Using Non-Deterministic Random Bit Generator (NRBG) Services in SmartFusion2 SoC and IGLOO2 FPGA Devices

Table of Contents

References

The following list of references is used in this document. The references complement and help in understanding the relevant Microsemi® SmartFusion®2 and IGLOO®2 field programmable gate array (FPGA) device flows and features that are demonstrated in this document.

- SmartFusion2 Microcontroller Subsystem User Guide
- SmartFusion2 System Controller User Guide
- SmartFusion2 Development Kit
- IGLOO2 FPGA High Performance Memory Subsystem User Guide
- IGLOO2 FPGA System Controller User Guide
- IGLOO2 Evaluation Kit

Tools Required

- The application design included with this application note uses Libero® System-on-Chip (SoC) v11.3

Introduction

The security-enabled SmartFusion2 system-on-chip (SoC) / IGLOO2 FPGA devices include a robust non-deterministic random bit generator (NRBG) service. NRBG is also sometimes called a true random number generator (TRNG). NRBG is used to generate random bit strings for various essential tasks, including the generation of the following:

- Secret or public keys
- Initialization vectors (for example, for use with various block-cipher encryption modes)

- Seeds for pseudo-random number generators
- Padding bits (for example, for RSA encrypted messages)
- Nonces (numbers used once)
- Non-cryptographic uses such as in gaming or Monte-Carlo scientific simulations

The NRBG block in SmartFusion2and IGLOO2 is designed to be compliant with the NIST SP800-90, NIST SP800-22, and BIS AIS-31 standards with a 256-bit security encryption. This application note provides a design example for using the NRBG service in SmartFusion2 and IGLOO2.

Overview of SmartFusion2/IGLOO2 NRBG Block

The NRBG block in SmartFusion2 SoC / IGLOO2 has the following two main components:

- A true random entropy source
- A deterministic random bit generator (DRBG), sometimes called a pseudo-random number generator (PRNG)

The entropy source is used to seed DRBG, which can generate many pseudo-random output bits from one seed. The NRBG block in SmartFusion2 SoC / IGLOO2 supports all commands defined in NIST SP800-90, such as creating an instantiation, generating random bits, and reseeding. They are supported at a design security strength of 256-bits. Up to 1,024, random bits can be returned per call to an instantiation.

The NRBG services are used for design security purposes by the System Controller, for example to generate nonces required in the various design security protocols, or to generate ephemeral design-security keys. You can additionally create up to two NRBG instantiations for any purpose, such as for data security end-applications.

Note: Access to the NRBG system services is only available on S version of the devices, such as M2S050TS and M2GL050TS.

Figure 1 shows the NRBG block in SmartFusion2 SoC / IGLOO2.

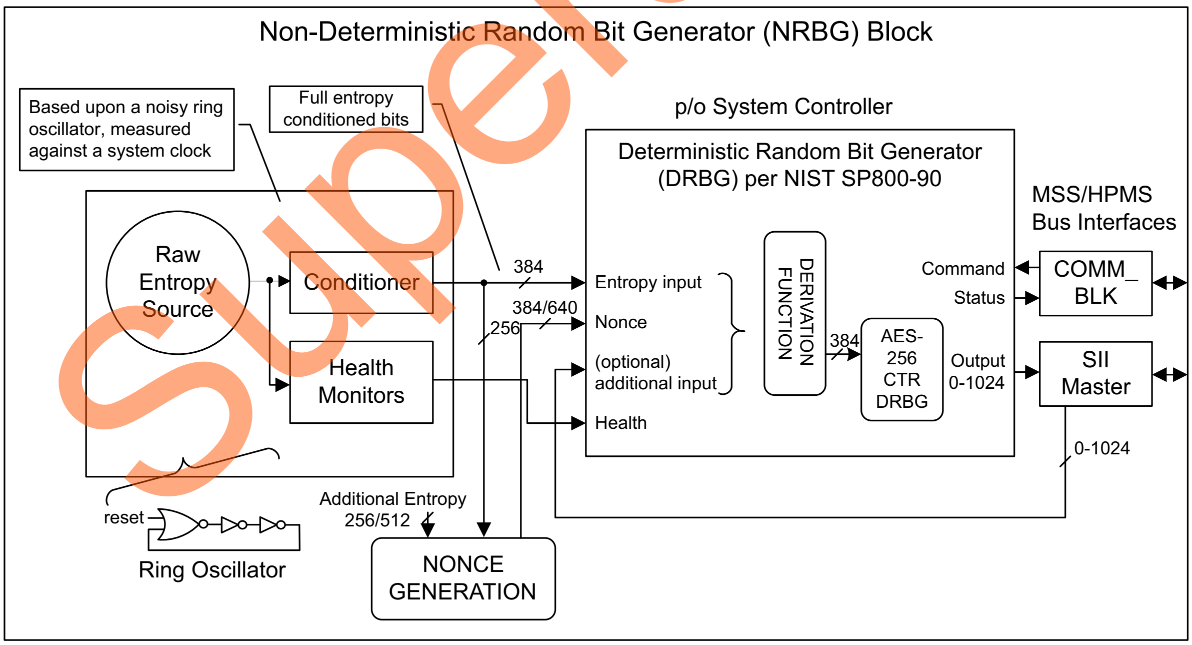

Non-Deterministic Random Bit Generator (NRBG) Block
Based upon a noisy ring oscillator, measured against a system clock
Full entropy conditioned bits
p/o System Controller
Deterministic Random Bit Generator (DRBG) per NIST SP800-90
MSS/HPMS Bus Interfaces
Raw Entropy Source
Conditioner
Health Monitors
384
384/640
256
Entropy input
Nonce
(optional) additional input
Health
DERIVATION FUNCTION
384
AES-256 CTR DRBG
Command
Status
Output 0-1024
COMM_ BLK
SII Master
0-1024
reset
Ring Oscillator
Additional Entropy 256/512
NONCE GENERATION

Figure 1 • **NRBG Block in SmartFusion2 SoC / IGLOO2**

The primary entropy source is ring-oscillator based, as shown in Figure 1. It supplies 384 true-random bits having full entropy to DRBG, upon instantiation and whenever DRBG is reseeded. Additional entropy is used in generating a 384-bit nonce, which is supplied as additional seed material when DRBG is first instantiated. In all devices, there is a minimum of 256-bits of additional entropy added to the nonce generation, which also uses 256-bits from the primary entropy source to randomize it for each new instantiation. For devices having the SRAM-PUF feature, the additional entropy is supplemented with another 256-bits (making a total of 512-bits additional entropy), and the nonce length is increased by 256-bits from 384-bits to 640-bits. You have the option of supplying up to 128 bytes of additional input with each call to the Generate() function.

Figure 2 shows the System Controller block in SmartFusion2. The NRBG block resides in the System Controller and accesses via the communication block (COMM_BLK). There are the following two COMM_BLK instances:

- One in the MSS that the user interfaces with
- One that communicates with the first one, which is located in the System Controller

The COMM_BLK consists of an APB interface, eight byte transmit-FIFO, and an eight byte receive-FIFO. It provides a bidirectional message passing facility between MSS and the System Controller. System services are initiated by the user using the COMM_BLK interface attached to the MSS, which can be read or written to by any master on AHB bus matrix; typically either the Cortex™-M3 or a design in the FPGA fabric. The System Controller then uses the SII Master, which is an MSS bus master controlled by the System Controller, to get the additional details and options of the NRBG services at an address supplied in the original COMM-BLK command, pointing where this structured data is stored in memory by the user before invoking the command. On the completion of the requested service, System Controller returns a status message via the COMM_BLK. Depending on the command, there might be other data and repercussions generated as a result of running the command.

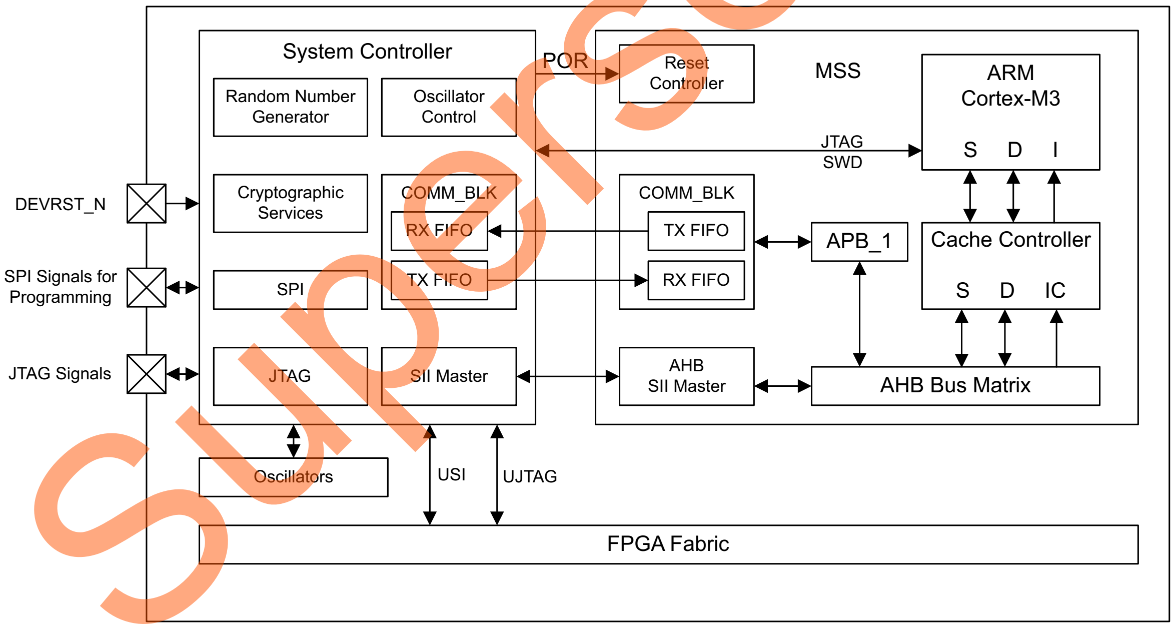


***Figure 2* • System Controller Interface to the MSS Block in SmartFusion2**

Figure 3 shows the System Controller block in IGLOO2. The architecture is similar to that in SmartFusion2, except that the COMM_BLK in System Controller communicates with COMM_BLK in HPMS. A fabric master is required to use the NRBG services. Microsemi provides the CoreSysService Directcore IP with a simple user interface to use various NRBG system services.

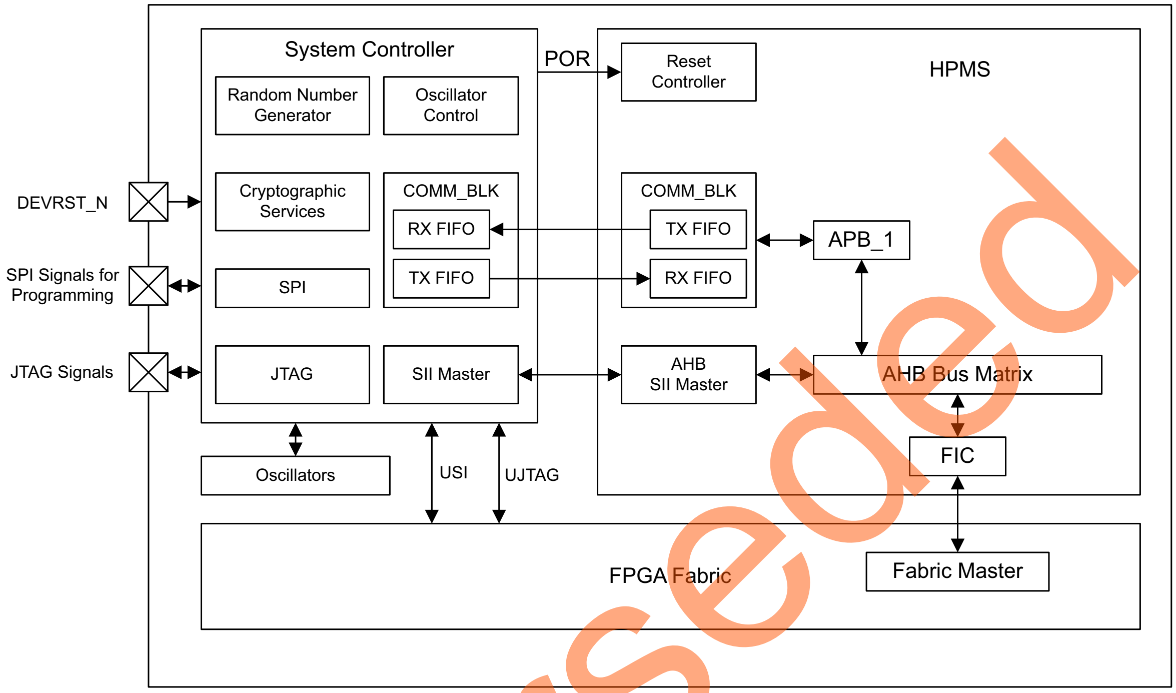


***Figure 3* • System Controller Interface to the HPMS Block in IGLOO2**

Refer to *the IGLOO2 FPGA System Controller User Guide* and *SmartFusion2 System Controller User Guide* for more information on System Controller. Refer Communication Block chapter in the *SmartFusion2 ARM Cortex-M3 and Microcontroller Subsystem User Guide* and *IGLOO2 FPGA High Performance Memory Subsystem User Guide* for more information on COMM_BLK.

Services in SmartFusion2 and IGLOO2 Devices

The NRBG block can provide random number services for data security in select models of SmartFusion2and IGLOO2 devices. These are designated by an “S” in the model number following the capacity indicator, as described in the Ordering Information section of the product brief. The random number services, also known as NRBG services, supported by the SmartFusion2 and IGLOO2 NRBG block are briefly described in "Using NRBG Services" section on page 5. Refer to the *Non-Deterministic Random Bit Generator (NRBG) Services* section of the *SmartFusion2 System Controller User Guide*, and *IGLOO2 FPGA System Controller User Guide* for more information on the system services.

- **Self Test**: This service invokes all DRBG health tests. If any health test fails, a fatal error is entered; otherwise the state of DRBG and all instantiations are not affected. The fatal error can only be removed by a device reset or user invocation of the NRBG reset service.
- **Instantiate**: This service instantiates DRBG with an optional personalization string. The personalization string must be in the range 0-128 bytes, inclusive. An error is returned from DRBG if this field is out of range.
- **Generate**: This service generates a random bit sequence up to 128 bytes long. An error is returned from DRBG if this field is out of range.
- **Reseed**: A reseed occurs automatically for every 65535 generate requests. This service may be used to force a Reseed operation.

- **Uninstantiate**: This operation removes a previously instantiated DRBG and releases the associated memory resources for later use by a new instantiation. The working state of the DRBG instantiation is zeroized before the release.
- **Reset**: This operation removes all DRBG instantiations and resets the DRBG. This service is the only mechanism to recover from a catastrophic DRBG error without physically resetting the device. All active instantiations are automatically purged.

Using NRBG Services

In SmartFusion2/IGLOO2, you can use CoreSysServices IP to initiate various NRBG services in the System Controller via the COMM_BLK in HPMS. If you want to write your own state machine to initiate various NRBG services, you can use the following steps. The NRBG services can also be accessed using the mss_sys_services driver in SmartFusion2.

The steps for running the various NRBG services are similar. However, you must run the NRBG Instantiate service before running the NRBG services.

Using the Instantiate Service

The following procedure describes the steps for using the Instantiate service:

1. Set up the DRBGINSTANTIATE descriptor in the user memory space (for example, eSRAM address 0x20001000) as shown in Table 1, containing two 4-byte words.
   a. Write PER_STRING_PTR (pointer to RBG personalization string, for example, 0x20002000) to eSRAM address 0x20001000: write 0x20002000 to eSRAM address 0x20001000.
   b. Write PER_STRING_LENGTH, DRBGHANDLE to eSRAM address 0x20001004. For example: write 0x00000004 (length of personalization string = 4 bytes) to eSRAM address 0x20001004.

Table 1 **• DRBGINSTANTIATE Structure**

Offset	Length (Bytes)	Field	Description
0	4	PER_STRING_PTR	Pointer to RBG personalization string
4	1	PER_STRING_LENGTH	Length of personalization string in bytes. Length must be in the range of 0-128 bytes inclusive.
5	1	RESERVED	Reserved
6	1	DRBGHANDLE	Returned DRBG handle

   c. Write PER_STRING value to PER_STRING_PTR address defined in "1.": write 0x00000000 (PER_STRING) to eSRAM address 0x20002000.
2. Enable the COMBLK_INTR interrupt from the COMM_BLK block to fabric by enabling COMBLK_INTR_ENBL bit (bit 29) in INTERRUPT_ENABLE0 register at address 0x40006000: write 0x20000000 to address 0x40006000.
3. Setup the registers in the COMM_BLK and send the command.
   a. Enable the COM_BLK by writing 1 to ENABLE bit of COMM_BLK CTRL register: write 0x00000010 to address 0x40016000.
   b. Enable TXTOKAY interrupt (TXT FIFO non full) in COMM_BLK by writing 1 to TXTOKAY bit of Interrupt Enable register: write 0x00000001 to address 0x40016008.
   c. Wait for COM_BLK_IN interrupt.
   d. Read COMM_BLK Status register (0x40016004) and check for TXTOKAY to be set. If set, proceed to the next step.

e. Send the command via the COMM_BLK FRAME_START8 register (0x40016018): write 0x29 to 0x40016018. Table 2 shows the NRBG services commands.

Table 2 • **NRBG Services Commands**

Non-Deterministic Random Bit Generator Services	Decimal	Hex
Self Test Service	40	0x28
Instantiate Service	41	0x29
Generate Service	42	0x2A
Reseed Service	43	0x2B
Uninstantiate Service	44	0x2C
Reset Service	45	0x2D

f. Wait for COM_BLK_IN interrupt.

g. Read COMM_BLK Status register (address 0x40016004) and check for TXTOKAY to be set. If set, proceed to next step.

h. Set Transmit FIFO in word (32-bit) mode using CONTROL register (0x40016000): write 0x00000014 to address 0x40016000.

i. Send the DRBGINSTANTIATE descriptor address via the DATA32 register (address 0x40016014): write 0x20001000 to address 0x40016014.

j. Enable COMMAND interrupt (receive FIFO has the command marker set) in COMM_BLK by writing 1 to COMMAND bit of Interrupt Enable register: write 0x00000080 to address 0x40016008.

k. Wait for COM_BLK_IN interrupt.

l. Set receive FIFO in byte (8-bit) mode using CONTROL register (0x40016000): write 0x00000010 to address 0x40016000.

System controller uses the command and DRBGINSTANTIATE descriptor address, and executes the DRBG instantiate service. It sends the response to COMM_BLK receive FIFO.

4. Check the RCVOKAY bit in the COMM_BLK STATUS register. Read bit 7 of the STATUS register (address 0x40016004) in the COMM_BLK. A value of 1 indicates that the command is executed.

5. Check the command, status code and DRBGINSTANTIATE descriptor pointer in the COMM_BLK STATUS register.

a. Read the Command Byte register (address 0x40016018) of the COMM_BLK.

b. Enable RCVOKAY (receive FIFO non empty) in COMM_BLK by writing 1 to RCVOKAY bit of Interrupt Enable register: write 0x00000002 to address 0x40016008.

c. Wait for COM_BLK_IN interrupt.

d. Read COMM_BLK Status register (address 0x40016004) and check for RCVOKAY to be set. If set, proceed to next step.

e. Read Byte Data register (address 0x40016010) and check the command (1st byte) and status code (2nd byte).

f. Set receive FIFO in word (32-bit) mode using CONTROL register (0x40016000): write 0x00000018 to address 0x40016000.

g. Wait for COM_BLK_IN interrupt.

h. Read COMM_BLK Status register (address 0x40016004) and check for COMMAND and RCVOKAY to be set. If set, proceed to next step.

i. Read Word Data register (address 0x40016014) and check the DRBGINSTANTIATE descriptor address.

6. Read the DRBGHANDLE value from COMM_BLK. Read the eSRAM address second byte location (0x20001000) to read DRBGHANDLE.

Table 3 • **DRBG Generate Service Response**

Offset	Length (Bytes)	Field	Description
0	1	CMD = 41	Command
1	1	STATUS	Command status
2	4	DRBGINSTANTIATEPTR	Pointer to DRBGINSTANTIATE structure

Table 4 • **NRBG Service Response Status Codes**

Status Code	Description
0x00	Successful completion (DRBGHANDLE is valid)
0x01	Catastrophic error
0x02	Maximum instantiations exceeded
0x03	Invalid handle
0x04	Generate request too big
0x05	Maximum length of additional data exceeded
0x7F	HRESP error occurred during MSS/HPMS transfer
0XFE	Service disabled by factory security
0XFF	Service disabled by user security

Figure 4 shows NRBG instantiate service data flow diagram in SmartFusion2 devices.

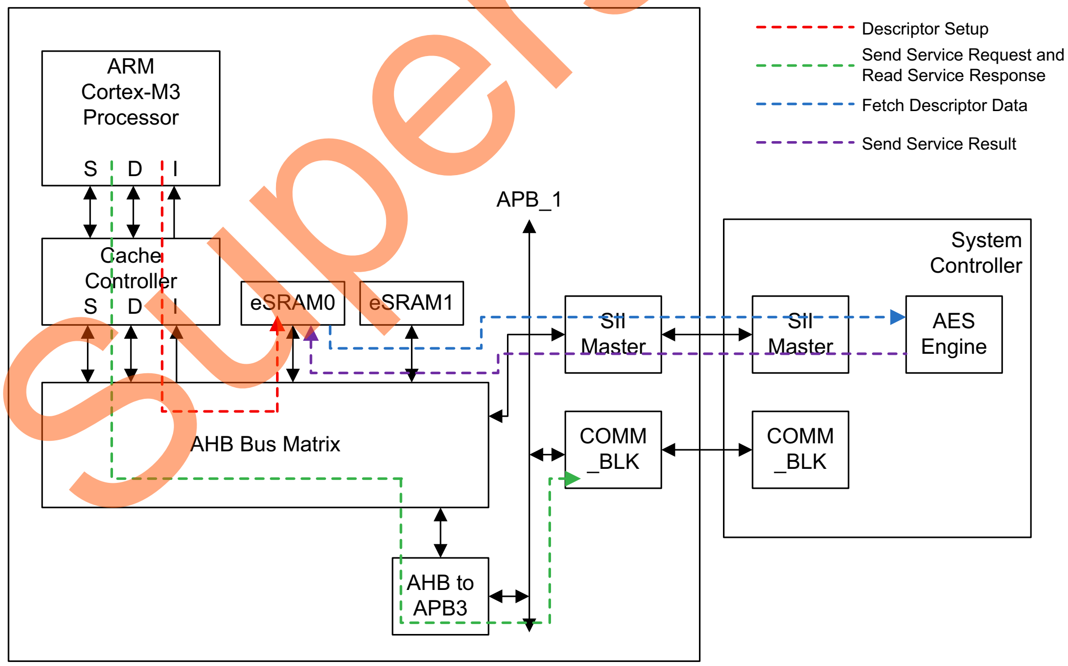


Figure 4 • **NRBG Instantiate Service Data Flow Diagram in SmartFusion2 Devices**

You can use similar steps for other services.

Note: When using a SmartFusion2 device, you can use system services driver in the *Firmware Catalog* of the Libero software to perform various services rather than going through these complex steps. Similarly, in IGLOO2 device you can use *CoreSysServices* IP to use these NRBG services rather than writing your own complex state machine.

Design Description

This design demonstrates using the NRBG services in SmartFusion2 and IGLOO2 devices to generate random bits and also run various NRBG services. The SmartFusion2 design is implemented on SmartFusion2 Development Kit Board using M2S050TS-FG896 device and the IGLOO2 design is implemented on IGLOO2 EVAL Kit Board using M2GL010TS-FG484 device.

SmartFusion2 NRBG Design

The design consists of RC Oscillator, fabric PLL, and MSS. The fabric PLL is used to provide the base clock for the MSS. The system services are run using various C routines in MSS, refer to Figure 5. The universal asynchronous receiver/transmitter (UART0) in the MSS is used to display the random number and to run other functions.

Hardware Implementation

Figure 5 shows the block diagram of the SmartFusion2 NRBG design example. The RC oscillator is used to generate a 50 MHz reference clock for the fabric PLL. The fabric PLL is used to generate 100 MHz base clock for MSS. The MMUART_0 signals are routed through the FPGA fabric for communicating with the serial terminal program.

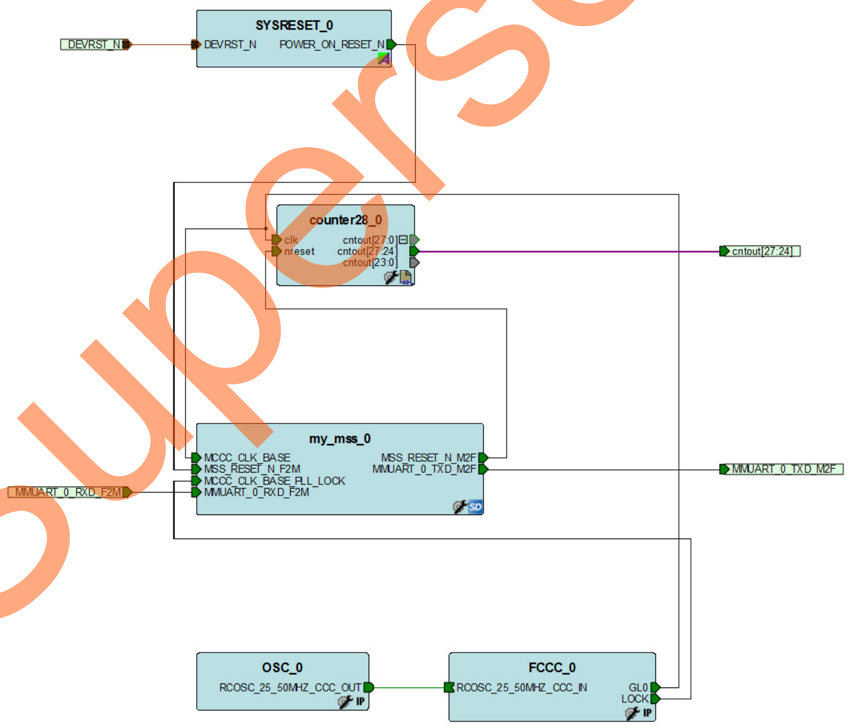


***Figure 5* • Block Diagram of SmartFusion2 NRBG Design Example**

Software Implementation

The design example performs the following operations:

- Initializes the System Controller Enable
- Initializes MMUART_0
- Performs various DRBG functions:
  - Generates a random bit string
  - Releases the DRBG instantiation
  - Runs self-test on the DRBG
  - Resets DRBG functions
  - Creates (reserves) a DRBG instantiation

generate_random_bits()

Generates a random-bit sequence. Enables to request a random bit string up to 1024-bits per word and a count of random words of a specified length. For example, it enables to generate 2000 words having 512 random bits in each.

release_drbg_service()

Removes a previously instantiated DRBG and releases the associated memory resources for later use by a new instantiation. The working state of the DRBG instantiation is zeroized before the release.

self_test_service()

Invokes all DRBG health tests. If any health test fails, a fatal error condition is served. It requires device reset to recover from the error.

reset_drbg_service()

Removes all DRBG instantiations and resets the DRBG. It is the only mechanism to recover from a catastrophic DRBG error without physically resetting the device.

reserve_drbg_service()

Instantiates a DRBG instance. A maximum of two concurrent user instances are available.

reseed_service()

Forces a reseed operation.

Note: A DRBG reseed service occurs automatically for every 65,535 generate requests, or if the prediction resistance flag is set in the generate data structure used with the Generate command.

Running the Design

The following procedure describes running the design in SmartFusion2 Development Kit - Microsemi using the M2S050TS-FG896 device.

1. Connect the power supply to the SmartFusion2 Development Kit board.
2. Plug the FlashPro4 ribbon cable into JTAG Programming Header on the SmartFusion2 Development Kit board.
3. Program the SmartFusion2 Development Kit board with the provided STAPL file (refer to "Appendix A - Design and Programming Files" on page 14) using FlashPro4.
4. Connect one end of the USB Mini-B cable to the J24 connector provided in the SmartFusion2 Development Kit board. Connect the other end of the USB cable to the host PC.
5. Invoke the SoftConsole integrated design environment (IDE) from the Libero SoC 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, any free serial terminal emulation program such as PuTTY or Tera Term can be used. Refer to *Configuring Serial Terminal Emulation Programs tutorial* for configuring HyperTerminal, Tera Term, or PuTTY.

7. Run the debugger in SoftConsole. The HyperTerminal window shows various options to run the DRBG functions.

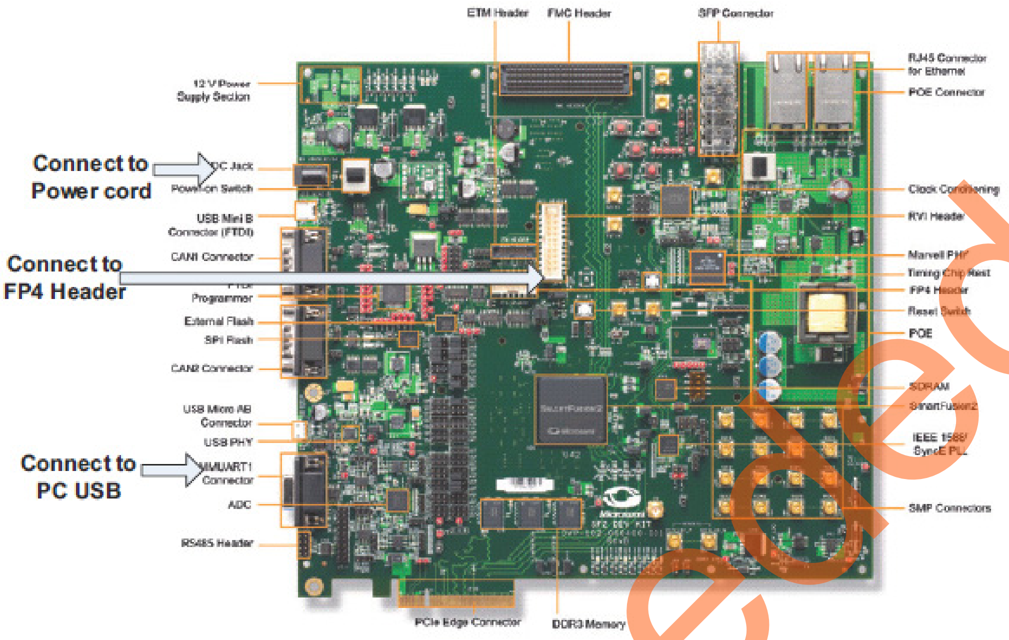


Figure 6 • **SmartFusion2 Development Kit Board**

Following is the code of the design example:

```
**********************************************************************
*******SmartFusion2 Random Bit Generator System Services Example*******
**********************************************************************
This example project exercises the random bit generator system services.
----------------------------------------------------------------------

DRBG reserve successful.
DRBG self test successful.
Press "1" to generate random numbers.
1
Number of random bytes to generate (1 to 128): 4
Total number of random number to generate (1 to 50000): 4
----------------------------------------------------
DRBG values are:
----------------------------------------------------
9eaf823f
ad922810
56d226ef
075f13c3

DRBG generate successful:
Press "1" to generate random numbers.
Press "2" to release DRBG.
Press "3" to run self test on DRBG.
Press "4" to reset DRBG.
Press "5" to reserve DRBG.
Press "6" to reseed DRBG.
2
DRBG release successful.
Press "1" to generate random numbers.
Press "2" to release DRBG.
Press "3" to run self test on DRBG.
Press "4" to reset DRBG.
Press "5" to reserve DRBG.
Press "6" to reseed DRBG.
5
DRBG reserve successful.
Press "1" to generate random numbers.
Press "2" to release DRBG.
Press "3" to run self test on DRBG.
Press "4" to reset DRBG.
Press "5" to reserve DRBG.
Press "6" to reseed DRBG.
3
DRBG self test successful.
Press "1" to generate random numbers.
Press "2" to release DRBG.
Press "3" to run self test on DRBG.
Press "4" to reset DRBG.
Press "5" to reserve DRBG.
Press "6" to reseed DRBG.
```

IGLOO2 NRBG Design

This design example consists of the IGLOO2 high-performance memory subsystem (HPMS), on-chip 50 MHz RC oscillator, Fabric CCC, CoreSysServices IP, CoreRESET, CoreABC, CoreUART_apb, fabric state machine to control block CoreSysServices IP, and an APB data block to capture NRBG data.

Hardware Implementation

Figure 7 on page 12 shows the block diagram of the design example. The 50 MHz RC oscillator is used as the main clock. It is used with CCC to provide a 100 MHz reference clock to the HPMS. This 100 MHz clock is also used as the main clock for the fabric blocks. The HPMS is configured to use CoreRESETP, to generate reset signals for all the blocks. The CoreSysServices IP is configured to use the NRBG services. It sends various DRBG commands to the System Controller via COMM_BLK block in the HPMS. The fabric SysService state control logic controls the sequence of system service commands and captures the NRBG data from CoreSysservices IP. The APB data block captures the NRBG data values and converts the Hex data to ASCII Hex data to display in the correct format to the HyperTerminal. CoreABC program controls initiating SysService state control logic and displaying the data via CoreUART_apb. The Fabric logic also consists of a counter block to display the counter value via LEDs to indicate that the design is up and running.

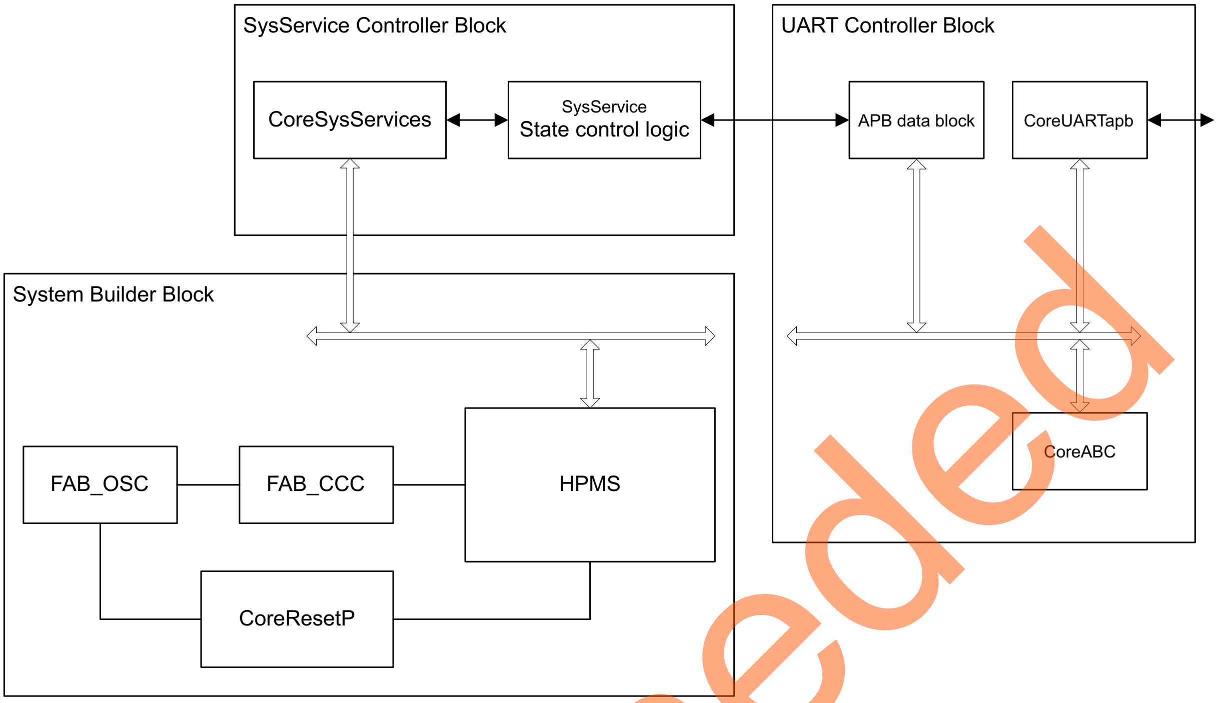


***Figure 7* • System Service State Machine Block Diagram**

Running the Design

The following procedure describes running the design example in IGLOO2 EVAL Kit Board using the M2GL010TS-FG484 device:

1. Plug the FlashPro4 ribbon cable into connector J5 (JTAG Programming Header) on the IGLOO2 EVAL Kit Board.
2. Connect the mini USB cable between the FlashPro4 and the USB port of the PC.
3. Connect one end of the USB mini cable to the J18 connector provided on the IGLOO2 EVAL kit. Connect the other end of the USB cable to the host PC. Ensure that the USB to UART bridge drivers are automatically detected (can be verified in the Device Manager). 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.
4. 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, any free serial terminal emulation program such as PuTTY or Tera Term can be used. Refer to *Configuring Serial Terminal Emulation Programs tutorial* for configuring HyperTerminal, Tera Term, or PuTTY.

5. Program the IGLOO2 EVAL Kit Board with the provided STAPL file using FlashPro4. Refer to "Appendix A - Design and Programming Files" on page 14 for more information.

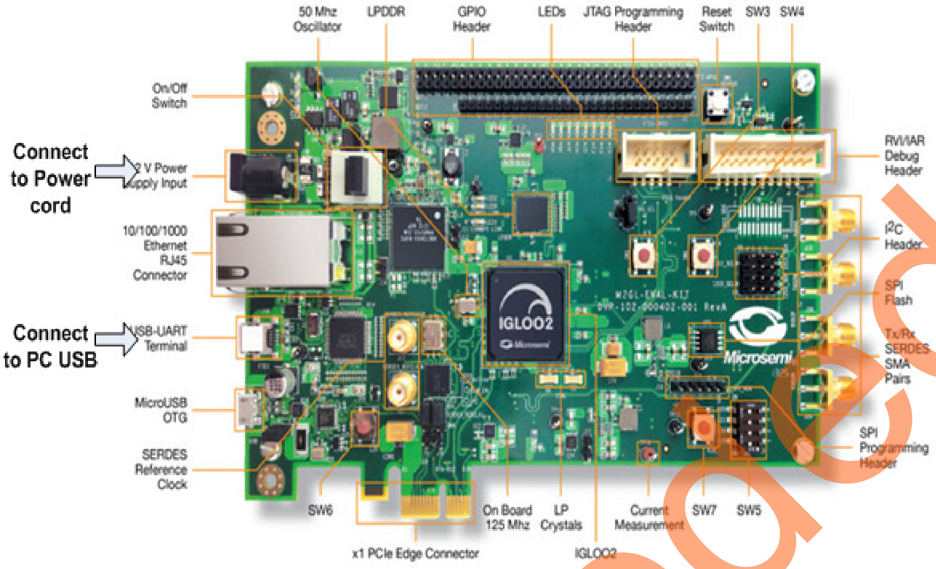
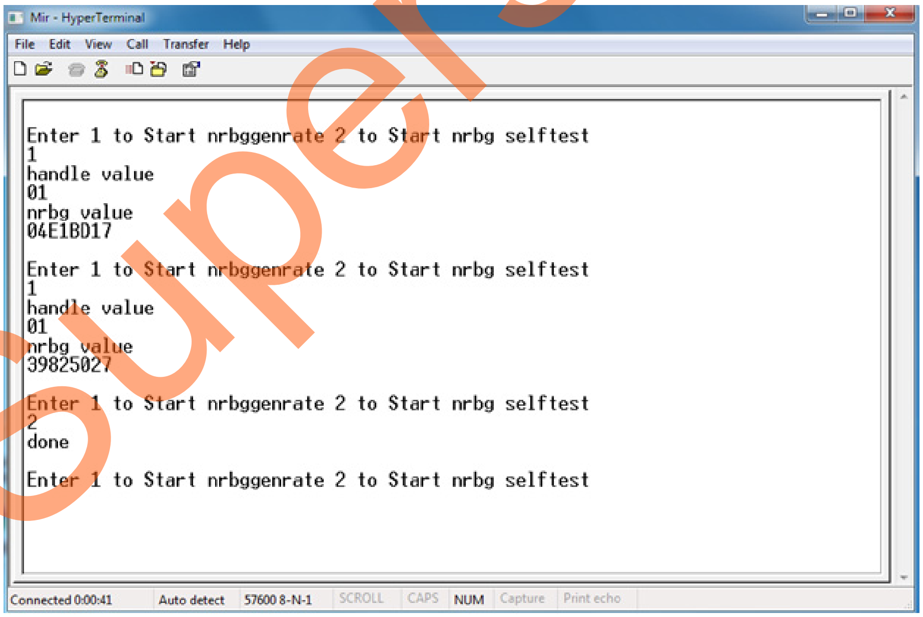


Figure 8 **• IGLOO2 Eval Kit**

After programming, HyperTerminal displays a message to run the NRBG system services. You can choose the required option to run NRBG generate and selftest services as shown in Figure 9.



```
Enter 1 to Start nrbggenrate 2 to Start nrbg selftest
1
handle value
01
nrbg value
04E1BD17

Enter 1 to Start nrbggenrate 2 to Start nrbg selftest
1
handle value
01
nrbg value
39825027

Enter 1 to Start nrbggenrate 2 to Start nrbg selftest
2
done

Enter 1 to Start nrbggenrate 2 to Start nrbg selftest
```

Figure 9 **• HyperTerminal showing IGLOO2 NRBG Design Result**

Conclusion

SmartFusion2 SoC / IGLOO2 FPGA devices include a robust NRBG service. The NRBG service is designed to be compliant with the NIST SP800-90, NIST SP800-22, and BIS AIS-31 standards, including all required health monitors. The DRBG incorporates DPA countermeasures for added security. This application note describes how to use various system services from MSS using Coretex™-M3 program and also with fabric logic using CoreSySservices IP.

Appendix A - Design and Programming Files

You can download the SmartFusion2 design files from
http://soc.microsemi.com/download/rsc/?f=SF2_IGL2_nrbg_DF. The design files consist of a Libero Verilog, SoftConsole software project, and programming files (*.stp) for the SmartFusion2 SoC FPGA Development Kit. Refer to the Readme.txt file included in the design files for the directory structure and description.

You can download the IGLOO2 design files from
http://soc.microsemi.com/download/rsc/?f=SF2_IGL2_nrbg_DF. The design files consist of a Libero Verilog project and programming files (*.stp) for the IGLOO2 Evaluation Kit. Refer to the Readme.txt file included in the design files for the directory structure and description.

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 1 (April 2014)	Updated the document for Libero v11.3 (SAR 56594).	NA
	Removed separate programming file links in "Appendix A - Design and Programming Files" section as these are part of the design file links now.	14
Revision 0 (November 2013)	Initial release.	NA

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and 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; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.