

# はじめての インテル<sup>®</sup> SoC FPGA 演習マニュアル (Atlas-SoC / DE10-Nano ボード版)

Ver.17.1



# はじめての インテル<sup>®</sup> SoC FPGA 演習マニュアル (Atlas-SoC / DE10-Nano ボード版)

# <u>目次</u>

1. 概要	4
1-1. 使用環境	5
2. ボードの設定	6
2-1. ボード・レイアウト	6
2-2. 電源およびケーブルの接続	6
2-3. SW10 の設定	6
3. 演習 1: ハードウェア演習	7
3-1. ステップ 1 : ハードウェア演習デザイン・プロジェクトのオープン	8
3-2. ステップ 2 : HPS コンポーネントの追加	13
3-3. ステップ 3 : HPS ペリフェラルの設定(MAC、UART、I2C、SDIO、USB)	
3-4. ステップ 4 : HPS クロックの設定	25
3-5. ステップ 5 ː SDRAM の設定	
3-6. ステップ 6 : HPS のクロックとエクスポート信号の設定	
3-7. ステップ 7 : HPS コンポーネントと他のコンポーネントの接続	
3-8. ステップ 8 : リセットの接続とベース・アドレスの割り当て	
3-9. ステップ 9 : Platform Designer システムの確認	
3-10. ステップ 10 : Platform Designer システムの生成	40
3-11. ステップ 11 : ピン・アサインメントの設定と Quartus <sup>®</sup> Prime プロジェクトのコンパイル	
3-12. ステップ 12 : 出力ファイルの確認	
3-13. 演習 1 ハードウェア演習のまとめ	
4. 演習 2: ソフトウェア演習(1) Preloader の生成	49
4-1. ステップ 1 : Embedded Command Shell の起動	
4-2. ステップ 2 : bsp-editor(Preloader Generator)の起動	50
4-3. ステップ 3 : プロジェクトの作成と設定	50
4-4. ステップ 4 : Preloader のビルド	54
5. 演習 3:ソフトウェア演習(2) ベアメタル・アプリケーション	55
5-1. FPGA デザインのダウンロード	
5-2. Hello World サンプル・アプリケーションの実行	59



# はじめての インテル<sup>®</sup> SoC FPGA 演習マニュアル

(Atlas-SoC / DE10-Nano ボード版)

5-3. LED Blink サンプル・アプリケーションの実行	. 68
5-4. 演習 2 で作成した Preloader による初期化(オプション演習)	. 75
5-5. システム・ヘッダ・ファイルによるアドレスの解決(オプション演習)	. 77
6. 演習 4: Linux アプリケーション演習 (オプション演習)	. 80
6-1. microSD カードの準備	. 80
6-2. Linux 起動とログイン	. 82
6-3. Linux での IP アドレスとパスワードの設定	. 84
6-4. ホスト PC 側のネットワーク設定	. 85
6-5. DS-5™ の起動と Linux サンプル・アプリケーションのインポートおよびビルド	. 88
6-6. リモート・システム・エクスプローラー(RSE)の設定	. 92
6-7. Linux アプリケーションの実行・デバッグ	. 96
<ol> <li>7. 今後の参考資料について</li> </ol>	102
改版履歴	103

# 

## 1. 概要

この演習では、Cyclone<sup>®</sup> V SoC FPGA 評価キット「DEO-Nano-SoC Kit / Atlas-SoC Kit」(以下、Atlas-SoC ボード)、 または「DE10-Nano Kit」(以下、DE10-Nano ボード)を使用して、Cyclone<sup>®</sup> V SoC のハードウェア、ソフトウェア それぞれの開発方法について解説します。

この演習を実行することにより、インテル® SoC FPGA の開発環境である インテル® Quartus® Prime 開発ソフト ウェアやシステム構成ツールである Platform Designer (旧: Qsys システム統合ツール)、およびソフトウェア開 発環境である インテル® SoC Embedded Design Suite (以下、SoC EDS)の基本的な操作を学ぶことができます。

演習は、以下の 4 つで構成されています。

- 演習 1: ハードウェア演習
- 演習 2: ソフトウェア演習(1)
- 演習 3: ソフトウェア演習(2)
- 演習 4: Linux アプリケーション演習(オプション演習)

演習 1 では、Quartus<sup>®</sup> Prime を使用して Arm<sup>®</sup> プロセッサを含むハードウェアを構成し、簡易的な SoC シ ステムを設計します。

演習 2 では、SoC EDS ツールを使用して 28nm 世代のブートローダである Preloader の生成を行います。

演習 3 では、Arm<sup>®</sup> Development Studio 5 Intel<sup>®</sup> SoC FPGA Edition (以下、DS-5<sup>™</sup>)を利用したソフトウェア開発 およびベアメタル・アプリケーションのデバッグを実施します。

演習 4 では、SD カードイメージを使用し、SoC デバイス上で Linux を動作させ、その上でアプリケーションを DS-5™ を使用して実行・デバッグします。

i Note:

演習 4 は、弊社開催の「SoC スタートアップ・トライアル・セミナー」では、時間の都合上実施しない オプション演習となります。



#### 1-1. 使用環境

この演習では、以下のソフトウェアを使用します。

- Quartus<sup>®</sup> Prime Standard Edition v17.1 (Lite Edition でも可能)
   また Device データとして Cyclone<sup>®</sup> V を登録しておく必要があります。
   インストール方法については以下のサイトをご参照ください。
   Quartus<sup>®</sup> Prime & ModelSim<sup>®</sup> インストール方法 (v17.1)
- SoC Embedded Design Suite v17.1(以下、SoC EDS)

インストール方法に関しては以下のサイトをご参照下さい。 SoCEDS のインストール方法 (v17.1)

● 演習データ( SoC-Trial\_Seminer\_Lab\_data\_atlas\_de10nano\_v17.1.exe )

演習データの .exe ファイルをダブルクリックすると、デフォルトでは次の場所に展開されます:

C:¥lab¥soc\_lab¥cv\_soc\_lab

本資料では演習データを上記の場所に展開したものとして説明しています。

ホスト PC の OS : Windows<sup>®</sup> 7 Professional

この演習では、Windows®7 Professional を使用して動作の確認を行っております。

▲ 注意事項:
<u>ホスト PC の OS が Windows® 10 の場合、Preloader の生成でエラーが発生する場合が確認さ</u>
した。 <u>れております。</u> していたので、 していたのでのでのでので、 していたので、 していたので、 していたので、 していたので、 していたので、 していたので
【エラー内容】
この問題は、SOC EDS ツールを使用してプリローダを生成するときに発生します。
新しい HPS および BSP 設定ファイルを作成した後、make コマンドが失敗します。
<pre>tar zxf /cygdrive/c/intelFPGA/18.0/embedded/host_tools/altera/preloader/uboot-socfpga.tar.gz</pre>
<pre>tar: Error opening archive: Failed to open '/cygdrive/c/intelFPGA/18.0/embedded/host_tools/altera/preloader/uboot-socfpga.tar.gz' make: *** [uboot-socfpga/.untar] Error 1</pre>
もしご使用の OS が Windows® 10 でエラーが発生する場合は、以下の参考情報サイトで説明され
ている対策が必要となりますのでご注意ください。
【参考情報サイト】
i Intel <sup>®</sup> Knowledge Base - <u>Unable to make preloader in Windows 10</u>
「」アルティマ技術サポート「 <u>Windows®10 における Preloader のビルドエラー</u> 」

## 2. ボードの設定

このセクションでは、演習 1、2、3 を実施するために必要なボードのセットアップに関して解説します。

2-1. ボード・レイアウト

本演習で使用する Atlas-SoC ボードのレイアウト図を以下に示します。

DE10-Nano ボードも基本的には同じです。



図 2-1 Atlas-SoC ボード・レイアウト図

2-2. 電源およびケーブルの接続

AC アダプタの接続や各種ケーブルは以下の通り接続してください。

- 電源(AC アダプタ)を DC 入力 (J14) に接続します。
- Mini USB ケーブルで作業用 PC とオン・ボード USB-Blaster™ II コネクタ(J13)を接続します。

2-3. SW10 の設定

SW10(MSEL 設定スイッチ)が以下の通り設定されていることを確認します。 この設定により、FPGA は FPPx32 モードとなります。

ボード・リファレンス	信 <del>号</del> 名	設定					
SW10. 1	MSELO	ON ("0")					
SW10. 2	MSEL1	OFF ("1")					
SW10. 3	MSEL2	ON ("0")					
SW10. 4	MSEL3	OFF ("1")					
SW10. 5	MSEL4	ON ("0")					
SW10. 6	N/A	N/A					







# 

## 3. 演習 1: ハードウェア演習

このセクションでは、Quartus<sup>®</sup> Prime および Platform Designer を使用し、以下に示す Arm<sup>®</sup> プロセッサを含むハードウェアの設計を行います。

インテル<sup>®</sup> Soc FPGA では Cyclone<sup>®</sup> V に限らず、Quartus<sup>®</sup> Prime に含まれている Platform Designer というツ ールを使用してシステムを構成します。この Platform Designer では Hard Processor System (以下、HPS)のブ ロックをはじめ、FPGA 側に実装することのできるコンポーネント群が用意されており、所望のコンポーネントの みを実装することでリソースの最適化を図ることができます。また作成したシステムはペリフェラルが対応してい れば、簡単に他のデバイスに移植できますので、それ自体も設計資産として活用していただくことが可能です。

本演習では演習時間を短縮するため、あらかじめ Platform Designer システム内にいくつかのコンポーネント とクロック・ソース・コンポーネントが実装してあります。このため、 HPS ブロック (太枠で囲われた青色のブロッ ク) の追加と既存コンポーネントの接続を実施します。演習内容は以下の通りです。

#### 演習内容:

- HPS コンポーネントを既存の Platform Designer システムへ追加
- HPS インタフェースと他のパラメータの設定
- 既存コンポーネントと HPS との接続
- Platform Designer システムの生成



図 3-1 演習 1 で設計する SoC システムのブロック図

3-1. ステップ 1 : ハードウェア演習デザイン・プロジェクトのオープン

演習を進めるにあたり、本演習マニュアルの各ステップに記載されている全ての説明をよく読み慎重に作業を 進めてください。

本資料では作業ディレクトリを C:¥lab¥soc\_lab フォルダとして説明をします。作業フォルダを変更された場合は 設定した環境に合わせて読み直してください。

では、はじめましょう。

\_\_1. インストールされている Quartus<sup>®</sup> Prime 17.1 Standard Edition 開発ソフトウェア内の、Quartus<sup>®</sup> Prime を起動しま す。デフォルトのままであれば下記にあります。

Windows スタート ⇒ すべてのプログラム ⇒ Intel FPGA 17.1.0.590 Standard Edition / Lite Edition ⇒ Quartus Prime Standard/Lite Edition 17.1.0.590 ⇒ Quartus Prime 17.1

\_2. Quartus<sup>®</sup> Prime メニュー・バーから、File ⇒ Open Project を選択し、C:¥lab¥soc\_lab¥cv\_soc\_lab にある soc\_system.qpf を選択します。この qpf ファイルは Quartus<sup>®</sup> Prime でのプロジェクト・ファイルとなっています。

0	Quartus Prime Standar	d Edition	Open Project				<b>×</b>
File	Edit View Project	Assignments Processing		.ーター 🔸 OS (C:) 🔸 lab 🔸 soc_lat	b ▶ cv_soc_lab ▶	✓ ← cv_soc_lab	の検索 👂
	New	Ctrl+N	整理 新しいフォル	ダー		. 18	0
10	Open	Ctrl+O	🍌 お気に入り 🔶	名前	更新目時	種類	サイズ
	Close	Ctrl+F4	🍌 ダウンロード	👢 .qsys_edit	2018/02/01 18:20	ファイル フォル	
	Nou Dealast Missed		🔳 デスクトップ 😑	🗼 db	2018/02/01 18:20	ファイル フォル	
12	New Project Wizdru		🔝 最近表示した場所	🐌 ip	2018/02/01 18:20	ファイル フォル	
Æ	Open Project	Ctrl+J	Box Sync	📙 License	2018/02/01 18:20	ファイル フォル	
_	Save Proiect			🐌 software_example	2018/02/01 18:20	ファイル フォル	
			*** ライブラリ	Streamline	2018/02/01 18:20	ファイル フォル	
				Soc_system.qpf	2018/01/25 11:02	QPF ファイル	2 KB
			💐 コンピューター				
			2 OS (C:)				
			· · · · · · · · · · · · · · · · · · ·	•	III		•
			ファ	イル名(N) soc_system.qpf	-	Quartus Prime Proje	ct File (* 🔻
						開<( <u>0</u> ) =	キャンセル

図 3-2 Quartus<sup>®</sup> Prime プロジェクトのオープン

- \_\_3. ボードの選択を行います。図を参考に、使用するボードを設定してください。
  - DEO-Nano-SoC / Atlas-SoC ボードの場合 : <u>Atlas</u> を選択
  - DE10-Nano ボードの場合 : <u>DE10-Nano</u> を選択

この設定を行うことにより、今回使用するボードに合わせ、あらかじめ設定済みのピンの配置や使用するデバイ スなどの情報を使用することができるようになります。

File Edit View Project A	ssignments Processing	Too	ls
D©∃/≁DD/⊃C	atlas	•	2
Project Novigator	atlas		-
-Toject Navigator	- DE10-Nano		1.11
	Entity:instance	_	•
Cyclone V: 5CSEMA4U23C6			
ehrd top <sup>1</sup> / <sub>1</sub>			





\_4. Quartus<sup>®</sup> Prime の **Tools ⇒ Platform Designer** を起動します。 または、ツール・バーにある Platform Designer のアイコン 👗 をクリックし、Platform Designer を起動します。

🕥 Quartus Prime Standard Edition - C:/lab/soc_la	b/cv_soc_lab/soc_system - atlas
File Edit View Project Assignments Processing	Tools Window Help
D た 日 ゲ D D つ C atlas Project Navigator Entity/Instance	Run Simulation Tool
Cvclone V: 5CSEMA4U23C6	S TimeQuest Timing Analyzer
▶ ghrd_top <sup>™</sup>	Advisors +
	<ul> <li>Chip Planner</li> <li>Design Partition Planner</li> <li>Netlist Viewers</li> </ul>
	<ul> <li>Signal Tap Logic Analyzer</li> <li>In-System Memory Content Editor</li> <li>Logic Analyzer Interface Editor</li> <li>In-System Sources and Probes Editor</li> <li>Signal Probe Pins</li> <li>Programmer</li> <li>JTAG Chain Debugger</li> <li>Fault Injection Debugger</li> <li>System Debugging Tools</li> </ul>
	IP Catalog
Fasks	Nios II Software Build Tools for Eclipse
Task	A Platform Designer
Compile Design	/ Tcl Scripts

図 3-4 Platform Designer の起動

\_\_\_\_5. soc\_system.qsys ファイルを開きます。



図 3-5 Platform Designer ファイルのオープン



まずは簡単に Platform Designer の使い方について説明します。

Platform Designer では主に IP Catalog と System Contents、そして Message Window の3つの画面があります。

IP Catalog には Platform Designer で使用できるコンポーネントがラインナップされています。この中から実装 したいコンポーネントを System Contents に追加します。そして System Contents 内のコンポーネント同士を接続 し、システムを作成します。HPS と呼ばれるチップ内のハード・マクロ化された部分に関してもソフト・コンポーネ ントとして IP Catalog 上にラインナップされており、このコンポーネントを Platform Designer システムに実装する ことで SoC デバイスの HPS 側が使用できるようになります。







オープンした Platform Designer システムは以下のコンポーネント(白色)が実装済みとなっています。この システムに対して HPS ブロック(青色)の追加と設定、そして実装済みコンポーネントの接続を行います。

実装済みのコンポーネント(白色):

- クロック・ソース
- オンチップ・メモリ
- LED/Button 制御用 PIO ペリフェラル
  - ➢ DIP スイッチ PIO
  - ▶ ボタン PIO
  - LED PIO

演習で追加するコンポーネント(青色):

• HPS



図 3-7 設計する Platform Designer システム

Platform Designer では各 IP ごとに設定画面が用意されており、System Contents 内のコンポーネントをダブ ルクリックすると、そのコンポーネントの設定画面を開くことができます。



- \_\_\_6. Clock Source コンポーネント (clk\_0) をダブルクリックして、Clock Frequency は開発ボード上の発振器と一致さ せるため、50 MHz に設定されていることを確認してください。
- \_\_\_\_7. Clock frequency is known がチェックされていることを確認してください。

ystem	Contents	🕴 Address Map 🛛	Interconnect Requirements 🛛				-	1 Presets 🕴 🧏 Parameters 🔯
× .	🛛 💓 Syste	em:soc_system Path:cl	k_0					System: soc_system Path: clk_U
Use	Connecti	Name	Description	Export	Clock	Base	End	Clock Source
V		🗆 clk_0	Clock Source					
	D	cik_in	Clock Input	clk	exported			Clock frequency
		clk_in_reset	Reset Input	reset				
	$\sim$	clk	Clock Output	Double-click to export	clk_0			Velock frequency is known
		clk_reset	Reset Output	Double-click to export				Reset synchronous edges: None 👻
V		□ onchip_memory2_0	On-Chip Memory (RAM or ROM)					
	$\phi \longrightarrow$	clk 1	Clock Input	Double-click to export	clk_0			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	۵		
	$   \downarrow \rightarrow$	reset1	Reset Input	Double-click to export	[clk1]			
V		🗆 led_pio	PIO (Parallel I/O)					
	$\bullet \longrightarrow$	clk	Clock Input	Double click to export	clk 0			

図 3-8 Clock Source の確認

\_\_\_\_8. Parameters タブの [閉じる] (× マーク) をクリックし、Parameters タブを閉じます。

Platform Designer の各コンポーネントの設定は **Parameters** タブを閉じても Platform Designer を閉じない限り 保持されます。



3-2. ステップ 2 : HPS コンポーネントの追加

HPS は、Dual-core Arm® Cortex<sup>™</sup>-A9 MPCore<sup>™</sup> プロセッサと様々なペリフェラルから構成されています。また、 以下に示す通り、インテル® SoC FPGA には、大きく分けて HPS 部と FPGA 部の 2 つのブロックから構成されま す。

このステップでは、Platform Designer システムに HPS ブロックの追加と設定を行います。この Platform Designer システム上の HPS ブロックにおいて、HPS 部の設定を行うことができます。

HPS を設定するために使用する GUI には複数のタブ (FPGA interfaces、Peripheral Pins、HPS Clocks、SDRAM) が用意されており、それぞれについて設定を行います。



図 3-9 Platform Designer システムに追加する HPS ブロック

次ページより、Platform Designer システムに HPS ブロックを追加および各種設定を行います。



\_1. IP Catalog タブの下の 検索ボックスに、processor と入力します。



図 3-10 IP Catalog の検索ボックス

\_\_\_\_ 2. Arria V/Cyclone V Hard Processor System をダブルクリックします。

このコンポーネントが HPS コンポーネントを設定するブロックです。これから設定する HPS コンポーネントのダ イアログ・ボックスが表示されます。このウィンドウは初回のみ別ウィンドウとして起動します。 [Finish] ボタンを クリック後、2 回目以降に再表示させる場合には、System Contents タブから HPS コンポーネントをダブルクリッ クしてください。

FPGA Interfaces タブではデバイス内部で接続される HPS と FPGA 間の信号の使用有無を設定することができます。設定次第で HPS 側のステータスを FPGA に通知したり、FPGA 側から HPS 側を制御したりすることができます。

ARM Cortex-A9 ARM NEON/FPU N L1 キャッシュ L1	I Cortex-A9 IEON/FPU 1 キャッシュ	USB OTG (2個)	EMAC (2個)	
L2キャッシュ		GPIO	l <sup>2</sup> C (4 個)	Ŧ
QSPI フラッシュ・ コントロール RAM	JTAG デバッグ/ トレース	SPI (4 個)	CAN (2 個)	O/I Sc
NAND フラッシュ SD / MMC	タイマ (6 個)	DMA (8チャネル)	UART (2 個)	
共有マルチボートDDR	HPS -	FPGA -	FPGA	
mm				
FPGA	・8 入ナ ・可変料 ・M10K MLAB ・fPLL	h ALM 椿度 DSP メモリおよび 6	40ビット	PFGA 乳用I

図 3-11 HPS のペリフェラルと FPGA との内部バス



\_\_3. FPGA Interfaces タブをクリックして、デフォルトで有効になっている Enable MPU standby and event signals のチ ェックを外して**無効**にします。

FPGA Interfaces Peripheral Pins HPS Clocks SDRAM	FPGA Interfaces Peripheral Pins HPS Clocks SDRAM
▼ General	* General
Enable MPU standby and event signals	Enable MPU standby and event signals
🗖 Enable general purpose signals	🗖 Enable general purpose signals
Enable Debug APB interface	Enable Debug APB interface
🗖 Enable System Trace Macrocell hardware events	Enable System Trace Macrocell hardware events
🗖 Enable FPGA Cross Trigger Interface	Enable FPGA Cross Trigger Interface
🗖 Enable FPGA Trace Port Interface Unit	n Enable FPGA Trace Port Interface Unit
🗖 Enable FPGA Trace Port Alternate FPGA Interface	🗖 Enable FPGA Trace Port Alternate FPGA Interface
Enable boot from fpga signals	Enable boot from fpga signals
Enable HLGPI Interface	Enable HLGPI Interface

図 3-12 FPGA Interface タブの設定

## i Note:

これは、マイクロ・プロセッサがスタンバイ・モードであるか、CPU がウェイク・アップ可能かを示す内部信号 です。恒久的に有効にするためこの入力信号を論理 High に接続する、もしくはプロセッサのイベントとし て接続することもできます。

\_\_\_\_4. Enable HLGPI Interface のチェックが外れて無効(デフォルト)になっていることを確認します。

i Note:

これは、SDRAM インタフェースで未使用のピン(14bit)を入力専用の汎用ピンとして使用する際のオプションです。この演習では、この信号は必要ありません。



次にHPS と FPGA 間のブリッジの設定を行います。

HPS と FPGA 間にはそれぞれがマスタ、スレーブになるポートがあります。ポート数としては HPS から FPGA へ 2 系統、FPGA から HPS へ 1 系統です。 HPS から FPGA への 2 系統ポートはそれぞれ HPS-to-FPGA interface、Lightweight HPS-to-FPGA interface です。 FPGA から HPS への1系統のポートは FPGA-to-HPS です。 すべてのポートについて、アクセスするパスに応じたバス幅の設定やポートの使用有無を設定することができま す。

Arm® プロセッサ や HPS 側の Master からアクセスする場合は、

「ブリッジのアドレス + FPGA 側のコンポーネントのオフセット・アドレス」のアドレスを指定することでアクセス することができます。ブリッジのアドレスは下記の図のように

HPS-to-FPGA interface が 0xC000\_0000

Lightweight HPS-to-FPGA interface t 0xFF20\_0000



と決まっています。

図 3-13 HPS と FPGA の内部バスと Arm から見たアドレスマップ

i Note:

HPS - FPGA 間のインタフェースに関しては、マクニカ・オンラインサービスにも資料がございますので、併 せてご参照ください。

Soc はじめてガイド - HPS-FPGA 間のアクセス方法

次ページより設定を行います。



\_ 5. AXI Bridges セクションにて、FPGA-to-HPS interface width を Unused、HPS-to-FPGA interface width を 64-bit、 Lightweight HPS-to-FPGA interface width を 32-bit に設定してください。

* AXI Bridges			* AXI Bridges	
FPGA-to-HPS interface width:	64-bit 👻		FPGA-to-HPS interface width:	Unused 👻
HPS-to-FPGA interface width:	64-bit 👻	$\Box$	HPS-to-FPGA interface width:	64-bit 👻
Lightweight HPS-to-FPGA interface width	32-bit 👻		Lightweight HPS-to-FPGA interface width	32-bit 👻



## i Note:

FPGA-to-HPS interfaces を有効にすると、FPGA 内のマスタが HPS のペリフェラルにアクセスすることがで きます。この演習では使用しません。

HPS-to-FPGA interface を有効にすると、HPS がマスタとなり FPGA のペリフェラルにアクセスすることがで きます。HPS-to-FPGA interfaces は、32/64/128 bit 幅を選択できますが、この演習では、中間の 64bit 幅 を使用します。

\_\_\_\_6. FPGA interface ページを下にスクロールすると、FPGA-to-HPS SDRAM interface、Resets および DMA Peripheral Request セクションなどさらに多くのオプションがあります。

\_\_\_\_7. FPGA to HPS SDRAM Interface が表示されるまで FPGA interface ウィンドウをスクロールします。

\_\_\_\_8. f2h\_sdram0 インタフェースをクリックし、 -- ボタンをクリックして、インタフェースを削除します。

こちらは FPGA から HPS 側の SDRAM ヘダイレクトにアクセスできる広帯域ポートです。インターコネクトと ACP(アクセラレータ・コヒーレンシ・ポート)を介さないので高速にアクセスできます。その反面、データのコヒー レンシはユーザがとる必要があります。

今回は使用しませんのでポートを削除します。

FPGA-to-HPS SDRA	M Interface		FPGA-to-HPS SDR.	AM Interface	
Click the '+' and '-' butt	ons to add and remove FPC	A-to-HPS SDRAM ports.	Click the '+' and '-' bu	ttons to add and remove Fl	PGA-to-HPS SDRAM ports.
Name	Туре	Width	Name	Туре	Width
f2h_sdram0	AXI-3	64			
+ -			+ -		

図 3-15 FPGA-to-HPS SDRAM インタフェースの設定

\_9. Resets セクションまでスクロール・ダウンします。

- \_\_\_\_10. Resets セクションでは、HPS リセットのためのすべてのオプションが無効になっていることを確認します。
- \_\_\_\_11. DMA Peripheral Request セクションでは、Enabled 列の下のすべての行が No と表示されていることを確認しま す。

i Note:

DMA peripheral request を有効にすると、HPS 側の DMA コントローラの Peripheral Request 信号を FPGA ファブリック側へ接続可能になります。 Peripheral Request 信号を利用した DMA 転送を行う場合を除き、 通常は No をセットします。

\_ 12. Interrupts セクションの、Enable FPGA-to-HPS interrupts オプションが無効になっていることを確認します。今回 は FPGA に実装したコンポーネントから Arm® プロセッサに対して割り込みは使用しません。

Resets / DMA / Interrupts の設定は以下の通りです(デフォルトから変更はありません):

T Resets					
Enable HPS-to-FPGA c	old reset output				
🔲 Enable HPS warm reset	handshake signals				
Enable FPGA-to-HPS c	lebug reset request				
🔲 Enable FPGA-to-HPS v	varm reset request				
🔲 Enable FPGA-to-HPS c	old reset request				
T DMA Peripheral Reque	st				
Peripheral Request ID	Enabled				
0	No				
2	No				
3	No				
4	No				
ļb	No				
Interrupts					
Enable FPGA-to-HPS I	nterrupts				
HPS-to-FPGA					
Enable CAN interrupt:	S				
Enable clock peripher	al interrupts				
Enable CTI interrupts					
🔲 Enable DMA interrupt	s				
Enable EMAC interrupts (for EMAC0 and EMAC1)					
Enable FPGA manager interrupt					
Enable GPIO interrupts					
Enable I2C-EMAC int	errupts (for I2C2 and I2C3)				
Enable I2C peripheral interrupts (for I2C0 and I2C1)					
🔲 Enable L4 timer interr	upts				
Enable NAND interrupt					
Enable OSC timer interrupts					
Enable Quad SPI interrupt					
Enable SD/MMC interrupt					
Enable SPI master interrupts					
Enable SPI slave interrupts					
Enable UART interrupts					
Enable USB interrupts					
Enable watchdog interrupts					

図 3-16 Resets / DMA / Interrupts の設定



3-3. ステップ 3 : HPS ペリフェラルの設定 (MAC、UART、I2C、SDIO、USB)

Peripheral Pins タブは HPS 内部にハードコーデットされている HPS ペリフェラルを有効にするタブです。

HPS のピンの多くは最大 4 つのペリフェラルで共有されています。しかしながら使用できるのは 1 つのペリ フェラルのみです。そのため、有効にするペリフェラル同士でピンが競合しないように、ピンの割り当てを指定す る必要があります。ピンの割り当ては最大 3 通りのパラメータ(HPS I/O Set 0~3)から選択することができます。



図 3-17 HPS I/O のピン・マルチプレクサ

- \_\_\_1. Peripheral Pins タブを選択します。
- 2. Ethernet Media Access Controller の EMAC1 pin を HPS I/O Set 0 に設定します。
- \_\_\_\_\_3. Ethernet Media Access Controller の EMAC1 mode を RGMII に設定します。
- 4. SD/MMC Controllerの SDIO pinを HPS I/O Set 0 に設定します。
- \_\_\_\_5. SD/MMC Controller の SDIO mode を 4-bit Data に設定します。
- \_\_\_\_6. USB Controllers の USB1 pin を HPS I/O Set 0 に設定します。
- \_\_\_\_\_7. USB Controllers の USB1 PHY interface mode を SDR with PHY clock output mode に設定します。
- \_\_\_\_\_8. SPI Controllers の SPIM1 pin を HPS I/O Set 0 に設定します。
- \_\_\_\_\_9. SPI Controllers の SPIM1 mode を Single Slave Select に設定します。
- \_\_\_\_10. UART Controllers の UARTO pin を HPS I/O Set 0 に設定します。
- \_\_\_\_ 11. UART Controllers の UARTO mode を No Flow Control に設定します。
- \_\_\_\_12. I2C Controllers の I2C0 pin を HPS I/O Set 0 に設定します。
- \_\_\_\_13. I2C Controllers の I2C0 mode を I2C に設定します。
- \_\_\_\_\_14. I2C Controllers の I2C1 pin を HPS I/O Set 0 に設定します。
- \_\_\_\_15. I2C Controllers の I2C1 mode を I2C に設定します。

設定後のパラメータは次ページを参照してください。



* Ethernet Media Acces	s Controller		Ethernet Media Acces	ss Controller
EMAC0 pin:	Unused 💌		EMACO pin:	Unused 💌
EMAC0 mode:	N/A 🔻		EMACO mode:	N/A -
EMAC1 pin:			EMAC1 pin:	HPS I/O Set 0
EMAC1 mode:			EMAC1 mode:	
2.4110 1 11000			Live of Indus.	
* NAND Flash Controlle	er		VAND Flash Controlle	er
NAND pin:	Unused 💌		NAND pin:	Unused 🔹
NAND mode:	N/A T		NAND mode:	N/A 🔻
Quad SPI Flash Contr	oller		▼ Quad SPJ Flash Contr	oller
QSPI pin:	Unused 💌		QSPI pin:	Unused
QSPI mode:	N/A 💌		QSPI mode:	N/A 💌
SD/MMC Controllor				
SDIO pin:			SDIO pin	
SDIO mode:			SDIO pin.	HPS I/O Set 0
3010 mode.	N/A 💌		SDIO mode:	4-bit Data
* USB Controllers			▼ USB Controllers	
USB0 pin:	Unused 💌		USBO pin:	
USB0 PHY interface mode	N/A T		LISBO PHY interface mode:	
USB1 pin:				
USB1 PHV interface mode			CSBI pin:	HPS I/O Set 0
CODITINI Internace mode	IN/A		USB1 PHY interface mode	SDR with PHY clock output mode 💌
* SPI Controllers			SPI Controllere	
SPIM0 pin:	Unused 🔻		SPIMO pin:	
SPIM0 mode:	N/A T		SPIMO mode:	
SPIM1 pin:	Unused			
SPIM1 mode:		۲	SPIMT pin:	HPS I/O Set 0
	N/A		SPIM1 mode:	Single Slave Select 💌
SPISU pin:	Unused 💌		SPISO pin:	Unused
SPIS0 mode:	N/A 💌		SPISO mode:	N/A 🔻
SPIS1 pin:	Unused 🔻		SPIS1 pin:	
SPIS1 mode:	N/A T		SPISt mode:	
			ST 101 11000.	
VART Controllers			- UART Controllers	
UART0 pin:	Unused 💌		UARTO pin:	HPS I/O Set 0
UART0 mode:	N/A 💌		UARTO mode:	No Flow Control
UART1 pin:	Unused 🔻		UART1 pin:	
UART1 mode:	N/A I		LIADT1 mode:	
			OARTT MODE.	
* I2C Controllers			- I2C Controllers	
I2C0 pin:	Unused 🗾		I2C0 pin:	HPS I/O Set 0
I2C0 mode:	N/A 💌		I2CO mode:	120 -
I2C1 pin:			I2C1 nin:	
I2C1 mode:			1201 pm.	HPS 1/U Set 0
19C2 pip:			1201 mode:	I2C 💌
1202 pm	Unused 🗾		I2C2 pin:	Unused 🔹
I2C2 mode:	N/A 💌		I2C2 mode:	N/A 💌
I2C3 pin:	Unused 💌		I2C3 pin:	
I2C3 mode:	N/A -		I2C3 mode:	
CAN Controllers			- CAN Controllers	
CANU pin:	Unused 💌		CANO pin:	Unused 💌
CAN0 mode:	N/A 💌		CANO mode:	N/A 🔻
CAN1 pin:	Unused 🗨		CAN1 pin:	
CAN1 mode:			OANI wada:	
			CANL MODE:	N/A 💌
Trace Port Interface	Unit		Trace Port Interface	Unit
TRACE pin:	Unused 💌		TRACE pin:	Unused
TRACE mode:	N/A -		TRACE mode	
			TRACE INDUE.	IV/A

### 図 3-18 HPS ペリフェラルの設定

Peripherals Mux Table セクションでは設定したピンの配置を確認することができます。

1 つのピンには 1 つの役割しか与えることはできません。そのため、 HPS の複数ペリフェラルが同じピンを 使用したり、同じピンに HPS ペリフェラルと GPIO の役割を与えたりすることはできません。そのため、この Peripherals Mux Table セクションで、各ピンの用途を確認してください。

左側の列はピン名を示しており、そのピンが使用されている場合は太字になります。ペリフェラルのピンとして 使用していないピンは HPS の GPIO ピンとして使用可能です。その場合はピンごとの各 GPIO のボタンを押す ことで有効になります。

ピンに対して競合が起きている場合は Message Window に Error が表示され、またそのピンの欄が赤くハイ ライトされますので、どのピンが競合を起こしているのかをリアルタイムに知ることができます。

では実際に使用していないピンを GPIO ピンに設定してみましょう。

16. Peripherals Mux Table セクションで GPIO09 をクリックすることにより、GPIO09 を有効にします。

! 反応に時間がかかる場合がありますので、何度も押さないように注意してください。

* Peripherals Mux Table						
RGMII0_TX_CLK			EMAC0.TX_CLK (Set0)	GPI000	LOANIO00	
RGMII0_TXD0		USB1.D0 (Set0)	EMAC0.TXD0 (Set0)	GPI001	LOANIO01	
RGMII0_TXD1		USB1.D1 (Set0)	EMAC0.TXD1 (Set0)	GP1002	LOANI002	
RGMIIO_TXD2		USB1.D2 (Set0)	EMAC0.TXD2 (Set0)	GP1003	LOANI003	
RGMII0_TXD3		USB1.D3 (Set0)	EMAC0.TXD3 (Set0)	GPIO04	LOANIO04	
RGMIIO_RXD0		USB1.D4 (Set0)	EMAC0.RXD0 (Set0)	GP1005	LOANI005	
RGMIIO_MDIO	I2C2.SDA (Set0)	USB1.D5 (Set0)	EMAC0 MDIO (Set0)	GPI008	LOANIO06	
RGMII0_MDC	I2C2.SCL (Set0)	USB1.D6 (Set0)	EMAC0 MDC (Set0)	GP1007	LOANI007	
RGMIIO_RX_CTL		USB1.D7 (Set0)	EMAC0.RX_CTL (Set0)	GPI008	LOANIO08	
RGMII0_TX_CTL			EMAC0.TX_CTL (Set0)	GP1009	LOANI009	
RGMIIO RX CLK		USB1.CLK (Set0)	EMACO.RX CLK (Set0)	GPIO10	LOANIO10	



* Peripherals Mux Table						
RGMII0_TX_CLK		11	EMAC0.TX_CLK (Set0)	GP1000	LOANIO00	
RGMII0_TXD0		USB1.D0 (Set0)	EMAC0.TXD0 (Set0)	GPIO01	LOANIO01	
RGMII0_TXD1		USB1.D1 (Set0)	EMAC0.TXD1 (Set0)	GPIO02	LOANIO02	
RGMII0_TXD2		USB1.D2 (Set0)	EMAC0.TXD2 (Set0)	GPIO03	LOANI003	
RGMII0_TXD3		USB1.D3 (Set0)	EMAC0.TXD3 (Set0)	GPIO04	LOANIO04	
RGMII0_RXD0		USB1.D4 (Set0)	EMACO.RXD0 (Set0)	GPIO05	LOANIO05	
RGMII0_MDIO	I2C2.SDA (Set0)	USB1.D5 (Set0)	EMACO MDIO (Set0)	GPIO08	LOANIO06	
RGMII0_MDC	I2C2.SCL (Set0)	USB1.D6 (Set0)	EMACO MDC (Set0)	GPIO07	LOANIO07	
RGMII0_RX_CTL		USB1.D7 (Set0)	EMACO.RX_CTL (Set0)	GPIO08	LOANIO08	
RGMII0_TX_CTL			EMACO.TX_CTL (Set0)	GPI009	LOANI009	
RGMII0_RX_CLK		USB1.CLK (Set0)	EMAC0.RX_CLK (Set0)	GPIO10	LOANIO10	

図 3-19 HPS GPIO09 の設定

#### 【重要:

もしクリックできなかった場合、HPS component ダイアログ・ボックスの右下にある [Finish] を選択して HPS component ダイアログ・ボックスを閉じた後、再度 hps\_0 コンポーネントをダブルクリックしてパラメータウィン ドウ を開き作業を続けてください。

もしくは、Parameters タブの "x"をクリックして閉じた後、再度 hps\_0 コンポーネントをダブルクリックしてパラ メータウィンドウ を開き作業を続けてください。



EMACD RX_CTL (Seb) EMACD RX_CTL (Seb) EMACD RX_CLK (Seb) EMACD RXD1 (Seb) EMACD RXD2 (Seb) EMACD RXD2 (Seb) EMACD RXD2 (Seb) NAND ALE (Seb) NAND CE (Seb)	6P1008 6P1009 6P1010 6P1010 6P1011	
EMACO.TX_CTL(Set0) EMAC0.RX_CLK(Set0) EMAC0.RXD1(Set0) EMAC0.RXD2 (Set0) EMAC0.RXD3 (Set0) NAND ALE (Set0) NAND AE (Set0)	GP1009 GP1010 GP1011	
EMACO RX_CLK (Sel0) EMACO RXD1 (Sel0) EMACO RXD2 (Sel0) EMACO RXD2 (Sel0) NAND ALE (Sel0) NAND ALE (Sel0)	GPI010 GPI011	MICRON MT42L128M321
EMACO.RXD1 (Set0) EMACO.RXD2 (Set0) EMACO.RXD0 (Set0) NAND.ALE (Set0) NAND.CE (Set0)	GPI011	
EMACO.RXD2 (Set0) EMACO.RXD3 (Set0) NAND.ALE (Set0) NAND.CE (Set0)		
EMACO.RXD3 (Set0) NAND.ALE (Set0) NAND.CE (Set0)	GPI012	MICRON MT42L128/0321
NAND.ALE (Set0) NAND.CE (Set0)	GPI013	WICHTAN WITH 128/110 System: Soc system: Dealer has 0
NAND.CE (Set0)	GPI014	Remove this view from the screen (Ctrl+F4)
	GPI015	Arria V/Cyclone V Hard Processor System
NAND.ULE (SetU)	GPI016	• MICRON MI47H32MI0F
NAND.RE (Set0)	GPI017	
NAND.RB (Set0)	GPI018	
NAND.DQ0 (Set0)	GPI019	MICRON MIL4/132/08-3 SPIS1 mode: N/A
NAND.DQ1 (Set0)	GPI020	MICRON MIA/HOAMIOF
NAND.DQ2 (Set0)	GPI021	MICHON M14/Hh4Mihe
NAND.DQ3 (Set0)	GPI022	UART Controllers
	Þ	HPS I/O Set 0 V
		UART0 mode: No Flow Control -
		UARTI pin: Unused 🔻
		HADTI water
		TI2C Controllers
		1200 pip
		HPS I/O Set U
		Cancel Finish

図 3-20 GPIO 設定時にクリックできなかった場合の対応方法

\_\_\_\_17. 同様に GPIO35、GPIO40、GPIO53、GPIO54、GPIO61 を有効にします。

		2 h		
QSPI_SS1			QSPI.SS1 (Set1)	GPIO35
SDMMC_CMD		USB0.D0 (Set0)	SDIO.CMD (Set0)	GPIO38
SDMMC_PWREN		USB0.D1 (Set0)	SDIO PWREN (Set0)	GPIO37
SDMMC_D0		USB0.D2 (Set0)	SDIO.D0 (Set0)	GPIO38
SDMMC_D1		USB0.D3 (Set0)	SDIO.D1 (Set0)	GPIO39
SDMMC_D4		USB0.D4 (Set0)	SDIO.D4 (Set0)	GPI040
SDMMC_D5		USB0.D5 (Set0)	SDIO.D5 (Set0)	GPIO41
SDMMC_D6		USB0.D6 (Set0)	SDIO.D6 (Set0)	GPIO42
SDMMC_D7		USB0.D7 (Set0)	SDIO.D7 (Set0)	GPIO43
SDMMC_FB_CLK_IN		USB0.CLK (Set0)	SDIO.CLK_IN (Set0)	GPIO44
SDMMC_CCLK_OUT		USB0.STP (Set0)	SDIO.CLK (Set0)	GPIO45
SDMMC_D2		USB0.DIR (Set0)	SDIO.D2 (Set0)	GPIO48
SDMMC_D3		USB0.NXT (Set0)	SDIO.D3 (Set0)	GPIO47
TRACE_CLK			TRACE.CLK (Set0)	GPIO48
TRACE_D0	UART0.RX (Set0)	SPIS0.CLK (Set0)	TRACE.D0 (Set0)	GPIO49
TRACE_D1	UART0.TX (Set0)	SPIS0.MOSI (Set0)	TRACE.D1 (Set0)	GPIO50
TRACE_D2	I2C1.SDA (Set0)	SPIS0.MISO (Set0)	TRACE.D2 (Set0)	GPIO51
TRACE_D3	12C1.SCL (Set0)	SPIS0.SS0 (Set0)	TRACE.D3 (Set0)	GPIO52
TRACE_D4	CAN1.RX (Set0)	SPIS1.CLK (Set0)	TRACE.D4 (Set0)	GPI053
TRACE_D5	CAN1.TX (Set0)	SPIS1.MOSI (Set0)	TRACE.D5 (Set0)	GPIQ54
TRACE_D6	12C0.SDA (Set0)	SPIS1.SS0 (Set0)	TRACE.D6 (Set0)	GPI055
TRACE_D7	12C0.SCL (Set0)	SPIS1.MISO (Set0)	TRACE.D7 (Set0)	GPIO56
SPIM0_CLK	UART0.CTS (Set2) (Set1) (Set0)	I2C1.SDA (Set1)	SPIM0.CLK (Set0)	GPIO57
SPIM0_MOSI	UARTO.RTS (Set2) (Set1) (Set0)	12C1.SCL (Set1)	SPIM0.MOSI (Set0)	GPIO58
SPIM0_MISO	UART1.CTS (Set0)	CAN1.RX (Set1)	SPIM0.MISO (Set0)	GPIO59
SPIM0_SS0	UART1.RTS (Set0)	CAN1.TX (Set1)	SPIM0.SS0 (Set0)	GPIO60
UART0_RX	SPIM0.SS1 (Set0)	CAN0.RX (Set0)	UART0.RX (Set1)	GP1061
UARTO TX	SPIM1.SS1 (Set0)	CAND.TX (Set0)	UART0.TX (Set1)	GPIO62

図 3-21 HPS GPIO の設定



\_18. 設定後、以下の図のようにエラーが出ていないことを確認してください。



図 3-22 エラーが無いときの表示例

例えば、<u>図 3-23</u> のようなエラーが出ている場合、SPISO と UARTO のピンの競合が起きていますので、設定に 誤りがないか確認し、修正を行ってください。

この例では、本来使用しない SPISO を使用することとなっているため、エラーになっています。設定を Unused とすると、エラーが消えます。



#### 図 3-23 エラーがあるときの表示例



3-4. ステップ 4 : HPS クロックの設定

**HPS Clocks** タブでは、Clock ソースと周波数が設定されます。これらのパラメータは、すべて Clock Manager Component で管理されます。

このタブで設定されたパラメータは、ブートローダ (Preloader ソフトウェア)の生成時に使用されます。 Preloader は「<u>演習 2: ソフトウェア演習(1) Preloader の生成</u>」で生成します。

- \_\_\_\_1. HPS Clocks タブを選択します。
- \_\_\_\_\_2. Input Clocks タブを選択します。
- 3. EOSC1 / EOSC2 clock frequency が 25MHz に設定されていることを確認します。また、すべての FPGA-to-HPS PLL Reference clocks が無効になっていることを確認します。 EOSC1 は HPS 側の専用ピンで HPS の MPU の クロックを生成するために必要なクロック・ソースです。今回使用している Atlas-SoC ボードや DE10-Nano ボード では、25MHz が入っているためこのように設定しています。

FPGA Interfaces Peripheral Pins HPS Clocks SDRAM		
Input Clocks Output Clocks		
External Clock Sources		
EOSC1 clock frequency:	25.0	ИНz
EOSC2 clock frequency:	25.0	ИНz
▼ FPGA-to-HPS PLL Reference Clocks		
Enable FPGA-to-HPS SDRAM PLL reference clock		
Enable FPGA-to-HPS peripheral PLL reference clock	ς	
FPGA-to-HPS SDRAM PLL reference clock frequency:	0.0	MHz
FPGA-to-HPS peripheral PLL reference clock frequency.	0.0	MHz
Peripheral EPGA Clocks		
EMAC0 emac0_md_clk clock frequency.	100	MHz
EMAC0 emac0_gtx_clk clock frequency	100	MHz
EMAC1 emac1_md_clk clock frequency.	100	MHz
EMAC1 emac1_gtx_clk clock frequency.	100	MHz
QSPI qspi_sclk_out clock frequency.	100	MHz
SPIM0 spim0_sclk_out clock frequency	100	MHz
SPIM1 spim1_sclk_out clock frequency	100	MHz
I2C0 i2c0_clk clock frequency.	100	MHz
I2C1 i2c1_clk clock frequency:	100	MHz
I2C2 i2c2_clk clock frequency:	100	MHz
I2C3 i2c3_clk clock frequency:	100	MHz



# 

\_\_\_\_4. Output Clocks タブを選択します。

\_\_\_5. 下図のように設定されていることを確認します(デフォルトから変更はありません)。こちらのタブでは HPS の各 ペリフェラルの動作周波数を設定することができます。設定した値に応じて PLL の設定値が自動計算されます。

FPGA Interfaces Peripheral Pins HPS Clocks SDRAM				
Input Clocks				
Clock Sources				
Peripheral PLL reference clock source:	ECSC1 clock		•	
SDMMC clock source:	Peripheral NAND	) SDMMC	) clock 📃 💌	
NAND clock source:	Peripheral NAND	) SDMMC	) clock 📃 🔽	
QSPI clock source:	Main QSPI clock		<b>v</b>	
L4 MP clock source:	Peripheral base (	clock 💌		
L4 SP clock source:	Peripheral base (	clock 💌		
Main DLL Output Clasks - Desired Freeman	!			
Default MPU clock frequency:	925.0	MHĐ	DE10-Nano の場合は、	800.0 MI
Like default MPLL clock frequency	1020.0	1011 12		
MPU clock frequency	800.0	Mee		
L3 MP clock frequency:	185.0 V MHz	DE10-N	Nano の場合は、200.0 MI	Hz
L3 SP clock frequency:	925 VIH2		しつの の 提合 (十 100.0.14)	47
Debug AT clock frequency:	25.0 V MHz			12
Debug clock frequency:	125 T MHz			
Debug trace clock frequency:	25.0 V MHz			
L4 MP clock frequency:	100.0	MHz		
L4 SP clock frequency:	100.0	MHz		
Configuration/HPS-to-FPGA user 0 clock frequency.	100.0	MHz		
Parinharal PLL Output Clocks - Desired Fr	aquencies			
SDMMC clock frequency	2000	MHH		
NAND clock frequency;	125	MHz		
QSPI clock frequency:	400.0	MHz		
EMACO clock frequency:	250.0 V MHz			
EMAC1 clock frequency:	250.0 VMHz			
USB clock frequency.	200.0	MHz		
SPI clock frequency:	200.0	MHz		
CANO clock frequency:	100.0	MHz		
CANI clock frequency:	100.0	MHz		
GPIO debounce clock frequency:	32000	Hz		
▼ HPS-to-FPGA User Clocks				
Enable HPS-to-FPGA user 0 clock				
🗖 Enable HPS-to-FPGA user 1 clock				
Enable HPS-to-FPGA user 2 clock				
HPS-to-FPGA user 0 clock frequency.	100.0	MHz		
HPS-to-FPGA user 1 clock frequency.	100.0	MHz		
HPS-to-FPGA user 2 clock frequency.	100.0	MHz		

図 3-25 HPS to FPGA Clock の設定

3-5. ステップ 5 : SDRAM の設定

SDRAM タブには、HPS 側の SDRAM コントローラおよび接続する DDR に関するパラメータを設定するオプションがあります。 SDRAM タブの内部には SDRAM 構成のためさらに 4 つのタブ (PHY Settings、Memory Parameters、Memory Timing、Board Settings) があります。

\_\_\_\_1. Arria V/Cyclone V Hard Processor System ウィンドウの下部の [Finish] をクリックします。

この操作により、HPS コンポーネントが Platform Designer システムに追加されます。(次の手順で Presets ウィンドウを表示するために必要な操作です)

# Arria V/Cyclone V Hard Processor System - hps_0				
Arria V/Cyclone V Hard Process altera_hps	or System		Documentation	
Block Diagram     Show signals     hps_0	FPGA Interfaces Peripheral Pins Hi SDRAM Protocol: DDR3 ▼ PHY Settings Memory Parameters	PS Clock SDRAM		
h2f axi clock conduit conduit h2f axi clock clock reset h2f lw axi clock clock axi axi	Clocks Memory clock frequency: Use specified frequency instea Achieved memory clock frequency; PLL reference clock frequency; Advanced PHY Settings Supply Voltage: I/O standard;	3000         MHz           d of calculated frequency         3000           32000         MHz           125.0         MHz           1.5V DDR3		
۲ III - ۲				
Warning hps 0 "Configuration/HPS-to-FPGA user 0 clock frequency" (desired cfg clk mitz) requested 1000 MHz, but only achieved 97:388421 Warning hps 0. 1 or more output clock frequencies cannot be achieved precisely, consider revising desired output clock frequencies. Warning hps 0. COT is disabled. Enabling COT (Mode Register 1) may improve signal integrity Info: hps 0. HPS peripherial PLL counter settings: n = 0 m = 39  Cancel Finish				

図 3-26 Parameters ウィンドウ表示の準備

- \_\_2. System Contents ウィンドウの HPS コンポーネントをダブルクリックして再度 HPS のオプション設定を Parameters ウィンドウ内に表示させます。
  - これは、次の手順で Preset ウィンドウを表示するために必要な操作です。



### 図 3-27 HPS パラメータ設定ウィンドウを再表示



\_3. Parameters ウィンドウの SDRAM タブをクリックします。

今回はあらかじめ用意されている Atlas-SoC ボード に載っている SDRAM の Preset を使用します。 Presets ウィンドウが表示されていることを確認します。

i Note:

**Preset** ウィンドウが表示されていない場合は、Platform Designer の **View** メニュー  $\Rightarrow$  **Presets** を選択し 表示させてください。

それでも表示されない場合は、Platform Designer の View メニュー  $\Rightarrow$  Reset to System Layout を選択後、 再度 Preset を選択してみてください。

\_\_\_\_4. Presets ウィンドウの Atlas\_HPS\_SDRAM プリセットを選択します。

Presets	
٩,	×
Project Attlas HPS_SDRAM Library ELPIDA EDJI108BASE-80 ELPIDA EDJ5308BASE-80 JEDEC DDR2-1066 256MB X8 ELPIDA DDR2 1055 510MD X8	

図 3-28 プリセットの選択

- \_\_\_\_5. Apply をクリックします。すると、Atlas\_HPS\_SDRAM が太字でハイライトされるはずです。この状態になっていれ ば設定が正しく反映されています。
- \_\_\_\_6. SDRAM タブが表示されていない場合は、SDRAM タブをクリックします。
- \_\_\_\_7. PHY Settings タブをクリックし、下図の設定となっていることを確認します。

FPGA Interfaces   Peripheral Pins   H	PS Clocks SDRAM	1		
SDRAM Protocol: DDR3 💌				
PHY Settings Memory Parameters	Memory Timing B	oard Settings		
Clocks				
Memory clock frequency.	400.0	MHz		
Use specified frequency instead of calculated frequency				
Achieved memory clock frequency:	400.0	MHz		
PLL reference clock frequency:	25.0	MHz		
Advanced PHY Settings				
Supply Voltage:	1.5V DDR3	[		
I/O standard:	SSTL-15 🔽			

図 3-29 PHY Settings の確認



\_8. Memory Parameters タブをクリックし、下図の設定となっていることを確認します。

F	FPGA Interfaces Peripheral Pins HPS Clocks SDRAM				
S	SDRAM Protocol: DDR3				
	PHY Setting Memory Parameters Mer	mory Timing Board Settings			
	Apply memory parameters from the man Apply device presets from the preset list	ufacturer data sheet t on the right.			
	Memory vendor:	Other 💌			
	Memory format:	Discrete Device 🔽			
	Memory device speed grade:	800.0 VMHz			
	Total interface width:	32			
	Number of DQS groups:	4			
	Number of chip select/depth expansion:	1 🔽			
	Number of clocks:	1 💌			
	Row address width:	15			
	Column address width:	10			
	Bank-address width:	3			
	🔽 Enable DM pins				
	🔽 DQS# Enable				

**3-30** Memory Parameters

\_\_9. Memory Initialization Options セクションまでスクロール・ダウンし、ODT Rtt nominal value に RZQ/6 が設定さ れていることを確認します。

Memory Initialization Options	
Mirror Addressing 1 per chip select:	0
🗖 Address and command parity	
Mode Register O	
Burst Length:	Burst chop 4 or 8 (on the fly) 💌
Read Burst Type:	Sequential 💌
DLL precharge power down:	DLL off
Memory CAS latency setting	7
Mode Register 1	
Output drive strength setting	RZQ/6 💌
ODT Rtt nominal value:	RZQ/6
Mode Register 2	
Auto selfrefresh method:	Manual 💌
Selfrefresh temperature:	Normal
Memory write CAS latency setting	7 💌
Dynamic ODT (Rtt_WR) value:	Dynamic ODT off

3-31 Memory Initialization Options



\_\_10. Memory Timing タブをクリックし、下図の通りの設定となっていることを確認します。

FPGA Interfaces Peripheral Pins HPS Clocks SDRAM						
SDRAM Prot	ocol: DDR3 🔻	[				
PHY Settin	PHY Settings Memory Parameters Memory Timing Board Settings					
Apply timin Apply devic	Apply timing parameters from the manufacturer data sheet Apply device presets from the preset list on the right.					
tIS (base):	base). 175 ps					
tIH (base):	250	ps				
tDS (base):	50	ps				
tDH (base):	125	ps				
tDQSQ:	120	ps				
tQH:	0.38	cycles				
tDQSCK:	400	ps				
tDQSS:	0.25	cycles				
tQSH:	0.38	cycles				
tDSH:	0.2	cycles				
tDSS:	0.2	cycles				
tINIT:	500	us				
tMRD:	4	cycles				
tRAS:	35.0	ns				
tROD:	13.75	ns				
tRP:	13.75	ns				
tREFI:	7.8	us				
tRFC:	300.0	ns				
tWR:	15.0	ns				
tWTR:	4	cycles				
tFAW:	37.5	ns				
tRRD:	7.5	ns				
tRTP:	7.5	ns				

🗷 3-32 Memory Timing



# 11. Board Settings タブをクリックし、Setup and Hold Derating セクションおよび Channel Signal Integrity セクションで、 Use Altera's default settings が選択されていることを確認します。

FPGA Interfaces Peripheral Pins HPS Clocks SDRAM					
SDRAM Protocol:					
BHV Setting Mamor Parameters Memory Timing Board Settin					
Use the Board Settings to model the board-level effects in the tir	ming analysis.				
The wizard supports single- and multi-rank configurations. Altera l effects on the output signaling of these configurations and has sto rate and the channel uncertainty within the UniPHY MegaWizard.	The wizard supports single- and multi-rank configurations. Altera has determined the effects on the output signaling of these configurations and has stored the effects on the output slew rate and the channel uncertainty within the UniPHY MegaWizard.				
These values are representative of specific Altera boards. You must change the values to account for the board level effects for your board. You can use HyperLynx or similar simulators to obtain values that are representative of your board.					
<ul> <li>Setup and Hold Derating</li> </ul>					
The slew rate of the output signals affects the setup and hold t	times of the memory device.				
You can specify the slew rate of the output signals to refer to their effect on the setup and hold times of both the address and command signals and the DQ signals, or specify the setup and hold times directly.					
Derating method:	Derating method:				
	© Specify slew rates to calculate setup and hold times				
	Specify setup and hold times directly				
CK/CK# slew rate (Differential):	2.0 V/ns				
Address and command slew rate:	1.0 V/ns				
DQS/DQS# slew rate (Differential):	2.0 V/ns				
DQ slew rate:	1.0 V/ns				
tIS:	0.325 ns				
tIH:	0.35 ns				
tDS:	0.2 ns				
tDH:	0.225 ns				
Channel Signal Integrity					
Obarral Circal Integrity	e due te interneted interference				
Channel Signal Integrity is a measure of the distortion of the eye due to intersymbol interference or crosstalk or other effects. Typically when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss as there are multiple stubs causing reflections. Please perform your channel signal integrity simulations and enter the extra channel uncertainty as compared to Altera's reference eye diagram.					
Derating Method:	OUse Altera's default settings				
	Specify channel uncertainty values				
Address and command eye reduction (setup):					

3-33 Board Settings (1)



#### \_12. Board Skew セクションまでスクロール・ダウンし、ボード・スキューが下図の通りであることを確認します。

- Oberne al Oirre al Intermiter						
Channel Signal Integrity						
Channel Signal Integrity is a measure of the distortion of the eye due to intersymbol interference or crosstalk or other effects. Typically when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss as there are multiple stubs causing reflections. Please perform your channel signal integrity simulations and enter the extra channel uncertainty as compared to Altera's reference eye diagram.						
Derating Method: © Use Altera's default settings						
	C Specify channel uncertainty values					
Address and command eye reduction (setup):	0.0	ns				
Address and command eye reduction (hold):	0.0	ns				
Write DQ eye reduction:	0.0	ns				
Write Delta DQS arrival time:	0.0	ns				
Read DQ eye reduction:	0.0	ns				
Read Delta DQS arrival time:	0.0	ns				
▼ Board Skews_						
PCB traces can have skews between them that can cause timing margins to be reduced. Furthermore skews between different ranks can further reduce the timing margin in multi–rank topologies.						
Restore default values						
Maximum CK delay to DIMM/device:	0.6	ns				
Maximum DQS delay to DIMM/device:	0.6	ns				
Minimum delay difference between CK and DQS:	-0.01	ns				
Maximum delay difference between CK and DQS:	0.01	ns				
Maximum skew within DQS group:	0.02	ns				
Maximum skew between DQS groups:	0.02	ns				
Average delay difference between DQ and DQS:	0.0	ns				
Maximum skew within address and command bus:	0.02	ns				
Average delay difference between address and command and CK	0.0	ns				

#### 3-34 Board Settings (2)

\_\_\_\_13. Platform Designer の File メニュー ⇒ Save を選択し、ここまでの手順で指定した HPS のパラメータ設定内容 を保存します。 3-6. ステップ 6 : HPS のクロックとエクスポート信号の設定

このステップでは、HPS の H2F ブリッジのクロックと LWH2F ブリッジのクロックの設定を行います。

ここで設定するクロックは、各ブリッジの FPGA 側のクロック(下図の h2f\_axi\_clk と h2f\_lw\_axi\_clk)です。 HPS 側のクロックは「<u>3-4. ステップ 4 : HPS クロックの設定</u>」で設定した I3\_main\_clk や I4\_mp\_clk となり、 これから設定するクロックとは異なります。クロックの違いは、ブリッジ内で吸収されます。

また、HPS のエクスポート信号の設定も行います。このエクスポート信号は Platform Designer 外部との通信に 使用されます。たとえば、Platform Designer と FPGA の他のロジックとの接続やピンへの配置に使用されます。



図 3-35 HPS と FPGA 間のクロック

- \_\_\_1. System Contents タブに移動します。
- \_\_\_\_2. Export 列に信号名を記述することで、 Platform Designer システムの外部に信号線を出すことができます。先ほ ど追加した HPS コンポーネントの hps\_io ポートが hps\_io という信号名でエクスポートされていることを確認し ます。
- \_\_\_\_3. 同様に HPS コンポーネントの memory ポートが memory という信号名でエクスポートされていることを確認し ます。こちらは先ほど設定した HPS 側の SDRAM の IO です。
- \_\_\_\_4. HPS コンポーネントの **h2f\_resest** をエクスポートします。 **h2f\_reset** の **Export** 列をダブルクリックし、"h2f\_reset" にリネームした後、Enter キーを押してエクスポートします。
- \_\_\_\_5. HPS 上の Clock Input インタフェース h2f\_axi\_clock の設定を行います。h2f\_axi\_clock の横の Clock 列のプル ダウン・メニューで clk\_0 を選択し、h2f\_axi\_clock に clk\_0 を接続します。
- 6. 同様に HPS 上の Clock Input インタフェース h2f\_lw\_axi\_clock の設定を行います。h2f\_lw\_axi\_clock の横の Clock 列のドロップ・ダウン・メニューで clk\_0 を選択し、h2f\_lw\_axi\_clock に clk\_0 を接続します。





3-7. ステップ 7 : HPS コンポーネントと他のコンポーネントの接続

このステップでは、Platform Designer システムに追加した HPS コンポーネントと Platform Designer システム に実装済みのコンポーネントを接続します。今回 FPGA 側は clk\_0(50MHz)で動作させるため、あらかじめ clk\_0 が各コンポーネントに接続されています。

- \_\_\_\_1. onchip\_memory2\_0 コンポーネントの Clock Input インタフェースが、 clk\_0 に接続されていることを確認します。
- \_\_\_\_2. led\_pio コンポーネントの Clock Input インタフェースが、 clk\_0 に接続されていることを確認します。
- \_\_\_\_3. dipsw\_pio コンポーネントの Clock Input インタフェースが、clk\_0 に接続されていることを確認します。
- \_\_\_\_4. button\_pio コンポーネントの Clock Input インタフェースが、 clk\_0 に接続されていることを確認します。
- \_\_\_5. onchip\_memory2\_0 の s1 を選択した後、右クリックをすることにより表示される接続サブメニューから hps\_0.h2f\_axi\_master を選択します。これにより HPS h2f\_axi\_master に、onchip\_memory2\_0 コンポーネント の s1 インタフェースが接続されます。この設定で Arm® プロセッサ から FPGA 側の onchip\_memory ヘアク セスすることができます。



図 3-37 コンポーネント間の接続

- 6. 同様に button\_pio の s1 を右クリックし、接続サブメニューから hps\_0.h2f\_lw\_axi\_master を選択します。これ により HPS h2f\_lw\_axi\_master に、 button\_pio コンポーネントの s1 インタフェースが接続されます。
  接続先が h2f\_lw\_axi\_master であることに注意してください。 続く各 PIO コンポーネントも同様です。
- \_\_\_\_7. 同様に dipsw\_pio の s1 を右クリックし、接続サブメニューから hps\_0.h2f\_lw\_axi\_master を選択します。これ により HPS h2f\_lw\_axi\_master に、dipsw\_pio コンポーネントの s1 インタフェースが接続されます。
  - \_\_8. 同様に led\_pio の s1 を右クリックし、接続サブメニューから hps\_0.h2f\_lw\_axi\_master を選択します。これに より HPS h2f\_lw\_axi\_master に、 led\_pio コンポーネントの s1 インタフェースが接続されます。



\_9. HPS コンポーネントを選択し System Contents ウィンドウの左側にある、Platform Designer ツール・バーの上下 ボタンを ヹ▽▲ヹ 使用して、HPS コンポーネントを clk\_0 の下に移動してください。

👫 System Contents 🔅 Address Map 😣 Interconnec		t Requirements 🛛							
	System: soc_system Path: hps_0h2f_lw_axi_master								
+ [	Use         Connections         Name           Image: Connections         Image: Connections         Image: Connections         Image: Connections           Image: Connections         Image: Connections         Image: Connections         Image: Connections         Image: Connections           Image: Connections         Image: Connections         Image: Connections         Image: Connections         Image: Connections           Image: Connections         Ima		Name	Description	Export	Clock	Base	End	
				⊟ clk_0	Clock Source				
X				clk_in	Clock Input	clk	exported		
				clk_in_reset	Reset Input	reset			
				clk	Clock Output	Double-click to export	clk_0		
-				clk_reset	Reset Output	Double-click to export			
-	◄			曰 🖳 hps_0	Arria V/Cyclone V Hard Processor S				
-			00	memory	Conduit	memory			
<b>•</b>			$\diamond \circ$	hps_io	Conduit	hps_io			
		$     \succ$	-0-	h2f_reset	Reset Output	h2f_reset			
		🛉	$\rightarrow$	h2f_axi_clock	Clock Input	Double-click to export	clk_0		
			+	h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_cloc		
		+	$\rightarrow$	h2f_lw_axi_clock	Clock Input	Double-click to export	clk_0		
		(	—	h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_axi_c		
	◄			onchip_memory2_0	On-Chip Memory (RAM or ROM)				
		• • • •	$\rightarrow$	clk1	Clock Input	Double-click to export	clk_0		
		• •	$\rightarrow$	sl	Avalon Memory Mapped Slave	Double-click to export	[clk1]	● 0×0000_0000	0×0000_ffff
			$\rightarrow$	reset1	Reset Input	Double-click to export	[clk1]		
	$\checkmark$			⊟ led_pio	PIO (Parallel I/O)				
			$\rightarrow$	clk	Clock Input	Double-click to export	clk_0		
			$\rightarrow$	reset	Reset Input	Double-click to export	[clk]		
			$\rightarrow$	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	● 0×0001_0040	0×0001_005f
	_		00	external_connection	Conduit	led_pio_external_conne			
	M			🗆 dipsw_pio	PIO (Parallel I/O)				
		•	$\rightarrow$	clk	Clock Input	Double-click to export	clk_0		
			$\rightarrow$	reset	Reset Input	Double-click to export			
			$\rightarrow$	sl	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0080	0×0001_008f
	_		00	external_connection	Conduit	dipsw_pio_external_con			
	M			🖯 button_pio	PIO (Parallel I/O)				
				clk	Clock Input	Double-click to export	CIK_U		
		- <u> </u>	$\square$	reset	Reset Input	Double-click to export			0 0001 00 0
		0		sl	Avaion Memory Mapped Slave	Double-click to export	[CIK]	• UxUUU1_UUcU	UXUUU1_UUct
			<u>~</u> ~	external_connection	Gonduit	button_pio_external_co			

設定完了後の Platform Designer システムは以下の通りです。

図 3-38 設定完了後の Platform Designer システム

led\_pio へのアクセスを考えてみます。

led\_pio の右から 2 つ目の Base 列を見ると 0x0001\_0040 と設定されています。これが led\_pio の Platform Designer でのオフセット・アドレスです。先の \_\_\_\_\_ 8. で設定したように led\_pio にアクセスするマスタは HPS h2f\_lw\_axi\_master です。Lightweight HPS-to-FPGA のブリッジのベース・アドレスは 0xFF20\_0000 でしたので、 この led\_pio にアクセスする場合は以下の値になります。

ブリッジのベース・アドレス(0xFF20\_0000) + Platform Designer のオフセット・アドレス (0x0001\_0040) = 0xFF21\_0040

ほかのコンポーネントも同様に考えることができ、dipsw\_pio であれば OxFF21\_0080 です。

次に、onchip\_memory へのアクセスを考えてみます。

HPS から FPGA に対してのもうひとつのパスである HPS h2f\_axi\_master ブリッジのベース・アドレスは 0xC000\_0000 でした。今回は HPS h2f\_axi\_master に接続した onchip\_memory の Platform Designer のオフセ ット・アドレスが 0x0 なので、この場合はブリッジのベース・アドレス (0xC000\_0000) がそのまま onchip\_memory にアクセスするベース・アドレスとなります。
3-8. ステップ 8 : リセットの接続とベース・アドレスの割り当て

このステップでは、リセットの一括接続とベース・アドレスの自動割り当てを行います。

- \_\_\_\_1. Platform Designer の System メニュー ⇒ Create Global Reset Network を選択し、デザインのすべてのリセット・ インタフェースを一括で接続します。
- \_\_\_\_2. 重複アドレスが存在しないように、すべてのコンポーネントのためにベース・アドレスを自動割り当てします。 System メニュー ⇒ Assign Base Addresses を選択します。



図 3-39 リセットの一括接続とベース・アドレスの自動割り当て

Assign Base Address を行っても、何も起こらなかったのではないでしょうか。

この演習では、事前に各ペリフェラルのベース・アドレスを固定してあったため、自動的にアドレスがアサインされませんでした。

図 3-40 に示すように、ベース・アドレスの横にある鍵マークを使用することにより、アドレス設定を固定できます。 クリックするごとに固定されるかどうかトグルします。アドレスを固定したい場合は、アドレス設定後に鍵マークで 固定してください。Platform Designer の Edit メニュー → Lock Base Address でも固定できます。

į.	(			ongreaer	Neser Ourpur	Double Glick to export			
L				⊡n_bps_0	Arria V/Cyclone V Hard Proce				
			00	memory	Conduit	memory			
			00	hps_io	Conduit	hps_io			
		$\succ$	-0-	h2f_reset	Reset Output	h2f_reset			
ŀ	$\vdash$			h2f_axi_clock	Clock Input	Double-click to export	clk_0		
		6		h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_cl		
ŀ				h2f_lw_axi_clock	Clock Input	Double-click to export	clk_0		
			$\prec$	h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_axi		
Π				onchip_memory2_0	On-Chip Memory (RAM or ROM)				
ł	$\vdash$		$\rightarrow$	clKl	Clock Input	Double-click to export	clk_0		
L		•	$\rightarrow$	s1	Avalon Memory Mapped Slave	Double-click to export	[clKl]	● 0x0000_0000	0x0000_ffff
		• •	$\rightarrow$	reset1	Reset Input	Double-click to export	[clKl]		
				🗆 led_pio	PIO (Parallel I/O)				
ŀ	$\vdash$		$\rightarrow$	clk	Clock Input	Double-click to export	clk_0		
	•	•	$\rightarrow$	reset	Reset Input	Double-click to export	[clk]		
		-	┢──	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0040	0x0001_005f
			00	external_connection	Conduit	led_pio_external_connec			
11									

図 3-40 ベース・アドレスの固定

3-9. ステップ 9 : Platform Designer システムの確認

 1. 設計した Platform Designer システムが以下の「<u>表 3-1 設計後の Platform Designer システムの接続状況</u>」の 通りであることを確認します。「<u>図 3-38 設定完了後の Platform Designer システム</u>」も参考にしてください。

演習用の Quartus<sup>®</sup> Prime プロジェクトとの整合性を取るため、エクスポート信号が適切にエクスポートされていること、および正しく命名されていることを確認してください。実際の設計においては任意の信号名を利用いただくことが可能です。また、コンポーネントの順序に規定はありません。

コンポーネント	ポート名	接続	
	clk_in	clk としてエクスポート	
dk 0	clk_in_reset	reset としてエクスポート	
	clk	すべてのコンポーネントに接続	
	clk_reset	hps_0 を除く、すべてのコンポーネントに接続	
led_pio	external_connection	led_pio_external_connection としてエクスポート	
dipsw_pio	external_connection	dipsw_pio_external_connection としてエクスポート	
button_pio	external_connection	button_pio_external_connection としてエクスポート	
	h2f_axi_master	onchip_memory2_0.s1 に接続	
hes 0		led_pio.s1 に接続	
nps_0	h2f_lw_axi_master	dipsw_pio.s1 に接続	
		button_pio.s1 に接続	

表	3-1	設計後の	<b>Platform Designer</b>	システムの接続状況
---	-----	------	--------------------------	-----------

\_ 2. Platform Designer の View メニュー ⇒ Device Family を選択し、

Device Family が Cyclone V になっていること、

Device が Atlas-SoC ボードの場合は 5CSEMA4U23C6、DE10-Nano ボードの場合は 5CSEBA6U23I7DK になっていることを確認します。



図 3-41 Device Family タブ



\_\_\_3. Platform Designer の View メニュー ⇒ Interconnect Requirements を選択し、

Limit interconnect pipeline stages to を 1 に設定します。段数を増やすとタイミングに余裕がでますが、同時に FPGA のロジックも大きくなります。

Clock crossing adapter type が Handshake になっていることを確認します。

File Edit System Generate	View Tools Help			
Processor	Address Map Assignments Block Symbol V- Clock Domains - Beta	ntents 🕺 Address Map 🛛 📴 Interconnect Requirer	nents 😤	
New Component Library     Processors and Periphers     Embedded Processors     Nios II (Classic)     Nios II Process     Hard Processor Comm	<ul> <li>Connections</li> <li>Details</li> <li>Device Family</li> <li>Generation Messages</li> <li>Hierarchy</li> </ul>	vide Requirements connect pipeline stages to sing adapter type: rements		
Altera HPS Em	IP Catalog	Identifier		
Altera HPS GM Altera HPS GM Altera HPS Tra	Instance Parameters     Instrumentation - Beta		Clock Limit i Enable	
<ul> <li>Arria 10 Externet</li> <li>Stratix 10 Externet</li> </ul>	6E Messages	master.master	Autom	

図 3-42 プロジェクト・パラメータの設定

3-10. ステップ 10 : Platform Designer システムの生成

完成した Platform Designer システムを生成します。

1. System Contents タブの Message ボックスに、残りのエラーがあるかどうかを確認します。エラーがある場合は、 続行する前にそれらを修正する必要があります。青字で表示される Warning に関しては、今回は無視してくだ さい。

🐉 Messag	es X	
Туре	Path	Message
EA	2 Warnings	
<u> </u>	soc_system.hps_0	"Configuration/HPS-to-FPGA user 0 clock frequency" (desired_cfg.clk_mhz) requested 100.0 MHz, but only achieved 97.368421 MHz
<u> </u>	soc_system.hps_0	1 or more output clock frequencies cannot be achieved precisely, consider revising desired output clock frequencies.
-0	5 Info Messages	
	soc_system.button_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
	soc_system.dipsw_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
	soc_system.led_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
	soc_system.hps_0	HPS Main PLL counter settings: n = 0 m = 73
	soc_system.hps_0	HPS peripherial PLL counter settings: n = 0 m = 39

図 3-43 Message ウィンドウの表示

\_\_\_\_2. File メニュー ⇒ Save を選択して、Platform Designer システムを保存します。 Save System Completed がポッ プアップされたら、[Close] します。

Info: C:/	intelfpga/ip/>>//> matched 0 files in 0	).00 seconds			
Info: Rea	ding index C:¥intelfpga¥17.1¥quartus	Sysopc builder¥builtin.ipx			
Info: C:¥	intelfpga¥17.1¥quartus¥sopc builde	r¥builtin.ipx described 83 t			
Info: C:/intelfpga/17.1/quartus/sopc_builder/**/* matched 8 files in C					
Info: Reading index C:¥intelfpga¥17.1¥quartus¥common¥librarian¥factc					
Info: C:¥	intelfpga¥17.1¥quartus¥common¥lib	orarian¥factories¥index.ir			
Info: C:/	intelfpga/17.1/quartus/common/lib	prarian/factories/**/* m			
Info: C:/	intelfpga/17.1/quartus/sopc builde	r/bin/\$IP IPX PATH mat			
Info: C:¥	intelfpga¥17.1¥quartus¥sopc_builde	r¥bin¥root_components.ir			
- C - C - C - C - C - C - C - C - C - C	intelfpga/17.1/quartus/sopc_builde	r/bin/root_components.ir *			
🕽 Info: C:/					
Info: C:/	incompany in the good too sope builde	is one root_components.it			

図 3-44 Platform Designer システムの保存

\_\_\_3. Generate メニュー ⇒ Generate HDL を選択します。

File Edit System	Generate	View	Tools	Help
🚛 TP Catalon	Generat	te HDL	···	
	Generate Testbench System			
🔍 processor	Generat	te Exar	nple De	isign 🕨
Project	Show Instantiation Template			

図 3-45 Platform Designer システムの生成



\_4. Generation ウィンドウの設定を確認し、[Generate] を実行します。

🖁 Generation	×				
<ul> <li>Synthesis</li> </ul>					
Synthesis files are used to compile th	he system in a Quartus project.				
Create HDL design files for synthesis	<sup>∞</sup> Verilog ▼				
Create timing and resource estin	nates for third-party EDA synthesis tools.				
☑ Create block symbol file (.bsf)					
<ul> <li>Simulation</li> </ul>					
The simulation model contains genera	ated HDL files for the simulator, and may include simulation-only features.				
Simulation scripts for this component	will be generated in a vendor-specific sub-directory in the specified output directory.				
Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the ip-setup-simulation and ip-make-simscript command-line utilities to compile all of the files needed for simulating all of the IP in your design.					
Create simulation model:	None 🔻				
<ul> <li>Output Directory</li> </ul>					
Path:	C:/lab/soc_lab/cv_soc_lab/soc_system				
•	4 111				
	Generate				

図 3-46 Platform Designer システムの Generate 実行画面

\_\_\_5. Platform Designer の Generate メニュー ⇒ Show Instantiation Template ではトップ・デザインにインスタンスする際に使用できるインスタンスの例が表示されます。今回はインスタンス済みですので、特に作業は発生しませんが実際に使用する場合にはとても有効です。

Instantiation Template	×
You can copy the example HDL below to declare an inst HDL Language: Verilog 💌	tance of <b>soc_system</b> .
Example HDL	
soc_system u0 ( .button_pio_external_connection_export .clk_clk .dipsw_pio_external_connection_export .h2f_reset_reset_n .hps_io_hps_io_emacl_inst_TX_CLK .hps_io_hps_io_emacl_inst_TXD0 .hps_io_hps_io_emacl_inst_TXD1 .hps_io_hps_io_emacl_inst_TXD2 .hps_io_hps_io_emacl_inst_TXD3 .hps_io_hps_io_emacl_inst_MD10	<pre>(<connected-to-button_pio_external_connection_export>), (<connected-to-clk_clk>), (<connected-to-clk_clk>), (<connected-to-h2f_reset_reset_n>), (<connected-to-h2f_reset_reset_n>), (<connected-to-h2f_reset_reset_n>), (<connected-to-h2f_reset_reset_n>), (<connected-to-h2f_reset_reset_n>), (<connected-to-h2f_reset_reset_nst_txdd>), (<connected-to-h2f_reset_reset_inst_txdd>), (<connected-to-h2f_reset_reset_inst_txd1>), (<connected-to-h2f_reset_reset_inst_txd2>), (<connected-to-h2f_reset_reset_inst_txd2>), (<connected-to-h2f_reset_res< th=""></connected-to-h2f_reset_res<></connected-to-h2f_reset_reset_inst_txd2></connected-to-h2f_reset_reset_inst_txd2></connected-to-h2f_reset_reset_inst_txd1></connected-to-h2f_reset_reset_inst_txdd></connected-to-h2f_reset_reset_nst_txdd></connected-to-h2f_reset_reset_n></connected-to-h2f_reset_reset_n></connected-to-h2f_reset_reset_n></connected-to-h2f_reset_reset_n></connected-to-h2f_reset_reset_n></connected-to-clk_clk></connected-to-clk_clk></connected-to-button_pio_external_connection_export></pre>

図 3-47 Platform Designer システムのインスタンス例



\_\_\_6. Platform Designer の生成が完了した後に、[Close] ボタンをクリックし、Platform Designer のシステム生成ダイア ログ・ボックスを閉じて Quartus<sup>®</sup> Prime に戻ります。

Info: border: "hps_io" instanti Info: error_adapter_0." avalo Info: error_adapter_0." avalo Info: soc_system: Dore "soc_s Info: qsys-generate succeeded.	ated altera_interface n_st_adapter <sup>®</sup> instant n_st_adapter <sup>®</sup> instant system <sup>®</sup> with 35 modu	_generator <sup>"</sup> boi iated error_ada; iated error_ada; es, 93 files esis	rder" oter "error_ oter "error_ [
🕕 Info: Finished: Create HDL de	esign files for synth		

図 3-48 Platform Designer システムの生成完了

- \_\_\_\_7. Quartus<sup>®</sup> Prime の Project メニュー ⇒ Add/Remove Files in Project を選択します (Settings ダイアログ・ボック スが Files カテゴリが選択された状態で表示されます)。
- \_\_\_\_8. Settings ダイアログ・ボックス内の File name フィールドの横にある 🔜 ボタンを押し、Select File ウィンドウか ら soc\_system/synthesis フォルダを参照します。



\_\_9. soc\_system.qip ファイルを選択し、[開く(O)] をクリックします。この qip ファイルは Platform Designer で Generate したコンポーネントを紐づけているファイルです。 Generate したファイルをひとつずつ登録するので はなく、こちらの qip ファイルの登録のみで、 Platform Designer システムをプロジェクトに追加することができま す。

整理▼ 新しいフォルダ・	_		•== •	
涬 お気に入り 🔶	名前	更新日時	種類	サイズ
🝌 ダウンロード	🗼 submodules	2018/02/06 17:25	ファイル フォル	
📰 デスクトップ 📰	soc_system.qip	2018/02/06 17:25	QIP ファイル	1,675 KB
딇 最近表示した場所	soc_system.v	2018/02/06 17:24	V ファイル	63 KB
ਙ Box Sync				
🍂 コンピューター				
👟 OS (C:) 👻 🗸	1	III		4
ファイ	ル名(N) soc_system.qip	-	Design Files (*.tdf *. 開く(O)	vhd *.vł ▼ ーヤンセル

図 3-49 qip ファイルの指定

### \_\_\_\_10. ファイルが追加されたことを確認します。

tegory:				Device/Boa
General	Files			
Files Libraries IP Settings IP Catalog Search Location Design Templates Operating Settings and Cond	Select the design files you oproject directory to the pro	want to include in the project. Click Add ject.	d All to add all design	files in the <u>A</u> dd Add All
Voltage Temperature Compilation Process Settings	File Name	Type	Library	Remove
EDA Tool Settings	gnrd_top.v	vernog HDL File	-	Цр

図 3-50 qip ファイルの登録

\_\_11. Settings ダイアログ・ボックスを [OK] ボタンで閉じます。

3-11. ステップ 11 : ピン・アサインメントの設定と Quartus<sup>®</sup> Prime プロジェクトのコンパイル

HPS 専用 IO に関しては、ピン配置が決まっているため基本的にピン・アサインメントはツールが自動で行い ます。例外として、SDRAM インタフェース は、ツールが生成したスクリプトを設計者が実行する必要があります。 スクリプトを実行するためには、まずネットリストを生成し、その後にスクリプトを実行します。

そのため、まずはネットリスト作成のための Analysis & Synthesis を実行後、スクリプトを実行し、再度 FPGA のコンパイルを行います。

\_\_1. Quartus<sup>®</sup> Prime の Processing メニュー ⇒ Start ⇒ Start Analysis & Synthesis を選択します。

(もしくは、GUI 上の Start Analysis & Synthesis ボタン 🔀 をクリックします)。

the Edit View Project Assignm	ins Pr	scessing Tools window Help		12.		
🗋 🔂 🖌 🗇 🛍 🏷 🤆 🛛 atlas	STOP	Stop Processing	Ctrl+Shift+C			
roject Navigator		Start Compilation	Ctrl+L			
	Entip	Analyze Current File				
Cyclone V: 5CSEMA4U23C6		Start	۲	A	Start Hierarchy Elaboration	
<ul> <li>▶ ghrd_top <sup>™</sup></li> </ul>		Update Memory Initialization File Compilation Report Ctrl+R		*	Start Analysis & Elaboration	
	÷		Ctrl+R	16.	Start Analysis & Synthesis	Ctrl+K
	Y	Dynamic Synthesis Report		1	Start Partition Merge	
🕥 Quartus Prim	e Stand	ard Edition - C:/lab/soc_lab/cv_soc	_lab/soc_system - a	atlas		
File Edit View	Projec	t Assignments Processing Tools	Window Help	_		
🗋 📂 🖶   🛩 🖸		C atlas	( 🗳 🎸 🐟 💿 🕨 I	K	• • • • •	
Project Navigator				2 -	Start Applyric & Synthesis Ctrl+V	
Project Navigator		Hi	ierarchy 🔹 🔍	No.	Start Analysis & Synthesis - Curry	

**3-51** Start Analysis & Synthesis

\_\_\_\_2. 終了後、エラーがないことを確認します。 ✔ があれば、エラーは出ていません。これでネットリストが作成されました。

File	Edit View Project Assignments Processing To	ols Window He
	🏹 🖳   🗲 🗋 💼   つ ୯ 📗 soc_system	- 🖌 🎸 🞸
Proje	ct Navigator 🔊 🔥 Hierarchy	▼QŢ₽×
	Entity:Instance	
A C)	clone V: 5CSEMA4U23C6	
🛨 - abo	ghrd_top 👛	
•		Þ
Tasks	Compilation	▼ ≡ ₽ ₽ ×
	Task	Time
	E Compile Design	
•	🖻 🕨 Analysis & Synthesis	00:03:40
	🕙 🕨 Fitter (Place & Route)	
	Assembler (Generate programming files)	
	TimeQuest Timing Analysis	
	EDA Netlist Writer	
	EDA Netlist Writer     Edit Settings	
	EDA Netlist Writer     Edit Settings     Program Device (Open Programmer)	

図 3-52 Start Analysis & Synthesis の正常終了確認



\_\_3. Quartus<sup>®</sup> Prime の Tools メニュー ⇒ TCL scripts を選択します。

□ 🗖 🛛 🗡 🛈 🛍 🤈 ୯ 🛛 atlas	Run Simulation Tool
Project Navigator	Launch Simulation Library Compiler
Entity:Ins	stance
A Cyclone V: 5CSEMA4U23C6	ImeQuest Timing Analyzer
ghrd_top	Advisors •
	🔷 Chip Planner
	Design Partition Planner
	Netlist Viewers
	🥠 Signal Tap Logic Analyzer
	<ul> <li>In-System Memory Content Editor</li> </ul>
	Logic Analyzer Interface Editor
	In-System Sources and Probes Editor
	Signal Probe Pins
	Programmer
	JTAG Chain Debugger
	Pault Injection Debugger
	System Debugging Tools
	🚺 IP Catalog
	Nios II Software Build Tools for Eclipse
•	
Tasks	/ Tcl Scripts

図 3-53 Tcl Scripts ウィンドウの起動

\_\_\_\_4. soc\_system ⇒ synthesis ⇒ submodules にある hps\_sdram\_p0\_pin\_assignments.tcl を選択し、[Run] ボタン をクリックします (反映されるまで少し時間がかかります)。

この作業により SDRAM の IO Standard の設定や OCT の設定など HPS の SDRAM Controller タブで設定した 内容が反映されます。

	W.Y.		
	synthesis		Edit
4 /	submodules		Add to Project
	hps_sdram_p0_parameters.tcl	=	ridd to rroject
	hps_sdram_p0_pin_assignments.tcl		
	hps_sdram_p0_pin_map.tcl		
# (C) 2001-2 # Your use c # software a # files from a # files), and a	017 Intel Corporation. All rights reserved of Intel Corporation's design tools, logi and tools, and its AMPP partner logic fu any of the foregoing (including device any associated documentation or infor	ved. c functions an inctions, and a programming rmation are ex	d other iny output or simulation pressly subject =

図 3-54 Tcl Script の実行



\_\_5. 完了したら [OK] ボタンをクリックします。

🕥 Qua	irtus Prime
0	Tcl Script File "C:/lab/soc_lab/cv_soc_lab/soc_system/synthesis/submodules/hps_sdram_p0_pin_assignments.tcl" executed.
	ОК

図 3-55 Tcl Script の完了

\_\_\_\_6. Tcl Scripts ウィンドウを [**Close**] します。

<ul> <li>Image: synthesis</li> <li>Image: submodules</li> </ul>		•	Edit
	Inps_sdram_p0_parameters.tcl	=	Add to Project.
	hps_sdram_p0_pin_assignments.tcl		
	hps_sdram_p0_pin_map.tcl	-	
# (C) 200 # Your u	D1-2017 Intel Corporation. All rights reserved use of Intel Corporation's design tools, logic fu are and tools, and its AMPP partner logic funct	Inctions and tions, and any ogramming of	other y output r simulation
# softwa # files fr # files), a	and any associated documentation or informa	ation are expr	essly subject -

図 3-56 Tcl Scripts ウィンドウの Close



\_7. Quartus<sup>®</sup> Prime の Processing メニュー ⇒ Start Compilation を選択(もしくは、GUI 上の Start Compilation ボ タン ▶ をクリック)し、FPGA のコンパイルを行います。このコンパイルで HW の動作イメージとなる .sof フ ァイル、そして次のソフトウェア開発に引き渡すハンドオフ・ファイルを作成します。



図 3-57 Start Compilation の実行

### \_\_\_8. コンパイルの完了を確認します。

File E	dit View Project Assignments Processing To	ols Window Help
] 🗋 🗖	▼ 🗄 🗲 🗅 🗈 🔊 ୯ 📗 soc_system	- 🖌 🗳 🗳 🚸
P <mark>roj</mark> ec	t Navigator 💧 Hierarchy	▼Q₽₽×
	Entity:Instance	
À Cyc	clone V: 5CSEMA4U23C6	
⊕ <mark>abc</mark>	ghrd_top 🛍	
•		Þ
Tasks	Compilation	▼ ≡ <u></u> ₽ ×
	Task	Time
<ul> <li>Image: A second s</li></ul>	Compile Design	00:06:23
× -	🖻 🕨 Analysis & Synthesis	00:03:44
<ul> <li>Image: A second s</li></ul>	🖻 🕨 Fitter (Place & Route)	00:01:44
<ul> <li>Image: A second s</li></ul>	Assembler (Generate programming files)	00:00:12
<ul> <li>Image: A second s</li></ul>	TimeQuest Timing Analysis	00:00:43
	🖻 🕨 EDA Netlist Writer	
	EDA Netlist Writer     Edit Settings	

図 3-58 コンパイルの完了



3-12. ステップ 12 : 出力ファイルの確認

Quartus<sup>®</sup> Prime および Platform Designer で出力したファイルを確認します。

- \_\_\_\_1. Windows<sup>®</sup> OS のエクスプローラを使用して、出力ファイルのフォルダ(下記)に移動します。 C:¥lab¥soc\_lab¥cv\_soc\_lab¥output\_files
- 2. 上記フォルダに .sof ファイルが出力されていることを確認します。<u>Atlas-SoC ボードの場合は atlas.sof</u>、<u>DE10-Nano ボードの場合は DE10-Nano.sof が出力されている事を確認してください。</u>先ほど説明したように、このファイルは FPGA の動作イメージ・ファイルです。このファイルは後の演習で Programmer というツールを使用してボード上の FPGA に書き込みます。
- 3. Windows<sup>®</sup>OS のエクスプローラを使用して、ハードウェア/ソフトウェアのハンドオフ・ディレクトリに移動します。 C:¥lab¥soc\_lab¥cv\_soc\_lab¥hps\_isw\_handoff¥soc\_system\_hps\_0

上記フォルダ以下にツールによって生成されたハードウェア・ソフトウェアのハンドオフ・ファイルがあります。 これらのファイルは Platform Designer の HPS コンポーネント画面で設定した各種データや HPS の SDRAM インタフェースの情報などの各種ファイルが格納されており、Preloader という HPS 側の初期化に使用されるフ ァイルの生成に利用します。これらのファイルを用いて後の演習で Preloader の作成を行います。

3-13. 演習 1 ハードウェア演習のまとめ

このセクションでは、以下の作業を実施し、Arm® プロセッサを含むハードウェアを構成しました。

- Platform Designer システムへの HPS コンポーネントの追加
- HPS コンポーネントの設定
- HPS コンポーネントと他のコンポーネントの接続
- Platform Designer システムの生成
- Quartus<sup>®</sup> Prime / Platform Designer が出力するファイルの確認

# 以上で 演習 1 は完了です。

# 4. <u>演習 2: ソフトウェア演習(1) Preloader の生成</u>

このセクションでは、「<u>演習 1: ハードウェア演習</u>」で作成したハンドオフ・ファイルを使用して Preloader を生成します。

Preloader は U-boot second program loader (以後、u-boot spl) をベースにインテル<sup>®</sup> SoC FPGA 向けにカスタ マイズが加えられたブートローダです。 Preloader の役割は以下の通りです。

- HPS ピン・マルチプレクスの設定
- HPS IOCSR の設定
- HPS PLL とクロックの設定
- HPS ペリフェラルのリセット解除
- SDRAM の初期化(キャリブレーションなど)
- SDRAM へ次ステージのブート・イメージの展開

上記の通り、Preloader は HPS ブロックの初期化と、U-boot や OS を SDRAM にロードする機能を提供しま す。Preloader は Quartus<sup>®</sup> Prime / Platform Designer の設計時に自動生成されるハンドオフ・ファイルを用いるこ とで自動生成されます。このため、ユーザ側で初期化用ソフトウェアの構築をすることなく Quartus<sup>®</sup> Prime / Platform Designer で設定した内容を HPS ブロックに反映することができます。先ほど確認した sof ファイルは FPGA 側の動作イメージでした。それに対して HPS 側の動作イメージがこの Preloader というファイルです。 FPGA 側、HPS 側でそれぞれ異なるファイルを使用して動作イメージを実行するという点にご注意ください。

では Preloader Generator というツールを使用し、 Preloader を作成する手順を行ってみましょう。

4-1. ステップ 1 : Embedded Command Shell の起動

\_1. SoC EDS に含まれている Embedded Command Shell 上より DS-5™ を起動します。 Embedded Command Shell は、Windows® のスタート・メニュー、または SoC EDS のインストール・フォルダ以下 に格納されている起動用スクリプト Embedded\_Command\_Shell.bat をダブルクリックして起動します。







4-2. ステップ 2 : bsp-editor (Preloader Generator)の起動

\_\_\_1. Embedded Command Shell に "bsp-editor" とタイプし、bsp-editor の GUI を起動します。

**bsp-editor** とタイプすることで Preloader Generator が起動します。



図 4-2 bsp-editor の起動

4-3. ステップ 3 : プロジェクトの作成と設定

\_1. File メニュー ⇒ New HPS BSP を選択し、プロジェクトを新規作成します。



図 4-3 新規プロジェクトの作成



\_\_\_2. Preloader settings directory: にハンドオフ・ファイルを指定します。 🔜 を押して、フォルダを指定します。

指定するフォルダは、c:¥lab¥soc\_lab¥cv\_soc\_lab¥hps\_isw\_handoff¥soc\_system\_hps\_0 です。 別のプロジェクト・ディレクトリを指定している場合は、プロジェクト・ディレクトリ以下の ¥hps\_isw\_handoff¥soc\_system\_hps\_0 です。

A New BSP	
Hardware	
Preloader settings directory:	
Software Operating system: BSD target directory:	U-Boot SPL Preloader (Cyclone V/  Version: default Use default locations
BCD Sottings Eile pages	
Additional Tcl script:	Enable Settings File relative paths     Enable Additional Tcl script
	OK. Cancel
👶 Open	vstem hns 0
最近使った デスクトップ マイドキュメ コンピューター ネットワーク Folder nam Files of typ	e: [\soc_lab\cv_soc_lab\hps_jsw_handoff\soc_system_hps_0] Qpen e: Preloader settings directory Cancel
Hardware Preloader settings directory:	C:\/ab\/soc_lab\/cv_soc_lab\/hps_jsw_handoff\/soc_system_hps_0
Software Operating system:	U-Boot SPL Preloader (Cyclone V/  Version: default
	Use default locations
BSP target directory:	C:\/ab\soc_lab\cv_soc_lab\software\spl_bsp
BSP Settings File name:	C:\Jab\soc_lab\cv_soc_lab\software\spl_bsp\settings.bsp
Additional Td script:	
	OK

### 図 4-4 ハンドオフ・ファイルの指定



\_\_3. Preloader のユーザ・オプション(Common)を確認します。本演習ではデフォルトのままで結構です。

こちらでは Preloader がどこのソースに格納されているか、また次段のソフトウェアの格納番地など、様々な設定を GUI で設定することができます。デフォルトでは SDMMC に格納されているものとして設定されており、本演習ではこの設定で行います。

	to the Record of The second test Records	
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone BSP target directory: .\	e V/Arria Version: default 🔹	
Common i -common i -spl IncLOADER_TGZ	Spl PRELOADER_TGZ: CROSS_COMPILE:	preloader/uboot-socfpga.tar.gz arm-altera-eabi-
boot     BOOT_FROM_QSPI     BOOT_FROM_SDMMC     BOOT_FROM_NAND     BOOT_FROM_RAM     QSPI_NEXT_BOOT_IMAGE     SDMMC_NEXT_BOOT_IMAGE     MAND_NEXT_BOOT_IMAGE     MAND_NEXT_MAND_NEXT_MAGE     MAND_NEXT_MAND_NEXT_MAND_NEXT_MAX     MAND_NEXT_NEXT_MAX     MAND_NEXT_MAX     MAXD_NEXT_MXD_NEXT_MXD_NXD_NXD_NXD_NXD_NXD_NXD_NXD_NXD_NXD_N	spl.boot	
-FAT_SUPPORT -FAT_BOOT_PARTITION -FAT_LOAD_PAYLOAD_NAME	QSPI_NEXT_BOOT_IMAGE: SDMMC_NEXT_BOOT_IMAGE:	0x60000 0x40000
er-Auvanceu	NAND_NEXT_BOOT_IMAGE:	0xc0000
	FAT LOAD PAYLOAD NAME	u bestime

図 4-5 ユーザ・オプションの確認

\_\_\_4. Preloader のユーザ・オプション(Advanced ⇒ spl ⇒ boot)を設定します。

「<u>演習 3: ソフトウェア演習(2) ベアメタル・アプリケーション</u>」では、ベアメタル・アプリケーションを使用するため、WATCHDOG\_ENABLE のチェックを外します。

(ベアメタル・アプリケーションでは WATCHDOG TIMER を使用できないというわけではありません。デバッグ初期 において、不用意に WATCHDOG TIMER によるリセットを発生させないために、ディセーブルにしておきます)。

BSP Editor - C:¥lab¥soc_lab¥cv_soc_lab¥sol	ution¥atlas¥software¥spl_bsp¥settings.bsp	
Main Software Packages, Drivers Linker Script, Enal	le File Generation Target BSP Directory	
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclon BSP target directory: .\	e V/Arria Version: default 🔹	
Advanced appl average assert	Spi.boot WATCHDOG_ENABLE CHECKSUM_NEXT_IMAGE EXE_ON_FPGA	
- handshake	FPGA_MAX_SIZE:	0x10000
WATCHDOG_ENABLE	FPGA_DATA_BASE:	0xffff0000
CHECKSUM_NEXT_IMAGE EXE_ON_FPGA	FPGA_DATA_MAX_SIZE:	0x10000
FPGA_DATA_BASE	STATE_REG_ENABLE	
FPGA_DATA_MAX_SIZE	BOOTROM_HANDSHAKE_CFGIO	
BOOTROM_HANDSHAKE_	WARMRST_SKIP_CFGIO	
WARMRST_SKIP_CFGIO	SDRAM_SCRUBBING	
SDRAM_SCRUB_BOOT_RE	SDRAM_SCRUB_BOOT_REGION_START:	0x1000000
	SDRAM_SCRUB_BOOT_REGION_END:	0x2000000
⊕ debug	SDRAM_SCRUB_REMAIN_REGION	
performance     *	RAMBOOT_PLLRESET	

図 4-6 Advanced 設定(1)



\_\_\_5. Preloader のユーザ・オプション(Advanced ⇒ spl ⇒ debug)を設定します。

「<u>演習 3: ソフトウェア演習(2) ベアメタル・アプリケーション</u>」では、DS-5™ のセミホスティング機能を使用する ので、SEMIHOSTING のチェックボックスを ON にします。このセミホスティング機能を使用すると UART などの コンソール入出力を DS-5™ のコンソールを用いて行うことができます。今回は DS-5™ でのデバッグを行うた め、チェックを入れますが、スタンドアローン動作 (DS-5™ などを使用せず、製品化時のように自律動作する場 合) の場合にはこちらのチェックを外した Preloader を使用してください。

ain Software Packages, Drivers, Linker Script, Enable	Efile Generation Target BSP Directory	
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone' BSP target directory: .\	V/Arria Version: default 🔹	
Advanced Advanced a reset_assert Avanced a reset_assert Construction Constructio	Spl.debug	0xffffd00 0x200

図 4-7 Advanced 設定(2)

\_\_6. [Generate] ボタンをクリックして、bsp プロジェクトを生成します。

生成完了を確認後、[Exit] ボタンをクリックし、BSP Editor を終了します。

Ele Edit Toole Hole	soc_lap+soleware+spi_psp+setting:		Ele Edit Teele Hele	soc_lab+soltware+spi_bsp+settings.bsp		
Main Software Packages Drivers Linker Solpt. En	hable File Generation. Target BSP Directory.		Main Software Packages, Drivers, Linker Script, E	nable File Generation. Target BSP Directory.		
SOPC Information file:			SOPC Information file:			
CPU name:			CPU name:			
Operating system: U-Boot SPL Preloader (Cyclo	one V/Arria Version: default 💌		Operating system: U-Boot SPL Preloader (Cycl	lone V/Arria Version: default 💌		
BSP target directory: .\			BSP target directory: .\			
- Settings - Common	spl.debug	1	E-Settings	spl.debug		
🖻 spl	DEBUG_MEMORY_WRITE		⊟spl	DEBUG_MEMORY_WRITE		
PRELOADER_TGZ CROSS_COMPILE	DEBUG_MEMORY_ADDR:	0xffffd00		DEBUG_MEMORY_ADDR:	0xffffd00	
🖻 boot	DEBUG_MEMORY_SIZE:	0x200	🖻 boot	DEBUG_MEMORY_SIZE:	0x200	
BOOT_FROM_QSPI	SEMILOSTING		BOOT_FROM_QSPI	2 SEMILOSTING		
BOOT_FROM_NAND			BOOT_FROM_SDMMC	C SEMINOSTING		
BOOT_FROM_RAM	HARDWARE_DIAGNOSTIC		BOOT_FROM_RAM	HARDWARE_DIAGNOSTIC		
QSPI_NEXT_BOOT_IMAGE	SKIP_SDRAM		QSPI_NEXT_BOOT_IMAGE	SKIP_SDRAM		
FAT_SUPPORT			FAT_SUPPORT			
-FAT_BOOT_PARTITION			FAT_BOOT_PARTITION			
FAT_LOAD_PAYLOAD_NAME			FAT_LOAD_PAYLOAD_NAME			
- spl			E-spl			
reset_assert			I reset_assert			
warm_reset_handshake			warm_reset_handshake			
# boot			i boot			
performance			E performance			
4 III +			4 III )			
Information Problems Procession			Information Problems Incompany	L		
Saarching for BSP components with ratemproved in the second	ar alamant		Trimersana: "Canaration file: C: /ab/coc lab/cv s	no lah/enftwara/enl.hen/nenerated/nimm.w.confin.cw/on	5.6.1	
Searching for BSP components with category: only Searching for BSP components with category: softy	ware package element		Tcl message: "Generating file: C:/lab/soc_lab/cv_s	soc_ab/software/sol_bsp/generated/reset_config.h.templa	te"	1
Added operating system component "spl:1.0".			Tcl message: "Generating file: C:/lab/soc_lab/cv_s	soc_lab/software/spl_bsp/generated Wakefile.template*		
O Generated file "C:\ab\soc_lab\cv_soc_lab\software	e\spl_bsp\settings.bsp*		Td message: "Generating file: C:/lab/soc_lab/cv_s	soc_lab/software/spl_bsp/generated/pll_config.h*		
			Tcl message: "Reading file: C:\lab\soc_lab\cv_soc.	_lab\hps_isw_handoff\soc_system_hps_0\soc_system_hps	0.hiof*	1
			Td message: "Generating file: C:/lab/soc_lab/cv_s	soc_lab/software/spl_bsp/generated\jocsr_config_cyclone5.	h"	8
			I Constant Constant Constant file: Collability of the form	soc_lab/coftware/spl_bsp/generated\jocsr_config_cyclone5.	c"	
			I I I I I I I I I I I I I I I I I I I	2 seconds		
		Generate Exit			Constata	Evit

### 図 4-8 bsp プロジェクトの生成

これにより Software フォルダ下の spl\_bsp に設定したデータ用のソース・ファイルが生成されます。このソー ス・ファイルと一緒に Makefile も自動生成されますので、こちらを使用して Preloader を作成します。



4-4. ステップ 4 : Preloader のビルド

- \_\_\_1. Embedded Command Shell のカレント・ディレクトリを bsp プロジェクト・ディレクトリへ移動します。
  - \$ cd "C:¥lab¥soc\_lab¥cv\_soc\_lab¥software¥spl\_bsp"

■ ~	×
Intel FPGA Embedded Command Shell	
Version 17.1 [Build 590]	
11242@HD11242A ~ \$ bsp-editor	
112420HD11242A ~ \$ cd ~C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp~	
	~

図 4-9 bsp プロジェクト・ディレクトリへの移動

\_2. make all コマンドを実行し、Preloader を生成します。

### ▲ 注意事項:

ホスト PC の OS が Windows<sup>®</sup> 10 の場合、Preloader の生成でエラーが発生する場合が確認されております。 こちらをご確認ください。



図 4-10 Preloader の生成

\_3. 実行後、エラーがなく終了したことを確認します。エラーなく終了したことを確認後 1s コマンドにて preloadermkpimage.bin が生成されていることを確認します。このファイルは、BootROM にて参照される Preloader 用の ヘッダ情報を付加したバイナリ・ファイルとなっており、SD カードや QSPI フラッシュ・メモリへ書き込むファイルと なります。

# 以上で 演習 2 は完了です。

![](_page_54_Picture_0.jpeg)

# 5. <u>演習 3: ソフトウェア演習(2) ベアメタル・アプリケーション</u>

このセクションでは、DS-5™ を使用し、SoC EDS に付属の Hello World サンプル・アプリケーションおよび本演 習用に用意された LED Blink サンプル・アプリケーションを実行し、ソフトウェアの開発手法およびデバッグ手法 について解説します。

以下にサンプル・アプリケーションの概要を記述します。

 Hello World サンプル・アプリケーションの概要 このサンプル・アプリケーションは、DS-5<sup>™</sup> が持つセミホスティング機能を使用して、デバッガ・コンソール に "Hello Tim" というメッセージを出力します。
 この方法ではデバイスのペリフェラルは使用されず、すべての通信は JTAG を通じて行われます。
 実行するアプリケーションは、64KB のオンチップ RAM にダウンロードされ実行が開始されます。このため、ボード上の SDRAM メモリの設定を必要としません。

上記の理由から、インテル® SoC FPGA が実装されたすべてのボードで実行することが可能です。

● LED Blink サンプル・アプリケーションの概要

このサンプル・アプリケーションでは、「<u>演習 1: ハードウェア演習</u>」にて作成した FPGA デザインを用い、 Arm<sup>®</sup> プロセッサから FPGA ファブリック側に実装された PIO ペリフェラルにアクセスし LED の点灯、消 灯を制御します。

このサンプル・アプリケーションはメイン・アプリケーションを実行する前に、Preloader と呼ばれる HPS ブロックの初期化ソフトウェアを実行し、SDRAM のキャリブレーション、クロックの設定、HPS-FPGA 間の ブリッジの初期化等を行います。これにより、FPGA ファブリック側のペリフェラルにアクセスすることが可 能な状態でメイン・アプリケーションを実行します。また、メイン・アプリケーションは SDRAM にロードされ 実行を開始します。

# <u>/ 注意事項</u>:

- □ この演習を行う前に、Linux<sup>®</sup>(または他の OS)が、ボード上で実行されていないことを確認してください。OS は、ベアメタル・アプリケーションのダウンロードおよびデバッグ機能を妨げる可能性があります(microSD カードが挿入されている場合は、外してください)。
- ↓ このセクションでの説明、画面スナップショットおよびコマンドは、SoC EDS の Windows<sup>®</sup> バージョン を使用して作成されたものですが、Linux ホスト PC 上でも同様の方法で実行することができます。
- このセクションで示すパスは、デフォルトのインストール・パスを使用したと仮定します。標準以外の場所が使用されている場合は、それに応じて調整してください。
- 【 ベアメタル・アプリケーションを DS-5™ でデバッグする場合、ライセンスが必要となります。ライセン スは、MAC Address に紐づけられています。 紐づけられているネットワーク・インタフェースを PC に認識させておいてください。

5-1. FPGA デザインのダウンロード

ソフトウェアの演習を開始する前に、「<u>3. 演習 1: ハードウェア演習</u>」で作成したハードウェア・デザイン (sof ファイル) を FPGA にダウンロードします。「<u>2. ボードの設定</u>」のセクションを参照し、ボードのセットアップが完 了していることを再度確認してください。セットアップに問題がなければ、 J14 に AC アダプタを接続してください。

- \_\_\_\_1. Quartus<sup>®</sup> Prime の **Tools** メニュー ⇒ **Programmer**、または **Programmer** アイコン 🌺 をクリックし、 Programmer を起動します。
- \_\_\_\_2. Programmer 内にある [Hardware Setup] ボタンをクリックし、Hardware Setup ウィンドウ内の Currently selected hardware のプルダウンリストから DE-SoC を選択し、ウィンドウを Close します。

Programmer - C:/lab/	soc_lab/cv_soc_lab/soc	c_system - atlas -	[atlas.cdf]*	
File Edit View Proces	sing Tools Window I	Help		Search altera
🔔 Hardware Setup D	E-SoC [USB-1]	Mode:	JTAG	Progress:
Enable real-time ISP to	o allow background progr	amming when avail	able	
B Start	👋 Hardware Setup			×
<sup>₽</sup> Stop	Hardware Settings	JTAG Settings		
Auto Detec	Select a programmin hardware setup appli	g hardware setup to ies only to the curre	o use when programming devices. ent programmer window.	This programming
× Delete	Currently selected ha	ardware: DE-SoC [	USB-1]	•
phi Add File	- Available hardware	items		

☑ 5-1 Hardware Setup

- \_\_\_3. [Auto Detect] ボタンをクリックし、基板上の JTAG チェインに接続されている FPGA を検出します。
- \_\_\_\_4. Select Device ウィンドウから <u>Atlas-SoC ボードの場合は 5CSEMA4</u> を、<u>DE10-Nano ボードの場合は 5CSEBA6</u> を選択し、[OK] をクリックします。

![](_page_55_Picture_10.jpeg)

図 5-2 デバイスの選択

![](_page_56_Picture_0.jpeg)

\_5. 以下のダイアログ・ボックスが表示された場合は、[Yes] を選択します。

![](_page_56_Picture_3.jpeg)

図 5-3 ダイアログ・ボックス

これにより、JTAG Chain 上に SOCVHPS と 5CSMA4/5CSEBA6 が表示されます。SOCVHPS は HPS 側、 5CSMA4/5CSEBA6 は FPGA 側が認識されたことをそれぞれ示しています。

\_\_\_\_6. ダウンロードするファイルを選択します。

Device 欄の 5CSEMA4/5CSEBA6 上で右クリックし、Change File をクリックします。 Select New Programming File ダイアログ・ボックスで、c:¥lab¥soc\_lab¥cv\_soc\_lab¥output\_files をブラウズし <u>Atlas-SoC ボードの場合は atlas.sof</u> を選択します。

![](_page_56_Figure_8.jpeg)

![](_page_56_Figure_9.jpeg)

![](_page_57_Picture_0.jpeg)

\_\_\_7. Program/Configure にチェックを入れた後、[Start] ボタンをクリックしてコンフィグレーションを行います。 右上の Progress パーが 100% になったら FPGA 側に動作イメージが書き込まれた状態となります。

Programmer <u>F</u> ile <u>E</u> dit <u>V</u> iew	- C:/lab/soc_lab/cv v P <u>r</u> ocessing <u>T</u> ools	_soc_lab/soc_s <u>W</u> indow <u>H</u> e	ystem - atla p	as - [atlas.o	cdf]*				Sear	ch alte	ra.com
Hardware So	etup) DE-SoC [USB	-1] ground program	Mo nming when a	de: JTAG available			•	Progress	5:		
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMF
Muto Detec	<none> output_files/atlas</none>	SOCVHPS 5CSEMA4U23	00000000 01A75534	<none> 01A75534</none>							
Add File		S 5CSEMA	4023								
J <sup>™</sup> Down											

![](_page_57_Picture_4.jpeg)

Programmer <u>F</u> ile <u>E</u> dit <u>V</u> iew	- C:/lab/soc_lab/cv v P <u>r</u> ocessing <u>T</u> ools	_soc_lab/soc_s <u>W</u> indow <u>H</u> e	ystem - atli p	as - [atlas.o	cdf]*				Sear	ch alte	ra.com
Hardware S	etup) DE-SoC [USB time ISP to allow back	-1] ground program	Mo nming when	ode: JTAG available			•	Progress	5: 10	0% (Su	ccessful)
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMF
Matto Detec	<none> output_files/atlas</none>	SOCVHPS 5CSEMA4U23	00000000 01A75534	<none> 01A75534</none>							
Add File Change File Save File Add Device The Up Management		S 5CSEMA	4U23								

### 図 5-5 sof のダウンロード

![](_page_58_Picture_0.jpeg)

5-2. Hello World サンプル・アプリケーションの実行

続いて HPS 上でサンプル・アプリケーションを動作させてみましょう。

はじめに、Eclipse を立ち上げます。

\_\_1. SoCEDS に含まれている Embedded Command Shell 上より DS-5™ を起動します。

Embedded Command Shell は、Windows<sup>®</sup> のスタート・メニュー、または SoC EDS のインストール・フォルダ以下 に格納されている起動用スクリプト Embedded\_Command\_Shell.bat をダブルクリックして起動します。

![](_page_58_Figure_7.jpeg)

図 5-6 Embedded Command Shell の起動

\_\_\_2. Embedded Command Shell 上で eclipse とタイプし、DS-5™ を起動します。このように Embedded Command Shell 上から起動することで、Intel<sup>®</sup> SoC FPGA Edition 用の環境変数が設定されます。

![](_page_58_Picture_10.jpeg)

![](_page_59_Picture_0.jpeg)

\_\_3. Eclipse ツールを使用するワークスペース・フォルダを設定します。

この演習では、「<u>3. 演習 1: ハードウェア演習</u>」の作業フォルダに workspace を作成します。

以下のパスを指定して [OK] をクリックします。(フォルダが存在しない場合は、自動的に作成されます)

C:¥lab¥soc\_lab¥cv\_soc\_lab¥workspace

●ワークスペース・ランチャー	×
ワークスペースの選択	
Eclipse プラットフォーム は、ワークスペースと呼ばれるフォルダにプロジェクトを保存し このセッションに使用するワークスペース・フォルダを選択してください。	します。
ワークスペース(W): C:¥lab¥soc_lab¥cv_soc_lab¥workspace ▼	参照( <u>B</u> )
■ この選切たゴフィルトトレア使用。 み後この筋頭をまニノ シロバロ	
□ この選択をナフォルトとして使用し、ラゼこの負向を衣示しない(豆)	
	OK キャンセル
ワークスペース(W): C:¥lab¥soc_lab¥cv_soc_lab¥workspace	参照(B) OK キャンセル

図 5-8 DS-5<sup>™</sup> のワークスペースの指定

\_4. DS-5™の Welcome 画面が表示されます。これは、ドキュメント、チュートリアルやビデオにアクセスするために 使用することができます。

[閉じる] (×マーク) をクリックします。

![](_page_59_Picture_10.jpeg)

続いて、Hello World サンプル・アプリケーションをインポートします。

Hello World サンプル・アプリケーションは SoC EDS に Software Example として入っています。

\_\_\_\_5. 「ファイル」メニュー⇒「インポート」を選択します。

新規(N)     Alt+シフト+N・       ファイルを聞く(.)       閉じる(C)     Ctrl+W       すべて閉じる(L)     Ctrl+シフト+W       保管(S)     Ctrl+シフト+W       マボて保管(E)     Ctrl+シフト+S       前回保管した状態に戻す(T)     移動(V)       営     和参変更(M)       野新(F)     F5       行区切り文字の変換(D)     ・       印刷(P)     Ctrl+P       ワークスペースの切り替え(W)     ・	ファ	ァイル(F) 編集(E) ソース(S) リ	リファクタリング(T)	ナビゲ-
閉じる(C)     Ctrl+W       すべて閉じる(L)     Ctrl+シフト+W       保管(S)     Ctrl+シフト+S       別名保存(A)     すべて保管(E)       すべて保管(E)     Ctrl+シフト+S       前回保管した状態に戻す(T)     移動(V)       図新(F)     F2       更新(F)     F5       行区切り文字の変換(D)     ・       印刷(P)     Ctrl+P       ワークスペースの切り替え(W)     ・		新現(N) ファイルを開く(.)	Alt+シフト+N・	• 0
保管(S)     Ctrl+S       別名保存(A)     すべて保管(E)       すべて保管(E)     Ctrl+シフト+S       前回保管した状態に戻す(T)     移動(V)       客動(V)        客動(V)     F2       更新(F)     F5       行区切り文字の変換(D)     、       印刷(P)     Ctrl+P       ワークスペースの切り替え(W)     、		閉じる(C) すべて閉じる(L)	Ctrl+W Ctrl+シフト+W	
移動(V)     F2       名前を変更(M)     F2       更新(F)     F5       行区切り文字の変換(D)     ・       印刷(P)     Ctrl+P       ワークスペースの切り替え(W)     ・		保管(S) 別名保存(A) すべて保管(E) 前回保管した状態に戻す(T)	Ctrl+S Ctrl+シフト+S	
○ 印刷(P) Ctrl+P ワークスペースの切り替え(W) →	2 81	移動(V) 名前を変更(M) 更新(F) 行区切り文字の変換(D)	F2 F5	e 8
	6)	印刷(P) ワークスペースの切り替え(W)	Ctrl+P	
		インボート(I) エンスポート(0)		J
≧ インボート(I) Ξ エシスポート(0)		プロパティ(R)	Alt+Enter	
<ul> <li>▲ インボート(I)</li> <li>▲ エンスボート(O)</li> <li>プロパティ(R)</li> <li>Alt+Enter</li> </ul>		終了/出口(X)		ロコン

図 5-10「インポート」メニュー

\_\_\_6. 「**一般」⇒「既存プロジェクトをワークスペースへ**」を選択し、[**次へ(N)**] をクリックします。

![](_page_60_Picture_8.jpeg)

# 図 5-11 既存プロジェクトのインポート

\_7. アーカイブ・ファイルの選択(A):オプションを選択します。

[参照(R)] ボタンより、サンプル・プロジェクトを指定します。

サンプル・プロジェクトは SoC EDS に含まれており、デフォルトでは以下のインストール・フォルダにあります。

 $C: \verb""" intelFPGA$17.1 \verb""" tembedded" examples \verb""" software \verb""" Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz" is a software \verb""" tempedded" examples \verb""" tempedded" examples \verb""" tempedded" examples \verb""" tempedded" examples \verb"" tempedded" examples "" tempedded"" examples "" tempedded" examples "" tempedded" examp$ 

(<SoC EDS インストール・ディレクトリ>¥examples¥software¥Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz を インポートしています)。

選択後、[終了(F)] ボタンをクリックします。

●インポート				
<b>プロジェクトのインボート</b> 既存の Eclipse プロジェクトを検索するディレクトリーを	選択します。			
○ルート・ディレクトリーの選択(I):	1 V h - d d - d V V - 6		and December Children	参照( <u>R</u> )
ーカイノ・ノアイルの選択( <u>A</u> ): C:#intelFPGA#17.	1#embedded#examples#sort	ware#Altera-SoCFPGA-Hellow	ond-Baremetal-GNU.tar.gz	●照( <u>K</u> )
フロジェクト( <u>P</u> ):	era-SoCFPGA-HelloWorld-Ba	remetal-GNU)		すべて選択( <u>S</u> ) 選択をすべて解除( <u>D</u> )
				更新(E)
オプション ジネストしたプロジェクトを検索(出) ジプロジェクトをワークスペースにコピー( <u>C</u> ) ■ ワークスペースに既に存在するプロジェクトを隠す(i)				
ワーキング・セット				
□ ワーキング・セットにプロジェクトを追加(I) ワーキング・セット(Ω):			v	選択(E)
0	< 戻る( <u>B</u> )	次へ(N) >	終了(E)	キャンセル

図 5-12 Hello World サンプル・アプリケーションの選択

この作業を実施すると、Eclipse 左側の プロジェクト・エクスプローラーにプロジェクトに含まれる各種ファイルが 表示されます。 次に、Hello World サンプル・アプリケーションをコンパイルします。

8. プロジェクト・エクスプローラ・タブよりプロジェクトを選択しハイライトします。

○C/C++ - Eclipse プラットフォーム

9. 「プロジェクト」⇒「プロジェクトのビルド」を選択します。もしくは、プロジェクト・エクスプローラー上でプロジェ クトを選択し、右クリック ⇒「プロジェクトのビルド」を実行します。

![](_page_62_Picture_5.jpeg)

図 5-13 Hello World プロジェクトのビルド

プロジェクトがコンパイルされ、上記の図に示すように、プロジェクト・エクスプローラーに hello.axf という DS-5<sup>™</sup> 上での実行可能バイナリが出力されます。コンソール・ウィンドウ上には、実行可能バイナリを生成する 際に実行されたコマンドが表示されています。

最後に、先ほど生成した Hello World サンプル・アプリケーション(hello.axf)を実行します。

\_\_10. 「**実行**」メニュー ⇒「**デバッグの構成**」を選択します。

✿ C/C++ - Eclipse プラットフォーム	_	_	
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P	) 実行	行(R) フィンドウ(W) ヘルプ(H)	
The set of th	-	Set Next Statement	Ctrl+Alt+R
La Južizoh・エクスプローラー № 日 物 マ ロ ロ C Altera CoCEDCA HalloWorld Personatal CNU	00 140	実行(R) デバッグ(D)	Ctrl+F11 F11
▲ 愛 Attra-SoccreSA-Reliovolid-barefileta-Givo ◎ 號 バイナリー ◎ 節 Includes		実行履歴(T) 実行(S) 実行構成(N)	> >
p ⊠ hello.axf - [arm/le] p ∰ hello.o - [arm/le]		デパッグ履歴(H) デパッグ(G)	•
📄 Altera-SoCFPGA-HelloWorld-Baremetal-Debuç		デバッグの構成(B)	
<ul> <li>Pello.axf.map</li> <li>Pello.axf.objdump</li> <li>Makefile</li> <li>Semihost_setup.ds</li> </ul>	00000	ブレークボイントの切り替え(K) 行ブレークボイントの切り替え(L) メソッド・ブレークボイントの切り替え(M) 監視ポイントの切り替え(W) すべてのブレークポイントをスキップ(I)	Ctrl+シフト+B Ctrl+Alt+B

図 5-14 デバッグの構成の選択

\_\_\_\_ 11. デバッグ構成ウィンドウにある左側のパネルから、

DS-5 デバッガ ⇒ Altera-SoCFPGA-HelloWorld- Baremetal-Debug を選択します (表示されない場合は、DS-5<sup>™</sup> デバッガの横にある (+) をクリックしてください)。 ターゲット接続は、インテル<sup>®</sup> FPGA ダウンロード・ケーブル (USB-Blaster<sup>™</sup>) を利用し、 Altera ⇒ Cyclone V SoC (Dual Core) ⇒ Bare Metal Debug ⇒ Debug Cortex-A9\_0 となるように設定されていま す。

\_ 12. 接続セクションの右側にある [**参照**] ボタンを押下し、USB-Blaster™ 接続の選択画面を表示させます。

● デバッグ構成	
構成の作成、管理、および実行	
	成は有効ではありません - Connection は空にできません.
	名前(N): Altera-SoCFPGA-HelloWorld-Baremetal-Debug
フィルタ入力	🦫 接続 📓 ファイル 🕸 デバッガ 🌘 OS 認識機能 № 引数 👼 環境 🛃 エクスポート
<ul> <li>■ C/C++ アブリケーション     <li>■ C/C++ アブリケーションへのアタッチ     <li>■ C/C++ ポストモーテム・デバッガー     <li>■ C/C++ リモート・アプリケーション     </li> </li></li></li></ul>	ターゲットの選択 使用する製造元、ボード、プロジェクトのタイプ、およびデバッグ操作を選択します。 現在の選択内容: Altera / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0
<ul> <li>▲ SS-5デバッガ</li> <li>◆ Altera-SoCFPGA-HelloWorld-Baremetal-Debug</li> <li>◆ IronPython Run</li> <li>● IronPython unittest</li> <li>③ Java アプリケーション</li> <li>⑤ Java アプレット</li> <li>● Jython run</li> <li>● Jython unittest</li> <li>⑥ PyDev Django</li> </ul>	プラットフォームのフィルタ            ・ Cyclone V SoC (Dual Core)         ・ Bare Metal Debug             ・ Debug Cortex-A9_0             Debug Cortex-A9_1             ターゲット接続         USB-Blaster             DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application.
& PyDev Google App Run ď Python Run ď Python unittest 및 リモート Java アプリケーション ▶ 起動グループ	接続 Bare Metal Debug Connection DTSL オプション 編集 USB-Blaster トレースまたはその他のターゲット オプションを構成します。"
フィルターー致: 18 / 18 項目 ⑦	syleryを届したくい思いにより(ビ)     JBJH(エ)     デバッグ(D)     閉じる

### 図 5-15 デバッグの構成

![](_page_64_Picture_0.jpeg)

\_\_13. 接続ブラウザ・ウィンドウで、目的の USB-Blaster™(この例では DE-SoC on localhost)をハイライトして、[**選択**] をクリックします。

● 接続ブラウザ			×
接続ブラウサ	2		
ターゲット接続	売を選択します		
DE-SAC USB-			_
DE-SoC on I	ocalhost [USB-1]		
?	選択	キャンセル	
_			

図 5-16 デバッグ・ケーブルの選択

\_\_\_\_14. デバッグ構成ウィンドウの右下にある [デバッグ(D)] ボタンをクリックします。

●デバッグ構成	
構成の作成、管理、および実行	1
構成の作成、管理、および実行	名前(N): Altera-SoCFPGA-HelloWorld-Baremetal-Debug ● 接続
💞 IronPython unittest 🗊 Java アプリケーション 🔄 Java アプレット 🔊 Jython run 🗳 Jython unittest 🗊 PyDev Django	Debug Cortex-A9_0     ・       Debug Cortex-A9_1     ・       ターゲット接続     USB-Blaster
& PyDev Google App Run ● Python Run ● Python unittest 型 リモート Java アプリケーション ▶ 起動グループ	接続 Bare Metal Debug Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1 DTSL オプション 編集 USB-Blaster トレースまたはその他のターゲット オプションを構成します。" ・
フィルター一致: 18 / 18 項目 ⑦	前回保管した状態に戻す(⊻) 適用(Y) デパッグ(D) 閉じる

図 5-17 デバッグの実行

![](_page_65_Picture_0.jpeg)

\_15. Eclipse は、デバッグ パースペクティブに切り替えるかどうかを尋ねます。 [**はい(Y)**] をクリックしてそれを受け 入れてください。

○1(->	マペクティブスイッチの確認	×
?	この 起動 は DS-5 デバッグ パースペクティブに関連付けられています。	
	このパースペクティブを今すぐ開きますか?	
■常(	にこの設定を使用する( <u>R</u> )	
		J

![](_page_65_Figure_4.jpeg)

### Windows ファイアウォールの警告が出た場合は、[アクセスを許可する(A)]をクリックします。

🚔 Windows セキュリティの重要な警告	🔐 Windows セキュリティの重要な警告
ごのプログラムの機能のいくつかが Windows ファイアウォールでブロックされています	このプログラムの機能のいくつかが Windows ファイアウォールでブロックされています
すべてのパブリック ネットワークとブライベート ネットワークで、Windows ファイアウォールにより vstrm_serverd_rddlexe の 機能の(X/つかがブロックとれています。 名前(N): vstrm_serverd_rddlexe 発行元(P): 不明 パス(日): C:¥altera¥14.0¥embedded¥ds=5¥sw¥debughw¥debug_server Vstrm_serverd_rddlexe (これらのネットワーク」とでの通信を許可する: 「 ブライベート ネットワーク (ホーム ネットワークや社内ネットワークなどXP) 「 パブリック ネットワーク (空港、喫茶店など) (非推奨/U) (このようなネットワーク」付き(の通信をおつする。	すべてのパブリック ネットワークとプライベート ネットワークで、Windows ファイアウオールにより eclipse exe の 機能のいくつかプロックされています。 名前(N): solipse exe 発行(兄P): 不明 パス(出): C:ValteraV14.0VembeddedVds-5VewVeclipseVeclipse exe eclipse exe にこれらのネットワーク上での通信を許可する: 「 プライベート ネットワーク (ホーム ネットワークや社内ネットワークなどXE) 「 パブリック ネットワーク (空港、喫茶店など)(非推奨)(U) 「 パブリック ネットワーク」(会会、学茶ンF7-6が低いかせる2.0F7-6がせる2.0F7-6が低いかせる2.0F7-6がほう2.0F7-6が低いかせる2.0F7-6がG0-6-77-77-77-77-77-77-77-77-77-77-77-77-7
プログラムにファイアウォールの経由を許可することの危険性の詳細	プログラムにファイアウォールの経由を許可することの危険性の詳細
	アクセスを許可する( <u>A</u> ) キャンセル

図 5-19 セキュリティの警告

### i Note:

ダウンロード時にエラーが発生した場合は、以下の確認を行ってください。

- (1) DS-5™ のライセンスが紐づけられているネットワーク・インタフェース(例えば USB-Ethernet Interface アダプタ)が有効になっているか確認してください。
- (2) 評価ボードの電源入切および PC の再起動で復旧しないか確認してください。評価ボードの電源を切った場合は、再度 FPGA のデータをダウンロードすることを忘れないでください。

デバッガは起動スクリプトの指示に従いセミホスティング機能を有効にした後、JTAG を経由してアプリケーショ ンをボードにダウンロードします。プログラム・カウンタ が main 関数に到達するとブレークされデバッグが開始 出来る状態となります。この段階では、DS-5™ のすべてのデバッグ機能を使用することができます(レジスタや 変数の表示と編集、逆アセンブリ・コードの参照、など)。

![](_page_66_Picture_0.jpeg)

\_\_\_\_16. 緑色の Continue ボタン 🕟 をクリック (または F8 キーを押して) アプリケーションを実行します。これによ

り、アプリケーション・コンソール に Hello Tim メッセージを表示します

✿ DS-5 デバッグ - Altera-SoCFPGA-HelloWorld-Ba	remetal-GNU/hello.c - Eclipse プラットフォーム	
ファイル(E) 編集(E) ソース(S) リファクタリング	(I) ナビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) 実行( <u>R</u> ) ウィンドウ	( <u>W</u> ) ヘルプ( <u>H</u> )
📸 🕶 📾 🐘 🏶 🕶 🔕 🗣 🧶 🖉 🗾	1 ★ 图 ★ ★ ★ ★ ★ ★	クイック・アクセス 🛛 🖻 🔜 🖏
<ul> <li>◆デパッ… ※ ●プロジ… 通りモー… □</li> <li>● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●</li></ul>	<ul> <li>■ コマンド ※ ■ 履歴 ※ スクリプト ● ● ■ ■ ● ジ マ 巻 □</li> <li>● リンク済み: Altera-SoCFPGA-HelloWorld-Baremetal-Debug next</li> <li>SVC モード S:0xFFFF32FC で実行停止で実行が停止しました cs3_premain 内 (デバッグ情報はありません)</li> <li>S:0xFFFF32FC LDR r3,[pc,#16]; [0xFFFF3314]</li> <li>●</li> </ul>	<ul> <li>●・変数 22 ●・ブ 亜レ 学式 60 関数 □ □</li> <li>◎ (************************************</li></ul>
ステータス: 接続	コマンド:コンテンツアシストを表示するには、Ctrl+スペース オ送信	変数の追加 <b>参照…</b>
le hello.c ⊠		Ⅲ逆 № 恒メ ■1 回イ 計ア □□
<pre>2* * Copyright Aftera 2013,2014[ 7 8 #include <stdio.h> 9 10 /* enable semihosting with gcc by def 11 intauto_semihosting; 12 130 int main(int argc, char** argv) { 14     printf("Hello Tim\n"); 15     return 0; 16 } 17 </stdio.h></pre>	<pre>fining anauto_semihosting symbol */ *</pre>	<ul> <li>ク済み: Altera-SoCFPGA-HelloWorld-Baremetal-E</li> <li>③ ● &lt;次の命令&gt; 100</li></ul>

図 5-20 Hello Tim の表示

- \_\_\_\_17. 接続の切断ボタン 🧏 をクリックし CPU との接続を切断します。
- \_\_\_\_18. 画面右上のパースペクティブボタン

🖹 🖥 🏘 をクリックし編集画面に戻ります。

![](_page_67_Picture_0.jpeg)

5-3. LED Blink サンプル・アプリケーションの実行

Hello World サンプル・アプリケーションと同様に事前に用意された LED Blink サンプル・アプリケーションを DS-5™ にインポートします。

\_\_\_\_1. 「**ファイル**」メニュー ⇒「インポート」を選択します。

![](_page_67_Picture_5.jpeg)

図 5-21「インポート」メニュー

\_\_2. 「**一般」⇒「既存プロジェクトをワークスペースへ**」を選択し、[**次へ(N)**] をクリックします。

![](_page_67_Picture_8.jpeg)

図 5-22 既存プロジェクトのインポート

![](_page_68_Picture_0.jpeg)

\_3. アーカイブ・ファイルの選択(A)オプションを選択します。

[参照(R)] ボタンより、以下のサンプル・プロジェクトを指定します。

C:¥lab¥soc\_lab¥cv\_soc\_lab¥software\_example¥Atlas-Blinking-LED-Baremetal-GNU.tar.gz

i Note:

これはツールのインストール・ディレクトリではなく 演習データのディレクトリ以下である ことに注意してく ださい。

選択後、[終了(F)] ボタンを押します。

●インポート				- • ×
<b>プロジェクトのインポート</b> 既存の Eclipse プロジェクトを検索する	ディレクトリーを選択します。			
◎ルート・ディレクトリーの選択( <u>工</u> ):				▼ 参照( <u>R</u> )
「アーカイブ・ファイルの選択(A):     「         の         の         の	C:¥lab¥soc_lab¥cv_soc_lab¥s	oftware_example¥Atlas-Blink	ing-LED-Baremetal-GNU.tar.gz	▼ 参照( <u>R</u> )
プロジェクト( <u>P</u> ):				
Atlas-Blinking-LED-Baremetal-G	NU(Atlas-Blinking-LED <mark>-</mark> Barer	metal-GNU)		すべて選択( <u>S</u> )
				選択をすべて解除( <u>D</u> )
				更新(E)
オプション				
<ul> <li>✓ ポストしたノロシェクトを検索(ロ)</li> <li>✓ プロジェクトをワークスペースにコピ</li> </ul>	-( <u>C</u> )			
□ ワークスペースに既に存在するプロジ	ェクトを隠す <u>(i</u> )			
ワーキング・セット				
ワーキング・セットにプロジェクトを	追加( <u>T</u> )			
ワーキング・セット( <u>0</u> ):			•	選択( <u>E</u> )
0	< 戻る( <u>B</u> )	次へ(N) >	終了( <u>E</u> )	キャンセル

図 5-23 LED Blink サンプル・アプリケーションの選択

この作業を実施すると、Eclipse 左側の プロジェクト・エクスプローラーにプロジェクトに含まれる各種ファイル が表示されます。 次に、LED Blink サンプル・アプリケーションをコンパイルします。

- \_\_\_\_4. プロジェクト・エクスプローラ・タブより Atlas-Blinking-LED-Baremetal-GNU プロジェクトを選択しハイライトします。
- \_\_\_\_5. 「プロジェクト」メニュー ⇒ 「プロジェクトのビルド」を選択します。もしくは、プロジェクト・エクスプローラー上で プロジェクトを選択し、右クリック ⇒ 「プロジェクトのビルド」を実行します。

![](_page_69_Figure_5.jpeg)

図 5-24 LED Blink サンプル・アプリケーションのビルド

最後に、LED Blink サンプル・アプリケーションを実行します。

- \_\_\_\_6. 「**実行**」メニュー ⇒「**デバッグの構成**」を選択します。サンプル・プロジェクトには、Atlas-SoC ボード上で実行 するための事前設定を付属しています。
- \_\_\_\_7. デバッグ構成ウィンドウにある左側のパネルから、

**DS-5 デバッガ**  $\rightarrow$  Atlas-Blinking-LED-Baremetal-Debug を選択します (表示されない場合は、DS-5<sup>M</sup> デバッガの 横にある (+) をクリックしてください)。

ターゲット接続は、USB-Blaster™ を利用し、

Altera  $\Rightarrow$  Cyclone V SoC (Dual Core)  $\Rightarrow$  Bare Metal Debug  $\Rightarrow$  Debug Cortex-A9\_0 となるように既に設定されています。

● デバッグ構成	×
構成の作成、管理、および実行	- A A A A A A A A A A A A A A A A A A A
C // レタスカ     C // レタスカ     C // レタスカ     C // レタスカ     C // レキ アプリケーションへのアタッチ     C // レキ アプリケーションへのアタッチ     C // レキ パストモーテム・デパリガー     C // レキート・アプリケーション     A las-Binking-LED-Baremetal-Debug     Atlas-Binking-LED-Baremetal-Debug     V IronPython Run     Y IronPython nultest     Java アプリケーション     Java アプリケーション     Jython run     Y Jython run     Y Jython Run     PyDev Google App Run     PyDev Google App Run     Y Python Run     Y Pytho	名前(M): Atlas-Blinking-LED-Baremetal-Debug ◆ 技統 @ ファイル ◆ デパッガ ● OS 認識機能 ● 引数 ■ 環境 △ エクスポート ターグットの選択 使用する製造元、ポード、プロジェクトのタイプ、およびデパッグ操作を選択します。現在の選択内容: Altera / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0 ブラットフォームのフィルタ Cyclone V SoC (Dual Core) ▲ Bare Metal Debug Debug Cortex-A9_0 Debug Cortex-A9_1 Debug Cortex-A9_1 Debug Cortex-A9_1 Debug Cortex-A92 SMP Ø-グット技装 USB-Blaster DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application. 技統 Bare Metal Debug Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1 使照 DTSL オブション 編集 USB-Blaster トレースまたはその他のターグット オブションを構成します。"default" コンフィキ * 前回保管した状態に戻す(Y) 道用(Y)
0	デバッグ( <u>D</u> ) 閉じる

図 5-25 LED Blink サンプル・アプリケーションのデバッグ構成

\_8. 以下の確認ポップアップが出た場合は、[はい] を選択してください。

<b>(</b> = 読d	か取り専用ファイルが見つかりました
?	ファイル '/Atlas-Blinking-LED-Baremetal-GNU/Atlas-Blinking-LED-Baremetal -Debuglaunch'は読み取り専用です。書き込み可能にしますか?
	【 ほい(文) 」 いいえ(N)

![](_page_70_Figure_12.jpeg)

- \_\_\_9. 接続セクションの右側にある [**参照**] ボタンを押下し、USB-Blaster™ 接続を選択します。
- \_\_\_\_10. 接続ブラウザ・ウィンドウで、目的の USB-Blaster™(この例では DE-SoC on localhost)をハイライトして、[**選択**] をクリックします。

●接続ブラウザ		×
接続ブラウザ		
ターゲット接続を選択し	)ます	
DE CACHICE 1		
DE-SoC on localhost	[USB-1]	
2	選択	キャンセル

図 5-27 デバッグ・ケーブルの選択

\_\_\_\_11. デバッグ構成ウィンドウの右下にある [デバッグ(D)] ボタンをクリックします。

ま成の作成 管理 お上び宇行	÷.
病成のTF成、目生、10よい天1]	3
	名前(N): Atlas-Blinking-LED-Baremetal-Debug
<ul> <li>イルタ入力</li> <li>C /C++ アプリケーション</li> <li>C /C++ アプリケーションへのアタッチ</li> <li>C /C++ アプリケーションへのアタッチ</li> <li>C /C++ リモート・アプリケーション</li> <li>DS-5デパッガ</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Plinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Atlas-Blinking-LED-Baremetal-Debug</li> <li>Y IronPython nunttest</li> <li>Java アプリケーション</li> <li>Java アプレット</li> <li>Jython nuntest</li> <li>PyDev Django</li> <li>PyDev Django</li> <li>PyDev Dogle App Run</li> <li>Python Run</li> <li>Python nunttest</li> </ul>	<ul> <li>◆ 接続 画 ファイル ◆ デバッガ ● OS 認識機能 ↔ 引数 ■ 環境 △ エクスポート</li> <li>ターゲットの選択 使用する製造元、ポード、プロジェクトのタイプ、およびデバッグ操作を選択します。現在の選択内容: Altera / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0</li> <li>ブラットフォームのフィルタ</li> <li>Cyclone V SoC (Dual Core)</li> <li>▲ Bare Metal Debug</li> <li>Debug Cortex-A9_1</li> <li>Debug Cortex-A9_1</li> <li>Debug Cortex-A9_2 SMP</li> <li>ターゲット接続</li> <li>USB-Blaster</li> <li>DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application.</li> <li>接続</li> <li>Bare Metal Debug</li> <li>Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1</li> </ul>
<ul> <li>■ 起動グループ</li> <li>(10.400 月) (10.10 円)</li> </ul>	DTSL オプション 編集 USB-Blaster トレースまたはその他のターゲット オプションを構成します。"default" コンフィキ 前回保管した状態に戻す(火) 適用(火)

図 5-28 LED Blink サンプル・アプリケーションのデバッグ


12. Eclipse は、デバッグ パースペクティブに切り替えるかどうかを尋ねます。 [**はい(Y)**] をクリックしてそれを受け入 れてください。





Windows ファイアウォールの警告が出た場合は、[アクセスを許可する(A)]をクリックします。



図 5-30 セキュリティの警告

### i Note:

ダウンロード時にエラーが発生した場合は、以下の確認を行ってください。

- (1) DS-5™ のライセンスが紐づけられているネットワーク・インタフェース (例えば USB-Ethernet Interface アダプタ) が有効になっているか確認してください。
- (2) 評価ボードの電源入切および PC の再起動で復旧しないか確認してください。評価ボードの電源を切った場合は、再度 FPGA のデータをダウンロードすることを忘れないでください。



\_\_13. ブレークポイントを設定します。

atlas\_main.c の 22 行目にブレークポイントを設定します。行数表示の左横のスペースをダブルクリックすることで設定可能です。

ファイル(E) 編集(E) ソース(S) リファクタリング(I) ナビグート(M) 検索(A) プロジェクト(P) 実行(B) ウィンドウ(W) ヘルブ(H)         マーロス (A)	● DS-5 デバッグ - Atlas-Blinking-LED-Baremetal-GNU/a	tlas_main.c - Eclipse プラットフォーム	
マール       クイック・アクセス       第1回後         マール       クイック・アクセス       第1回後         マール       アフ/ドッグ	ファイル(E) 編集(E) ソース(S) リファクタリング(I) :	+ビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) 実行( <u>R</u> ) ウィンドウ( <u>W</u> ) ヘルプ( <u>H</u> )	
◆デパツグ 3 レブロジェ 書リモート □ ■ コマンド 3 ■ 滞電 多 スクリブト 単晶 副 夢 多 ◆ ● □ ● 全 な な 本 論   キ ◆ e) ◆ ご ◆   ▶ Ⅲ 3, 0 ● は as _main.c. (行 22 * Correst-A9_0 # 1 プレークポイントで停止 (SVC) * Correst-A9_0 # 1 プレークポイントで停止 (SVC) * Correst-A9_0 # 1 プレークポイントで停止 (SVC) * * Atlas-Blinking-LED-Baremetal-Debug 提続 ■ Cortex-A9_0 # 1 プレークポイントで停止 (SVC) * * Atlas-Blinking-LED-Baremetal-Debug 提続 ■ Cortex-A9_0 # 1 プレークポイントで停止 (SVC) * * * Correst-A9_0 # 1 プレークポイントで停止 (SVC) * * * Correst-A9_0 # 1 プレークポイントで停止 (SVC) * * * * Correst-A9_0 # 1 プレークポイントで停止 (SVC) * * * Correst-A9_0 # 1 プレークポイントで停止 (SVC) * * * * * * * * * * * * * * * * * * *		Σ × t+ φ × + ×	クイック・アクセス 🔢 🗟 🖉
	<pre> * デバッグ ※ 「フロジエ ヨリモート □ □ * デバッグ ※ 「フロジエ ヨリモート □ □ * 私tas-Binking-LED-Baremetal-Debug 切 * Atas-Binking-LED-Baremetal-Debug 投 # Cortex-A9_0 #1 ブレークポイントで停止 (SVC)  Zテータス: 接続 * Cortex-A9_0 #1 ブレークポイントで停止 (SVC)  Zテータス: 接続 * Copyright Altera 2013 7 8 #include 'sscal.h" 10 11 #define LED_BASE_ADDR (0xFF210040) 12 13= int main(int argc, char** argv) *14 { 15 int i; 16 17 printf("Hello from Atlas. \n"); 18 19 while(1) 20 { for(i=0; i &lt; 16; i++){ 31 L.write_word(LED_BASE_ADDR, printf("LED [%x] \n",i); 24 } ************************************</pre>	<ul> <li>□マンド※ ● 履歴 多スクリプト ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●</li></ul>	

#### 図 5-31 ブレークポイントの設定

- \_\_\_\_14. 緑色の Continue ボタン をクリックして(または F8 キーを押して)アプリケーションを実行します。これに より、アプリケーション・コンソール に Hello from Atlas. メッセージ が表示されます。
- \_\_\_\_15. もう 2 回、緑色の Continue ボタン 🕨 をクリックして(または F8 キーを押して)アプリケーションを実行し

ます。これにより、アプリケーション・コンソール に LED [0] メッセージ が表示され、Atlas-SoC ボード上のユー ザ LED ( LED [3:0] )の点灯状態が変化することを確認します。

- \_\_\_\_16. さらに ▶ を押すごとに LED の状態が変化することを確認してください。
- \_\_\_\_17. 接続の切断ボタン 🔯 をクリックし CPU との接続を切断します。

#### 以上で 演習 3 は完了です。 お疲れ様でした。

次のページ以降にオプション演習があります。時間がある方は、こちらも実施してみてください。

5-4. 演習 2 で作成した Preloader による初期化(オプション演習)

演習 3 では、前もって準備されていた Preloader を使用して HPS を初期化していました。

ここでは、「<u>演習 2: ソフトウェア演習(1) Preloader の生成</u>」で作成した Preloader にて HPS の初期化を実施します。

\_\_\_\_1. 演習 2 で Preloader イメージが作成されていることを確認します。

Preloader は、C:¥lab¥soc\_lab¥cv\_soc\_lab¥software¥spl\_bsp¥uboot-socfpga¥spl ディレクトリの下に、"u-boot-spl" という名で作成されているはずです。このファイルが生成されていることを確認してください。

もし、生成されていない場合は、再度 演習 2 を実施してください。

	- 📜 🕨 コンピューター	- OS (C:) ▶ lab ▶	soc_lab ▶ cv_soc_l	ab 🕨 software 🕨 s	spl_bsp ▶ ubc
整理▼	] 開く 新しい	フォルダー			
1 44	夕前	^	<b>审新</b> 口時	種類	サイズ
关 223	נמוח.		X-WI LINO	1±AA	
しょう ち	🐌 arch		2018/02/13 09:01	ファイル フォル	
🥅 デ	👠 board		2018/02/13 09:01	ファイル フォル	
🛬 ธ	👢 common		2018/02/13 09:02	ファイル フォル	
B	👠 drivers		2018/02/13 09:02	ファイル フォル	
	👢 lib		2018/02/13 09:02	ファイル フォル	
Ma =.	👢 spl		2018/02/13 09:01	ファイル フォル	
	depend		2018/02/13 09:01	DEPEND ファイル	0 KB
-	.gitignore		2017/10/27 23:35	GITIGNORE 7	1 KB
	Makefile		2017/10/27 23:35	ファイル	6 KB
0 📣	u-boot.lst		2018/02/13 09:09	WDExpress.lst.1	0 KB
SD SI	u-boot-spl		2018/02/13 09:10	ファイル	574 KB
्र pı	u-boot-spl.bin		2018/02/13 09:10	BIN ファイル	36 KB
🛫 El	u-boot-spl.lds		2018/02/13 09:09	LDS ファイル	1 KB
SP CI	u-boot-spl.map		2018/02/13 09:10	WDExpress.map	90 KB
-					

図 5-32 u-boot-spl ファイルの確認

### i Note:

ここで確認した、"u-boot-spl"ファイルは、Arm® Executable and Linkable Format (ELF)ファイルです。DS-5™の初期化スクリプトにて読み出され、ユーザ・アプリケーション実行前に実行されています。

詳細は、「soc はじめてガイド - DS-5 によるベアメタル・アプリケーション・デバッグ」の「カスタム・ボード への対応方法」をご参照ください。

Soc はじめてガイド - DS-5 によるベアメタル・アプリケーション・デバッグ

\_\_\_2. 演習 3 で使用した "u-boot-spl.axf" ファイルをリネームします。

C:¥lab¥soc\_lab¥cv\_soc\_lab¥workspace¥Atlas-Blinking-LED-Baremetal-GNU に "u-boot-spl.axf" ファイルがありま すので、このファイルを、"\_u-boot-spl.axf" 等にリネームします。

\_\_\_\_3. 演習 2 で作成した "u-boot-spl" をコピーします。

C:¥lab¥soc\_lab¥cv\_soc\_lab¥software¥spl\_bsp¥uboot-socfpga¥spl ディレクトリの下にある、"u-boot-spl" ファイルを、 C:¥lab¥soc\_lab¥cv\_soc\_lab¥workspace¥Atlas-Blinking-LED-Baremetal-GNU ディレクトリにコピーします。

\_\_\_\_4. コピーした "u-boot-spl" ファイルを "u-boot-spl.axf" という名前にリネームします。

ここまでの作業で、デバッグ時に使用する Preloader が変更されました。 実際に、動作するか確認していきます。

\_\_\_\_5. LED Blink サンプル・アプリケーションを再度実行します。

<u>71</u> ページの "\_\_\_\_ 6.「実行」メニュー ⇒「デバッグの構成」を選択します。サンプル・プロジェクトには、 Atlas-SoC ボード上で実行するための事前設定を付属しています。" から実行してください。

i Note:

演習 2 で作成した Preloader はセミホスティング機能が有効となっているので、先程のデバッグ実行時と 異なり、DS-5™ の アプリケーション・コンソール (App Console) ウィンドウに Preloader のログが表示され ることが確認できるはずです。 5-5. システム・ヘッダ・ファイルによるアドレスの解決(オプション演習)

演習 3 の LED Blink サンプル・アプリケーションでは、ソースコード上で、LED PIO のアドレスを直接指定していました。



図 5-33 今までのアドレス指定方法

ここでは、 SoC EDS のシステム・ヘッダ・ファイル生成コマンド(sopc-create-header-files)を使用してシステム・ ヘッダ・ファイルを生成し使用してみましょう。

- \_\_\_\_1. Embedded Command Shell が起動していない場合は起動します。
- \_\_\_2. C:¥lab¥soc\_lab¥cv\_soc\_lab に移動します。

\$ cd "C:¥lab¥soc\_lab¥cv\_soc\_lab"



図 5-34 ディレクトリの移動



\_3. Embedded Command Shell 上で、システム・ヘッダ・ファイル生成コマンド(sopc-create-header-files)を実行しま す。

\$ sopc-create-header-files soc\_system.sopcinfo



図 5-35 システム・ヘッダ・ファイル生成コマンドの実行

5 つのファイルが生成されたことを確認します。

soc\_system.h : Platform Designer 内のすべてのマスタに対するモジュール情報を定義 hps\_0.h : HPS の各ブリッジ(H2F, LWH2F)に接続されているモジュール情報を定義 hps\_0\_bridges.h : HPS の各ブリッジ(F2H, H2F, LWH2F)に接続されているモジュール情報を定義 hps\_0\_arm\_a9\_0.h: hps\_0\_arm\_a9\_0 向けのモジュール情報を定義。各ブリッジのオフセットも付加されている hps\_0\_arm\_a9\_1.h: hps\_0\_arm\_a9\_1 向けのモジュール情報を定義。各ブリッジのオフセットも付加されている

ここでは、 hps\_0\_arm\_a9\_0.h を使用します。

\_\_\_\_4. システム・ヘッダ・ファイルを LED Blink サンプル・アプリケーション・プロジェクトにコピーします。

ファイル名: hps\_0\_arm\_a9\_0.h

コピー元: C:¥lab¥soc\_lab¥cv\_soc\_lab

コピー先: C:¥lab¥soc\_lab¥cv\_soc\_lab¥workspace¥Atlas-Blinking-LED-Baremetal-GNU

\_\_\_\_5. LED Blink サンプル・アプリケーションのソースコードを変更します。

変更時に「書き込み可能にしますか?」というポップアップが表示される場合は「はい」を選択します。

記述追加:

#include "hps\_0\_arm\_a9\_0.h"

記述変更:

<変更前> #define LED\_BASE\_ADDR (0xFF210040)

<変更後> #define LED\_BASE\_ADDR LED\_PIO\_BASE

以下の図では、比較しやすいように以前の LED\_BASE\_ADDR 記述をコメントアウトしてあります。

参考までに、"hps\_0\_arm\_a9\_0.h"の該当箇所も図示します。



図 5-36 ソースコードの変更箇所とシステム・ヘッダ・ファイルの該当箇所

\_\_\_\_6. LED Blink サンプル・アプリケーションをビルドします。

\_\_\_\_ 7. ビルド後、LED Blink サンプル・アプリケーションを実行し、演習 3 と同様の結果となることを確認します。

### 以上で 演習 3 (オプション) は完了です。



# 6. <u>演習 4: Linux アプリケーション演習 (オプション演習)</u>

この演習では DS-5<sup>™</sup> 上から Linux のアプリケーションのひとつとして用意されている Hello World を実行、 デバッグします。

i Note:

この演習では、弊社がお貸し出しする「SoC FPGA Seminar in a Box」をご利用のお客様は、同梱されている microSD カードを使用します。この microSD カードには Linux OS を起動するためのデザインが入ってい ます。

SoC FPGA Seminar in a Box 以外でこの演習を実行されるお客様は、以下の「6-1. microSD カードの準備」 の手順により、ご自身で microSD カードをご用意ください。

6-1. microSD カードの準備

SoC FPGA Seminar in a Box に同梱されている microSD カードを使用する場合は、このセクションはとばして次の「<u>6-2. Linux 起動とログイン</u>」に進んでください。ご自身で microSD カードを書き込む場合は以下の手順 で行ってください。

- \_1. 下記のサイトから使用するボード向けの SD カードイメージ・ファイルをダウンロードします。 ダウンロードしたファイルは任意のフォルダに解凍しておきます。解凍したフォルダ内に .img イメージ・ファイ ルがあることを確認します。
  - Atlas-SoC ボード向け SD カードイメージ・ファイル
  - DE10-Nano ボード向け SD カードイメージ・ファイル
- \_\_\_\_2. Windows をご使用の場合、 SD カードイメージ・ファイルの書き込みには汎用のソフトウェアを利用します。 ここでは Win32DiskImager を紹介します。以下よりダウンロード可能です。
  - Win32DiskImager
- \_\_\_\_3. microSD カード (8GB 以上を推奨) を PC の SD カード・スロットに挿入します (または USB カード・リーダ/ ライタを使用します)。 自動再生の表示が出たら microSD カードに割り当てられたドライブ (この例では、ドライ ブ E) を確認して閉じます。



図 6-1 microSD カードに割り当てられたドライブの確認

# 

\_\_\_4. あらかじめ PC にインストールしておいた Win32DiskImager を起動します。

- ① Device として PC に挿入した microSD カードのドライブが選択されていることを確認します。
- ② 先ほど解凍した SD カードイメージ・ファイルを選択して開きます。
- ③ [Write] ボタンをクリックして イメージ・ファイルを書き込みます。
- ④ 書き込みが完了したら [OK] ボタンをクリックします。
- ⑤ [Exit] ボタンをクリックして Win32DiskImager を終了します。

[[E¥] ▼ ①
1
-
Exit
5
plete - 1.0
Write Su



\_\_\_5. PC から microSD カードを安全に取り外します。

$\Lambda$	注意事項:
	ホスト PC の OS が Windows® 10 の場合、SD カードの書き込みの際に、カード内に FAT 以外のパー
	ティション(ボリューム)が存在している場合は、以下の現象が発生することがあります。
	● カード挿入時に警告ウィンドウが表示される
	● SD カードイメージの書き込みに失敗する
	これらの現象への対処方法については、以下の参考情報サイトの記事をご参照ください。
	【参考情報サイト】
	i アルティマ技術サポート 「 <u>Windows® 10 で SD カードイメージの書き込みに失敗する場合の対処法</u> 」



6-2. Linux 起動とログイン

この演習では、以下のインタフェースを使用します。

DE10-Nano ボードも基本的には同じです。



図 6-3 本演習で使用するインタフェース

- \_\_\_\_1. ボードの 5V DC ジャック (J14) に電源アダプタが接続されている場合は、一旦ケーブルを抜きます。
- \_\_\_\_2. ボードの UART USB コネクタ(J4)へ USB Mini-B ケーブルを接続します。ケーブルの反対側のコネクタを PC の USB コネクタへ接続します。
- \_\_\_\_3. ボードの HPS Ethernet コネクタ(J10) ヘイーサーネット・ケーブルを接続します。ケーブルの反対側のコネクタ を PC のイーサーネット・コネクタへ接続します。
- \_\_\_\_4. ボード裏側の microSD カード・スロット(J11)に microSD カードを挿入します。
- \_\_\_\_5. 電源アダプタケーブルをボードの 5V DC ジャック(J14)に接続し、ボードに電源を投入します。

\_\_6. Windows の「デバイス マネージャー」を開きます。デバイスマネージャーの「ポート(COM と LPT)」を展開し てボードの UART が何番の COM ポートに接続されているかを確認します(この例では COM5)。確認できた らデバイスマネージャーを閉じます。



図 6-4 COM ポートの確認

\_7. あらかじめインストールしておいたターミナル・ソフトを起動して、シリアル・ポートの設定を行います。 先ほど確認した COM ポートを選択して下図のように設定します(この例では COM5)。

ポート(P):	COM5 •	ок
ボー•レート(B):	115200 -	
データ(D):	8 bit 🔹	キャンセ
バリティ(A):	none 🔹	
ストップ(S):	1 bit 🔹	ヘルプ(
フロー制御(F)	none 🔹	

図 6-5 シリアル・ポートの設定

\_\_\_8. ボードの WARM リセット・ボタン(KEY3)を押します。ターミナルに起動メッセージが表示されます。



図 6-6 WARM リセット・ボタン (KEY3)

\_\_\_9. Linux カーネルが起動したら、<mark>root</mark> でログインします。



図 6-7 root でログイン



6-3. Linux での IP アドレスとパスワードの設定

\_\_\_\_1. ターミナルから ifconfig コマンドで、ボードの IP アドレスを設定します (この例では 192.168.1.30 を設 定しています)。

# ifconfig eth0 192.168.1.30

\_\_\_\_2. その後 ifconfig コマンドで設定内容を確認します。

# ifconfig eth0

\_\_\_\_3. passwd コマンドで任意のパスワードを設定します。このパスワードは後でリモート・システムによるデバッグで 使用します。

# passwd

\_\_\_\_4. 再度パスワードを入力します。

The Angstrom Distribution cyclone5 ttyS0
Angstrom v2018.06 - Kernel
cyclone5 login: root root@cyclone5: <sup>*</sup> # ifconfig eth0 192.168.1.30 root@cyclone5: <sup>*</sup> # ifconfig eth0 eth0 Link encap:Ethernet HWaddr 36:6f:9b:f9:91:89 inet addr:192.168.1.30 Bcast:192.168.1.255 Mask:255.255.255.0 UP BROADCAST MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B) Interrupt:27 Base address:0xa000
root@cyclone5:~# passwd New password:
Retype new password: passwd: password updated successfully root@cyclone5:~~#

図 6-8 IP アドレスとパスワードの設定

6-4. ホスト PC 側のネットワーク設定

DS-5<sup>™</sup> での リモート・システム・エクスプローラー (RSE) を使用した Linux アプリの実行・デバッグを行う上で、 ホスト PC 側のネットワーク設定を行います。

\_\_1. まずホスト PC 側の IP アドレスを設定します。「コントロール パネル」から「ネットワークと共有センター」を クリックし、左側の「**アダプタの設定の変更**」をクリックします。



図 6-9 アダプタの設定の変更

\_\_\_2. 「**ローカル エリア接続**」をダブルクリックします。



図 6-10「ローカル エリア接続」をダブルクリック



- \_\_\_\_3. [プロパティ] ボタンをクリックします。
- \_\_\_\_4. 「**インターネット プロトコル バージョン 4(TCP/IPv4)**」をダブルクリックします。

全般	🖳 ローカル エリア接続のプロパティ
接続 インターネット IPv6 接続 インターネット IPv6 接続 ネットワーク メディアの状態: 期間: 速度: IF###(E) Monthanking Constraints IF###(E) IF##(E) I	ネットワーク       共有         接続の方法: <ul> <li></li></ul>
	OK ##21211

図 6-11 ローカル エリア接続のプロパティ

\_\_\_5. 「次の IP アドレスを使う(S):」にチェックを入れて「IP アドレス」と「サブネット マスク」を設定します(この例 では、IP アドレスを 192.168.1.31、サブネット マスクを 255.255.255.0 に設定しています)。

設定後、[OK] をクリックします。

ターネットンロトコルバーショ. 股	
トットワークでこの機能がサポートされてで ます。 サポートされていない場合は、 ネ こください。	いる場合は、IP 設定を自動的に取得することがで 、ットワーク管理者に適切な IP 設定を問い合わせ
○ IP アドレスを自動的に取得する(C)	0)
(③)欠の IP アドレスを使う(S):	100 160 1 01
IF アドレス(D) サブネット マスク(11):	255 255 255 0
デフォルト ゲートウェイ(D):	
DNS サーバーのアドレスを自動的	hに取得する(B)
③ 次の DNS サーバーのアドレスを使	更う(E):
優先 DNS サーバー(P):	67 (B) (B)
代替 DNS サーバー(A):	4: (i) (j)
🗐 終了時に設定を検証する(L)	【詳細設定(V)

図 6-12 「IP アドレス」と「サブネット マスク」の設定



- \_\_6. ネットワークの接続を確認します。ボードの Linux から、ホスト PC に対して ping を実行して接続を確認して みます(この例では、PC の IP アドレスを 192.168.1.31 に設定しています)。
   # ping 192.168.1.31
- \_\_\_\_7. [Ctrl] + [C] をキー入力して ping を停止します。

root@cyclone5:~# ping 192.168.1.31	
PING 192.168.1.31 (192.168.1.31): 56 data bytes	
64 bytes from 192.168.1.31: seq=0 ttl=128 time=1.137	ms
64 bytes from 192.168.1.31: seq=1 ttl=128 time=0.379	ms
64 bytes from 192.168.1.31: seq=2 ttl=128 time=0.688	ms
64 bytes from 192.168.1.31: seq=3 ttl=128 time=0.337	ms
°C <sup>™</sup>	
192.168.1.31 ping statistics	
4 packets transmitted, 4 packets received, 0% packet	loss
round-trip min/avg/max = 0.337/0.635/1.137 ms	
root@cyclone5:~#	

図 6-13 PC に対して ping を実行して接続を確認

\_\_\_8. もし ping 応答が無い場合は Windows ファイアウォール設定を確認します。 「**パブリックネットワークの場所の設定**」を確認し、Windows ファイアウォールが "**有効**" に設定されている場 合は "**無効**" に設定して、再度 ping を実行して接続を確認してください。



図 6-14 Windows ファイアウォール設定



6-5. DS-5™の起動と Linux サンプル・アプリケーションのインポートおよびビルド

\_1. Windows® のスタート・メニューまたは、SoC EDS のインストール・フォルダ (intelFPGA¥<バージョン>¥embedded) に格納されている起動用スクリプト Embedded\_Command\_Shell.bat をダブルクリックして、Embedded Command Shell を起動します。



図 6-15 Embedded Command Shell を起動

\_2. Embedded Command Shell から eclipse と入力して DS-5™ を起動します。

L w	
Intel FPGA Embedded Command Shell	á
Version 17.1 [Build 585]	
111490HD11149A/~ \$ eclipse_	

図 6-16 Embedded Command Shell から eclipse と入力

\_\_\_3. Eclipse ツールを使用するワークスペース・フォルダを設定します。 この演習では、「<u>3. 演習 1: ハードウェア演習</u>」の作業フォルダに workspace を作成します。 以下のパスを指定して [**OK**] をクリックします(フォルダが存在しない場合は自動的に作成されます)。

C:¥lab¥soc\_lab¥cv\_soc\_lab¥workspace

● ワークスペース・ランチャー	×
ワークスペースの選択	
Eclipse プラットフォーム は、ワークスペースと呼ばれるフォルダにプロジェクトを保存しま このセッションに使用するワークスペース・フォルダを選択してください。	ŧ.
ワークスペース( <u>W</u> ): C:¥lab¥soc_lab¥cv_soc_lab¥workspace  ・	参照(臣)
□ この選択をデフォルトとして使用し、今後この質問を表示しない(山)	
	OK キャンセル

図 6-17 workspace の作成



.4. DS-5<sup>™</sup>の Welcome 画面が表示された場合は、[**閉じる**] (×マーク) をクリックして閉じます。



- \_\_\_\_5. DS-5™ のメニューから「**ファイル」⇒「インポート**」を選択します。
- \_\_\_6. 「**一般」⇒「既存プロジェクトをワークスペースへ**」を選択し [**次へ(N)**] をクリックします。



図 6-19 「ファイル」⇒「インポート」

 7.「アーカイブ・ファイルの選択(A):」オプションを選択し、「参照(R)」ボタンよりサンプル・プロジェクトを指定します。 サンプル・プロジェクトは SoC EDS に含まれており、デフォルトでは以下のインストール・フォルダにあります。
 C:¥intelFPGA¥17.1¥embedded¥examples¥software¥Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz (<SoC EDS インストール・ディレクトリ>¥examples¥software¥Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz をインポ ートしています)。

選択後、[終了(F)] ボタンをクリックします。

🛆 ALTIMA



図 6-20 サンプル・プロジェクトのインポート



\_8. Eclipse 左側のプロジェクト・エクスプローラーに Altera-SoCFPGA-HelloWorld-Linux-GNU プロジェクトが追加され、 Altera-SoCFPGA-HelloWorld-Linux-GNU をクリックして展開するとプロジェクトに含まれる各種ファイルが表示され ます。





\_\_9. Altera-SoCFPGA-HelloWorld-Linux-GNU アプリケーションをビルドします。

プロジェクト・エクスプローラーより Altera-SoCFPGA-HelloWorld-Linux-GNU プロジェクトをハイライトし、 「プロジェクト」⇒「プロジェクトのビルド」を選択します。または、プロジェクト・エクスプローラー上でプロジェク トを選択し、右クリック ⇒「プロジェクトのビルド」を実行します。

プロジェクト・エクスプローラーに新たに生成された hello 実行可能ファイルが出力されます。



図 6-22 プロジェクトのビルド

6-6. リモート・システム・エクスプローラー (RSE) の設定

DS-5<sup>™</sup> では、リモート・システム・エクスプローラー (RSE) を使用する事で、Linux アプリケーション・プログラム をターゲット上で実行・デバッグすることが可能です。

\_\_\_\_1. 「ウィンドウ」メニュー ⇒「Perspective」⇒「パースペクティブを開く」⇒「その他」を選択します。

新規ウィンドウ(N) Editor ツールバーの非表示(T)	•			
ビューの表示(V)	÷			
Perspective		パースペクティブを開く(0)	▶ 襟	DS-5 デバッグ
ナビゲーション(G)	•	パースペクティブのカスタマイズ(Z)	L	LDRAlite for ARM DS-5 software9.5.6
設定(P) パースペクティブの別名保管(A) パースペクティブのリセット(R)	パースペクティブの別名保管(A) パースペクティブのリセット(R)	台谷	デームIDJAINL デバッグ	
		パースペクティブを閉じる(C)		その他(0)
		すべてのパースペクティブを閉じる(L)		

図 6-23 「パースペクティブを開く」⇒「その他」を選択

\_\_\_\_2. 「リモート・システム・エクスプローラー」を選択して [OK] をクリックします。

<ul> <li>         CVS リポジトリー・エクスプローラー         ② DS-5 コンフィギュレーション         ※ DS-5 デバッグ         Git             Git             Java             ② Java の型階層             ③ Java の型階層             ③ Java の型階層             〖」LDRAlite for ARM DS-5 software9.5.6             @ PyDev             〖③ チーム同期化             ↓ 二 の用化      </li> </ul>	暍C/C++ (デフォルト)	
<ul> <li>② DS-5 コンフィギュレーション</li> <li>登 DS-5 デバッグ</li> <li>Git</li> <li>認 Java</li> <li>③ Java の型階層</li> <li>③ Java の型階層</li> <li>③ Java 参照</li> <li>☑ LDRAlite for ARM DS-5 software9.5.6</li> <li>④ PyDev</li> <li>⑤ チーム同期化</li> </ul>	品CVS リポジトリー・エクスプローラ・	78
<ul> <li>♣ DS-5 デバッグ</li> <li>♣ Git</li> <li>♥ Java</li> <li>♥ Java の型階層</li> <li>♥ Java 参照</li> <li>▲ LDRAlite for ARM DS-5 software9.5.6</li> <li>● PyDev</li> <li>■ チーム同期化</li> <li>♥ = = = = = =</li> </ul>	🤣 DS-5 コンフィギュレーション	
<ul> <li>Git</li> <li>Java</li> <li>Java の型階層</li> <li>Java 参照</li> <li>LDRAlite for ARM DS-5 software9.5.6</li> <li>PyDev</li> <li>デーム同期化</li> </ul>	🐥 DS-5 デバッグ	
費 Java ◎ Java の型階層 ◎ Java 参照 ■ LDRAlite for ARM DS-5 software9.5.6 ● PyDev 音 <sup>0</sup> チーム同期化	🔐 Git	
は Java の型階層 ぼ Java 参照 I LDRAlite for ARM DS-5 software9.5.6 愛 PyDev 旨 <sup>0</sup> チーム同期化	🐉 Java	
記 Java 参照 ■ LDRAlite for ARM DS-5 software9.5.6 ● PyDev 音 <sup>0</sup> チーム同期化	と Java の型階層	
✔ LDRAlite for ARM DS-5 software9.5.6 PyDev ● ケーム同期化	Java 参照	
● PyDev 音 <sup>0</sup> チーム同期化 サーマーの 第	LDRAlite for ARM DS-5 software9.	5.6
	appev and the second se	
12	≦ <sup>8</sup> チーム同期化	
なテハック	検デバッグ	
13- リソース	■ リソース	_
闘リモート・システム・エクスプローラー	111 リモート・システム・エクスプローラ	<b>5</b> —

図 6-24 「リモート・システム・エクスプローラー」の選択



\_\_\_3. リモート・システム・エクスプローラーのビューで 👔 ボタンまたは、空白部分を右クリックして「新規接続」を 選択します。

		Ut-ト・:	システム・	エクスノローコ	2— - Eclips	eノフットノ
A リモート・システム・エクスプローラー - Eclinge プラットフォーム		ファイル(F)	編集(E) 🖯	ナビゲート(N)	検索(A)	プロジェク
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R)		🖻 • 🖬 🚯	010	. 10 10 10	9. <del>0</del> . e	<b>B R B</b>
	または	週リモート・	システム 8	3 <del>95</del> 7-4	£ 81	(† †) (éj
周リモート・システム 🛛 🕾 チーム 🜃 🕄   🌣 🗟   🕞 📚 🌣 🤅	01210					
リモート・システムの接続を定義			1	新規接続(A)		
				接続をインオ	ペート(B)	
	空白部	分を右クリック	7			

図 6-25 「リモート・システム・エクスプローラー」 での新規接続

\_\_4. リモート・システム・タイプの選択のビューで「SSH のみ」を選択し [次へ(N)] をクリックします。

♥ 新規接	続	
リモー	ト・システム・タイプの選択	
リモー	ト・システムへ SSH アクセスするための接続	
システム	・タイプ:	
フィルち	9入力	
	一般 S FTP のみ ス SSH のみ	

図 6-26 「SSH のみ」を選択

\_\_\_5. 「ホスト名:」の欄には設定しておいたボードの IP アドレス(この例では 192.168.1.30)を入力し、「接続名:」 と「記述/説明:」には"Atlas SoC"または"DE10 Nano"と入力し「ホスト名を検証」にチェックを入れて [終了(F)] ボタンをクリックします。

● 新規接続				
リモート1シス 接続情報の定義	、テム接続(SSH のみ)			
親プロファイル:	HD11149A			•]
ホスト名:	192.168.1.30			
接続名:	Atlas SoC			
記述/説明:	Atlas SoC	ホスト名:	192.168.1.30	
☑ たスト名を検証		接続名:	Atlas SoC または	DE10 Nano
プロキシー設定を構	<u>電成</u>	記述/説明:	Atlas SoC または	DE10 Nano
?	< 戻る(B)	次へ(N) >	終了(F)	キャンセル
0				

図 6-27 接続設定

- \_\_\_\_6. 「ssh.files」にチェックが入っていることを確認して、[次へ(N)] をクリックします。
- \_\_\_\_7. リモート・システム・エクスプローラーのビューで、「Atlas SoC」(または「DE10 Nano」)⇒「Sftp ファイル」⇒ 「ルート」をクリックすると、ユーザ ID とパスワードを入力するウィンドウが表示されます。
- \_\_\_\_\_8. 「ユーザ ID:」には "root"、「Password」には設定したパスワードを入力して [OK] をクリックします。

構成	プロパティー		
✓ ssh.files	プロパティ	値	
使用可能なサービス			
<ul> <li></li></ul>			

\_\_\_9. 下図の警告が出た場合は [**はい**] をクリックします。

_		
2	WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!	
J	IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!	
	Someone could be eavesdropping on you right now (man-in-the-middle attack)!	
	It is also possible that the RSA host key has just been changed.	
	The fingerprint for the RSA key sent by the remote host 192.168.1.30 is	
	1e:13:ee:bc:8b:e7:a5:d8:05:bd:2a:65:43:c2:77:8a.	
	Please contact your system administrator.	
	Add correct host key in C:¥Users¥11149¥ ssh¥known hosts to get rid of this message	
	Do you want to delate the old key and insert the new key?	
	by you want to delete the old key and insert the new key?	
	(はい(Y) いいえ(N)	
		-

### 図 6-29 警告表示



\_10. 接続が成功すると、リモート・システム・エクスプローラーに現在のボード上のファイル群が表示されます。



図 6-30 リモート・システム・エクスプローラーに現在のボード上のファイル群が表示

- \_\_\_\_11. エラーが出て接続できない場合は、ホスト PC のプロキシ設定の問題が考えられます。この場合は「**コントロー** ル パネル」→「インターネット オプション」 をクリックし、「接続」 タブの「LAN の設定」 をクリックします。
- \_\_\_\_ 12. "LAN にプロキシサーバーを使用する" にチェックが入っている場合は、このチェックを外して [OK] をクリックし





図 6-31 プロキシサーバーの設定

\_13. 再度 Atlas SoC(または DE10 Nano )のルートへの接続を試みてください。

6-7. Linux アプリケーションの実行・デバッグ

ここからは、デバッガ設定方法と実行・デバッグ方法について確認します。

- \_\_\_\_1. メニュー・バーの「C/C++ パースペクティブ」ボタン 10 をクリックして C/C++ パースペクティブに戻ります。
- \_\_\_\_2. プロジェクト・エクスプローラ・タブより Altera-SoCFPGA-HelloWorld-Linux-GNU を**右クリック**して 「**デバッグ**」⇒「**デバッグの構成**」を選択します。

と プロジェク	ト・エクスプローラー 🛛 🕞 🧐 🌱	-		- 8
	新規(N) 次ヘジャンプ(I)	右クリック	۲	
	新規ウィンドウで開く(N)			
	Copy Paste 削除(D) 移動(V) 名前を変更(M)			
24 24	インボート(I) エクスポート(0)			
	プロジェクトのビルド(B) プロジェクトをクリーンにする			
2	更新(F) プロジェクトを閉じる(S) 無関係なプロジェクトを閉じる(U)		F5	
	Make ターゲット インデックス ビルド構成		6 16 16	
	リモートシステムビューで表示 実行(R)		*	
	デバッグ(D)		۰ ا	■ 1 ローカル C/C++ アプリケーション
	フロファイル(P)			デバッグ の構成(B)

図 6-32「デバッグ」⇒「デバッグの構成」を選択

\_\_\_\_3. 「DS-5 デバッガ」を右クリックし「新規」を選択して、新しいデバッグ・コンフィギュレーションを作成します。

構成の作成、管理、およ	はび実行
DS-5 デバッグセッションを開	間始するための
- 🤹 🗆 🕅 🗱	このダイア
フィルタ入力	📑 - 選択
C/C++ アプリケー: ▲	┣┓- 選択
C/C++ アプリケー:	
C/C++ ポストモー:	右クリック
C/C++ UE-h.	
参 DS-5デバッガ	
a IronPython F 「新活	見(W)
ar IronPython u 视题	및(D)
🗊 Java アプリク 💥 削除	≩(T) <sup>≤</sup>

図 6-33 新しいデバッグ・コンフィギュレーションを作成

- \_4. 「名前」 フィールドに "HelloWorld" と入力します。
- 5. 「接続」タブの「ターゲットの選択」フィールドにおいて、

Altera  $\Rightarrow$  Cyclone V SoC (Dual Core)  $\Rightarrow$  Linux Application Debug  $\Rightarrow$  Download and debug application を選択します。

\_\_\_6. 「接続」フィールドでは、生成した RSE 接続(この例では Atlas SoC)を選択し、その他はデフォルト値を使用し ます。



図 6-34 デバッグ構成の設定(1)



\_\_7. 「ファイル」タブの「ダウンロードするホスト上のアプリケーション」に Hello World の実行体を設定します。 「ワークスペース」ボタンを使用して hello を選択し [OK] をクリックします。

名前(N): HelloWorld	
🐠 接続  🚮 ファイル 🌾 デバッガ 🧐 OS 認識機能 🖉 引数 属 環境 🗹	
ターゲットコンフィギュレーション	ファイルを選択します:
ダウンロードするホスト上のアプリケーション:	Altera-SoCFPGA-HelloWorld-Linux-GNU
ファイルシステム ターゲットダウンロードディレクトリ: ワークスペースからファイルを選	<ul> <li>cproject</li> <li>project</li> <li>Makefile</li> <li>hello</li> <li>hello.c</li> <li>hello.map</li> <li>hello.o</li> <li>RemoteSystemsTempFiles</li> </ul>
	OK キャンセル

図 6-35 デバッグ構成の設定 (2)

\_\_\_8. 「ターゲットダウンロードディレクトリ」と「ターゲット作業ディレクトリ」には"/home/root"を設定します。

名前(N): HelloWorld		
▲ 接続 📾 ファイル 🛛 🏘 デノ	(ッガ) 🇐 OS 認識機能 😡= 引数 🚾 環境 🖾 エクスポート	
ターゲットコンフィギュレー	ション	
ダウンロードするホスト上の	アプリケーション:	
\${workspace_loc:/Altera-S	SoCFPGA-HelloWorld-Linux-GNU/hello}	
ファイルシステム	クスペース 🗹 シンボルをロードします	
ターゲットダウンロードディ	レクトリ:	
/home/root		
ターゲット作業ディレクトリ		
/home/root		
,		

図 6-36 デバッグ構成の設定(3)

\_\_\_9. 「**デバッガ**」タブで、実行制御フィールドは「**シンボルからデバッグします**」を選択し、シンボル名に"main"と 入力します。

名前(N): Hellov	World					
🔷 接続 ြ フ	ァイル 👫 デバッガ	i 🔪 🏠 OS 認識機能	(※) 引数 属 環境	ি 🖾 エクスポー।	F)	
実行制御						
<ul> <li></li></ul>	◎ エントリポイン	トからデバッグしま	す )のシンボルカ	らデバッグします	main	





ロパース	ペクティブスイッチの確認		
?	この 起動 は DS-5 デバッグ パースペクティブに関連付けられています。		
	このパースペクティブを今すぐ開きますか?		
□常に	この設定を使用する(R)		
		(\$U\(Y)	いいえ(N)

図 6-38 パースペクティブ切り替えのプロンプト

\_\_\_\_12. アプリケーションは、ロードされてから main 関数でブレークします。

20	* (	Copyright (c) 2013. Altera Corporation <www.altera.com< th=""></www.altera.com<>
27		
28	#ind	clude <stdio.h></stdio.h>
► 30G	int	<pre>main(int argc, char** argv) {</pre>
31		<pre>printf("Hello SoC FPGA!\n");</pre>
32		return 0;
33	}	
3.4	93	

図 6-39 main 関数でブレーク

\_\_\_\_13. ソースコードの左余白をダブルクリックすると、赤い点 🧿 で示すようにデバッガがそこにブレークポイントを設定 します。



#### 図 6-40 ブレークポイントの設定

\_\_\_\_14. [**続行**] ボタン ▶ を押すと、アプリケーションが実行されてブレークポイントで停止します。



### 図 6-41 ブレークポイントで停止

\_\_\_15. ソースコードの左余白に赤い点 • で示されたブレークポイントをダブルクリックすると、ブレークポイント設定が 解除されます。

- \_\_16. [**ソース行のステップ実行**] ボタン 💫 (または F5)を押すと、実行コードが 1 ライン進みます。
- \_\_\_17. 「**レジスタ**」ビューは、ターゲット・レジスタの内容を表示します。また、書き込み可能なレジスタの値を変更できます。

		00-215		155
レシスター セッ	r: 97 COLSA	9-		
名前	値	<b>サイ</b> ス*	アクセス	
🛛 🔁 Core	17/17 レジスタ			
- 🛛 R0	0x00000001	32	R/W	
- 🛛 R1	0x7EFFFD24	32	R/W	
- 🛛 R2	0x7EFFFD2C	32	R/W	
- 🛛 R3	0x000083CD	32	R/W	
- 🛛 R4	0x000083ED	32	R/W	
- 🛛 R5	0x00000000	32	R/W	
- 🛛 R6	0x000082F1	32	R/W	
- 🛛 R7	0x7EFFFBC0	32	R/W	1
- 🛛 R8	0x00000000	32	R/W	
- 🛛 R9	0x00000000	32	R/W	
- @ R10	0x76FFF000	32	R/W	
- 🛛 R11	0x00000000	32	R/W	
- 🛛 R12	0x00000000	32	R/W	
- O SP	0x7EFFFBC0	32	R/W	
- 🛛 LR	0x76EAA4BC	32	R/W	-
- O PC	0x000083D6	32	R/W	
E S CPSR	0x60070030	32	R/W	

図 6-42 「レジスタ」 ビュー

\_\_18. 「**変数**」ビューは、現在有効範囲にある変数の内容を表示します。また、現在有効範囲にある変数の値を変更 できます。

(生) リン・	ク済み: HelloWo	0x orld <del>v</del>	× 18 8	
名前	値	型	カウント	サイス、
早 🗁 ローカル	変数: 2		11907/11290-11291	
e Pargc	1	int		32
🗄 💔 argv	0x7EFFFD24	char**	1	32
- > ファイルスタティック変数	変数: 0/0			
- 00-10L	変数: 0/0			

図 6-43「変数」ビュー

- \_\_\_\_19. 「App Console」(アプリケーション・コンソール)ビューは、Arm C ライブラリでのセミホスティングの実装によっ て提供されるコンソール I/O 機能を使用できます。アプリケーション内の print 文の内容が表示されます。
- \_\_\_\_ 20. [**続行**] ボタン ▶ を押すと、アプリケーションが続行され Hello SoC FPGA! と表示されます。



図 6-44「App Console」(アプリケーション・コンソール)ビュー



- \_\_21. [エントリポイントからデバッグ] ボタン 斜 をクリックすると、アプリケーションの先頭 main に戻ってブレークし ます。
- \_\_\_\_22. 再度 [**続行**] ボタン ▶ を押すと、アプリケーションが先頭から実行され、「App Console」 ビューに Hello SoC FPGA! と表示されます。
- \_\_\_\_23. [ターゲットから切断] ボタン 🔯 をクリックしてデバッグ・セッションを終了します。
- \_\_\_\_\_24. [すべての接続の削除] ボタン 🙀 をクリックしてデバッグ・セッションを削除します。
- \_\_\_\_25. メニュー・バーの [C/C++ パースペクティブ] ボタン 🐻 をクリックすると C/C++ パースペクティブに戻ります。

おめでとうございます。これで全ての演習が完了しました。

## 7. 今後の参考資料について

本演習ではインテル<sup>®</sup> SoC FPGA の開発環境であるインテル<sup>®</sup> Quartus<sup>®</sup> Prime 開発ソフトウェアやシステム構成 ツールである Platform Designer システム統合ツール、およびソフトウェア開発環境である SoC EDS の基本的な 操作を学ぶことを中心に紹介しました。今後さらなる知識向上につなげたい場合はさまざまな情報源があります のでこちらをご利用ください。

また同様の内容として「SoC のはじめてガイド」をご参照いただくと、さらに理解が深まると思いますのでご利用ください。

• SoC はじめてガイド

SoC はじめてガイド - HPS-FPGA 間のアクセス方法 SoC はじめてガイド - Preloader Generator の使用方法 SoC はじめてガイド - DS-5 によるベアメタル・アプリケーション・デバッグ

- SoC 関連情報
   マクニカ・オンラインサービス: SoC 関連記事/ドキュメント
   マクニカ・オンラインサービス: SoC 関連 FAQ
- デバイスやツールについての説明
   Intel FPGA and SoC
- SoC デバイスで Linux を使用する上で参考となる各種ドキュメントやプロジェクト <u>RocketBoards.org</u>
- 代理店からの各種情報サイト
   マクニカ・オンラインサービス
   アルティマ技術サポート
   アルティマ技術データベース



# 改版履歴

Revision	年月	概要
1	2018 年 6 月	初版
2	2018 年 9 月	Windows®10 使用の際の Preloader 生成における注記を追加
3	2019 年 1 月	注記の体裁および文言の見直し。 アイコンの追加。 相互参照の追加。
4	2019 年 7 月	p80 - Atlas-SoC ボード向け SD カードイメージ・ファイルのダウンロード URL 変更
		p81- 注意事項として、「Windows®10 で SD カードイメージの書き込みに 失敗する場合の対処法」の紹介を追記
		p83-図 6-7 の差し替え
		p84- 図 6-8 の差し替え
		p87- 図 6-13 の差し替え

#### 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。
   株式会社マクニカ アルティマ カンパニー <u>https://www.alt.macnica.co.jp/</u> 技術情報サイト アルティマ技術データベース <u>http://www.altima.jp/members/</u>
- 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。