

Table of Contents

Purpose	1
Introduction	1
Accessing eNVM from FPGA Fabric	1
Accessing eSRAM from FPGA Fabric	2
References	2
Microsemi Publications	2
Design Requirements	3
Design Description.	3
Hardware Implementation	3
SmartDesign Components	4
Simulation	13
eNVM Simulation	13
eSRAM Simulation	15
Setting Up the Design	15
Programming the Demo Design	17
Configuring the Device	18
Running the Design	20
eNVM Write and Read Operations	20
eSRAM Write and Read Operation.	23
Conclusion	24
Appendix A – Design Files.	25
Appendix B	26
eNVM Write Operation	26
eNVM Read, eSRAM Write and Read Operations	26
	27

Purpose

This application note describes how to access the embedded nonvolatile memory (eNVM) and embedded static random access memory (eSRAM) from FPGA fabric in SmartFusion[®]2 system-on-chip (SoC) FPGA and IGLOO[®]2 FPGA devices.

Introduction

This application note describes the following:

- Accessing the eNVM from FPGA fabric
- · Accessing the eSRAM from FPGA fabric



Accessing eNVM from FPGA Fabric

SmartFusion2 SoC FPGA and IGLOO2 FPGA devices have a maximum of two on-chip 256 KB flash memories called eNVM. The eNVM stores the application code image or data required to be stored by the end application. The eNVM block is interfaced through the eNVM controller to the AHB bus matrix.

In SmartFusion2 SoC FPGA and IGLOO2 FPGA devices, the eNVM can be initialized by custom logic in FPGA fabric (Fabric Master).

In this application note, the Fabric master writes and reads from 25th page (address starting from 0x60000C80 to 0x60000CFC) of the eNVM.

For more information about eNVM initialization methods, refer to Application Note AC391.

Accessing eSRAM from FPGA Fabric

SmartFusion2 SoC FPGA and IGLOO2 FPGA devices have two eSRAM blocks, each of 32 Kbytes, for data read and write operations. These eSRAM blocks are interfaced through eSRAM controllers to the AHB bus matrix.

In SmartFusion2 SoC FPGA and IGLOO2 FPGA devices, the eSRAM can be accessed by custom logic in FPGA fabric (Fabric Master).

In this application note, Fabric master write and reads from 32 eSRAM locations (0x20000000 to 0x20000080).

References

The following list of references is used in this document

Microsemi Publications

- IGLOO2 System Builder User Guide
- IGLOO2 High Performance Memory Subsystem User Guide
- IGLOO2 FPGA Programming User Guide
- SmartFusion2 MSS Embedded Nonvolatile Memory (eNVM) Simulation
- SmartFusion2 Microcontroller Subsystem User Guide
- SmartFusion2 System Builder User Guide





Design Requirements

Table 1 lists the design requirements.

Table 1 • Design Requirements

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Evaluation Kit (M2S025T - FGG484):	Rev C or later
 12 V adapter (provided along with the kit) 	
 FlashPro4 programmer (provided along with the kit) 	
IGLOO2 Evaluation Kit:	Rev C or later
 12 V adapter (provided along with the kit) 	
 FlashPro4 programmer (provided along with the kit) 	
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero [®] SoC (System-on-Chip)	11.4

Design Description

The design examples included with this application note uses the following DIP switches:

- SW5-1 DIP switch: Used to start the eNVM write and read operations. An incremental data
 pattern starting from 0x0000000 to 0x0000001F is written to 25th page of the eNVM.
- SW5-2 DIP switch: Used to start the eSRAM write and read operations. An incremental data starting from 0xA1B2C300 to 0xA1B2C31F is written to the eSRAM locations starting from 0x20000000 to 0x20000080 respectively.

During the eSRAM or eNVM read operation, the read data from the eNVM or eSRAM is stored in the Fabric SRAM. The SmartDebug tool in Libero SoC verifies the write and read operations performed on the eNVM and eSRAM.

The design example uses two RTL FSMs—one FSM provides the eNVM or eSRAM write and read commands. The other FSM is an AHB master which receives these commands and communicates with the selected memory using AHB bus matrix through FIC_0 (Fabric Interface Controller) interface.

The read operations of the eNVM and the read and write operations of the eSRAM are simple AHB transactions. eNVM write requires a separate set of command sequences. For more information about this, see "Appendix B" on page 26

Hardware Implementation

The hardware implementation involves configuring the Device Features, Memory, Peripheral, and Clocks pages using System Builder. Configuring the TPSRAM IP and adding fabric logic are done at the top level using Smart Design.



Figure 1 shows the top level hardware design in SmartDesign for IGLOO2.



Figure 2 • Top level SmartDesign for SmartFusion2

SmartDesign Components

The top level SmartDesign has four components (as shown in Figure 1 and Figure 2):

- eSRAM_eNVM_access_0: System Builder generated component.
- **AHB_IF_0**: User generated RTL FSM, which performs AHB master function. This FSM interacts with the eNVM and eSRAM controller using AHB switch matrix through the FIC_0 interface.
- **eSRAM_eNVM_RW_0**:User generated RTL FSM, which takes inputs from the user and provides the required commands to AHB_IF_0 (AHB master).
- **TPSRAM_0**: Fabric IP core. Stores the data read from the eNVM or eSRAM.

System Builder Configuration for IGLOO2

1. In the **Device Features** page, ensure that **HPMS On-chip Flash Memory (eNVM)** and **On-chip SRAM (eSRAM)** check boxes are checked, as shown in Figure 3.





Figure 3 • IGLOO2 - Device Features Page



2. In the **Memories** page, add **zeros_client** to initialize the eNVM with zeros. Figure 4 shows the **Memories** page with zeros client.

m Builder - Memories) a winch a sure la						
Device Features / / M	remones / / F	emprierais Config		rnal and embedded	memories	<u>JEU / 2</u>	> Security / 2	метногу мар
ΛM		Coning	Jure your exte		memories			
Available Client types				User Clients in e	NVM			
Data Storage Serialization	Client Type	Client Name	DepthxWidth	Start Address(Hex)	Page Start	Page End	Initialization Order	Lock Start A
	1 Data Storage	Zeros_dient	32 x 32	c80	25	25	N/A	
Add to System Usage Statistics Available Pages: 2000 Used Pages: 1 Free Pages: 1999							0	
	•				Θ			4
	< Optimize			C			Edit	Delete

Figure 4 • IGLOO2 - Memories Page

3. In the **Peripherals** page, add **HPMS FIC_0-Fabric Master Subsystem** to provide the AHBL master interface to the user logic, as highlighted in Figure 5.





stem Builder - Peripherais		
Device Features > > Mem	ories > > Peripherals > > Clocks	> HPMS Options > SECDED > Security > Memory Map >
	Allocate and configure master	and slave components for your subsystems
	Fabric Slave Cores	Subsystems
Core	Version	HPMS FIC_0 - Fabric Master Subsystem
Fabric AMBA Slave 0.0.102		Configure Quantity Name
		V 1 HPMS_FIC_0_OSER_MASTER
	Fabric Mactor Coros	
Core	Version	
Fabric AMBA Master 0.0.102		
move a peripheral from one subsystem to	another, drag it from its present location and drop it onto the des	sired susbsystem.
asters are in bold and blue.		
Help Cancel		Back Next

Figure 5 • IGLOO2 - Peripherals Page

c S



Device Features	>> Memor	ries 🔷 🔪 Perij	oherals > 🔀 Clo	ks	> HPMS Options > >	SECDED	Security	>> Memo	ry Map
			Config	ure your su	bsystem clocks				
ock Fabric CCC	Chip Oscillators								
stem Clock									
50.0	MHz								
On-chip 25/50 MHz RC (Oscillator		•				DDR Bridge COI	MI/_BLK @SRAM	eNVM
IPMS Clock						_	¢		1
HPMS_CLK	=	100.00	MHz 100.000		HPMS_CCC	Ľ		AHB Bus Matrix	
MDDR Clocks					Innus_cex				
MDDR_CLK	= HPMS_CLK *	1 -							
DDR/SMC_FIC_CLK	= MDDR_CLK /	1 -							
			5		APB_0_CLK				
Fabric Interface Clocks					,—— <u>ВС_0_</u> ак				
FIC_0_CLK	= HPMS_CLK /	4	25.000						
		AHBLite Bypass Mode			IIPWS+		FIC_0		
FIC_1_CLK	= HPMS_CLK /	1 -]						
		AHBLite Bypass Mode			CCOPLL		FIC_0		
Fabric DDR Clocks					FIC_0_CLK		Cubsystem	I	
FDDR_CLK	=	100	MHz						
FDDR_SUBSYSTEM_CLK	= FDDR_CLK /	1 -	J						
					OSC				
					Fabric				
						m			Þ

Figure 6 shows the clocks configuration page for IGLOO2.

Figure 6 • IGLOO2 - Clocks Page

For more information about how to generate a complete system builder component for IGLOO2, refer to IGLOO2 System Builder User Guide.

System Builder Configuration for SmartFusion2

1. In the **Device Features** page, ensure that **MSS On-chip Flash Memory (eNVM)** check box is checked, as shown in Figure 7.







Figure 7 • SmartFusion2 - Device Features Page

2. In the Memories page, add zeros_client and dummy_client, as shown in Figure 8.







Figure 8 • SmartFusion2 - Memories Page

3. In the **Peripherals** page, add **MSS FIC_0 - Fabric Master Subsystem** to provide the AHBL master interface to the user logic, as highlighted in Figure 9.





	Select the nerinher	s and masters for each subsystem
	Fabric Slave Cores	Subsystems
Core	Version	MSS FIC_0 - MSS Master Subsystem
CoreAHBLSRAM	2.0.113	drag and drop here to add to subsystem
CoreGPIO	3.0.120	MSS FIC () - Fabric Master Subsystem
CoreI2C	7.0.102	Configure Quantity Name
CorePWM	4.1.106	1 AMBA MASTER 0
CoreSPI	3.0.156	MCC Daripharala
CoreTimer	1.1.101	Configure Enable Name
CoreUARTapb	5.2.2	
Fabric AMBA Slave	0.0.102	
	Fabric Master Cores	MSS SPL0
Core	Version	
Fabric AMBA Mast	er 0.0.102	MSS_GPIO
		MSS_USB
		MSS_MAC
		MSS_CAN
move a peripheral fra	m one subsystem to another, drag it from its present location and drop it onto the	sired subsystem.

Figure 9 • SmartFusion2 - Peripherals Page

c S



bric CCC Chip Oscillators stem Clock	Cortex.M3
Core Core Stem Clock Stem Clock So.0 WHz On-chip 25/50 MHz RC Oscillator Image: Core Cortex-M3 and MSS Main Clock Image: Core M3_CLK = MDDR_ClcK = MDDR_CLK = MDDR_CLK = MDDR_CLK = MDDR_CLK = MDDR_CLK = MDDR_CLK = MSS_CCC Image: Core MDDR_CLK = MDDR_CLK = MSS_CCC Image: Core MDDR_CLK = MSS_CCC Image: Core MDDR_CLK = MSS_CLK = MS_CLK = MS_CLK <td< th=""><th>Contex-M3 Hge Cache Controller AHB Bus Methy HB Sus Met</th></td<>	Contex-M3 Hge Cache Controller AHB Bus Methy HB Sus Met
S0.0 MHz On-chip 25/50 MHz RC Oscillator Dork-tip 25/50 MHz RC Oscillator Cortex-M3 and MSS Main Clock M3_CLK m0R_CLK = 100.00 MHz DOR Clocks MDOR_CLK = MBB_0_CLK = M3	Active Controller Active Contro
On-chip 25/50 MHz RC Oscillator DOR Bridge Carbo Co Dortex-M3 and MSS Main Clock M3_CLK = 100.00 MDDR Clocks MSS_CCC N1_C C MDDR_CLK = M3_CLK ' 1 DDR/SMC_FIC_CLK = M30_CLK / 1 MSS_CCC N1_C C N1_C C MSS Main Clocks MS MS APB_D_CLK = M3_CLK / APB_L_CLK = M3_CLK /	Alter Bas Martix
Cortex:M3 and MSS Main Clock M3_CLK = 100.00 MHz 100.000 MDDR Clocks MDDR_CLK = M3_CLK * 1 ~ DDR/SMC_FIC_CLK = MDDR_CLK / 1 ~ MSS_CCC MSS_CCC NLST NLST MDDR_CLK = M3_CLK / 1 ~ APB_0_CLK = M3_CLK / 1 100.000 APB_1_CLK = M3_CLK / 1 100.000 abric Interface Clocks US3 US3 US3	AHB Bus Matrix
John Key Milling Miss Main Clock Milling Miss Main Clock Milling Miss Main Clock M3_CLK = 100.00 MHz 100.000 MDDR_Clock MILLing K Milling K Milling K MDDR_CLK = MILLing K Milling K DDR/SMC_FLC_CLK = MDDR_CLK / 1 * DDR/SMC_FLC_CLK = MIDDR_CLK / 1 * APB_0_CLK = M3_CLK / 1 100.000 APB_1_CLK = M3_CLK / 1 100.000 abric Interface Clocks MISS MISS	ArtB Bus Metrix ArtB Sus Metrix ArtB - SPL0 SPL0 DCc.0 Data
M3_LLK = 100.00 MH2 100.000 MDDR dods MDDR_LLK = M3_LLK* 1 DDR/SMC_FIC_LLK = MDDR_LLK/ 1 APB_0_LLK = M3_LLK/ 1 APB_1_LLK = M3_LLK/ 1 M3_LK 1 M3_	
MDDR_CLK = M3_CLK * 1 * DDR_SMC_FIC_CLK = MDDR_CLK / 1 * ASS_APB_0/L Clocks APB_0_CLK = M3_CLK / 1 * 100.000 APB_1_CLK = M3_CLK / 1 * 100.000 ib/ic Interface Clocks	019 0.01 Na
mLUR_LIK = M_1_LK I DDR/SMC_FIC_CLK = MDDR_CLK / 1 ASS APB_0/L Glods APB_0_CLK = M3_CLK / 1 APB_1_CLK = M3_CLK / 1 abitic Interface Glods	- 196,0 DAK
DDR,RMC_FLC_LK = MDDR_LK / 1 * MSS APB_0/L Clocks APB_0_CLK = M3_CLK / 1 * 100.000 APB_1_CLK = M3_CLK / 1 * 100.000 abbic Interface Clocks MSS *	
APB_0/1 Clocks APB_0/2 LK = M3_CLK / 1 100.000 APB_1_CLK = M3_CLK / 1 100.000 abric Interface Clocks M3.3	
APB_0_CLK = M3_CLK / 1 100.000 APB_1_CLK = M3_CLK / 1 100.000 abric Interface Gods M3.3	100
APB_1_CLK = M3_CLK / 1 100.000	
abric Interface Clocks	
FIC_0_CLK = M3_CLK / 1 100.000	
AHBLITE Bypass Mode	FIC_0 Subsystem
FIC_1_QLK = M3_QLK / 1	a a a a a a a a a a a a a a a a a a a
AHBLite Bypass Mode	
iabric DDR Clocks	
FDDR_CLK = 100 MHz	

Figure 10 shows the clocks configuration page for SmartFusion2.

Figure 10 • SmartFusion2 - Clocks Page

For more information about how to generate a complete system builder component for SmartFusion2, refer to SmartFusion2 System Builder User Guide.

TPSRAM IP Configuration

In SmartDesign, TPSRAM IP is configured as Write Port 32 (Depth) X 32 (Width) and Read port as 32 (Depth) X 32 (Width). Figure 11 shows the TPSRAM IP configuration.





Optimized for	Composed Low Power	Initialize RAM for Simulation	Customize RAM Content
Write Port			
Depth: 32	► WD [31	u]	
	WADDR [4:0	1	
width: 32	► WEN		
🖾 Grada Chada			
СІК :=		[31:0] RD	
Dead Deat	REN 🔽	Pipeline	
		1	
Depth: 32			
Width: 32	RD_EN		
	RD_SRST_N		
	ARST_N		

Figure 11 • TPSRAM Configuration

Simulation

This section describes the design simulation.

eNVM Simulation

To perform the eNVM write and read simulation operation, provide "1" to "0" transition on the start_envm input signal.

Figure 12 shows the eNVM write command sequence. For more information about this, see "Appendix B" on page 26.



.

SmartFusion2 and IGLOO2 - Accessing eNVM and eSRAM from FPGA Fabric - Libero SoC v11.4

- Default														
		Msgs												
start envm	1			Check	eNVM busy			Request	exclusive ac	res	Chec	k access gr	anted Set FR	EQRNG parame
								/	exercisive de		form	ht	/	
			hnhn	hinn	لمممد				mm	hnh	تمطره	ىىلىت	ւիւրի/	hunu
				/			/							
		2					· · · · · ·							
	00000000	00000000	60080120	00000000	(60080120)(00000	0000	(60080 1fc)00	000000	60080 1fc (000	00000	600801fc	00000000)4003800c)0	000000
		0	12	0	2)0)2 0		2 0		<u>)2</u>	0	2)0	
WRITE	0													
SIZE	2	2												
WDATA		000000000						000001					p	0001#1
READT	00000000	0000000		Yaa	0000000 00000	2000 0 10	0000000 000	000000		-	0000000	0000000 10	0000000 00	0000000
IDECD	0	0000000		,po	,00000000 <u>(</u> 00000	0		00000		<i>p</i>	00100000	0000000 0		000000
	`													
ve - Default :		ge												
start_envm	0	ه کار کار کار												
		و المربع ال			 Fill Write Da 	ta outter								
			nnnn		nnn	uyuu		yuuuu	<u>ununu</u>	unn	nnn	uyun		LULUU
HRESETN														
HADOR	20000000	2	60090090 000000	10 16000	084 00000000	60090099	00000000	feaneticer You	2000000	60090000 000	20000	60030004 000	000000	60080008 0000000
HTRANS	0	0	12 10		2 10)2	10	22 10	~~~~	12 10	~~~~	2 10		12 10
		2												
	00001ff1	00001ff1	0000000	00	00000001		00000002))	000003	000	20004)00	000005	00000006
HREADY	1 00000000													
HRESP	0	0000000												
sue Default														
ave - Default														
ave - Default	Msgs			Contraction Contract	man d									
ave - Default	Msgs 0			rogram Com	mand		_	check for	eNVM bus	/				
ave - Default	Msgs 0 z			rogram Com	mand		- 1	check for	eNVM bus					
ave - Default	0 z 0			rogram Com	mand			check for	NVM bus					
ave - Default ===== start_ervm start_esram HCLK HRESETn	0 2 0 1			rogram Com			n dr	check for						
ave - Default	Msgs 0 z 0 1 2			rogram Com				check for				<u> </u>		
ave - Default	Msgs 0 2 0 1 2 00000000		F00000148_1000000					check for		6008012				
ave - Default	Msgs 0 2 0 1 2 00000000 0		60080148 10000000	x	mand			check for (6008011: Y2)60080120 10	
ave - Default	0 Msgs 0 z 0 1 2 0000000 0 0		(60080148)0000000 22)0	x x x x x x x x x x x x x x x x x x x	mand 60080120 (000 2 2 0			check for (6008011 12			\$60080 120 10 12 10	
ave - Default	Maga 0 z 1 2 00000000 0 0		(60080148 (000000 2 10	x x x x x x x x x x x x x x x x x x x	mand 60080120 000 12 0		~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	check for 080120 000000 2 10		6008011 22	0.00000000)60080 120 J0 2 J0	
ave - Default start_ervm start_eram + HCLK + HCEETIN + HCLK + HCLKK + HCLKKK + HCLKKK + HCLKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK	0 Maga 0 z 1 2 00000000 0 0 2 z		(60080 148 (0000000 2 10	x x x x x x x x x x x x x x x x x x x	mand 60080120 000 2 2 0			check for (080120 1000000 12 10	eNVM dust	5008011 22	0)60080120 10 2 10	
start_envm start_ervm start_eram start_eram HCBETn HESETn HADER HATRANS HATRANS HATRANS HATRANS HATRANS HATRANS HATRANS	Maga 0 2 0 1 2 00000000 0 0 0 2 0000001f	2 2 000000000 2 2 2 2 2 2 2000001f	(60080148 (6000000 22 /0 10000006	x x x x x x x x x x x x x x x x x x x	mand 60080120 000 X2 0			check for 		\$008011 22	0 0000000)60080 120 10 2 10	
Ave - Default	Mage 0 2 0 1 2 00000000 0 0 2 00000001f 1 		(60080148 (6000000 2 /0 0000000	xogram Com	600 <u>80120</u> 000			check for (680120 000000 12 10		6008017 22	0		\$60080 2)0	
ave - Default : start_envm start_envm start_envm HCLK HRESETN HADR HARDR HRITE		2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	60090148 10000000 2 10 100000000	xograni Com				check for 080120 1000000 12 10 0 1000000		600801: 	0 00000000		120 10 120 10 2 10	

Figure 12 • eNVM Write Command Sequence

Figure 13 shows the eNVM read simulation (AHB read operation).



Figure 13 • eNVM Read Simulation

Figure 14 shows the release exclusive access to eNVM.

I											
Wave - Default	Msgs										
₄ start_envm ₄ start esram	0 z			Rel	ease exclusi	ve access			-		
A HCLK	0 1			<u> </u>					_	 	<u> </u>
	2	2	1600801fc				00000000				
	0	0			2		0				
	2	2					20000000				
	0	00000000					00000000				
	0	0					20000000				

Figure 14 • Release Exclusive Access to eNVM



eSRAM Simulation

To perform the eSRAM write and read simulation operation, provide "1" to "0" transition on the start_esram input signal.

Figure 15 shows the eSRAM write simulation (AHB write operation).



Figure 16 • eSRAM Read Simulation

Setting Up the Design

The following steps describe how to setup the hardware demo for IGLOO2 Evaluation Kit board:

- 1. Connect the jumpers on the IGLOO2 Evaluation Kit board as perTable 2.
- Table 2 shows the jumper settings.

Table 2 • Jumper Setting	IS
--------------------------	----

Jumper	Pin (From)	Pin (To)	Comments
J22	1	2	default
J23	1	2	default
J24	1	2	default
J8	1	2	default
J3	1	2	default

Note: Ensure that the power supply switch SW7 is switched off while connecting the jumpers on the IGLOO2 Evaluation Kit.

- 2. Connect the power supply to the J6 connector.
- 3. Switch on the power supply switch SW7.
- 4. Connect the FlashPro4 programmer to the J5 connector of the IGLOO2 Evaluation Kit board.



The following steps describe how to setup the hardware demo for SmartFusion2 Evaluation Kit board:

- 1. Connect the jumpers on the SmartFusion2 Evaluation Kit board as perTable 3.
- Table 3 shows the jumper settings.

Table 3 • Jumper Settings

Jumper	Function									
J23	Selects switch-side Mux inputs of A or B to the line side									
	Pin 1-2 (Input A to the line side) that is on board 125 MHz differential clock oscillator output is routed to the line side									
	Pin 2-3 (Input B to the line side) that is external clock required to source through SMA connectors to the line side									
J22	2 Selects the output enables control for the line side outputs									
	Pin 1-2 (line side output enabled)									
	Pin 2-3 (line side output disabled)	Open								
J24	Provides VBUS supply to USB when using in Host mode	Open								
J8	JTAG selection jumper to select between RVI header or FP4 header for application debug	—								
	Pin 1-2 FP4 for SoftConsole/FlashPro	Closed								
	Pin 2-3 RVI for Keil ULINK™/IAR J-Link [®]	Open								
	Pin 2-4 to remotely toggle JTAG_SEL signal using GPIO capability of FT4232 chip	Open								
J3	Selects SW2 input or ENABLE_FT4232 signal from FT4232H chip	_								

Note: Ensure that the power supply switch SW7 is switched off while connecting the jumpers on the SmartFusion2 Evaluation Kit.

- 2. Connect the power supply to the J6 connector.
- 3. Switch on the power supply switch SW7.

3

4. Connect the FlashPro4 programmer to the J5 connector of the SmartFusion2 Evaluation Kit board.





Figure 17 shows the demo setup for SmartFusion2 Evaluation Kit / IGLOO2 Evaluation Kit.

Figure 17 • Accessing eNVM and eSRAM Demo Setup

Programming the Demo Design

The following steps describe how to program the demo design:

- 1. Download the demo design from the following link:
 - http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_ac429_accessing_envm_esram_fpga_libe rov11p4_an_df
- 2. Switch on the power supply switch SW7.
- 3. Launch the FlashPro software.
- 4. Click New Project (see Figure 18).
- 5. In the New Project window, enter the project name as eNVM_eSRAM_RW.
- 6. Click **Browse** and navigate to the location where the project needs to be saved.
- 7. Select Single device as the Programming mode.
- 8. Click **OK** to save the project.



File Edit View Tools Programmers Configuration Customize Help Image: I	P FlashPro										
New Project Dpen Project Wew Project Wew Project Wew Project Project <td>File Edit View Tools Programmers Configuration Customize Help</td>	File Edit View Tools Programmers Configuration Customize Help										
New Project Open Project Wew Project mark Wew Project Location: C LeWWM_eSRAM_RW Project Location: C LeWWM_eSRAM_RW Werston: Not optication Werston: List Add Leros > Warnings / Info / Ready No project located											
New Project Image: Configure Device Image: Configure Device Image: Configure Device Vew Programmers Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Configure Device Image: Con											
Image: Construction of the second	New Project										
Image: Stand RW Project Name: Project Name: Project Name: Single device Chan Device Chan Image: Single device Chan Device Chan Image: Single device Chan Device Chan Image: Single device Chan Device Chan	Open Project										
Ready No project loaded	New Project Project Name: eWM_ESRAM_RW Project Location: Clear Programming mode © Chan Heb OK OK Cance										
All / Errors / Warnings / Info / Ready	Release: v11.4										
Ready No project loaded	III All & Errors & Warnings & Info /										
	Ready No project loaded										

Figure 18 • FlashPro - New Project

Configuring the Device

The following steps describe how to configure the device:

- 1. Click Configure Device on the FlashPro GUI (see Figure 19).
- 2. Click **Browse** and navigate to the location where the **ENVM_RW_Fabric_Top.stp** programming file is located, and select the file.
 - The default location of the programming file is shown below:
 - For SmartFusion2: <download folder>\eSRAM_eNVM_RW_Fabric\Programming_file\SF2\eSR
 AM_eNVM_access_top.stp
 - For IGLOO2: <download folder>\eSRAM_eNVM_RW_Fabric\Programming_file\IGL2\eSRAM_e NVM_access_top.stp



P FlashPro - [eNVM_eSRAM_RW] *								
<u>File Edit View Iools Programmers Configuration Customize Help</u>								
New Project Image: Configure Device Image: Configure Device <td< th=""></td<>								
x Programming file								
ENVM_RW_Fabric_Top.stp Browse Modify								
DATE_MODIFIED Thu Jul 31 18:54:45 2014 STAPL_FILE_NAME E:\Libero_11.4_updates\11.4_working_backu CREATOR FlashPro Version: v11.4 DEVICE M2GL010T-fq484 DATE 2014/07/31 STAPL_VERSION JESD71 IDCODE 0F8031CF DESIGN ENVM_RW_Fabric_Top CHECKSUM 9513 SECURITY Disable ALG_VERSION 2 Chain Parameter Inspect Device								
X Created new project 'C:\eNVM_eSRAM_RW\eNVM_eSRAM_RW.pro' STAPL_file 'E:\Libero 11.4 updatea\11.4 working backup IGD2 ENVM RW Fabric\IGL2 ENVM RW Fabric\designer\FNVM R								
DESIGN : ENVM_RW_Fabric_Top; CHECKSUM : 9E13; ALG VERSION : 2								
Ready E:\Libero_11.4_updates\11.4_working_backup_IGL2_ENVM_RW_Fabric\IGL2_ENVM_RW_Fabric\designer\ENVM_RW_Fabric_Top\export\ENVM_RW_Fabric_Top.stp_SINGLE								

3. Select **Basic** as Mode and **PROGRAM** as Action (highlighted in Figure 19).

Figure 19 • FlashPro - Project Configuration





4. Click **PROGRAM** to start programming the device. Wait until the Programmer Status is changed to **RUN PASSED** (highlighted in Figure 20).



Figure 20 • FlashPro - RUN PASSED

Running the Design

eNVM Write and Read Operations

The following steps describe how to perform read and write operations from the eNVM:

- Make "1" to "0" transition using the SW5-1 DIP switch (FPGA pin number: L19) to perform write and read operation from the eNVM. An incremental data starting from 0x00000000 to 0x0000001F is written to page 25 of the eNVM, and the data is read back from the eNVM and stored in the Fabric SRAM.
- 2. The SmartDebug tool in Libero SoC verifies the write and read operations. The following steps describe how to verify the write and read operations:
- a. In Libero SoC, go to **Designflow > DebugDesign > SmartDebugDesign**. Right-click and open the **SmartDebug** window (see Figure 21).



Figure 21 shows the SmartDebug window.

SmartDebug	
Device: M2GL010T (M2GL010T)	Programmer: 73590 (usb73590) 🔻
ID code read from device: F8031CF	
View Device Status View Flash Memory Content	Debug FPGA Array Debug SERDES
Help	Close

Figure 21 • SmartDebug Window

S

- b. Click **Debug FPGA Array** (highlighted in Figure 21) and open the **Debug FPGA Array** window (see Figure 22).
- c. Browse for eNVM_eSRAM_RW_Top_debug.txt file and click Read Block (highlighted in Figure 22) in Memory Blocks tab.



🔝 Debug FPGA Array Debug File: dkup_IGL2_ENVM_RW_Fabric/IGL2_ENVM_RW_Fabric/designer/ENVM_RW_Fabric_Top/ENVM_RW_Fabric_Top_debug.txt Browse... FPGA Array Debug Data Live Probes Active Probes Memory Blocks TPSRAM_0/ENVM_RW_Fabric_Top_TPSRAM_0_TPSRAM_R0C0/INST_RAM1K18_IP • Read Block Write Block Memory Block Data Close Help Figure 22 • Debug FPGA Array Window 3

Figure 22 shows the **Debug FPGA Array** window.



Figure 23 shows the displayed fabric SRAM (TPSRAM) memory content. It shows that eNVM Write and Read is successful.

Live Prob	es	Active F	Probes	Mem	ory Bloc	ks											
TPSRAM	_0/eSR	AM_eN\	/M_acce	ess_top_	TPSRAM	1_0_TPS	RAM_R	OCO/INS	ST_RAM	1K18_IF)					•	Read Block Write Block
Memory	Block	Data															
0000	000	000	000	000	001	000	000	000	002	000	000	000	003	000	000	000	
0010	004	000	000	000	005	000	000	000	006	000	000	000	007	000	000	000	
0020	008	000	000	000	009	000	000	000	00A	000	000	000	00B	000	000	000	
0030	00C	000	000	000	00D	000	000	000	00E	000	000	000	00F	000	000	000	
0040	010	000	000	000	011	000	000	000	012	000	000	000	013	000	000	000	
0050	014	000	000	000	015	000	000	000	016	000	000	000	017	000	000	000	
0060	018	000	000	000	019	000	000	000	01A	000	000	000	01B	000	000	000	
		000	000	000	01D	000	000	000	01F	000	000	000	01F	000	000	000	

Figure 23 • Debug FPGA Array - Memory Blocks

- d. Close the Debug FPGA Array window.
- e. Close the SmartDebug window.

For more information about how to interpret the Memory Block Data, go to Help > Help Topics > Debug Design > SmartFusion2 and IGLOO2 SmartDebug > Debug FPGA Array - Memory Blocks Tab.

eSRAM Write and Read Operation

The following steps describe how to perform read and write operations from the eSRAM:

- Make "1" to "0" transition using the SW5-2 DIP switch (FPGA pin number: L18) to write and read from the eSRAM.An incremental data pattern starting from 0xA1B2C300 to 0xA1B2C31F is written to 32 locations in eSRAM address starting from 0x20000000 to 0x20000080, and the data is read back from that eSRAM and stored in the Fabric SRAM.
- 2. To verify the write and read operations using the SmartDebug tool in Libero SoC, follow the steps (Step a to Step e) from the eNVM write and read operation.



Figure 24 shows the displayed fabric SRAM (TPSRAM) memory content. It shows that eSRAM write and read operations are successful.



Figure 24 • Debug FPGA Array - Memory Blocks

For more information about how to interpret the Memory Block Data, go to Help > Help Topics > Debug Design > SmartFusion2 and IGLOO2 SmartDebug > Debug FPGA Array - Memory Blocks Tab.

Conclusion

This application note describes how to write and read to the eNVM and eSRAM from FPGA fabric. This application note also describes the usage of SmartDebug tool to verify the design functionality.



Appendix A – Design Files

Download the design files from the Microsemi SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_ac429_accessing_envm_esram_fpga_liberov11p 4_an_df

The design file consists of Libero Verilog projects and programming files (*.stp) for SmartFusion2 Evaluation Kit and IGLOO2 FPGA Evaluation Kit. Refer to the Readme.txt file included in the design file for the directory structure and description.



Appendix B

eNVM Write Operation

The following steps describe how to perform the eNVM write operation:

- 1. Wait for Bit 0 of status register (address: 0x60080120) to become 1. If this bit is 0, it implies that the eNVM is busy.
- Request exclusive access to the eNVM. This is required to ensure that no two Masters can write to the eNVM at the same time. This is done by writing 0x1 to the REQACCESS register (address: 0x600801FC).
- 3. Check if request has been granted, by reading back from the REQACCESS register. On read back, check for Bit 2 (counting up from 0):
 - if Bit 2 is 1, it implies that the request is successful.
 - If Bit 2 is 0, the request for exclusive access is denied. The eNVM cannot be written at this time
- 4. Write 0x000001FF1 to ENVM_CR register (address: 0x4003800C). This changes the FREQRNG register field to 15 decimal.
- 5. Writes to the eNVM are buffered. First, write data into the write data buffer (WDB It is a byte addressable 1024-bit buffer. Its base address is 0x60080080 for eNVM_0 and 0x600C0080 for eNVM_1), and then use a single command to commit (aka program) data into one page of the eNVM. Write the data into the WDB.
- 6. Compute the values of bits that needs to be written into the eNVM command register:
 - Bits 31-20 should be 0x080.
 - Bit 19 should be 0x0.
 - Bits 18-7 corresponds to the number of page to be written (for 25th page, the bit field is 000 0000 1100 1).
 - Bits 6-0 should be 0x0.
- 7. Write the eNVM command register (address: 0x60080148) with the data computed in Step 6. eNVM does not respond to further commands until the write is complete.
- 8. Release exclusive access to the eNVM by writing 0x0 to the REQACCESS register (address: 0x600801FC).

eNVM Read, eSRAM Write and Read Operations

No special command sequences are required for the eNVM read, eSRAM read, and eSRAM write operations. eNVM read and eSRAM read are performed using AHB read operation, and eSRAM writes are performed using AHB write operations.



List of Changes

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

Revision*	Changes	Page
Revision 1 (September 2014)	Initial release	NA

Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.



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**.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.