTR (INTISR[19]) in Cortex-M3 processor interrupts. For fabric design in SmartFusion2 or IGLOO2, enable the COMBLK_INTR interrupt from the COMM_BLK block to fabric by enabling COMBLK_INTR_ENBL bit (2th bit) in INTERRUPT_ENABLE0 register at address 0x40006000.

4. Set up the COMM_BLK register in byte mode and send the SHA-256 command. The command value of the SHA-256 cryptographic service is 0x0A (decimal 10).

5. Wait for RCVOKAY bit to be set in the COMM_BLK STATUS register.

   The system controller receives the command through the COMM_BLK block in the system controller. The system controller reads the data from the address pointer and generates the message digest. On completion, the service system controller returns a status message through the COMM_BLK. After receiving the status message, RCVOKAY bit is set.

6. Read the Word Data register in the COMM_BLK and check the command, status code, and SHA256DATAPTR descriptor pointer as listed in Table 4.

Table 4 • **SHA-256 Service Response**

Offset	Length (bytes)	Field	Description
0	1	CMD = 10	Command
1	1	STATUS	Command status: • 0: Success • 127: HRESP error occurred during MSS transfer • 253: Not licensed • 254: Service disabled by factory security • 255: Service disabled by user security
2	4	SHA256DATAPTR	Pointer to SHA256DATA structure

7. If STATUS is successful, read the message digest data from user memory space (the return data buffer address is specified in DATAINPTR).

Figure 4 shows the system service data flow diagram in the SmartFusion2 device from the Cortex-M3 processor using eSRAM0 for input data and message digest output.
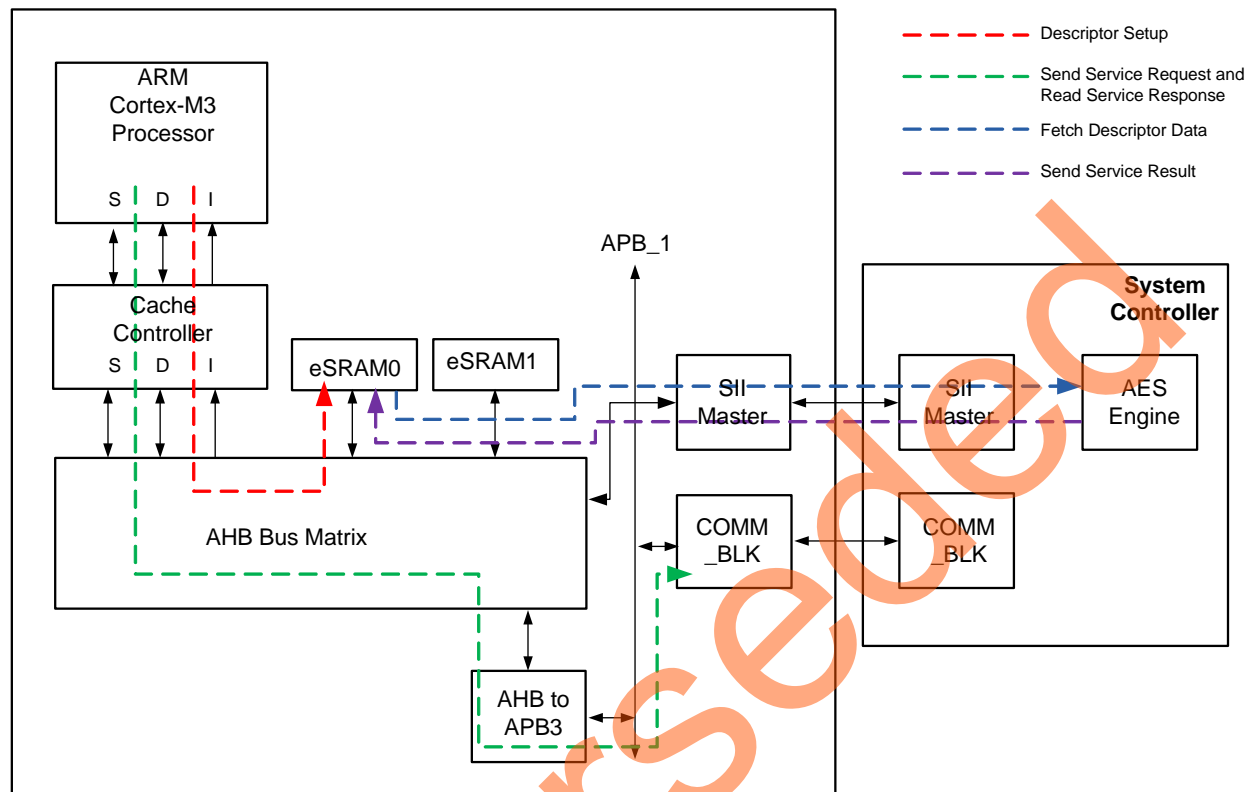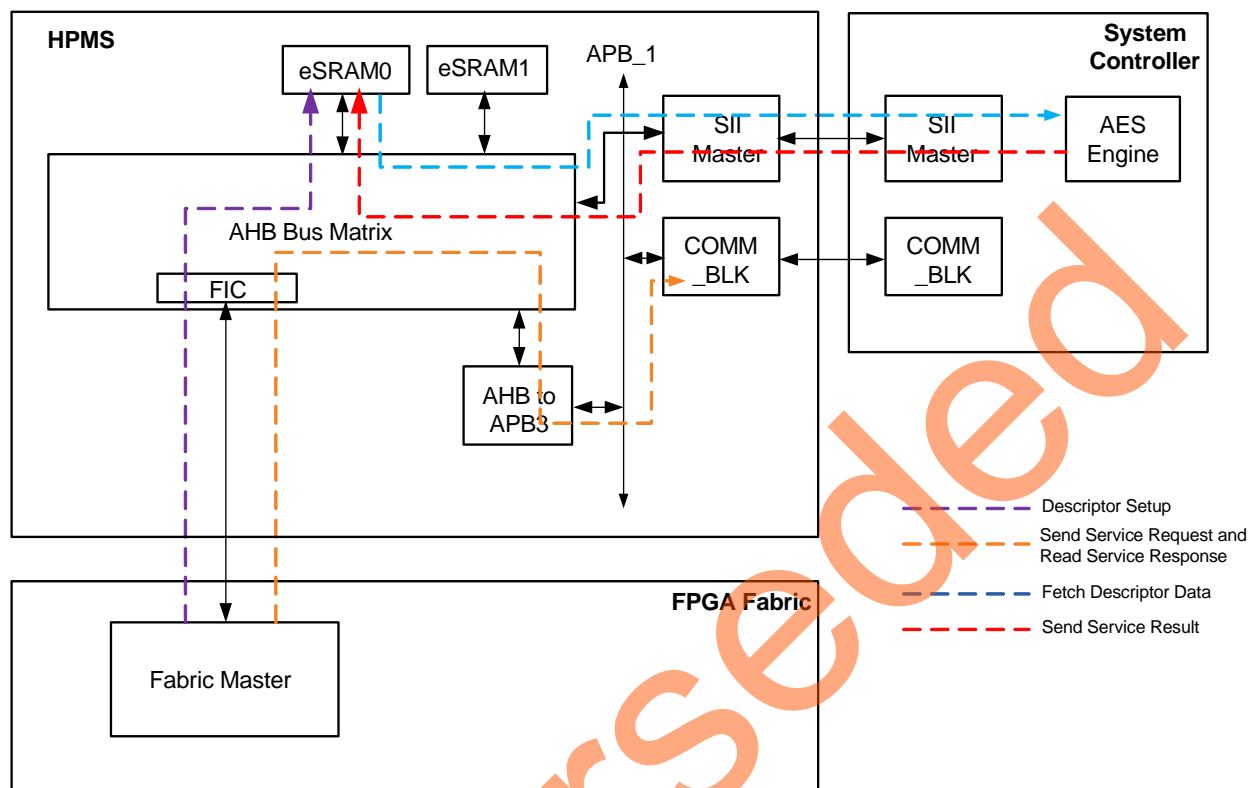

Figure 4 • **SHA-256 Service Flow Diagram in SmartFusion2 Device**

Figure 5 shows the system service data flow diagram in the IGLOO2 device using eSRAM0 for input data and message digest output.

Figure 5 • **SHA-256 Service Flow Diagram in IGLOO2 Device**

Microsemi provides a system service driver and CoreSysServices soft IP to run the SHA-256 operation in SmartFusion2 and IGLOO2. So, it is not required to write software code from scratch or create a complex master in fabric to run the SHA-256 system services.

In SmartfFusion2, the SHA-256 system can be accessed:

- using mss_sys_services driver in the firmware core configurator
- using CoreSysServices as fabric master

In the IGLOO2 device, CoreSysServices soft IP can be used as fabric master to run SHA-256 system service. CoreSysServices provides a simple user interface on one side and an AHB-Lite master interface to connect to the FIC to use system services through the COMM_BLK block. The IGLOO2 approach can be used in the SmartFusion2 device too. Refer to the *CoreSysServices Handbook* for more information about the CoreSysServices soft IP.

Design Description

This application note includes the following design examples to use the SHA-256 system service:

- **SHA_256_Services_SF2 Design Example**—demonstrates the running SHA-256 system service in the SmartFusion2 device using the system driver firmware code supplied by Microsemi.
- **SHA_256_Services_IGL2 Design Example**—demonstrates the running SHA-256 system service in the IGLOO2 device using the CoreSysServices IP core supplied by Microsemi.

The SmartFusion2 device design is implemented on the *SmartFusion2 Security Evaluation Kit Board* using an M2S090TS-FG484 device and the IGLOO2 device design is implemented on the *IGLOO2 Evaluation Kit Board* using an M2GL090TS-FGG484 device.

Design Example 1: Using SHA-256 Services in SmartFusion2 Device

The design consists an RC oscillator, a fabric CCC (FCCC), and the MSS. The fabric PLL(FPLL) is used to provide the base clock for the MSS. The system services run using C routine in the MSS, as described in the "Software Implementation" on page 10. In addition, a universal asynchronous receiver/transmitter (MMUART1) in the MSS is used to display the operation of the SHA-256 system service.

Hardware Implementation

The RC oscillator generates a 50 MHz input clock and the FPLL generates a 100 MHz clock from the RC oscillator. This 100 MHz clock is used as the base clock for the MSS. The MMUART_1 signals are routed through the FPGA fabric to communicate with the serial terminal program. The counter block is used to show that the device is up and running.

Figure 6 shows a block diagram of the first design example.
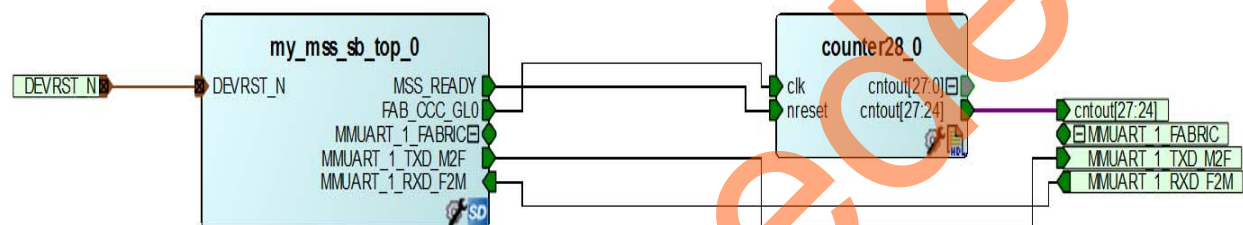
Figure 6 • **SmartFusion2 SHA-256 System Service Design**

Software Implementation

The software design performs the following operations:

- Initializes the system service driver
- Initializes MMUART_1
- Performs SHA-256 cryptographic services

sha_256_checksum()

The sha_256_checksum() function provides access to the SmartFusion2 SHA-256 encryption cryptographic service. It allows the user to provide 64 bytes (512-bits) of input data/text. This input is used for hashing to generate 256-bit message digest.

Running the Design

The following steps describe how to run the design on the SmartFusion2 Security Evaluation Kit board using the M2S090TS-FG484 device:

1. Connect the power supply to the M2S_EVAL_Kit board and power on the board using **SW7**.
2. Plug the FlashPro4 ribbon cable into the JTAG programming header on the SmartFusion2 Security Evaluation Kit board.
3. Program the SmartFusion2 Security Evaluation Kit board with the provided STAPL file (refer to "Appendix A: Design and Programming Files " on page 17) using FlashPro4.
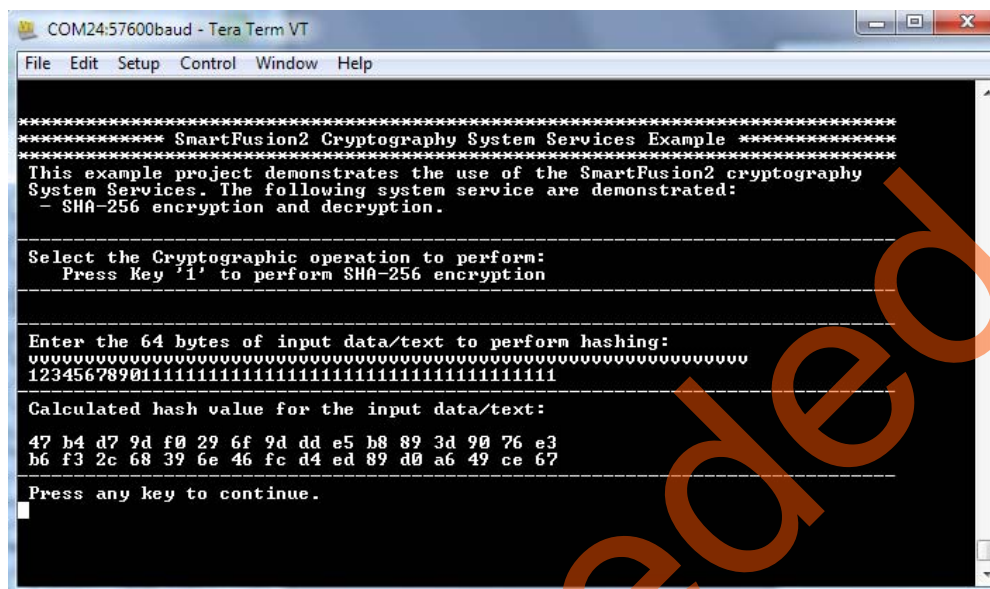
4. Connect the host PC to the J18 connector using the USB min-B cable.
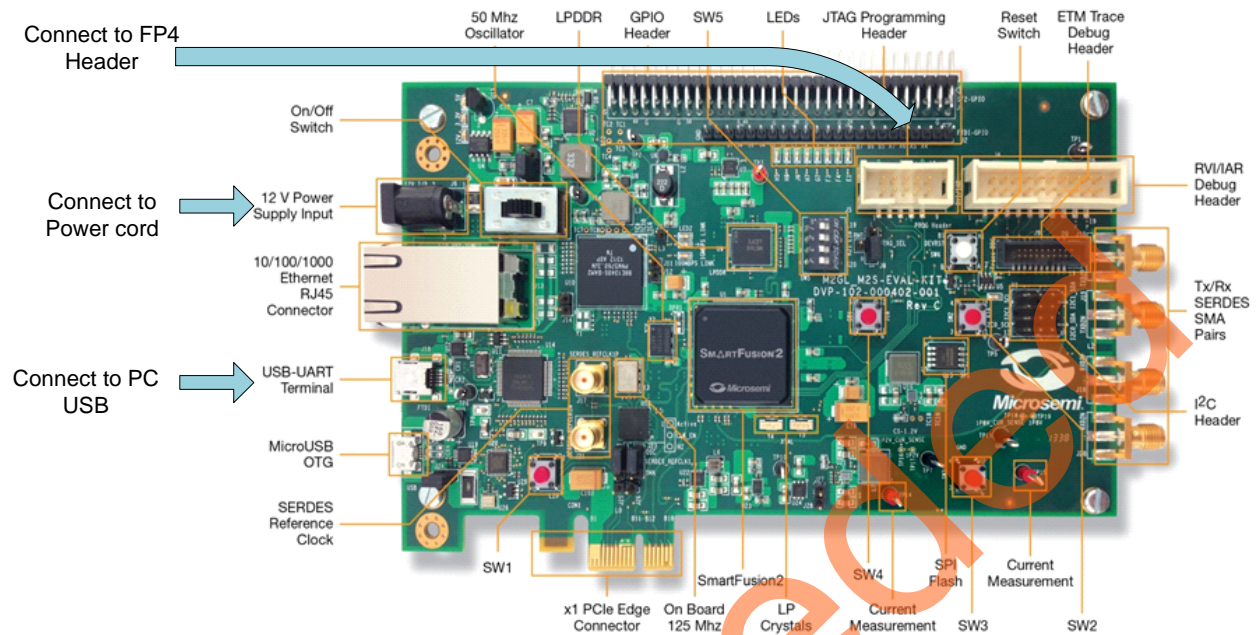
Figure 7 • **SmartFusion2 Security Evaluation Kit Board**

5. Invoke the SoftConsole integrated design environment (IDE), open the included SoftConsole project and launch the debugger.

6. Start a HyperTerminal session with 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If the computer does not have the HyperTerminal program, use other free serial terminal emulation program such as PuTTY or TeraTerm. For configuring HyperTerminal, TeraTerm, or PuTTY, refer to *Configuring Serial Terminal Emulation Programs Tutorial.*

7. Run the debugger in the SoftConsole tool. The **HyperTerminal** window shows various options to run the SHA-256 operation. Follow the instruction on the screen to run the example. Figure 8 shows the **HyperTerminal** window showing SHA-256 operation.

Figure 8 • **SHA-256 System Service Design using HyperTerminal**

Note: The ASCII-Hex notation is used as input by the program so that the data is easily readable. The ASCII characters are selected by the value from the ordered sixteen character set 0-9 and a-f. The input data goes from the first byte to the last byte of the multi-byte message entered and displayed from left to right as shown by the terminal emulator. Each input ASCII byte represents two Hex bytes. For example, if 1 is entered as ASCII input, it is converted into Hex input 31 for SHA-256 input. So, the 64-byte ASCII input is used as 512-bit Hex input.The additional firmware code can be added to convert ASCII-to-hex so that the input from HyperTerminal can be treated as Hex input instead of ASCII input. However, the message digest output is displayed in Hex format in endian-order (from left to right and then top to bottom), as shown in Figure 9.

Figure 9 • **Message Digest Output Mapping between SoftConsole Memory Viewer and HyperTerminal**

Design Example 2: Using SHA-256 Services in IGLOO2 Device

The design consists the following:

- IGLOO2 HPMS
- An on-chip 50 MHz RC oscillator
- A FCCC
- A CoreSysServices IP block
- A CoreRESET IP block
- A CoreABC IP block
- A CoreUARTapb IP block
- A Fabric state machine to control the CoreSysServices block
- An APB data block to reformat the message digest output, so that it can be displayed by a terminal emulator.

Hardware Implementation

Figure 10 shows the block diagram of the IGLOO2 design. In this design example, the main blocks are system builder block, Sysservice controller block, and UART controller block.
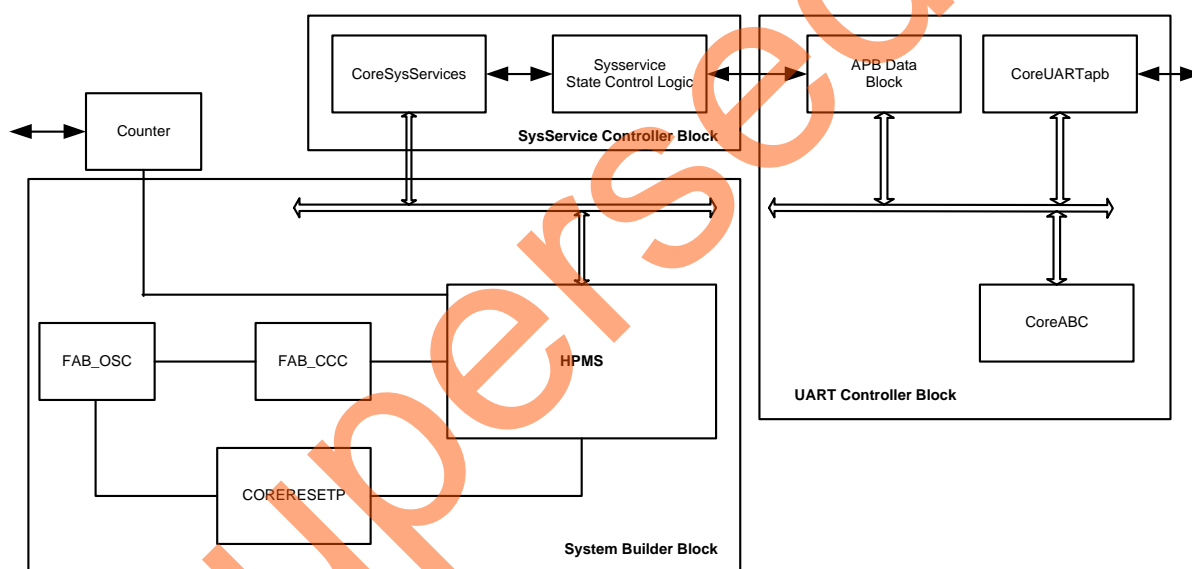
Figure 10 • **IGLOO2 SHA-256 System Service Design**

System Builder Block

This block includes a 50 MHz RC oscillator that is used as the main clock. It is used with a CCC to provide a 100 MHz reference clock to the HPMS. This 100 MHz clock is also used as main clock for the fabric blocks. CoreResetP generates reset signals for all the blocks.

Sysservice Controller Block

This is the main block that uses CoreSysServices IP to run the SHA-256 system services. CoreSysServices IP sends the SHA-256 command to the system controller through the COMM_BLK block in the HPMS. The fabric Sysservice state control logic connects to the user interface of CoreSysServices IP. The state control logic initiates the SHA-256 system service and captures the digest data from CoreSysService. The fabric Sysservice state block sends the input data/message that is basically a big-endian binary counter and the four MSBs are connected 4-bit DIP switch input in the board. The counter increments after every SHA-256 operation. The incremented value is used as input for the next operation.

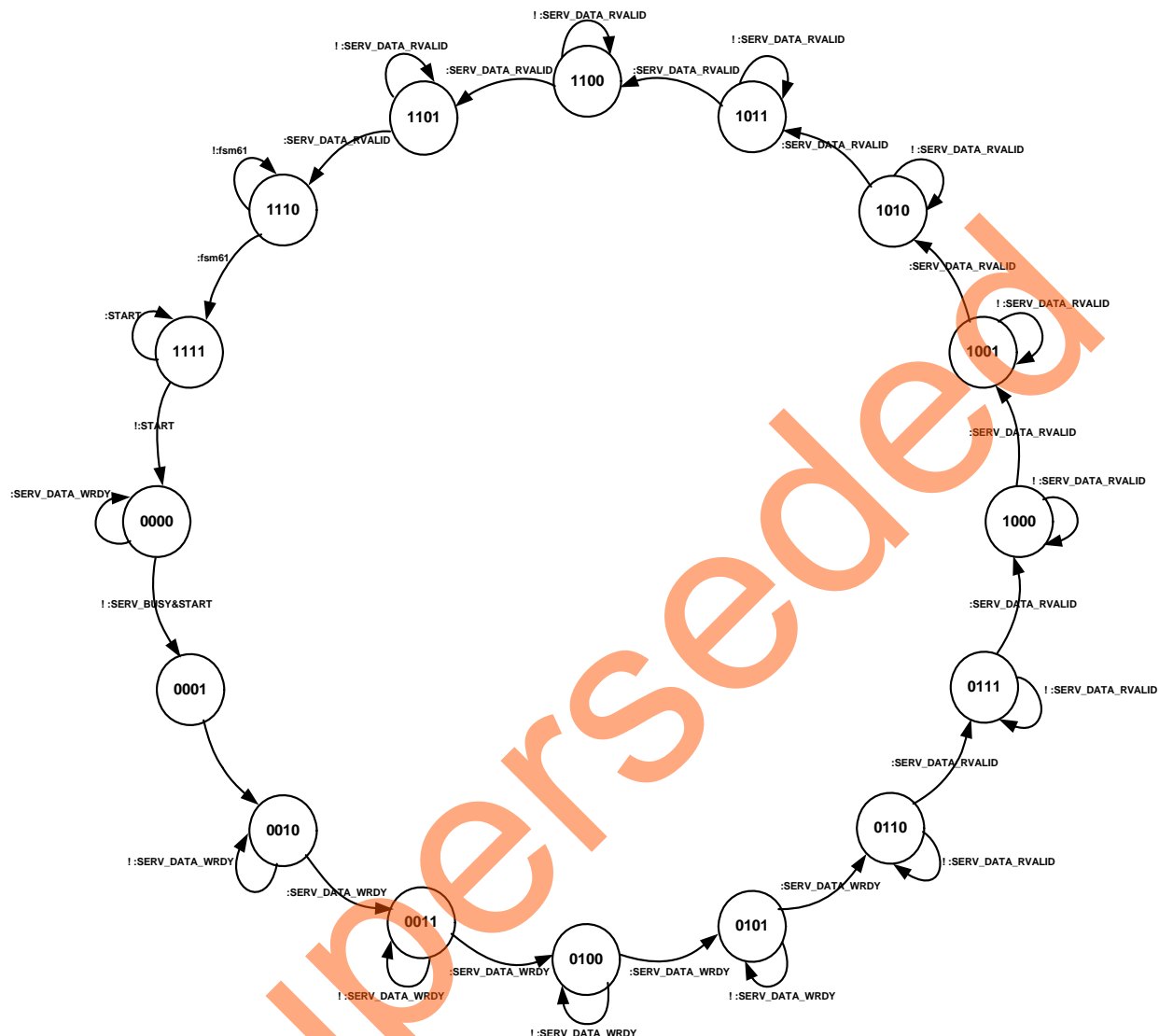Figure 11 shows the state machine for the Sysservice block.

Figure 11 • **Sysservice State block - State Diagram**

UART Controller Block

The UART controller block takes the user input and initiates the SHA-256 operation. It is also used to display the hashed output to HyperTerminal. The UART controller block has the following three main components:

- APB data block
- CoreABC block
- CoreUARTapb block

The APB data block captures the message digest data and converts the binary data to ASCII Hex data to display in human readable format on the HyperTerminal. The CoreABC program controls initiating fabric state machine and displays the data through the CoreUARTapb interface. In addition, there is a counter block used to show that the device is up and running.

Simulating IGLOO2 SHA-256 System Service Design

The design file includes the testbench files and ModelSim do-file to run simulation in the Libero SoC software. The simulation uses the system service memory model in the SmartFusion2 simulation library to exercise the data transfer between the HPMS and the fabric. The ModelSim do file, named `run_sha.do`, forces START signal in the CoreSysCtrl block to initiate the state machine for SHA-256 operation.

Figure 12 shows the simulation waveform displaying CoreSysServices IP user interface and AHB-lite interface.
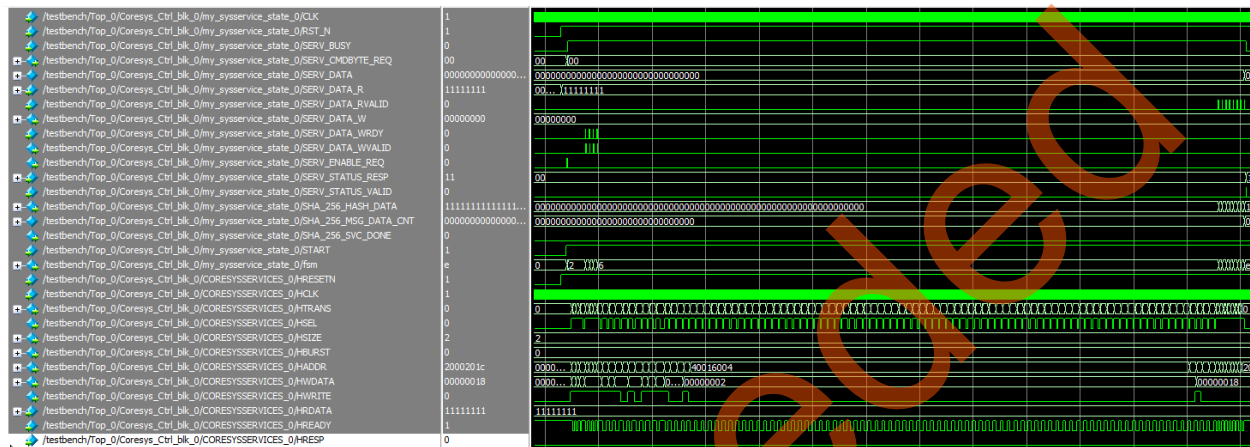
Figure 12 • **Sysservice State Block Simulation Waveform**

Running the Design

This section describes how to run the design in the IGLOO2 Evaluation Kit board using the M2GL090TS-1FGG484 device:

1. Connect the power supply to the IGLOO2 Evaluation Kit board and power on the board using **SW7**.

2. Plug the FlashPro4 ribbon cable into connector J5 (JTAG Programming Header) on the IGLOO2 Evaluation Kit board (refer to Figure 13 on page 16).

3. Connect the mini USB cable between the FlashPro4 and the USB port of the host PC.

4. Connect the host PC to the J18 connector using the USB min-B cable. Ensure that the USB to UART bridge drivers are automatically detected (can be verified in the Device Manager).
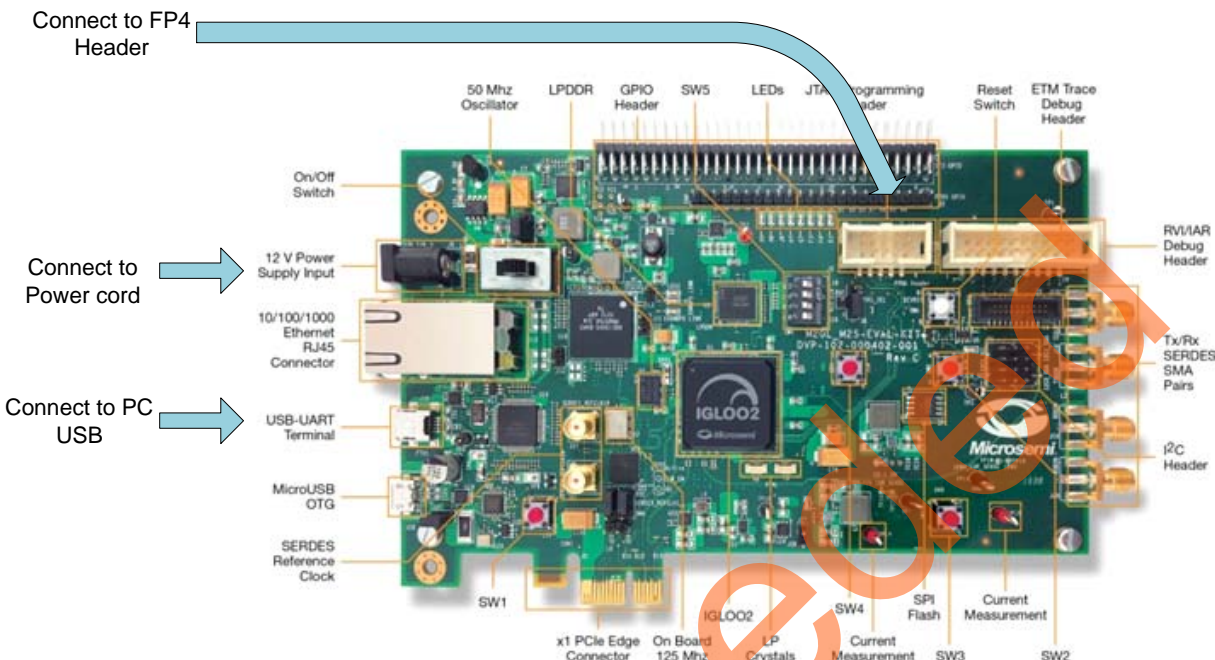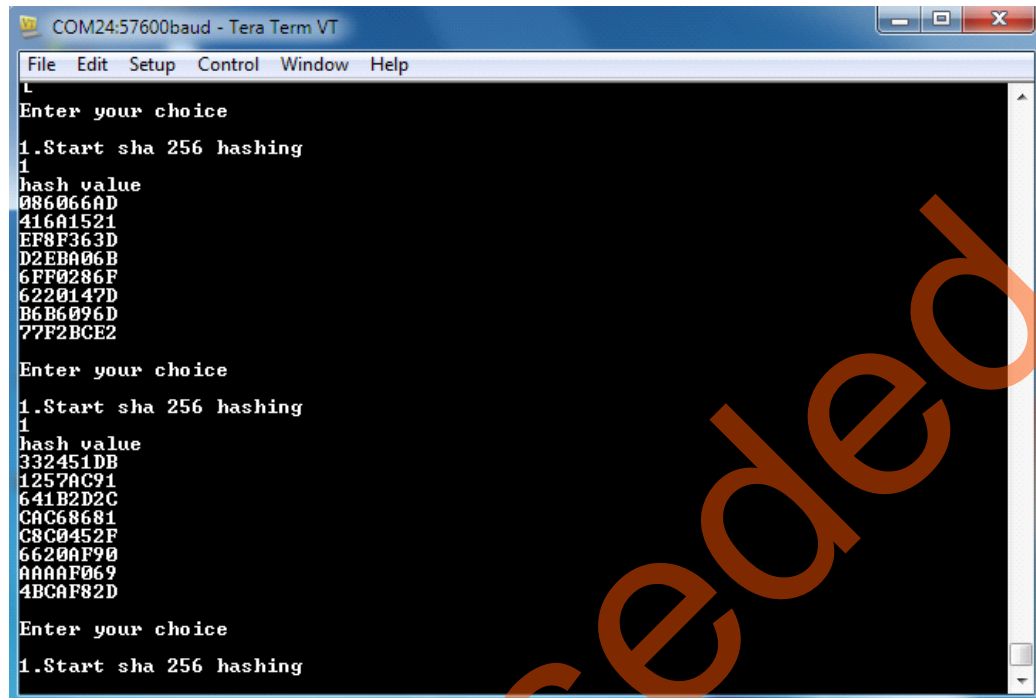
Figure 13 • **IGLOO2 Evaluation Kit Board**

5. If USB to UART bridge drivers are not installed, download and install the drivers from *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*.

6. Start a HyperTerminal session with 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If the computer does not have the HyperTerminal program, use other free serial terminal emulation program such as PuTTY or TeraTerm. Refer to *Configuring Serial Terminal Emulation Programs Tutorial* for configuring HyperTerminal, TeraTerm, or PuTTY.

7. Program the IGLOO2 Evaluation Kit board with the provided STAPL file (refer to "Appendix A: Design and Programming Files " on page 17) using FlashPro4.

After programming, the HyperTerminal displays a message to run the SHA-256 system services, as shown in Figure 14.

Figure 14 • **SHA-256 System Service Design in IGLOO2 using HyperTerminal**

Conclusion

The SmartFusion2 and IGLOO2 family of FPGAs are the most secure programmable logic devices. In the selected SmartFusion2 and IGLOO2 devices, the SHA-256 accelerator block can perform SHA-256 operation as defined in NIST FIPS180-3. The SHA-256 system services, along with the other Cryptographic Services offered, allows to use the SmartFusion2 and IGLOO2 devices in various secure applications.

Appendix A: Design and Programming Files

The SmartFusion2 and IGLOO2 SHA-256 design files can be downloaded from the Microsemi website:

http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_ac432_sha_256_liberov11p6_df

The SmartFusion2 design file consists both Libero VHDL and Verilog project, SoftConsole software project, and programming files (*.stp) for the SmartFusion2 Security Evaluation Kit. The IGLOO2 design file consists Libero VHDL and Verilog project and programming files (*.stp) for the IGLOO2 Evaluation Kit. Refer to the Readme.txt file included in the design file folder for the directory structure and description.

List of Changes

The following table shows the important changes made in this document for each revision.

Revision*	Changes	Page
Revision 3 (October 2015)	Updated the document for Libero v11.6 software release changes (SAR 71461)	NA
Revision 2 (March 2015)	Updated the document for Libero v11.5 software release (SAR 64228).	NA
Revision 1 (September 2014)	Initial release.	NA

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Ethernet Solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,600 employees globally. Learn more at **www.microsemi.com**.