

Nios II Software Build Tool を使用した マルチコア・システムの実装

ver.14



2015年5月 Rev.1

ELSENA,Inc.





Nios II Software Build Tool を使用した マルチコア・システムの実装

<u>目次</u>

1.	はじめに	3
2.	適用条件	3
3.	システムの構成	3
3	-1. 検証デザインの概略	3
3	-2. ブロック・ダイアグラム	4
3	-3. Qsys ブロック	4
	3-3-1. Nios II プロセッサの設定	5
	3-3-2. BUTTON PIO の設定	7
	3-3-3. LED PIO の設定	8
	3-3-4. Interval Timer の設定	8
	3-3-5. Dual-port Onchip RAM の設定	9
	3-3-6. Vectored Interrupt Controller の設定	9
	3-3-7. Flash Memory Interface (CFI)の設定	. 10
	3-3-8. SSRAM の設定	. 10
3	-4. システム・メモリ・マップ	11
4.	ソフトウェアの構成	12
4	-1. 検証ソフトウェアの概略	. 12
4	-2. ソフトウェア・プロジェクトの生成	. 14
4	-3. ソフトウェアの実行	. 20
4	-4. ハードウェア・イメージとソフトウェアの Flash ROM への書き込み	. 24
5.	複数の Nios II を構成する際の注意事項	28
5	-1. 排他制御に関して	. 29
5	-2. 自動で生成される初期化コードがサポートされる構成	. 30
5	-3. まとめ	. 32
改覑	反履歴	33

1. <u>はじめに</u>

この資料は、複数の Nios[®] II を使用したマルチコア・システムにおいて、メモリやペリフェラルを共有して使用する 方法や注意点、マルチコア・システムならではのデバッグの方法、Flash メモリへの書き込み方や注意点などを、検 証を元に解説しています。

2. <u>適用条件</u>

本資料では、Common Flash Interface (CFI) Flash ROM を使用したファスト・パッシブ・パラレル・コンフィグレーション(FPP)を使用し、ハードウェア・イメージと各ソフトウェア・イメージを1つの Flash ROM に格納、起動させ、ソフトウェアの実行は外部の SSRAM を共有で使用して動作させます。

- 対応バージョン
- Quartus[®] II 開発ソフトウェア v14.1
- Nios II Software Build Tools (Nios II SBT) v14.1
- 検証ハードウェア
- Cyclone V E FPGA Development Board
 - · FPGA : Cyclone[®] V 5CEFA7F31I7ES
 - · Flash Memory : 64Mbyte/16bit (CFI)
 - · SSRAM : 2Mbyte/18bit
 - ※ Flash Memory と SSRAM は共有バスを使用
 - ※ SSRAMは、バス幅16ビットとして使用

3. <u>システムの構成</u>

3-1. 検証デザインの概略

本検証では、3 つの Nios II コアを使ったマルチコア・システムを使用します。それぞれの Nios II には 1 つの BUTTON からの入力と 1 つの LED への出力、インターバル・タイマを持ちます。また、全ての Nios II に共有のペリ フェラルとして 1 つの BUTTON と LED を持ちます。

検証のためのソフトウェアでは、それぞれの Nios II はタイマからの割り込みによって LED を等間隔に点滅させ ます。そして BUTTON の押下で、その点滅間隔を徐々に延ばす処理を行い、全ての Nios II が確実に動作してい ることを確認します。また、共有 BUTTON の押下では全ての LED の点滅間隔を初期状態に戻す処理を行い、全 ての Nios II が共有のペリフェラルからの入力も正しく処理できることを確認します。

その他に、2つの Nios II の間に共有メモリを実装し、プロセッサ間通信が行えることも確認しています。



3-2. ブロック・ダイアグラム

本検証では、Qsys ブロックを図 3-2-1 に示す構成としています。なお、図の中では割り込みコントローラやデバッグ 関連のモジュールについては省略しています。



⊠ 3-2-1

3-3. Qsys ブロック

図 3-3-1 に Qsys の接続を示します。

Use	Connections	Name	Description	Ex	Clock	Base	End	IRQ
		😐 pli	Altera PLL		clk			
v		🗆 🖳 nios2_cpu_0	Nios II Gen2 Processor		[clk]			
L .	<	data_master	Avaion Memory Mapped Master	Dop	sys_clk			
L .		instruction_master	Avaion Memory Mapped Master	Dau	[clk]			
L .		interrupt_controller_in	Avalon Streaming Sink	Dau	[clk]			
L .		debug_mem_slave	Avaion Memory Mapped Slave	Dau	[clk]	● 0×0600_0000	0×0600_07ff	
L .		custom_instruction_master	Custom Instruction Master	Dau				
	$ \bullet \circ \circ \circ \circ \rightarrow$	⊞ jtag_uart_0	JTAG UART		sys_clk	● 0×0600_0800	0×0600_0807	
•	$ \bullet \bullet$	⊞ button_pio_0	PIO (Parallel I/O)		sys_clk		0x0600_090f	
•	$ \bullet \bullet$	🗈 led_pio_0	PIO (Parallel I/O)		sys_clk	● 0×0600_0a00	0x0600_0a0f	
•	$ \bullet \bullet$	🗉 timer_0	Interval Timer		sys_clk	● 0×0600_0500	0x0600_0b1f	
•		🗆 🖳 nios2_cpu_1	Nios II Gen2 Processor		[clk]			
		data_master	Avalon Memory Mapped Master	Dop	sys_clk			
		instruction_master	Avaion Memory Mapped Master	Dau	[clk]			
		interrupt_controller_in	Avalon Streaming Sink	Dapi	[clk]			
		debug_mem_slave	Avaion Memory Mapped Slave	Dapi	[clk]	● 0×0700_0000	0×0700_07ff	
	×	custom_instruction_master	Custom Instruction Master	Dau				
•		⊞ jtag_uart_1	JTAG UART		sys_clk	0×0700_0800	0×0700_0807	
•		⊞ button_pio_1	PIO (Parallel I/O)		sys_clk	● 0×0700_0900	0×0700_090f	► <u></u>
•		⊞ led_pio_1	PIO (Parallel I/O)		sys_clk		0x0700_0a0f	
•		⊞ timer_1	Interval Timer		sys_clk	● 0×0700_0b00	0x0700_0b1f	⊨ b
•		🗉 🖳 nios2_cpu_2	Nios II Gen2 Processor		[clk]			
		data_master	Avalon Memory Mapped Master	Dop	sys_clk			
		instruction_master	Avalon Memory Mapped Master	Dop	[clk]			
		interrupt_controller_in	Avalon Streaming Sink	Dop	[clk]			
		debug_mem_slave	Avalon Memory Mapped Slave	Dop	[clk]	● 0×0800_0000	0×0800_07ff	
	×	custom_instruction_master	Custom Instruction Master	Dop				
	$ \diamond - \diamond - \diamond - \diamond - \diamond - \diamond - \rightarrow$	⊞ jtag_uart_2	JTAG UART		sys_clk	● 0×0800_0800	0×0800_0807	>++3
		button_pio_2	PIO (Parallel I/O)		sys_clk		0×0800_090f	->++(i
		⊡ led_pio_2	PIO (Parallel I/O)		sys_clk		0x0800_0a0f	
		⊞ timer_2	Interval Timer		sys_clk		0x0800_0b1f	<u>}++</u> ₫
		button_pio_3	PIO (Parallel I/O)		sys_clk		0x0510_000f	2-2-2
		■ led_pio_3	PIO (Parallel I/O)		sys_clk	● 0x0510_0100	0x0510_010f	
	│∲╶∲╶∲ ∲──∲─────────────────────────────	🗉 🖳 ext_flash	Generic Tri-State Controller		sys_clk	● 0x0000_0000	0x03ff_fff	
		🗉 🖳 ext_ssram	Generic Tri-State Controller		sys_clk	● 0×0400_0000	0x041f_ffff	
v		🗉 🖳 tristate_conduit_pin_sharer_0	Tri-State Conduit Pin Sharer		sys_clk			
v		tristate_conduit_bridge	Tri-State Conduit Bridge		sys_clk			
v	$ \bullet \bullet$	⊡ com_mem_0_1	On-Chip Memory (RAM or ROM)		multiple	≜ muitipie	muitipie	
		⊡ com_mem_1_2	On-Chip Memory (RAM or ROM)		multiple	≜ multiple	muitiple	
		⊡ com_mem_2_0	On-Chip Memory (RAM or ROM)		multiple	≜ multiple	muitiple	
	$ \diamond \diamond \diamond \diamond \diamond \rightarrow$	🖽 啦 vic_0	Vectored Interrupt Controller		sys_clk		0x0600_0fff	\sim
	$ \diamond \diamond \bullet \diamond \bullet \diamond \rightarrow$	🗉 🖳 vic_1	Vectored Interrupt Controller		sys_clk		0x0700_0fff	\leftarrow
	$ \diamond - \diamond $	⊞ Щ vic_2	Vectored Interrupt Controller		sys_clk	● 0×0800_0c00	0x0800_0fff	<i>←</i>



3-3-1. Nios II プロセッサの設定

下記に Nios II の設定を示します。基本的に3つの Nios II は同一の設定としますが、表 3-3-1-1 のとおり、Reset Vector Offset および Exception Vector Offset が異なることに注意してください。



ً 3-3-1-1



🗵 3-3-1-2

🔨 Parameters 🕺		- 5 🗆
System: nios2_system Path: nios2_cpu_0		
Nios II Gen2 Processor altera_nios2_gen2		
Main Vectors Caches and Memory Interfaces	Arithmetic Instructions MMU and MPU Settings JTAG	Debug Advanced Features
Instruction Cache		
Size: Add burstcount signal to instruction_master:	4 Kbytes 💌 🗲	Instruction キャッシュ サイズ:4kbyte
Flash Accelerator		
Line Size:	None	
Number of Cache Lines:	2 -	
T Data Cache		
Size: Victim buffer implementation: Add hurstcount signal to data master:	2 Kbytes	Data キャッシュサイズ :2kbyte
☑ Use most-significant address bit in process	ror to bypass data cache	
Tightly-coupled Memories		
Number of tightly coupled instruction master po	rts: None 💌	
Number of tightly coupled data master ports:	None 💌	
Peripheral Region		
Size:	None	
Base Address:	0×00000000	

⊠ 3-3-1-3

堕 Parameters 🛞 🗕 🗖 🗖
System: nios2_system Path: nios2_cpu_0
Nios II Gen2 Processor altera_nios2_gen2
Main Vectors Caches and Memory Interfaces Arithmetic Instructions MMU and MPU Settings JTAG Debug Advanced Features
General
In ECC Present Interrupt controller: External I Controller: External I Controller: M部割込みコントローラ Number of shadow register sets (0-63): 7 を選択
Include cpu_resetrequest and cpu_resettaken signals
These signals appear on the top-level Qsys system. You must manually connect these signals to logic external to the Qsys system
CPUID control register value: 0x0000000
Assign unique values for CPUID if system has multiple Nios II cores sharing code
Generate trace file during RTL simulation
Exception Checking
Misaligned memory access
Branch Prediction
Branch prediction type: Dynamic 💌
Number of entries (2-bits wide): 256 Entries
TRAM Memory Protection
✓ Include reset_req signal for OCI RAM and Multi-Cycle Custom Instructions

⊠ 3-3-1-4

表 3-3-1-1

	Reset Vector Offset	ROM Size	Exception Vector Offset	RAM Size
Nios II #0	0x400000	384Kbyte	0x00020	384Kbyte
Nios II #1	0x460000	384Kbyte	0x60020	384Kbyte
Nios II #2	0x4C0000	384Kbyte	0xC0020	384Kbyte

3-3-2. BUTTON PIO の設定

下記に BUTTON PIO の設定を示します。BUTTON0~BUTTON3 の設定は同一とします。

🙍 Parameters 🛛					
System:nios2_system P	Path: button_pio_1				
PIO (Parallel I/O)					
altera_avalon_pio		<u>D</u> etails			
Basic Settings					
Width (1-32 bits):	1 4			バス唱:1bit	
Direction:	O Bidir				
	💿 Input 🔫			方向:入力のみ	
	C InOut				
	C Output				
Output Port Reset Value	e: 0×000000000000000				
Output Register					
Enable individual bit	t setting/clearing				
TEdge capture regist	ter		_ •		A 1 / #11 50
Synchronously captu	ure		- エッ ジ	ン・キャフチャ・レジス	、タ:立ち上かりエッジ
Edge Type:	RISING				
Enable bit-clearing t	for edge capture register				
Tinterrupt					
🔽 Generate IRQ			i	割り込み:エッジ	
IRQ Type:	EDGE 🔽 🗲				
Level: Interrupt CPU wh Edge: Interrupt CPU wh register is logic true. Av	hen any unmasked I∕O pin is logic true nen any unmasked bit in the edge-capture ∕ailable when synchronous capture is enabled				
Test bench wiring				テストベンチへの初	朝値
🔽 Hardwire PIO inputs	s in test bench			:0x0	
Drive inputs to:	0×00000000000000				
-		-			

⊠ 3-3-2



3-3-3. LED PIO の設定

	h
Parameters 🗧 🗕 🗖 🗖	<u>.</u>
iystem: nios2_system Path: led_pio_1	
PIO (Parallel I/O)	
▼ Basic Settings	
Width (1-32 bits): 1	パス幅:1bit
Direction: O Bidir	
O Input	方向:出力のみ
O InOut	
Output	
Output Port Reset Value: 0x0000000000000	Reset 時の初期値
Output Register	
Enable individual bit setting/clearing	
* Edge capture register	
Synchronously capture	
Edge Type:	
Enable bit-clearing for edge capture register	
▼ Interrupt	
Generate IRQ	
IRQ Type:	
level: Interrupt CPU when any unmasked I/O pin is logic true	
Edge: Interrupt CPU when any unmasked bit in the edge-capture	
register is logic true. Available when synchronous capture is enabled	
Test bench wiring	
Hardwire HU inputs in test bench	
Drive inputs to.	

下記に LED PIO の設定を示します。LED0~LED3 の設定は同一とします。

⊠ 3-3-3

3-3-4. Interval Timer の設定

下記に Interval Timer の設定を示します。全ての Interval Timer の設定は同一とします。

👱 Parameters 🙁 _ 🗂 🗖	5
System:nios2_system Path:timer_1	
Interval Timer	· · · · · · · · · · · · · · · · · · ·
	タイムアウト周期:10ms
Timeout period	
Period: 10 Units: ms T	カウンタ・サイズ : 32bit
Timer counter size]
Registers No Start/Stop control bits	ハードウェア・オブション
Fixed period	:ノロセック前仰可能な可愛向我ワイマ
Readable snapshot	
Output signals	
🗌 System reset on timeout (Watchdog)	
Timeout pulse (1 clock wide)	



3-3-5. Dual-port Onchip RAM の設定

下記に On-Chip Memory の設定を示します。全ての On-Chip Memory の設定は同一とします。

🙋 Parameters 🛛	I		1
System: nics2_system Path: com_mem_0_	1		
On-Chip Memory (RAM or RO	M)		
altera_avalon_onchip_memory2		<u>D</u> etails	
Type:	DAM (W.2-61-)		- Dual-port access にチェック
Dual-port access	True mename) - 23 -		
Single clock operation			
Read During Write Mode:	DONT_CARE		
Block type:	AUTO		
			メモリ・サイス: 4KDyte ※本検証では 4byte しか使用しませんので、 4Kbyte でなくても違いません。
▼ Size			
Data width:	32 💌		
Total memory size:	4096 bytes		
Minimize memory block usage (may in	npact fmax)		
Read latency			
Slave s1 Latency:			
Slave s2 Latency	1		
ROM/RAM Memory Protection			
Reset Request:	Enabled 💌		
ECC Parameter			
Extend the data width to support ECC bits	S: Disabled		
Memory initialization			
I Initialize memory content			
Enable non-default initialization file			
Type the filename (e.g. my_ram.he)	;) or select the hex file using the file browser button.		
User created initialization file:	onchip_mem.hex		
Enable In-System Memory Content Ec	litor feature		
Instance ID:	NONE		
Memory will be initialized from nic	s2_system_com_mem_0_1.hex		

🗵 3-3-5

3-3-6. Vectored Interrupt Controller の設定

下記に Vectored Interrupt Controller の設定を示します。全ての Vectored Interrupt Controller の設定は同一としま





3-3-7. Flash Memory Interface (CFI) の設定

Flash Memory Interface の設定については、付属のデザインを確認ください。また、詳細については、担当する代理店の技術情報サイトにおいて、以下の資料をご参照ください。

「Qsys におけるオフチップ・メモリ・インタフェースの接続方法」

3-3-8. SSRAM の設定

SSRAM Interface の設定については、付属のデザインを確認ください。また、詳細については、担当する代理店の 技術情報サイトにおいて、以下の資料をご参照ください。

「Qsys におけるオフチップ・メモリ・インタフェースの接続方法」



3-4. システム・メモリ・マップ



4. <u>ソフトウェアの構成</u>

4-1. 検証ソフトウェアの概略

本検証のソフトウェアは、main()関数の他にインターバル・タイマ割り込み関数、専用 BUTTON 割り込み関数、 共有 BUTTON 割り込み関数の 4 つ関数で構成されています。main()関数では各ペリフェラルの設定と割り込み関 数の登録を行っています。また、共有 On-chip RAM を使ったプロセッサ間でのデータの受け渡しも行っています。 専用 BUTTON 割り込み関数では LED の点滅間隔を決定する変数のインクリメント処理をインターバル・タイマ割 り込み関数では点滅間隔を決定する変数に従い LED の点滅処理を、共有 BUTTON 割り込み関数では LED の 点滅間隔を決定する変数を初期化しています。



<< main 関数 >>>

void button_isr(void* context) ł // LED点滅間隔の変数をインクリメント nTime ++: if(nTime > 10)nTime = 1;// エッジ・キャプチャ・レジスタのリセット IOWR_ALTERA_AVALON_PIO_EDGE_CAP(BUTTON_x_BASE, 0x00);

<< 専用 BUTTON 割り込み関数 >>

void {	l timer_isr(void* context)
ι.	nCount ++;
	if(nCount > (nTime * 10))
	{ // LEDを反転 nLed ^= 0x01; // LED出力 IOWR_ALTERA_AVALON_PIO_DATA(LED_x_BASE, nLed);
,	nCount = 0; } // タイマ割り込みのリセット IOWR_ALTERA_AVALON_TIMER_STATUS(TIMER_x_BASE, 0x00); return;
}	<pre>// LEDを反転 nLed ^= 0x01; // LED出力 IOWR_ALTERA_AVALON_PIO_DATA(LED_x_BASE, nLed); nCount = 0; } // タイマ割り込みのリセット IOWR_ALTERA_AVALON_TIMER_STATUS(TIMER_x_BASE, 0x00 return;</pre>

<< インターバル・タイマ割り込み関数 >>



<< 共有 BUTTON 割り込み関数 >>

※ 1. ソースコード中の com_mem_xxx および com_mem_yyy、IRQ や BASE で定義されている x は、表
 4-1 のとおり、Nios II ごとに変更する必要があります。

	表 4-1		
	com_mem_xxx	com_mem_yyy	x
Nios II #0	com_mem_2_0	com_mem_0_1	0
Nios II #1	com_mem_0_1	com_mem_1_2	1
Nios II #2	com_mem_1_2	com_mem_2_0	2

 ※ 2. 共有メモリが正しくインクリメント処理されていることを確認できるように、共有メモリの変数を LED に出 カする処理を実装します。いずれかの Nios II の main()関数のループに下記のコードを追加してください。



<< プロセッサ間通信動作状況出力処理 >>

※ 3. ソフトウェア処理の詳細につきましては、別途、各ペリフェラルの資料などを参照ください。



4-2. ソフトウェア・プロジェクトの生成

 Nios II SBT を起動して Workspace フォルダを Quartus II プロジェクトのフォルダの直下に作成します。そして、 Project Explorer の上にマウスを移動し右クリックし図 4-2-1 のように New ⇒ Nios II Application and BSP from Template を選択します。

🖨 Nios II - Eclipse			
File Edit Navigate Search Project Nios	II Run Window Help		
i 🖬 • 🖩 🖻 è i 📽 • थि • 📽 • 🕑 •	🎋 • 🔘 • 🍋 • 🗁 🛷 • 🖢 - 🔁 • 😓 •		cess 📑 🖬 🚺 Nios II
Project Explorer XX Project Explorer XX New Magnetic Science Project Explorer XX Project Explorer Project Explorer XX Project	S Project Nios II Application Nios II Application and BSP from Template Nios II Lobard Support Package Nios II Library Other Ctrl+N		E Outline
	Problems Ø Tasks	25	ਲ਼ੑੑਗ਼ੑੑੑੑ੶ੑੑੑੑਖ਼ੑੑੑ੶ੑੑ
0 items selected			

凶 4-2-1

表示された Nios II Application and BSP from Template ウィンドウに図 4-2-2 のように設定し Finish をクリックします。



⊠ 4-2-2

③ Project Explorer のプロジェクトを選択した状態で右クリックし、図 4-2-3-1 のように New ⇒ File を選択します。 New File ウィンドウが表示されたら図 4-2-3-2 のようにファイル名を入力して Finish をクリックします。



🗵 4-2-3-1

☑ 4-2-3-2

④ 作成したファイルが開くのでソースコードを入力します。前述のソースコードの他にも適時インクルード・ファイル やプロトタイプ宣言なども追加してください。



🗵 4-2-4

- ⑤ Project Explorer の BSP プロジェクト(nios_0_bsp の方)を選択した状態で右クリックし、Nios II ⇒ BSP Editor... を選択します。
- ⑥ Nios II BSP Editor ウィンドウが表示されたら図 4-2-5 のように Main タブの sys_clk_timer: を none に設定し ます。

🚠 Nios II BSP Editor - settings.bsp			
File Edit Tools Help			
Main Software Packages Drivers Linker Script Enable File	Generation Target BSP Directory		
SOPC Information file:\\nios2_system.sopcinfo			
CPU name: nios2_cpu_0			
Operating system: Altera HAL	Version: default		
BSP target directory: C:\Lab\nios2_multi_core_sample\soft	ware\nios_0_bsp		
E-Settings	hal		<u> </u>
⊟ <u>Lommon</u> ⊡hal	sys_clk_timer:	none	
sys_clk_timer	timestamp_timer:	none 💐	
	stdin:		
stdout			
	stdout:	jtag_uart_0 💌 🔨	大调 机
enable_gprof	stderr:	jtag_uart_0 🔻	none ぞ迭代
	🗖 anabla small s library	,	
⊟-linker	enable_smail_c_library		
enable_exception_stack	l enable_gprof		
exception_stack_size	<pre>enable_reduced_device_drivers</pre>		
enable interrupt stack	🗖 enable_sim_optimize		
interrupt_stack_size	hal.linker		
interrupt_stack_memory_region_na	enable_exception_stack		-
	٠. بر		
Information Problems Processing			
Set setting property: altera_vic_driver.VIC_0.irq3_rnmi iden	tifier to altera_vic_driver_VIC_0_IRQ3_RNMI		



- ⑦ Linker Script タブを開いて図 4-2-6 のようにセクションとリージョンを確認します。Linker Section Name では は entry を除く全てのセクション(.bss,heap,text 等)が ext_ssram に設定されていることを確認し、Linker Region Name では ext_ssram がソフトウェア実行領域に、ext_flash がソフトウェア・イメージ領域(ヘッダ部除く) に設定されていることを確認します。※1
 - ※ 1. 複数の Nios II を持つシステムでも、BSP はあらかじめリージョンが重ならないようなメモリ・スケジューリングを行いますので、本検証では変更の必要はありません。

🚠 Nios II BSP Editor – settings.bsp					_ 🗆 >					
File Edit Tools Help										
Main Software Packages Drivers Linker	Script Enable File Generation Target	BSP Directory			ext_ssram					
Linker Section Mappings										
Linker Section Name	Linker Re	gion Name	Memory Devic	e Name	Add					
bss	ext ssram	ext s	sram		Remove					
entry	reset	ext f	lash		Deskeye Defeudke					
.exceptions	ext ssram	ext s	sram		Restore Deradits					
.heap	ext_ssram	ext_s	sram							
.rodata	ext_ssram	ext_s	sram							
.rwdata	ext_ssram	ext_s	sram							
.stack	ext_ssram	éxt_s	sram							
.text	ext ssram	ext s	sram							

			1	975	フェノ美行領域					
Linker Memory Regions			<u>/</u>		Linker Memory Regions					
Linker Region Name	Address Range 🛛 🗸	Memory Device Name	Size (bytes)	Offset (bytes)	Add					
Linker Region Name	Address Range ∇ 0×09002000 - 0×09002FFF	Memory Device Name	Size (bytes) 4096	Offset (bytes) 0	Add Remove					
Linker Region Name com_mem_2_0 com_mem_0_1	Address Range 0x09002000 - 0x09002FFF 0x09000000 - 0x09000FFF	Memory Device Name	Size (bytes) 4096 4096	Offset (bytes) 0 0	Add Remove Restore Defaults					
Linker Region Name com_mem_2_0 com_mem_0_1 ext_ssram	Address Range 0x09002000 - 0x09002FFF 0x09000000 - 0x09000FFF 0x04000020 - 0x0406001F	Memory Device Name com_mem_2_0 com_mem_0_1 ext_ssram	Size (bytes) 4096 4096 393216	Offset (bytes) 0 0 32	Add Remove Restore Defaults					
Linker Region Name com_mem_2_0 com_mem_0_1 ext_ssram_BEEORE_EXCEPTION	Address Range ∇ 0x09002000 - 0x09002FFF 0x09000000 - 0x09000FFF 0x04000020 - 0x0406001F 0x04000000 - 0x0406001F	Memory Device Name com_mem_2_0 com_mem_0_1 ext_ssram ext_ssram	Size (bytes) 4096 4096 393216 32	Offset (bytes) 0 0 32	Add Remove Restore Defaults					
Linker Region Name com_mem_2_0 com_mem_0_1 ext_ssram ext_ssram ext_ssram_BEFORE_EXCEPTION ext_flash	Address Range ♥ 0x09002000 - 0x09002FFF 0x09000000 - 0x09000FFF 0x04000020 - 0x0406001F 0x04000020 - 0x0406001F 0x04000020 - 0x0405FFFF	Memory Device Name com_mem_2_0 com_mem_0_1 ext_ssram ext_ssram ext_flash	Size (bytes) 4096 4096 393216 32 32 333184	Offset (bytes) 0 32 32 4194338	Add Remove Restore Defaults Add Memory Device					
Linker Region Name com_mem_2_0 com_mem_0_1 ext_ssram ext_sram_BEFORE_EXCEPTION ext_flash reset	Address Range ♥ 0x08002000 0x08002FFF 0x08000000 0x08000FFF 0x04000020 0x0406001F 0x004000020 0x0406001F 0x00400020 0x040500FFF 0x00400020 0x040500FFFF 0x00400020 0x040500FFFF 0x00400020 0x040500FFFF	Memory Device Name com_mem_2_0 ext_ssram ext_ssram ext_flash ext_flash	Size (bytes) 4096 4096 393216 393184 393184 32	Offset (bytes) 0 32 4194336 4194336 4194304	Add Remove Restore Defaults Add Memory Device Remove Memory Device					
Linker Region Name com_mem_2_0 com_mem_2_1 ext_ssram ext_ssram ext_ssram_ ext_flash reset ext_flash BEFORE_RESET	Address Range ♥ 0x08002000 0x08002FFF 0x08000000 0x08000FFF 0x04000000 0x0408001F 0x04000000 0x0408001F 0x00400000 0x0040001F 0x00400000 0x00405FFFF 0x00400000 0x00405FFFF 0x00400000 0x0040000 0x00400000 0x0040000	Memory Device Name com_mem_2.0 ext_ssram ext_flash ext_flash ext_flash	Size (bytes) 4096 398216 38218 383184 383184 32 4194304	Offset (bytes) 0 0 32 4194330 4194334 4194304 0	Add Remove Restore Defaults Add Memory Device Remove Memory Device Memory Usage					
Linker Region Name com_mem_2_0 com_mem_0_1 ext_stram ext_stram ext_stram ext_stram ext_flash reset ext_flash BEFORE RESET	Address Range ♥ 0x09002000 0x090020FF 0x0900000 0x09000FFF 0x04000000 0x040000FF 0x04000000 0x040000FF 0x04000000 0x040000FF 0x00400000 0x040000FF 0x00400000 0x040000FF 0x00400000 0x004000FF 0x00400000 0x004000FF 0x00400000 0x004000FF	Memory Device Name com_mem_2_0 com_mem_0_1 ext_sram ext_sram ext_flash ext_flash ext_flash	Size (bytes) 4096 4096 39216 32 32 4194304	Offset (bytes) 0 32 4194335 4194304 0 0	Add Remove Restore Defaults Add Memory Device Remove Memory Device Memory Usage					
Linker Region Name com_mem_2_0 com_mem_0_1 ext_sram bxt_sram	Address Range ♥ 0x09002000 0x090020FF 0x0900000 0x09000FFF 0x04000020 0x04000FFF 0x04000020 0x040000FF 0x04000020 0x040000FF 0x04000000 0x040000FF 0x00400000 0x040000FF 0x00400000 0x00405FFFF 0x00400000 0x00405FFFF 0x00000000 0x003FFFFF	Memory Device Name com_mem_2_0 com_mem_0_1 ext_sram ext_sram ext_flash ext_flash ext_flash ext_flash	Size (bytes) 4096 4096 393216 32 333184 32 4194304 ソフトウェア・イ	Offset (bytes) 0 32 4194335 41943404 0 メージ格納領域	Add Remove Restore Defaults Add Memory Device Remove Memory Device Memory Usage Memory Map					

⊠ 4-2-6

⑧ 続いて Drivers タブを開きます。本検証のハードウェア構成では割り込み制御をベクタ割込みコントローラ (Vectored Interrupt Controller)で実装していますので、多重割り込みの許可やシャドウ・レジスタの設定、ノンマ スカブル割り込みの割り当てなどが行えます。本検証では変更の必要はありません。

🚠 Nios II BSP Editor – settings.bsp						_	
File Edit Tools Help							
Main Software Packages Drivers Linker Script Enable File	Generation Ta	arget BSP Directory					
Module Name 🛆 Module Class Name		Module Version	Driver Name	Drive	er Version	Enable	
putton_pio_0 altera_avalon_pio		14.1	altera_avalon_pio_driver	de	efault	v	
putton_pio_3 altera_avalon_pio		14.1	altera_avalon_pio_driver	de	efault		
com_mem_0_1 altera_avalon_onchip_memory2	14.1	none		none			
com_mem_2_0 altera_avalon_onchip_memory2		14.1	none		none		-
jaxt_irash jartera_generic_tristate_contr	oner	14.1	artera_avaron_crit_itasin_oriver	j de	arauri j	I v	-
🖻 Settings 📃	irq2_	rrs:	5				
E-Advanced	ira3	ril	4				
⊡ratera_avaion_tag_uart_driver	1143_		14				
enable_smail_uniter	. ⊑ir	a3 mmi					
□ altera vic driver	ira3	rre:	4				
enable_preemption	1 190_		14				
-enable_preemption_into_new_registe	vec_s	size:	16				
-enable_preemption_rs_1	<u> </u>		,				
enable_preemption_rs_2	altera_vic_	driver					
enable_preemption_rs_3		nable_preemption					
enable preemption rs 5	—	□ □ enable presemption into new register set					
enable_preemption_rs_6		nable_preemption_into_	ilew_legister_set				
-enable_preemption_rs_7	C e	nable_preemption_rs_1					
-linker_section	□ □ e	nable_preemption_rs_2					
E-VIC_0		noble preemption re 2					
ing romi	1.6						
ira0 rrs	_ □ e	enable_preemption_rs_4					
irq1_ril	, 🗆 e	nable_preemption_rs_5					
irg1_mm - Problems 97	_ e	nable_preemption_rs_6					
-irq2_ril	Гe	nable preemption rs 7					
-irq2_rnmi -	linker	r section:	text				
			LIGAL				-
Toformation In 11 In 1						Exit =	
	and a la			enerate			
Set setting property: altera_vic_driver.VIC_0.irq3_rnmi ider	ntirier to altera_	vic_ariver_VIC_0_IRQ3_RN		cincian	N.	- <u>L</u>	
Set setting property: altera_vic_driver.VIC_0.irq3_rnmi des	unation to syste	em_n_denne			<u> </u>		
					Generate	E⊻it	

幺 4-2-7

- ⑨ Problems タブを開いてエラーが無いことを確認したら Generate をクリックし、Exit で BSP Editor を閉じます。
- ① Generate が正常に終了すると、BSP フォルダにリンカ・スクリプト・ファイル(linker.x)が生成されます。このファ イルを開いて(開く際に確認画面が開く場合がありますが、F5 を押して開いてください)、共有メモリのセクション に (NOLOAD) を加えてください。

<pre>.com_mem_0_1 (NOLOAD): { PROVIDE (_alt_partition_com_mem_0_1_start = ABSOLUTE(.)); *(.com_mem_0_1. com_mem_0_1.*) . = ALIGN(4); PROVIDE (_alt_partition_com_mem_0_1_end = ABSOLUTE(.)); } > com_mem_0_1</pre>	(NOLOAD) を追加
PROVIDE (_alt_partition_com_mem_0_1_load_addr = LOADADDR(.com_mem_0_1));	
.com_mem_2_0 (<i>NOLOAD</i>) :	
<pre>PROVIDE (_alt_partition_com_mem_2_0_start = ABSOLUTE(.)); *(.com_mem_2_0. com_mem_2_0.*)</pre>	
. = ALIGN(4); PROVIDE (_alt_partition_com_mem_2_0_end = ABSOLUTE(.));	
} > com_mem_2_0	
PROVIDE (_alt_partition_com_mem_2_0_load_addr = LOADADDR(.com_mem_2_0));	

<< リンカ・スクリプト >>

複数のプロセッサが共有メモリを相互にアクセスする処理を実装した際、Nios II SBT からのソフトウェア・イメージ のロード完了後に他のプロセッサが共有メモリの領域を書き替えてしまうため、ベリファイ・エラーが発生する場合が あります。そのため、共有メモリをロード対象から外すことでベリファイを実行させないようにする必要があります。

なお、上記の処置を行うことで、ソフトウェア・イメージの中に共有メモリの初期化データが格納されなくなりますの で、ソフトウェア処理の中で動的に初期化を行う必要があることに注意してください。

 再び BSP Editor を起動し、Enable File Generation タブを開きます。Generated Files の下の linker.x を選択した 状態で右クリックし、Disable generation を選択します。こうすることで、BSP の Generate 対象から外すことがで きます。



🗵 4-2-8-1

义 4-2-8-2

⑫ Problems タブを開いてエラーが無いことを確認したら Generate をクリックし、Exit で BSP Editor を閉じます。



 Nios II SBT に戻り、Project Explorer のアプリケーション・プロジェクトを選択した状態で右クリックし、Build Project を選択し、ソフトウェアの Build を行います。



⊠ 4-2-9

 ・ソフトウェアのビルドが成功したら、再び 4-2 の先頭からその他の Nios II (nios_1 および nios_2) のプロジェクト を作成します。

€ Nios II - nios_2/multi_core_test2.c - E	clipse	
<u>File E</u> dit <u>S</u> ource Refac <u>t</u> or <u>N</u> avigate Se	<u>a</u> rch <u>P</u> roject Ni <u>o</u> s II <u>R</u> un <u>W</u> indow <u>H</u> elp	
📑 • 🖩 🦷 🛆 🗟 i 🌶 📚 i 💣 • 6	3 • 🖸 • ☞ • ॐ • Ø • 💁 • 😂 🗁 🖋 • ½ • ½ • 🏷 ↔ • ⇒ • 🖆 Ouick Access	🖬 🔯 Nios II
🎦 Project Explorer 🛛 🗖 🗖	🖻 multi_core_test0.c 🔹 multi_core_test1.c 💽 multi_core_test2.c 🕱 🖓 🗖 📑	🗄 Outline 🔀 🗖 🗖
<pre> Project Explorer 33 Project Explorer 33 Project Explorer 33 Project Explorer 33 Project Explorer 34 Project Explorer 34</pre>	<pre>mult_core_testLc @ mult_core_testLc 23 #include (stdio.h> #include (stdio.h> #include "sys/alt_irq.h" #include "sys/alt_irq.h" #include "sys/alt_irq.h" #include "altera_avalon_pio_regs.h" #include "altera_avalon_timer_regs.h" unsigned long src _attribute_ ((section (".com_mem_1_2"))) = 0; unsigned long dst _attribute_ ((section (".com_mem_2_0"))) = 0; short nCount = 0; short nLed = 0; mm -f - f libhal_bsp.a nios2-elf-ar -src libhal_bsp.a obj/HAL/src/alt_alarm_start.o obj/HAL/src/alt_busy. [BSP build complete] Info: Compiling multi_core_test2.c to obj/default/multi_core_test2.o nios2-elf-gc -xc -MP -MND -c -I/nios_2_bsp//HAL/inc -I/nios_2_bsp/ -I/nios_ Info: Linking nios_2.elf -thread_model halcpu_name nios2_cpu_2qsys true Info: (nios_2.elf -jS KBytes program size (code + initialized data). Info: (reating nios_2.elf -thread_model halcpu_name nios2_cpu_2qsys true Info: (nios_2.elf -objdumpdisassemblesymsall-headersource nios_2.elf >nios_2.od [nios_2 build complete]</pre>	<pre>Stoutine SS = Stoutine Stout Stoutine Stout S</pre>
	1	
0 items selected		

🗷 4-2-10



4-3. ソフトウェアの実行

 Nios II SBT の Nios II メニューから Quartus II Programmer を選択します。Quartus II Programmer が起動した ら Auto Detect をクリックしデバイスを認識させます。続いて File 欄の <none> をダブルクリックしてファイル を選択し、Program/Configure にチェックを入れて Start をクリックし FPGA のコンフィグレーションを行います。



🗷 4-3-1

 Nios II SBT に戻り、Run メニューから Run Configuration... を選択し、Run Configurations を起動し、Nios II Hardware を選択した状態で右クリック、New を選択します。



凶 4-3-2



③ New Configuration が生成されたら Name に任意の名前を設定します。続いて Project タブの Project name を選択すると、自動的に ELF file name に .elf ファイルのパスが設定されます。

🚝 Run Configurations	×
Create, manage, and run configu Create, manage, and run configu Target Connection]: No Nios II target downloaded. Contemport type filter text Contemport Co	Instions et connection paths were located. Check connections and that a Nios I sof is
Filter matched 8 of 8 items	ApplyRevert
?	Run Close

፼ 4-3-3

④ Target Connection タブを開いて、Processors をターゲット・プロセッサに、Byte Stream Devices をターゲット・プロセッサに接続されている JTAG-UART を選択します。それ以外はデフォルトのままとします。Processors やByte Stream Devices が表示されていない場合は、Refresh Connections をクリックします。

ERun Configurations	
Create, manage, and run config The expected Stdout device name do	urations es not match the selected target byte stream device name. ターゲット・プロセッサを選択
Ype filter text © C/C++ Application © C/C++ Remote Application © Launch Group Image: State of the application Image: State of the	Name: nics_0 Project Target Connection Connections Processors: Cable Device Device ID ISB-Blaster11 on local host IUSS-I) SOE(BA7 F ISB-Blaster11 on local host IUSS-I) SOE(BA7 F I ISB-Blaster11 on local host IUSB-I) ISE (BA7 F I nics2_cpu_l Nics2:3 ISB-Blaster11 on local host IUSB-I) ISE (BA7 F I nics2_cpu_l Nics2:3 Byte Stream Devices: Cable Device Device ID Instance ID Name Version ISB-Blaster11 on local host IUSB-I) SOE(BA7 F I I Issuert_1 I ISB-Blaster11 on local host IUSB-I) SOE(BA7 F I I taguart_0 I ISB-Blaster11 on local host IUSB-I) SOE(BA7 F I I taguart_1 I ISB-Blaster11 on local host IUSB-I) SOE(BA7 F I I taguart_2 I ISB-Blaster11 on local host IUSB-I) SOE(BA7 F I I taguart_1 I <td< td=""></td<>
Filter matched 8 of 8 items	Apply Revert
?	Run Close

₩ 4-3-4

- ⑤ Apply をクリックして設定を確定します。
- ⑥ その他の Nios II の Run Configuration も、上記の操作で同様に設定したら Close をクリックしてウィンドウを 閉じます。

🖶 Run Configurations							×
Create, manage, and run configurations							
The expected Stdout device name do	es not match the selected target byte stre	am device name.					
	Name: nios_2						
type filter text	Project 🛄 Target Connection	🏇 Debugger) 🍹	🖉 Source 🕅 🛄	<u>C</u> ommon			
C/C++ Application	Connections						
C/C++ Remote Application	Processors:						
Eacher Group	Cable	Device	Device ID	Instance ID	Name	Architecture	Refresh Connections
nios_0	JSB-BlasterII on localhost [US	5CE(BA7]FA	1	1	nios2_cpu_U nios2_cpu_1	Nios2:3 Nios2:3	Resolve Names
nios_1	JSB-BlasterII on localhost [US	5CE(BA7 FA	1	2	nios2 cpu 2	Nios2:3	System ID Properties
- mios_2							
- TRINGS II Hardware V2 (beta) - TRINGS II ModelSim	Byte Stream Devices:						
Nios II ModelSim v2 (beta)	Cable	Device	Device ID	Instance ID	Name	Version	
	JSB-BlasterII on localhost [US	5CE(BA7 FA	1	0	jtaguart_0	1	
	USB-BlasterII on localhost [US	5CE(BA7 FA	1	2	itaguart_1	1	
	Disable 'Nios II Console' view						
	Quartus Project File name: < Using default	.sopcinfo & .jdi file	s extracted from	n ELF >			
	System ID checks						
	Ignore mismatched system ID						
	Ignore mismatched system timestamp						
	Download						
	Download ELF to selected target syst	em					
	Start processor						
	Reset the selected target system						
	<u></u>						
							Apply Revert
Filter matched 10 of 10 items							
?							<u>R</u> un Close
			_				

🗵 4-3-5

⑦ Nios II SBT に戻り、Project Explorer の各プロジェクトを選択した状態で右クリック、Run As ⇒ 3 Nios II Hardware を選択します。



☑ 4-3-6

⑧ 全てのプロセッサの Run を行うと、全ての LED が点滅を始めます。

LED0 から LED2 は各プロセッサが点滅周期を制御しており、BUTTON0 から BUTTON2 の押下で点滅周期 を切り替えます。また、BUTTON3 の押下により、全てのプロセッサに対して点滅周期の初期化を行います。

LED3 は共有メモリを使ったプロセッサ間通信が正しく動作しているかどうかを示します。動作の仕組みに関して は図 4-3-7 を参照してください。



図 4-3-7



4-4. ハードウェア・イメージとソフトウェアの Flash ROM への書き込み

 デフォルトでは CFI Flash ROM の初期化処理が全ての Nios II ソフトウェアに含まれることになり、Flash ROM から起動させた場合、処理が競合してしまい正常に動作しませんので、Nios II #0 にのみ残し、Nios II #1 と Nios II #2 の初期化処理をコメント・アウトします。Nios II SBT の Nios II #1 プロジェクトと Nios II #2 プロ ジェクトの BSP フォルダの alt_sys_init.c ファイルを開き、下記の箇所を修正します。

: void alt_sys_init(void) { ALTERA_AVALON_TIMER_INIT (TIMER_1, timer_1); // ALTERA_AVALON_CFI_FLASH_INIT (EXT_FLASH, ext_flash); ALTERA_AVALON_JTAG_UART_INIT (JTAG_UART_1, jtag_uart_1); } :				
< <nios #1="" ii="" の修正コード="">></nios>				
: void alt_sys_init(void) { ALTERA_AVALON_TIMER_INIT (TIMER_2, timer_2); // ALTERA_AVALON_CFI_FLASH_INIT (EXT_FLASH, ext_flash); ALTERA_AVALON_JTAG_UART_INIT (JTAG_UART_2, jtag_uart_2); } :				

<< Nios II #2 の修正コード >>

修正が終了したら、Nios II #1 と Nios II #2 のプロジェクトを再ビルドします。

※詳細については、5章. 複数の Nios II を構成する際の注意事項を参照ください。

② Nios II SBT の Nios II メニューから Flash Programmer を選択し、Nios II Flash Programmer を起動します。

🔬 Nios II Flash Programmer	
File Options Tools Help	
Target hardware information	
BSP Settings File name:	
SOPC Information File name:	
CPU to program flash:	
Hardware connection:	Connections
Nios II Flash Programmer	
This tool is the GUI version of the command line tool: nios2-flash-programmer-generate.	
It parses a SOPC Information File or Board Support Package (BSP) Settings File, and dynamically creates a tab for each flash device for	und.
Each Flash tab allows multiple files to be added to a list for eventual conversion to flash and programming to its target flash device.	
Use the File-New menu item from the top level menu bar to create a new flash settings project, or open an existing project using the File->Open menu item.	
Information Problems Processing	
- 	Start Exit

ً ⊈ 4-4-1

※ Flash Programmer を使って Flash ROM に書き込む際は、直前に本検証で作成したコンフィグレーション・デ ータ(.sof)を書き込んでから行ってください。



③ Nios II Flash Programmer の File メニューから New を選択し、New Flash Programmer Setting File が起動した ら、Get flash programmer system details from BSP Settings File または Get flash programmer system details from SOPC information File にチェックを入れ、各項目に BSP セッティング・ファイル_{※1}、または、.sopcinfo ファイル を設定し、OK をクリックします。なお、選択する Nios II はどれでも結構です。

搔 New Flash Programmer Set	tings File X					
🔘 Get flash programmer system details from BSP Settings File						
• Get flash programmer system deta	ails from SOPC Information File					
BSP Settings File name:						
SOPC Information File name:	C:\Lab\nios2_multi_core_sample\nios2_system.sopcinfo					
Master CPU name:	nios2_cpu_0					
Flash memory:	ext_flash					
	OK Cancel					
	Ĩ 1.1.2.1					
🛓 New Flash Programmer Set	tings File					
New Flash Programmer Set G Get flash programmer system det	tings File					
New Flash Programmer Set Get flash programmer system det C Get flash programmer system det	Itings File					
 New Flash Programmer Set Get flash programmer system det Get flash programmer system det BSP Settings File name: 	times File ails from BSP Settings File ails from SOPC Information File C:\Lab\nios2_multi_core_sample\software\nios_0_bsp\settings.bsp					
 New Flash Programmer Set Get flash programmer system det Get flash programmer system det BSP Settings File name: SOPC Information File name: 	Image:					
 New Flash Programmer Set Get flash programmer system det Get flash programmer system det BSP Settings File name: SOPC Information File name: Master CPU name: 	tings File X alls from BSP Settings File X alls from SOPC Information File C:\Lab\nios2_multi_core_sample\software\nios_0_bsp\settings.bsp C:\Lab\nios2_multi_core_sample\nios2_system.sopcinfo nios2_cpu_0					
 New Flash Programmer Set Get flash programmer system det Get flash programmer system det BSP Settings File name: SOPC Information File name: Master CPU name: Flash memory: 	times File Image: Settings File ails from BSP Settings File Image: Settings File ails from SOPC Information File Image: Settings.bsp [::Lab\nios2_multi_core_sample\nios2_system.sopcinfo Image: Settings.bsp c:Lab\nios2_multi_core_sample\nios2_system.sopcinfo Image: Settings.bsp nios2_cpu_0 ext_flash					

図 4-4-2-2

- ※ 1. BSP セッティング・ファイルは、デフォルトでは Nios II ソフトウェア・プロジェクト(software フォルダ)の BSP フォルダの中に settings.bsp と言うファイル名で存在します。
- ④ Nios II Flash Programmer の Connections... ボタンをクリックします。Hardware Connections ウィンドウが表示されたら、Processors: のリストから③で選択した Nios II を選択し、Close をクリックします。

🔬 Hardware Connections 📃 🔀	1
Connections	- Refresh Connections
Processors:	
Cable Device Device ID Instance ID Name Architecture Refresh Connections	
USB-BlasterII on localhost [USB-1] 5CE(BA7 F 1 0 nios2_cpu_0 Nios2:3 Resolve Names	
JSB-BlasterII on localhost [USB-1] 5CE(BA7 F 1 1 nios2_cpu_1 Nios2:3	
JSB-BlasterII on localhost [USB-1] 50E(BA7 F 1 2 nios2 cpu 2 Nios2:3 System ID Properties	
4-4-2 で選択した Nios II を選択	
Quartus Project File name: C:\Lab\nios2_multi_core_sample\nios2_basic_lab.qpf	
System ID checks	
🔽 Ignore mismatched system ID	
Ignore mismatched system timestamp	
Close	í l
	-

⊠ 4-4-3

- ※ Processors: にリストが正しく表示されない場合は、Refresh Connections をクリックしてください。
- ※ System ID ミスマッチのエラーが出た場合には、System ID checks の 2 項目をチェックしてください。

⑤ Nios II Flash Programmer の Add ボタンをクリックします。ファイル選択ウィンドウが表示されたら Nios II #0 の ソフトウェア・プロジェクト・フォルダに生成されている .elf ファイルを選択します。続いて Properties... ボタンを クリックして Properties ウィンドウを開きます。Properties ウィンドウの CPU reset address: には 0x4000000、 Flash base address: に 0x0、Flash end address: に 0x400000 が設定されていることを確認します。Boot loader: には標準のブート・コピアが設定されていますのでそのままにして、Close で閉じます。

🖇 Nios II Flash Programmer					1
File Options Tools Help					
Target hardware information					7
BSP Settings File name:					
SOPC Information File name: C:\Lab\nios2_multi_core_sample\nios2_s	system.sopcinfo				
CPU to program flash: nios2_cpu_0					
Hardware connection: Connection: USB-BlasterII on localhost [USB-1] Device: 5CE(B	3A7 FA7 FA7ES)@1 Device :	ID: 1 CPU Instance ID: 0 .	Connections	
Flash: ext_flash					
Base address: 0x0 Memory span: 0x40	00000				
Master CPU: nios2_cpu_0 ,zip file system offset in BSP:					
Files for flash conversion:					
File Name		Conversion Type	Flash Offset	Nios II #0 の起動	アドレス:0x400000
E:¥Lab¥nios2 multi core sample¥software¥nios O¥nios 0.elf ELF <no offset<="" td=""></no>					
		1.1			
l	Properties	and the second sec	×	1	
File generation command:	CDU waart adde			-	
elf2flashinput="C:/Lab/nios2_multi_core_sample/software/	CPO reset adure			Properties	
output="C:/Lab/nios2_multi_core_sample/flash/nios_0_ext_f	Flash base addre	ess: 0x0	_		
boot="nios2eds/components/altera_nios2/boot_loader_cfi.sr	Elash end addre	ess: 0x4000000	▼.	1.8	Propertiesをクリック
File programming command:			<u> </u>		1
nios2-flash-programmer "C:/Lab/nios2 multi core sample/flas	Boot load	der: D:/altera/14.1/nios2eds,	/components/altera_r	Flash ROM	の失雨ノ終端アドレス
'cable=USB-BlasterII on localhost [USB-1]'programve	Additional argume	nts:			
				で設定	
			Close		

፼ 4-4-4

⑥ 再び Add ボタンをクリックし、ファイル選択ウィンドウが表示されたら Nios II #1 のソフトウェア・プロジェクト・フォルダに生成されている .elf ファイルを選択します。続いて Properties... ボタンをクリックして Properties ウィンドウを開きます。Properties ウィンドウの CPU reset address: には 0x460000 を設定します。Boot loader: には 標準のブート・コピアが設定されていますのでそのままにして、Close で閉じます。Nios II #2 についても同様の 手順で CPU reset address: には 0x4C0000 を設定します。

		N	ios II #1 の起動アドレス:0x460000			
🕌 Properties	×	を	設定		Nios II #0 の起動アドレス:0	x4C0000
CPU reset address:	0x460000			/	を設定	
Flash base address:	0x0		🛓 Properties 💦 💡	,	×	
Flash end address:	0×4000000		CPU reset address: 0x4C0000			
Boot loader:	D:/altera/14.1/nios2eds/components/altera_r		Flash base address: 0x0			
Additional arguments:			Flash end address: 0x4000000			
	Close		Boot loader: D:/altera/14	1/nios2eds,	/components/altera_r	
			Additional arguments:			
×	4-4-3				Close	

₩ 4-4-6

⑦ 再び Add ボタンをクリックし、ファイル選択ウィンドウが表示されたら本検証で作成した .sof ファイルを選択します。Flash Offset にはファスト・パッシブ・パラレル・モードでの User Hardware 1 コンフィグレーション・アドレスである 0x800000 を設定します。続いて Properties... ボタンをクリックして Properties ウィンドウを開きます。 Properties ウィンドウの Additional arguments: に --pfl --optionbit=0x18000 --programmingmode=FPP と設定してください。

🛃 Nios II Flash Programmer				×	
File Options Tools Help					
Target hardware information					
BSP Settings File name:					
SOPC Information File name: C:\Lab\nios2_multi_core_sample\nios2_system.sopcinfo					
CPU to program flash: nios2_cpu_0					
Hardware connection: Connection: USB-BlasterII on localhost [USB-1] Device: 5CE(BA7 FA7 FA7ES)@1 Device I	D: 1 CPU Instance ID: 0	Connections	1	
Flash: ext_flash					
Base address: 0x0 Memory span: 0x4000000					
Master CPU: nios2_cpu_0 .zip file system offset in B5P:					
Files for flash conversion:					
File Name	Conversion Type	Flash Offset	Add		
D:¥Lab¥nios2_multi_core_sample¥software¥soft_test0¥soft_test0.elf	ELF	<no offset=""></no>	Remove		
C:¥Lab¥nios2_multi_core_sample¥software¥nios_1¥nios_1.elf	ELF	<no offset=""></no>			
C:¥Lab¥nios2_multi_core_sample¥software¥nios_2¥nios_2.elf	ELF	<no offset=""></no>			
C:¥Lab¥nios2 multi core sample¥output files¥nios2 basic lab.sof	SOF	0×800000	4]	Flash Offset に 0x800000 を設定	
File generation command:		×			
sof2flashinput="C:/Lab/nios2_multi_core_sample/output_fi Additional argume	nts:pfloptionbit=0x1800)programmingmode=FPP	Properties		
output="C:/Lab/nios2_multi_core_sample/flash/nios2_basic_					
optionbit=0x18000programmingmode=FPP Propertiesをクリック					
File programming command:					
nios2-flash-programmer "C:/Lab/nios2_multi_core_sample/flas		1. N. 1.			
device=1instance=0 'cable=USB-BlasterII on localhost		×-		- FPP に必要なオプション を設定	
Information Problems Processing		Close		-1	

図 4-4-7

※User Hardware 1 コンフィグレーション・アドレスである 0x800000 や optionbit で指定している 0x18000 は、アル テラ社評価用ボードの設定となります。詳しくは、評価用ボードのマニュアルを参照ください。

- ⑧ start ボタンをクリックして Flash ROM にソフトウェア・イメージとコンフィグレーション・データを書き込みます。 Processing にエラー・メッセージが表示されなければ終了です。ボードの PGM_RECONFIG ボタンを押下して、 正しく動作するか確認してください。
 - ※上記の設定では、評価用ボードの設定が、FPP(ファスト・パッシブ・パラレル)に設定されており、User Hardware1 のイメージがデフォルトでロードされる設定になっていることを確認してください。詳しくは、評価用 ボードのマニュアルを参照ください。

5. <u>複数の Nios II を構成する際の注意事項</u>

Nios II SBT で、ソフトウェアをビルドすると、Qsys の GUI 上で、Nios II と接続されているペリフェラル用の初 期化コードを自動で生成します。この各ペリフェラルに対応した初期化コードで、ペリフェラルを制御するためにア ルテラ社から提供された HAL を利用するための準備や各ペリフェラルの一般的な理想と考えられる初期状態の レジスタ設定等を行います。しかし、この自動で生成される各ペリフェラルの初期化コードは、複数の Nios II が構 成されたシステムで、且つ複数の Nios II が同一のペリフェラルを共有させたシステムに対応したものでありませ ん。

複数の Nios II を構成する場合においての注意事項に関して説明します。

```
#include "system.h"
#include "sys/alt_sys_init.h"
/*
 * device headers
 */
#include "altera_avalon_timer.h"
#include "altera_avalon_uart.h"
/*
 * Allocate the device storage
 */
ALTERA_AVALON_UART_INSTANCE( UART1, uart1 );
ALTERA_AVALON_TIMER_INSTANCE( SYSCLK, sysclk );
/*
 * Initialize the devices
 */
void alt_sys_init( void )
{
    ALTERA_AVALON_UART_INIT( UART1, uart1 );
    ALTERA_AVALON_TIMER_INIT( SYSCLK, sysclk );
}
```

5-1. 排他制御に関して

周知のとおり、前提として複数のプロセッサがペリフェラルを共有する場合、排他処理が必要です。片方のプロ セッサがあるペリフェラルに対して、何らかの処理をするためにアクセスしている最中に、他のプロセッサから同一 ペリフェラルに対してのアクセスは避ける必要があります(フラグ情報を複数のプロセッサが共通で管理すること等 で排他アクセスを実現する必要があります)。前述したように、Nios II SBT では Qsys で接続されているペリフェ ラル用に、初期化コードを自動で生成します(Qsys の GUI 上で、Nios II と接続されたペリフェラル用の初期化コ ードを自動生成します)。下の図の例のように、複数の Nios II に共通で接続されたペリフェラルが存在する場合 には、それぞれの Nios II の初期化コード内に、共通で接続されたペリフェラル用の初期化コードが自動でリンクさ れますが、この初期化コードはこのような構成時に対応したものではありません。自動で生成されたコードを変更 なく採用してしまうと、それぞれの、Nios II は同一ペリフェラルに対し排他制御をせず初期化を実行してしまいま す。つまり、このような構成は、自動で生成される初期化コードをそのまま変更なく採用する場合において、許され た構成ではありません。



図 5-1-1

以下の例では、それぞれの Nios II から、共通のペリフェラルへアクセスができるようにハードウェアが接続されていますが、排他制御が非常に手間となり、また、割り込みがかかると目的でない方の Nios II も割り込みを認識してしまう理想的でない構成です。また、自動で生成される初期化コードでは動作保証がされない非推奨の構成 となっています。





5-2. 自動で生成される初期化コードがサポートされる構成

複数の Nios II をシステムに実現する場合、それぞれの Nios II が単独で利用できるペリフェラルを複数個用 意し、バスを完全に分離させた構成にします。このような構成であれば、ツールにより自動生成されたペリフェラル 用の初期化コードを変更することなく利用できます。この自動生成される初期化コードは、それぞれの Nios II に 接続されたペリフェラルに対してのみ用意されます。



🗵 5-2-1

以下の例は、それぞれの、Nios II から個々のペリフェラルにのみアクセスができるハードウェア構成になって いて、割り込みポートも接続された Nios II にのみ有効になります。下の図で示されている DMA コンポーネント のみならず、UART、タイマ、SPI 等、すべてのペリフェラルは共有せず、それぞれの Nios II 用に専用で独立して 接続することが推奨の構成となります。

Use	Connections	Module Name	Description	Clock	Base	End	IRQ
	•	 ⊂ cpu_0 ✓ instruction_master ✓ data_master → jtag_debug_module 	Nios II Processor Avalon Memory Mapped Master Avalon Memory Mapped Master Avalon Memory Mapped Slave	clk_0	il ⊯°0×00000800	RQ 0 IRQ 3 0x00000fff	1
	*****		DMA Controller Avalon Memory Mapped Slave Avalon Memory Mapped Master Avalon Memory Mapped Master	clk_0	0x0000020	0x000003f	10
			Nios II Processor Avalon Memory Mapped Master Avalon Memory Mapped Master Avalon Memory Mapped Slave	clk_0	ii ⊯°0x20000800	RQ 0 IRQ 3 0x20000fff	1
	┝┿┿┿┿ ╸ 	Control_port_slave read_master ≺ write_master	DMA Controller Avalon Memory Mapped Slave Avalon Memory Mapped Master Avalon Memory Mapped Master	clk_0	₩° 0x0000060	0×0000007f	



システム全体のスループットの向上、排他制御の不要等の利点より、すべてのペリフェラルは、それぞれのプロ セッサごとに個別に用意し構成するのが理想的です。しかし、そのような構成がとれず、ペリフェラルを複数の Nios II で共有する場合には、前述のとおり自動で生成される初期化コードをそのまま採用すると問題を起こす可 能性があります。

初期化コードを自動で生成しないペリフェラルと自動で生成するペリフェラルに関して次に示します。

<<自動で初期化コードを生成しないペリフェラル>>

- ・ ユーザ・ペリフェラル
- ・ Generic Tri-State Controller (CFI Flash を構成したもの) 以外のメモリコンポーネント

(例: SDRAM 、DDR SDRAM コントローラ・コンポーネント、オンチップ・メモリコンポーネント等)

<<自動で初期化コードを生成するペリフェラル>>

・ メモリ以外のすべてのアルテラ社から提供されるペリフェラル

詳細は、Nios II Software Developer's Handbook を確認してください。

https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2sw_nii5v2.pdf

多くのペリフェラル用に初期化コードが用意されています。この自動で生成される初期化コードの処理内容を 十分把握する必要があります。

例外として、System ID Core に関しては、自動で生成される初期化コード内に、マルチコア・システムにおい て、問題を引き起こす処理が含まれていなく、割り込みのポートも持たないため、複数の Nios II で共有すること ができます。また、もともと排他処理を目的として用意されているペリフェラル(ミューテックス・コア、メールボック ス・コア)についても、複数の Nios II で共有することができます。Generic Tri-State Controller(CFI Flash を構成した もの)以外のメモリコンポーネントは、複数の Nios II でアクセスする領域が重ならないように利用する場合は共 有できます。

それ以外のコンポーネントに関しては、複数の Nios II から同一のペリフェラルとして共有する場合には注意が 必要です。

各初期化コード内で行われるもともとの処理内容を理解し、必要に応じてユーザ側で削除、または変更等をする 必要があります。具体的な例としては、Generic Tri-State Controller(CFI Flash を構成したもの)であれば、フラッシ ュ・メモリをクエリモードに入れるコマンドやフラッシュ・メモリに存在するプログラムを RAM へ展開するブート動 作が複数の Nios II から行われないように対処します。DMA コンポーネントであれば、初期化コード内の処理で 割り込みの登録がされているので、1つの Nios II が割り込みを受け付ける状態に入った後、異なる Nios II が初 期化コードを実行することで IRQ のノードが一瞬だけ成立してしまい、先に割り込みを受け付けた Nios II が例 外処理を実行するが、その IRQ のパルス幅が狭いことが原因で正しい例外処理が行われず先に割り込みを受 け付けた Nios II が無限ループして停止してしまうことを防ぐために、自動で生成される初期化コードは削除し、カ スタムで初期化を行います。これらの例のように各ペリフェラルに応じて対処が必要です。

5-3. まとめ

Nios II でマルチ・プロセッサ・システムを構成する際、様々なペリフェラルはそれぞれの Nios II ごとに個別に 用意しバスを分離することを推奨します。また、複数の Nios II から、同一のペリフェラルを共有する場合には、自 動で生成されるそのペリフェラル用の初期化コードでは、システムの動作保証がされていないため、ユーザの方で 問題が起きないように対処が必要となります。



<u> 改版履歴</u>

Revision	年月	概要
1	2015 年 5 月	初版

免責およびご利用上の注意

弊社より資料を入手されましたお客様におかれましては、下記の使用上の注意を一読いただいた上でご使用ください。

- 1. 本資料は非売品です。許可無く転売することや無断複製することを禁じます。
- 2. 本資料は予告なく変更することがあります。
- 本資料の作成には万全を期していますが、万一ご不明な点や誤り、記載漏れなどお気づきの点がありましたら、本資料を入手されました下記代理店までご一報いただければ幸いです。
 株式会社アルティマ ホームページ: http://www.altima.co.jp
 技術情報サイト EDISON: http://www.altima.jp/members/index.cfm
 株式会社エルセナ ホームページ: http://www.elsena.co.jp
 技術情報サイト ETS : http://www.elsena.co.jp/elspear/members/index.cfm
- 4. 本資料で取り扱っている回路、技術、プログラムに関して運用した結果の影響については、責任を負いかねますのであらかじめご了承ください。
- 5. 本資料は製品を利用する際の補助的な資料です。製品をご使用になる際は、各メーカ発行の英語版の資料もあわせてご利用ください。