

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

Ver.18.1



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

<u>目次</u>

目次	2
<mark>本書をお読みになる前に</mark>	4
1. 概要	5
1-1. 使用環境	6
2.ボードの設定	7
2-1. ボードレイアウト	7
2-2. 電源およびケーブルの接続	7
2-3. SW10 の設定	7
3. 演習 1: ハードウェア演習	8
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	9
3-2. ステップ 2 : HPS コンポーネントの追加	15
3-3. ステップ 3 : HPS ペリフェラルの設定(MAC、UART、I2C、SDIO、USB)	21
3-4. ステップ 4 : HPS クロックの設定	26
3-5. ステップ 5 ː SDRAM の設定	28
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	34
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	36
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	38
3-9. ステップ 9 : Platform Designer システムの確認	39
3-10. ステップ 10 : Platform Designer システムの生成	41
3-11. ステップ 11 : ピン・アサインメントの設定と Quartus [®] Prime プロジェクトのコンパイル	45
3-12. ステップ 12 : 出力ファイルの確認	49
4. 演習 2: ソフトウェア演習(1) Preloader の生成	50
4-1. ステップ 1 : Embedded Command Shell の起動	51
4-2. ステップ 2 : bsp-editor(Preloader Generator)の起動	52
4-3. ステップ 3 : プロジェクトの作成と設定	52
4-4. ステップ 4 : Preloader のビルド	56
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	57
5-1. FPGA デザインのダウンロード	58



はじめての インテル[®] SoC FPGA 演習マニュアル

(Atlas-SoC / DE10-Nano ボード版)

5-2. Hello World サンプル・アプリケーションの実行61
5-3. LED Blink サンプル・アプリケーションの実行70
5-4. 演習 2 で作成した Preloader による初期化(オプション演習)
5-5. システム・ヘッダーファイルによるアドレスの解決(オプション演習)
6. 演習 4: Linux アプリケーション演習 (オプション演習)82
6-1. microSD カードの準備82
6-2. Linux 起動とログイン84
6-3. Linux での IP アドレスとパスワードの設定86
6-4. ホスト PC 側のネットワーク設定87
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
6-6. リモート・システム・エクスプローラー(RSE)の設定95
6-7. Linux アプリケーションの実行・デバッグ
7. 今後の参考資料について 105
改版履歴

本書をお読みになる前に

この資料の内容は 2020 年 6 月現在のものです。

この資料で紹介しているソフトウェアやハードウェア、操作手順などは、指定バージョンやデバイス等以外でも 共通のものもありますが、一部については共通にならないものもありますので、ご注意ください。

文書中の記号

i Note	補足情報などを記載しています。
Point	重要なポイントを記載しています。
🖹 参考	理解を深めるため、参考となる資料やサイトを紹介しています。
▲ 注記	この資料の中では具体的には触れませんが、必要となる知識や情報を記載しています。
○ 禁止	注意点や、してはいけないことを記載しています。

文中の表記

<u>下線</u>	クリックする事で、資料中の別の章や、外部のサイトにジャンプします。
太字斜体	画面の操作をする際の、メニューやウィンドウなどに表示されている文字を示しています。
XXXXXX	入力するコマンド文字列を示しています。
網掛け	使用するツールを示しています。

1. 概要

この演習では、Cyclone[®] 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[®] 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 FPGA エンベデッド開発スイート (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[®] Prime を使用して Arm[®] プロセッサー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ーである Preloader の生成を行います。

演習 3 では、Arm[®] Development Studio 5 Intel[®] SoC FPGA Edition (以下、DS-5[™])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。

- インテル[®] Quartus[®] Prime Standard Edition v18.1 (Lite Edition でも可能)
 また Device データとして Cyclone[®] V を登録しておく必要があります。
 ダウンロードと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。
 <u>インテル[®] Quartus[®] Prime 開発ソフトウェアおよび ModelSim[®] Intel[®] FPGA Edition のダウンロード方法</u>
 インテル[®] Quartus[®] Prime 開発ソフトウェアおよび ModelSim[®] Intel[®] FPGA Edition のインストール方法
- インテル[®] SoC FPGA エンベデッド開発スイート Standard Edition v18.1 (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。
 インテル[®] SoC FPGA エンベデッド・デベロップメント・スイート(SoC EDS)のインストール方法
- 演習データ(SoC-Trial_Seminer_Lab_data_atlas_de10nano_v18.1_r1.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[®] 10 Enterprise

この演習では、Windows[®] 10 Enterprise を使用して動作の確認を行っております。

<u> 入</u> 注記:

<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れております。</u>

【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ー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。

tar zxf /cygdrive/c/intelFPGA/18.1/embedded/host_tools/altera/preloader/uboot-socfpga.tar.gz
tar: Error opening archive: Failed to open '/cygdrive/c/intelFPGA/18.1/embedded/host_tools/altera/preloader/uboot-socfpga.tar.gz'
make: *** [uboot-socfpga/.untar] Error 1

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】

■ 参考: アルティマ技術サポート「Windows® 10 における Preloader のビルドエラー」

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 アダプター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 アダプター)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 コネクター(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。

表 2-1. SW10 の設定

ボード・リファレンス	信 号 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. <u>演習 1: ハードウェア演習</u>

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

インテル[®] Soc FPGA では Cyclone[®] V に限らず、Quartus[®] 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 インターフェイスと他のパラメーターの設定
- 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ーを C:¥lab¥soc_lab フォルダーとして説明をします。作業フォルダーを変更された 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[®] Prime 18.1 Standard Edition (Lite Edition でも可能)開発ソフトウェア内の、 Quartus[®] 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 $\mathcal{A}\mathcal{P} \rightarrow \mathcal{P}$ Intel FPGA 18.1.0.625 Standard Edition / Lite Edition \Rightarrow Quartus (Quartus Prime 18.1)



図 3-2. Quartus[®] Prime の起動

__2. Quartus[®] 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[®] Prime でのプロジェクト・ファイルとなっています。

8	Quartus Prime Standard Edition	Open Project				×
File	Edit View Project Assignments	Processi $\leftarrow \rightarrow \cdot \uparrow $	lab > soc_lab > cv_soc_lab	~ ひ	cv_soc_labの検索	م
	New Ctrl+	N 整理 ▼ 新しいフォル	ダー		812	• 🔳 🔞
5	Open Ctrl+ Close Ctrl+	O	名前	更新日時	種類	サイズ
	-	📙 3D オブジェクト	📜 .qsys_edit	2020/01/10 16:09	ファイル フォルダー	
å	New Project Wizard	📜 ダウンロード	📕 db	2020/01/10 16:09	ファイル フォルダー	
×.	Open Project Ctrl+	J デスクトップ	📕 ip	2020/01/10 16:09	ファイル フォルダー	
	Save Project	■ ドキュメント	software_example	2020/01/10 16:09	ファイル フォルダー	
	Close Project	■ ピクチャ	soc_system.qpf	2018/01/25 11:02	QPF ファイル	2 KB
		🚺 ミュージック				
		🐛 Windows (C:) 🗸	<			>
		עב	イル名(N): soc_system.qpf	~	Quartus Prime Proje 開く(O)	ect File (*.qp 〜 キャンセル

図 3-3. Quartus[®] 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 : atlas 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 I	P <mark>roj</mark> ect	Assignments	Processing	
	って	atlas	•	
Project Navigator 🛛 🔺 Hier		atlas		
		DE10-Nano		
E Cyclone V: 5CSE	ntity:Ins MA4U2	tance 3C6		
abrd top				

図 3-4. 使用ボードの選択



__4. Quartus[®] 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_lab/cv	/_soc_lab/soc_system - atlas
File Edit	View Project Assignme	nts Processing	Tools Window Help
🗋 🔭 🖬 🖓	イロロクマ atlas gator A Hierarchy	- Q. 4 8 ×	Run Simulation Tool
Cyclone \ P ghrd_te	Entity:Instance /: 5CSEMA4U23C6 op ⁴	_	 Timing Analyzer Advisors
			 Chip Planner Design Partition Planner Netlist Viewers
			 Signal Tap Logic Analyzer In-System Memory Content Editor Logic Analyzer Interface Editor In-System Sources and Probes Editor Signal Probe Pins Programmer JTAG Chain Debugger Fault Injection Debugger System Debugging Tools
T <mark>asks</mark>	Compilation	* ≡ ₽ *	😫 IP Catalog
∨ ⊳ c	Task Compile Design	<u>^</u>	Nios II Software Build Tools for Eclipse
		```	🤌 Tcl Scripts

図 3-5. Platform Designer の起動

____5. soc_system.qsys ファイルを開きます。



図 3-6. 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。



図 3-7. Platform Designer 画面



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・メモリー
- 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 3-8. 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。

<b>1</b>	Syste	em Conte	nts 🛛 🕅 Address Ma	ap 🛛 Interconnect Requ	irements 🕺			- 5 🗆	Presets 🕺 🧏 Parameters 🖉 🗕 – 🗗 🗖
	3 4	Sys	stem: soc_system	Path: dk_0					System: soc_system Path: clk_0
1	U	Conn	Name	Description	Export	Clock	Base	End	Clock Source.
X	$\checkmark$		■ clk_0	Clock Source					
N 14			cik_in_reset clk_in_reset clk clk_reset	Clock Input Reset Input Clock Output Reset Output	clk reset Double-click to Double-click to	exported			Parameters Clock frequency.      5000000      Dock frequency is known
×	$\mathbf{N}$	$\begin{array}{c} \bullet \\ \bullet $	□ onchip_memo clk1 s1 reset1 □ led_pio clk reset	On-Chip Memory (RAM Clock Input Avalon Memory Mapped Reset Input PIO (Parallel I/O) Intel F Clock Input Poart Input	Double-click to Double-click to Double-click to Double-click to	<b>clk_0</b> [dk1] [dk1] <b>clk_0</b>	4		Reset synchronous edges: None v

図 3-9. 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[™]-A9 MPCore[™] プロセッサー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 3-10. 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。

📫 IP Catalog 🛛 🗕 🗗 🗖
🔍 processor 🕺 🔯
Project New Component Library Processors and Peripherals
Embedded Processors     Nios II (Classic) Processor     Nios II Processor     Nios II Processor     Hard Processor Components     Altera HPS Emac Interface Splitter     Altera HPS Trace IP     Ama 10 External Memory Interfaces for HPS     HPS GMII to RGMII Converter Intel FPGA IP     HPS GMII to RGMII Converter Intel FPGA IP     HPS GMII to TSE 1000BASE–X/SGMII PCS Bridge Intel FPGA IP     Stratix 10 External Memory Interfaces for HPS     Hard Processor Systems
Ania 10 Hard De cosser Statem Ania V/Ovclone V Hard Processor System Stratix 10 Hard Processor System
New Edit

図 3-11. 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。

ハード	・プロも	ヹッサ・シ	ステム(	HPS)					
ARM Cortex NEON/FPI L1キャッシ	-A9 ARM J Ni a L1	USB OTG (2個)	EMAC (2 個)						
	L2キャッシュ	GPIO	l ² C (4 個)	Ŧ					
QSPI フラッシュ・ コントロール	64KB RAM	SPI (4 個)	CAN (2 個)	OII S					
NAND フラッシュ	SD / SDIO/ MMC	DMA (8チャネル)	UART (2 個)						
共有マルチ	⊀-⊦DDR	HPS -	FPGA -	FPGA コンノフィギュ					
-28mm LP フロセス ・8入力 ALM ・可変精度 DSP ・M10K メモリあよび 640ビット MLA ・FPLL									
ハード・マルチポート DDR SDRAM コントローラ PCle [®] 用 ハード IP ドランシーバ									

図 3-12. 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。



図 3-13. FPGA Interface タブの設定

#### i Note:

これは、マイクロ・プロセッサー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ーのイベントとし 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 インターフェイスで未使用のピン(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ー、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® プロセッサー 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 が 0xFF20_0000



と決まっています。

図 3-14. HPS と FPGA の内部バスと Arm から見たアドレスマップ

#### 📄 参考:

HPS-FPGA 間のインターフェイスに関しては、マクニカ・ホームページ技術情報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 🖂	HPS-to-FPGA interface width:	64-bit 🖂
Lightweight HPS-to-FPGA interface width	32-bit ~	Lightweight HPS-to-FPGA interface width	32-bit 🗸

図 3-15. AXI Bridges の設定

#### i Note:

FPGA-to-HPS interfaces を有効にすると、FPGA 内のマスターが 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 がマスターとなり 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** インターフェイスをクリックし、 - ボタンをクリックして、インターフェイスを削除します。

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(アクセラレーター・コヒーレンシー・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 コヒーレンシーはユーザー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 SDRAM Interface	FPGA-to-HF	PS SDRAM Interfa	ice	
Click the '+' and '-' buttons to add and remove FPGA-to-HPS SDRAM ports.	Click the '+' a	nd '' buttons to add	d and remove FPG	iA—to—HPS SDRAM ports.
Name Type Width	Name	Туре	Width	]
<mark>f2h_sdram0 AXI−3 × <u>64</u> ×</mark>				
+ <b>-</b>	+ -			1

図 3-16. FPGA-to-HPS SDRAM インターフェイスの設定

## 

_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 コントローラーの 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® プロセッサー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):

Enable	HPS-to-FPGA cold	l reset output
Enable	HPS warm reset ha	ndshake signals
Enable	FPGA-to-HPS deb	ug reset request
Enable	FPGA-to-HPS war	m reset request
Enable	FPGA-to-HPS cold	l reset request
MA Per	ipheral Request	
Periphera	al Reg., Enabled	
)	No	^
	No	
1	No	
	No	~
nterrupt	5	
Enable	FPGA-to-HPS Inte	mupts
HPS-to	-FPGA	
Enat	ole CAN interrupts	
Le	il landshood land	San
Enat	ne dock beriblieigi i	nterrupts
Enat	ble CTI interrupts	nterrupts
Enat	ble CTI interrupts ble DMA interrupts	nterrupts
Enat	ble CTI interrupts ble DMA interrupts ble EMAC interrupts	(for EMAC() and EMAC()
Enal	ole CTI interrupts ole DMA interrupts ole EMAC interrupts ole FPGA manager in	(for EMACO and EMAC1 itemupt
Enal	ole COUR peripheral i ole CTI interrupts ole DMA interrupts ole EMAC interrupts ole EPGA manager in ole GPIO interrupts	(for EMACO and EMAC1 itemupt
Enal	ole CCT interrupts ole DMA interrupts ole EMAC interrupts ole FPGA manager in ole GPIO interrupts ole I2C-EMAC interr	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3)
Enal	ole CCT interrupts ole CTT interrupts ole EMAC interrupts ole FPGA manager in ole GPIO interrupts ole I2C-EMAC interr ole I2C peripheral int	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3) iemupts (for 12C0 and 12(
Enal Enal Enal Enal Enal Enal Enal Enal	de GOCK peripheral i de CTI interrupts de DMA interrupts de EMAC interrupts de FPGA manager in de GPIO interrupts de I2O-EMAC interr de I2O peripheral int de L4 timer interrupt	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3) emupts (for 12C0 and 120 ts
Enat Enat Enat Enat Enat Enat Enat Enat	ble COCK peripheral i ble CTI interrupts ble DMA interrupts ble EMAC interrupts ble EPGA manager in ble GPIO interrupts ble I2C-EMAC interr ble I2C peripheral int ble L4 timer interrupt ble NAND interrupt	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3) iemupts (for 12C0 and 120 is
Enat Enat Enat Enat Enat Enat Enat Enat	ble COCK peripheral in ble CTI interrupts ble DMA interrupts ble EPGA manager in ble GPI0 interrupts ble I2C-EMAC interru ble I2C peripheral int ble I2C peripheral int ble L4 timer interrupt ble NAND interrupt ble OSC timer interru	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3) emupts (for 12C0 and 120 ts
Enet Enet Enet Enet Enet Enet Enet Enet	ble COUR peripheral in ble CTI interrupts ble DMA interrupts ble EPGA manager in ble GPI0 interrupts ble I2C -EMAC interrupt ble I2C peripheral int ble L4 timer interrupt ble NAND interrupt ble OSC timer interrupt ble Quad SPI interrupt	(for EMACO and EMAC1 temupt upts (for 12C2 and 12C3) emupts (for 12C0 and 12( ts upts ot
Enet Enet Enet Enet Enet Enet Enet Enet	ble COUR peripheral in ble CTI interrupts ble DMA interrupts ble EMAC interrupts ble FPGA manager in ble GPI0 interrupts ble I2C peripheral int ble I2C peripheral int ble I2C peripheral int ble L4 timer interrupt ble NAND interrupt ble OSC timer interru ble Quad SPI interrupt ble SD/MMC interrupt	(for EMACO and EMAC1 temupt upts (for 12C2 and 12C3) emupts (for 12C0 and 120 ts upts ot
Enet Enet Enet Enet Enet Enet Enet Enet	ble COUR peripheral in ble CTI interrupts ble DMA interrupts ble EMAC interrupts ble FPGA manager in ble GPI0 interrupts ble I2C peripheral int ble L4 timer interrupt ble NAND interrupt ble OSC timer interrupt ble SD/MMC interrupt ble SD/MMC interrupt ble SD/MMC interrupt	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3) iemupts (for 12C0 and 120 is upts ot ot
Enet Enet Enet Enet Enet Enet Enet Enet	ble COCK peripheral ble CTI interrupts ble DMA interrupts ble EMAC interrupts ble FPGA manager in ble GPI0 interrupts ble GPI0 interrupt ble GPI0 interrupt ble L4 timer interrupt ble OSC timer interrupt ble OSC timer interrupt ble SD/MMC interrupt ble SPI master interru ble SPI slave interrupt	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3) emupts (for 12C0 and 120 ts upts ot ot ot
Enst Enst Enst Enst Enst Enst Enst Enst	de COCK peripheran de CTI interrupts de DMA interrupts de EMAC interrupts de FPGA manager in de GPIO interrupts de I2C-EMAC interru de I2C peripheral int de L4 timer interrupt de NAND interrupt de OSC timer interru de GSC timer interru de SPI master interru de SPI master interru de SPI slave interrupts	(for EMACO and EMAC1 itemupt upts (for 12C2 and 12C3) emupts (for 12C0 and 120 ts upts ot ot ot upts ots
Enet Enet Enet Enet Enet Enet Enet Enet	ble CUCK peripheral in ble CTI interrupts ble DMA interrupts ble EMAC interrupts ble EPGA manager in ble GPIO interrupts ble I2C-EMAC interrupt ble I2C peripheral int ble L4 timer interrupt ble OSC timer interrupt ble OSC timer interrupt ble SD/MMC interrupt ble SPI master interrupts ble UART interrupts ble USB interrupts	(for EMACO and EMAC1 temupt upts (for 12C2 and 12C3) emupts (for 12C0 and 120 ts upts ot upts ot

#### 図 3-17. 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 通りのパラメーター (HPSI/O Set 0~3) から選択することができま す。



図 3-18. HPS I/O のピン・マルチプレクサー

- __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 O 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 I2CO pin を HPS I/O Set O に設定します。
- ____13. I2C Controllers の I2CO 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ーは次ページを参照してください。



GA Interfaces [Peripheral F	THE HES Clo	OCKS SURAM	HHCiA Interfaces Henp	nerai Fins HPS Clocks SE
Ethernet Media Access (	Controller	10	Ethernet Media Ace	cess Controller
MAGO pin:	Unused	×	EMACO pro	Unused ~
:MACU mode:	N/A ~		EMACO mode:	N/A ~
:MAC1 pin:	Unused	~	EMAC1 pin:	HPS I/O Set 0 ~
MAC1 mode:	N/A ~		EMAC1 mode:	RGMII ~
AND Flash Controller	-		- NAND Flash Contro	oller
IAND pin:	Unused	~	NAND pin:	Unused 🗸 🗸
IAND mode:	N/A ~		NAND mode:	N/A ~
uad SPI Flash Controll	er		Quad SPI Flash Co	ntroller
SPI pin:	Unused	<i>•</i>	QSPI pin	Unused 🗸
SPI mode:	N/A ~		QSPI mode	N/A ~
0/MMC Controller	•		SD/MMC Controlle	r
DIO pin:	Unused	~	SDIO pin	HPS I/O Set 0 ~
IO mode:	N/A ~		SDIO mode:	4-bit Data
B Controllers			USB Controllers	
B0 pin:	Unused	~	USB0 pin	Unused ~
BO PHY interface mode	N/A v		USB0 PHY interface	mode N/A V
SB1 pin:	Linumod	~	USB1 pin	HPS 1/0 Set 0 ~
SB1 PHY interface mode.	N/A ~		USB1 PHY interface	mode SDR with PHY cloc
Controllor	ALCONCER!		- SPI Controller	
1 Controllers	Downed	31	SPIM0 pin:	Unused
MAN DAT	Unused	×.	SPIMO mode	N/A N
IND TOOR:	N/A V		SDM ele	HPC 1/0 CHA
1MI pin:	Unused	Ŷ	SPIMI pri	HPS I/U Set U V
IMI mode:	N/A v		SPIMI mode:	Single Slave Selec
PISO pin:	Unused	~	SPISU pin	Unused ~
4S0 mode.	N/A ~		SPIS0 mode	N/A ~
PIS1 pin:	Unused	~	SPIS1 pin:	Unused 🗸 🗸
IS1 mode:	N/A ~		SPIS1 mode	N/A $\sim$
ART Controllers			UART Controllers	
ARTO pin:	Unused	~	UARTO pin	HPS I/O Set 0 V
ART0 mode	N/A v	201	UARTO mode:	No Flow Control
IART1 rin	Linumat		UART1 pin	Unused
ART1 mode:	N/A v	~	UART1 mode:	N/A ~
0. Controllors	and a second second		= 12C Controllers	
C0 nin	Linused	0	I2C0 pin	HPS I/O Set 0 v
C0 mode	N/A		I2C0 mode	12C ×
CO mode.	N/A V		12C1 pig	HPS 1/0 Set 0 1
of pin	Unused	~	12C1 mode	120
201 mode:	N/A ~		1000	460
2CZ pin	Unused	~	1202 pm	Unused
2C2 mode:	N/A ~		12UZ mode:	[N/A ~]
2C3 pin:	Unused	~	I2C3 pin:	Unused
2C3 mode:	N/A ~		I2C3 mode	N/A ~
AN Controllers	-		CAN Controllers	
ANO pin:	Unused	×.	CAN0 pin	Unused
ANO mode:	N/A ~		CAN0 mode	N/A ~
AN1 pin:	Unused	~	CAN1 pin	Unused ~
DAN1 mode:	N/A ~	0	CAN1 mode	N/A ~
	it		Trace Port Interfac	æ Unit
ace Fort Intertace Lin	172		TOLOT	102
ACE pin	Unused	~	I RACE pin:	Unused V

図 3-19. 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。



#### 図 3-21. 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。





_18. 設定後、下図のように2つのエラー以外に、エラーが出ていないことを確認してください(この2つのエラーは後 で解決します)。

🏅 Mess	ages 🛛	- d'a
Туре	Path	Message
-8	2 Errors	
8	soc_system.hps_0	hps_0.h2f_axi_clock must be connected to a clock output
8	soc_system.hps_0	hps_0.h2f_lw_axi_clock must be connected to a clock output
	7 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.
<u> </u>	soc_system.hps_0	ODT is disabled. Enabling ODT (Mode Register 1) may improve signal integrity
<u> </u>	soc_system.button_pio	button_pio.s1 must be connected to an Avalon-MM master
<u> </u>	soc_system.dipsw_pio	dipsw_pio.s1 must be connected to an Avalon-MM master
<u> </u>	soc_system.led_pio	led_pio.s1 must be connected to an Avalon-MM master
<u> </u>	soc_system.onchip_memory2_	0 <mark> onchip_memory2_0.s1</mark> must be connected to an Avalon-MM master
	5 Info Messages	
0	soc_system.button_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
0	soc_system.dipsw_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
0	soc_system.hps_0	HPS Main PLL counter settings: n = 0 m = 73
0	soc_system.hps_0	HPS peripherial PLL counter settings: n = 0 m = 39
0	soc_system.led_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

図 3-23. ピン競合エラーが無いときの表示例

例えば、<u>図 3-24</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。

Na Par	ameters 🖄		
Syster	n:soc_system Path:	hps_0	
Amia ' altera	V/Cyclone V Hard Proc _hps	cessor System	
- SF	PI Controllers		
SF	PIMO pin:	Unused ~	
SF	PIM0 mode:	N/A ~	
SF	PIM1 pin:	HPS I/O Set 0 ~	
SF	PIM1 mode:	Single Slave Select ~	
SF	PIS0 pin:	HPS I/O Set 0 V	
SF	PIS0 mode:	SPI ~	使用しない SPISU を誤つ (設定しににの)・・・
SF	PIS1 pin:	Unused ~	
SF	PIS1 mode:	N/A ~	
- U/	ART Controllers		
UA	ARTO pin:	HPS I/O Set 0 🗸	
UA	ARTO mode:	No Flow Control $$	
UA	ART1 pin:	Unused ~	
U/	ART1 mode:	N/A ~	
<			spico と UAPTO が諳会するというエラーが発生
6 Mes	sages 🙁		
Type	Path	Message	
	2 Empre		
8	soc_system.hps_0	Refer to the Periphera	Is Mux Table for more details. The selected peripherals 'SPISO' and 'UARTO' are conflicting.
G	soc_system.hps_0	hps_U.hZt_axi_clock r	hust be connected to a clock output
8	Soc_system.hps_0 hps_0.h2f_lw_axi_clock must be connected to a clock output		

図 3-24. ピン競合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ーは、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ーは、ブートローダー(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	MHz	
EOSC2 clock frequency.	25.0	MHz	
▼ <u>FP</u> GA-to-HPS PLL Reference Clocks			
Enable FPGA-to-HPS SDRAM PLL reference clock			
Enable FPGA-to-HPS peripheral PLL reference clock	(		
FPGA-to-HPS SDRAM PLL reference clock frequency.	0.0	MHz	
FPGA-to-HPS peripheral PLL reference clock frequency.	0.0	] MHz	
Perinheral FPGA Clocks			
EMACO emacO_md_clk clock frequency.	2.5	MHz	
EMAC0 emac0_gtx_clk clock frequency.	125	MHz	
EMAC1 emac1_md_clk clock frequency.	2.5	MHz	
EMAC1 emac1_gtx_clk clock frequency:	125	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	

図 3-25. HPS to FPGA Clock の設定

# 

___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。

PGA Interfaces Peripheral Pins HPS Clocks SDRA	M	
Input Clocks		
Clock Sources		
Peripheral PLL reference clock source:	EOSC1 d	lock 🗸
SDMMC clock source.	Periphera	I NAND SDMMC dock 🛛 🗸
NAND dock source	Periphera	I NAND SDMMC clock
QSPI dock source	Main QSF	N dock
L4 MP clock source	Periphera	l base clock 🛩
L4 SP dock source:	Peripheral base clock 🗠	
Main PLL Output Clocks - Desired Frequence	ies	
Default MPU clock frequency:	925.0	DE10-Nano の場合は、800.0 MHz
Use default MPU clock frequency		
MPU clock frequency.	800.0	Mitz
L3 MP clock frequency.	185.0 ~	DE10-Nano の場合は、200.0 MHz
L3 SP dock frequency	92.5 V N	
Debug AT clock frequency	250 V N	
Debug clock frequency:	12.5 ~ N	AHz
Debug trace dock frequency	25.0 ~	MHz
L4 MP clock frequency.	100.0	MHz
L4 SP dock frequency.	100.0	MHz
Configuration/HPS-to-FPGA user 0 clock frequen	SY 100.0	MHz
Peripheral PLL Output Clocks - Desired Free	quencies	
SDMMC dock frequency:	200.0	MHz
NAND clock frequency:	12.5	MHz
QSPI clock frequency:	400.0	MHz
EMACO dock frequency:	250.0	MHz
EMAC1 clock frequency	250.0 ~	MHz
USB clock frequency:	200.0	MHz
SPI clock frequency	200.0	MHz
CAND clock frequency:	100.0	MHz
CAN1 clock frequency	100.0	MHz
GP10 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 3-26. HPS to FPGA Clock の設定

3-5. ステップ 5 : SDRAM の設定

\Lambda ALTIMA

*SDRAM* タブには、HPS 側の SDRAM コントローラーおよび接続する DDR に関するパラメーターを設定するオ 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)。

🌲 Platfo	orm Designer - soc_system.qsy	s (C:¥la	b¥soc_lab¥cv_soc_lab¥soc_system.qs	ys)	- C	) X	
Eile Edit ;	<u>System</u> <u>Generate</u> <u>V</u> iew <u>T</u> ools	Help					
👑 Block	Symbol ≈	- 🗗 🗆	Nerameters 🛛			- 5'	
	u simala		System: soc_system Path: hps_0				
	v sigi dis		Arria V/Cyclone V Hard Processor Sys	stem		Detaile	
	hps 0		altera_hps			Details	
hps_io					ר		^
h2f_ax	i_clock clock reset		FPGA Interfaces Peripheral Pins Fil	-S Clock SDIVAW	J		
h2f_lw	_axi_clockaxi	h2f_i	SDRAM Protocol: DDR3 ~				
	axi	2f_lw_i	PHY Settings Memory Parameters	Memory Timing B	oard Settings		
			- Ola ale				
			Memory clock frequency.	300.0	MH ₂		
				ed of calculated fra			
			Achieved memory clock frequence	/ 200.0	MHH		
			PLL reference clock frequency	125.0			
				120.0	IVII 12		
			* Advanced PHY Settings		***		
			Supply Voltage:	1.5V DDR3 ~			
			I/O standard:	SSTL-15			
<		>				~	1
			<			>	
🏅 Messa	ges 🛛					- ದ್	
Type	Path	Mes	sage				
-0	BIC 2 Errors			1	~		
8	soc_system.hps_0	hps_	0.h2f_axi_clock must be connected to a	l clock output			
8	soc_system.hps_0	hps	0.h2f_lw_axi_clock must be connected	to a clock output			1
<		_			1	~	
2 Errors,	2 Errors, 7 Warnings Generate HDL Finish						

図 3-27. 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。

File Edit System Generate View Tools Help IP Catalog 🗧 🕺 Interconnect Requi 🖇 🛛 🖬 🗖 Parameters - 🗗 🗖 🖾 System Content: 🔅 Address Map - ರೆ ⊏ 🛯 🛋 😻 System: soc_system 🛛 Path: hps 0 ystem: soc_system Path: hps_0 Arria V/Cyclone V Hard Processor System altera_hps Project New Component... Details 再度 HPS **Parameters** 設定ウィンドウが表示される Bostern
 Basic Functions
 Bost
 Dop
 Dop
 Dop
 Interface Protocols
 Low Power
 Bemory Interfaces and Controllers
 Processors and Peripherals
 Processors and Peripherals
 Tri-State Components
 Hubuensity Program Avaion Memory Mapped ... Reset Input PIO (Parallel I/O) Intel F... Clock Input Avaion Memory Mapped ... Conduit PIO (Parallel I/O) Intel F... Clock Input Avaion Memory Mapped ... Conduit PIO (Parallel I/O) Intel F... Clock Input Reset Input Avaion Memory Mapped ... Conduit Ama W/Oyclone V Hard FPGA Interfaces Peripheral Pins HPS Clocks SDRAM set1 led_pio clk reset  $\checkmark$ * General Enable MPU standby and event signals s1 Enable general purpose signals ed_pic ⊟ dipsw_pio Enable Debug APB interface Enable System Trace Macrocell hardware events System Contents ウィンドウに追加された Enable FPGA Cross Trigger Interface dipsw hps コンポーネントをダブルクリックすると・・・ Enable FPGA Trace Port Interface Unit Enable FPGA Trace Port Alternate FPGA Interface Enable boot from fpga signals 😫 Hierarchy 🖾 Device Fami 🛛 button ರೆ ⊏ Clear Contractions
 Contractions 💷 hps_0 ~ Enable HLGPI Interface 
 Ips 0
 Arris V/Cxd0

 memory
 Conduit

 hps io
 Conduit

 h2Fresst
 Reset 0 utput

 h2faxi_dock
 Clock Input

 h2faxi_dock
 Clock Input

 h2fiv_axi_mas...
 AXI Master
 memor hps_io * AXI Bridges EPGA-to-HPS interface width hps io led pio external connection Unused ~ HPS-to-EPGA interface width: 64-bit ~ memory
 reset
 button pio Lightweight HPS-to-FPGA interface width: 32-bit

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。

_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。

# 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 ⇒ 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 ⇒ 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。



図 3-29. プリセットの選択

- __5. Apply をクリックすると、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 HP	S Clock <mark>;</mark> [SDRAM]				
SDRAM Protocol: DDR3 V					
PHY Settings Memory Parameters 1	Memory Timing Bo	ard Settings			
Clocks					
Memory clock frequency:	400.0	MHz			
Use specified frequency instead	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 V				

図 3-30. 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。

FPGA Interfaces Peripheral Pins HPS Clock <mark>s</mark> SDRAM				
SDRAM Protocol: DDR3 🗸				
PHY Settings Memory Parameters Men	nory Timing Board Settings			
Apply memory parameters from the manufacturer data sheet Apply device presets from the preset list on the right.				
Memory vendor:	Other ~			
Memory format:	Discrete Device 🗸			
Memory device speed grade:	800.0 ~ MHz			
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				

図 3-31. 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	
Burst Length:	Burst chop 4 or 8 (on the fly) $ \sim$
Read Burst Type:	Sequential ~
DLL precharge power down:	DLL off ~
Memory CAS latency setting:	7
Output drive strength setting:	RZQ/6 ~
ODT Rtt nominal value:	RZQ/6 ~
Auto selfrefresh method:	Manual 🗠
Selfrefresh temperature:	Normal ~
Memory write CAS latency setting:	7 ~
Dynamic ODT (Rtt_WR) value:	Dynamic ODT off ~

☑ 3-32. 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 H	IPS Clocks SDRAM
SDRAM Protocol:	DDR3 ~	
PHY Settings M	emory Parameters	Memory Timing Board Settings
Apply timing para Apply device pre	ameters from the r sets from the pres	nanufacturer data sheet et list on the right
tIS (base):	175	ps
tIH (base)	250	pe
tDS (base):	50	ps
tDH (base):	125	De
tDQSQ:	120	pe
tQH:	0.38	cycles
tDQSCK:	400	pe
tDQSS:	0.25	cycles
tQSH:	0.38	cycles
tDSH:	0.2	cycles
tDSS	0.2	cycles
tINIT:	500	us
tMRD (tMRW):	4	cycles
tRAS:	35.0	ns
tRCD:	13.75	ns
tRP.	13.75	ns
tREFI (tREFIab)	7.8	us
tRFC (tRFCab):	300.0	ns
tWR:	15.0	ns
tWTR:	4	cycles
tFAW:	37.5	ns
tRRD:	7.5	ns
tRTP:	7.5	ns

図 3-33. 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: DDR3 V					
PHY Settings Memory Parameters Memory Timing Board Set	tings				
Use the Board Settings to model the board-level effects in the	timing analysis.				
The wizard supports single- and multi-rank configurations. Alte effects on the output signaling of these configurations and has rate and the channel uncertainty within the UniPHY MegaWizar	ra has determined stored the effects o d.	the on the output slew			
These values are representative of specific Altera boards. You the board level effects for your board. You can use HyperLynx values that are representative of your board.	<i>must change the v</i> or similar simulator	<i>alues to account for</i> s to obtain			
<ul> <li>Setup and Hold Derating</li> </ul>					
The slew rate of the output signals affects the setup and hold	d times of the mem	ory device.			
You can specify the slew rate of the output signals to refer to both the address and command signals and the DQ signals, o	o their effect on the r specify the setup	e setup and hold times of and hold times directly.			
Derating method					
	OSpecifys	ew rates to calculate setup and hold times			
	O 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				
tDH:	0.225	ns			
- Olamont Circuit Intermite					
Channel Signal Integrity is a measure of the distortion of the or crosstalk or other effects. Typically when going from a sing configuration there is an increase in the channel loss as there reflections. Please perform your channel signal integrity simul uncertainty as compared to Altera's reference eye diagram.	eye due to intersyr ge-rank configurati e are multiple stube lations and enter th	nbol interference on to a multi−rank s causing ne extra channel			
Derating Method	Jse Alter	a's default settings			
Address and command eye reduction (setup)	0.0	namei uncertainty values			
図 3-34. Boar	d 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。

or crosstalk or other effects. Typically when going from a single- configuration there is an increase in the channel loss as there are reflections. Please perform your channel signal integrity simulation uncertainty as compared to Altera's reference eye diagram.	due to inters ank configura multiple stu ns and enter	ymbol interference ition to a multi-rank bs causing the extra channel		
Derating Method	• Use Altera's default settings			
	O 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			
Read DQ eye reduction:	0.0	ns		
Read Delta DQS arrival time:	0.0	ns		
PCB traces can have skews between them that can cause timing	manzins to b	e reduced. Eurthermore		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values	margins to t in in multi-ra	e reduced. Furthermore ank topologies.		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values Maximum CK delay to DIMM/device:	mangins to b in in multi-m	e reduced. Furthermore ank topologies. ns		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values Maximum CK delay to DIMM/device: Maximum DQS delay to DIMM/device:	mangins to b in in multi <del>na</del> 0.6 0.6	e reduced. Furthermore ank topologies. ns		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values Maximum CK delay to DIMM/device: Maximum DQS delay to DIMM/device: Minimum delay difference between CK and DQS:	margins to b in in multi-ra 0.6 0.6	e reduced. Furthermore ank topologies. ns ns		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values Maximum CK delay to DIMM/device: Maximum DQS delay to DIMM/device: Minimum delay difference between CK and DQS: Maximum delay difference between CK and DQS:	margins to b in in multi-ra 0.6 0.6 -0.01 0.01	e reduced. Furthermore ank topologies. ns ns ns ns		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values Maximum CK delay to DIMM/device: Maximum DQS delay to DIMM/device: Minimum delay difference between CK and DQS: Maximum delay difference between CK and DQS: Maximum skew within DQS group:	margins to b in in multi-ra 0.6 0.6 -0.01 0.01 0.02	e reduced. Furthermore ank topologies. ns ns ns ns ns		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Maximum CK delay to DIMM/device: Maximum DQS delay to DIMM/device: Minimum delay difference between CK and DQS: Maximum delay difference between CK and DQS: Maximum skew within DQS group: Maximum skew between DQS groups:	margins to b in in multi-ra 0.6 -0.01 0.01 0.02 0.02	e reduced. Furthermore ank topologies. ns ns ns ns ns ns		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values Maximum CK delay to DIMM/device: Maximum DQS delay to DIMM/device: Minimum delay difference between CK and DQS: Maximum delay difference between CK and DQS: Maximum skew within DQS group: Maximum skew between DQS groups: Average delay difference between DQ and DQS:	margins to b in in multi-ra 0.6 -0.01 0.01 0.02 0.02 0.02	e reduced. Furthermore ank topologies. ns ns ns ns ns ns ns ns		
PCB traces can have skews between them that can cause timing skews between different ranks can further reduce the timing marg Restore default values Maximum CK delay to DIMM/device: Maximum DQS delay to DIMM/device. Minimum delay difference between CK and DQS: Maximum delay difference between CK and DQS: Maximum skew within DQS group: Maximum skew between DQS groups: Average delay difference between DQ and DQS: Maximum skew within address and command bus:	margins to b in in multi-ra 0.6 -0.01 0.02 0.02 0.02 0.02	e reduced. Furthermore ank topologies. ns ns ns ns ns ns ns ns ns ns		

図 3-35. 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 のパラメーター設定内 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 3-36. 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 インターフェイス *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 インターフェイス h2f_lw_axi_clock の設定を行います。 h2f_lw_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_lw_axi_clock に clk_0 を接続します。





図 3-37. クロックとエクスポート信号の設定

この時点で *clk0* が接続されたので、図 3-23 において *Message Window* に出ていた<u>2つのエラーメッセー</u> ジが無くなっているはずです。 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 インターフェイスが、 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 インターフェイスが、 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 インターフェイスが、 *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 インターフェイスが、 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 インターフェイスが接続されます。この設定で Arm® プロセッサー から FPGA 側の onchip_memory へ アクセスすることができます。

Ħ	System Contents 🖇	ৈ Address Map  ଛ Inter soc_system <b>Path</b> : onchi	roonnect Requirements     ≋ p_memory2_0.s1					
1	U Connections	Name	Description	Export	Clock	Base	End	
		□ clk_0 clk_in clk_in_reset clk_reset □ onchip_memory2_0	Clock Source Clock Input Reset Input Clock Output Reset Output On-Chip Memory (RAM Clock Input	clk reset Double-click to Double-click to	exported dk_0			
-	Si Connections: onchip_memory2_0.s1			Export as: onchip_memory2_0_s1 hps_0.h2f_axi_master				
	onchip_memory2_0 の s1 を選択し右クリック 表示された接続サブメニューから接続先を指定							

図 3-38. 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 インターフェイスが接続されます。 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 インターフェイスが接続されます。
- ____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 インターフェイスが接続されます。


_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 ▼▼▲▼ 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。

<b>1</b> 23	Syste	m Contents ک	ै Address Map 🛛 Inte	rconnect Requirements 🛛				
	<b>Z</b>	🛛 🕷 System:	soc_system Path: hps_(	).h2f_lw_axi_master				
1	U	Connections	Name	Description	Export	Clock	Base	End
× 1			□ <b>clk_0</b> clk_in clk_in_reset clk _clk_reset	Clock Source Clock Input Reset Input Clock Output Reset Output	clk reset Double-click to export Double-click to export	<b>exported</b> dk_0		
• •			<b>¤[®] hps_0</b> memory hps_io h2f_reset h2f_axi_dock h2f_axi_master h2f_lw_axi_clock	Arria V/Cyclone V Hard Process Conduit Conduit Reset Output Clock Input AXI Master Clock Input	memory hps_io h2f_reset Double-click to export Double-click to export Double-click to export	clk_0 [h2f_axi_clock] clk_0		
			h2flw aximaster	AXI Master	Double-click to export	[h2flwaxiclock]		
			□ onchip_memory2_0 clk1 s1 reset1	On-Chip Memory (RAM or ROM) Clock Input Avalon Memory Mapped Slave Reset Input	Double-click to export Double-click to export Double-click to export	<b>clk_0</b> [clk1] [clk1]	• 0×0	0x0000_ffff
			□ <b>led_pio</b> clk reset s1 external connection	PIO (Parallel I/O) Intel FPGA IP Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export led pic external connection	<b>ak_0</b> [ak] [ak]	• 0×1_0040	0×0001_005f
			□ dipsw_pio clk reset s1 external connection	PIO (Parallel I/O) Intel FPGA IP Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export dipsw pio external connection	<b>alk_0</b> [alk] [alk]	• 0x1_0080	0×0001_008f
			□ button_pio clk reset s1 external_connection	PIO (Parallel I/O) Intel FPGA IP Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export button_pio_external_connection	<b>clk_0</b> [dk] [dk]	• 0x1_00c0	0x0001_00cf

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

図 3-39. 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ーは 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) = <u>0xFF21_0040</u>

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・ インターフェイスを**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 3-40. 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 3-41 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。

■ Pg hps_0 memory hps_io h2f_reset h2f_axi_dock h2f_axi_dock h2f_axi_dock	Arria V/Cyclone V Hard Process. Conduit Conduit Reset Output Clock Input AXI Master Clock Input	memory hps_io h2f_reset Double-click to export Double-click to export Double-click to export	clk 0 [h2f.axi_clock] clk 0		
h2flw axi mas	er AXI Master	Double-click to export	[h2t_lw_axi_clock]		
□ □ onchip_memor	2_0 On-Chip Memory (RAM or ROM).				
<b>♦         →</b>   clk1	Clock Input	Double-click to export	clk_0		
l l <b>♦ <del>I</del> → s</b> 1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	≜ 0x0	0x0000_ffff
reset1	Reset Input	Double-click to export	[dk1]		_
□ led pio	PIO (Parallel I/O) Intel FPGA IP				
	Clock Input	Double-click to export	clk 0		
reset	Reset Input	Double-click to export	[dk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[dk]	● 0×1 0040	0x0001 005f
external_conne	ction Conduit	led_pio_external_connection		<u> </u>	_

図 3-41. 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 3-39. 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® 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				
	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				
has 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 Platform Designer システムの接続状況

_2. Platform Designer の View メニュー  $\Rightarrow$  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 3-42. 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。



図 3-43. プロジェクト・パラメーターの設定



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。

🏅 Messa	ages 🛛	
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
	soc_system.hps_0	1 or more output clock frequencies cannot be achieved precisely, consider revising desired output clock frequencies.
	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.hps_0	HPS Main PLL counter settings: n = 0 m = 73
	soc_system.hps_0	HPS peripherial PLL counter settings: n = 0 m = 39
0	soc_system.led_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

図 3-44. 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。

AII 😣 🛆 🕕		
<ul> <li>Info: C:¥intelFF</li> <li>Info: C:/intelFF</li> <li>Info: Reading ind</li> <li>Info: C:¥intelFF</li> <li>Info: C:/intelFF</li> <li>Info: C:/intelFF</li> <li>Info: C:¥intelFF</li> <li>Info: C:/intelFF</li> <li>Info: C:/intelFF</li> </ul>	GA¥18.1¥quartus¥sopc_builder¥builtir GA/18.1/quartus/sopc_builder/**/*r × C:¥intelFPGA¥18.1¥quartus¥commo GA¥18.1¥quartus¥common¥librarian¥1 GA/18.1/quartus/common/librarian/1 GA/18.1/quartus/sopc_builder/bin/\$1 GA/18.1¥quartus¥sopc_builder/bin/rc GA/18.1/quartus/sopc_builder/bin/rc	n.il ma bn fac fac IP pot vot >
9 Save System: co	npleted successfully.	
	Clos	se

図 3-45. Platform Designer システムの保存

___3. Generate メニュー ⇒ Generate HDL を選択します。



図 3-46. 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 X
Synthesis
Synthesis files are used to compile the system in a Quartus project.
Create HDL design files for synthesis: Verilog 🗸
Create timing and resource estimates for third-party EDA synthesis tools.
└── └── └── └── └── └── └── └── └── └──
* Simulation
The simulation model contains generated 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 <i>ip-setup-simulation</i> and <i>ip-make-simscript</i> command-line utilities to compile all of the files needed for simulating all of the IP in your design.
Create simulation model: None V
Output Directory
Path: C:/lab/soc_lab/cv_soc_lab/soc_system
Generate

図 3-47. 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 instance of <b>soc_system</b> . HDL Language: Verilog ✓ Example HDL	
<pre>soc_system u0 (     .button_pio_external_connection_export (<connected-to-button_pio_external_connection_export), (<connected-to-clk_clk="" .clk_clk="">),     .dipsw_pio_external_connection_export (<connected-to-dipsw_pio_external_connection_export), (<connected-to-hps_io_hps_io_emacl_inst_tx_clk="" .hps_io_hps_io_emacl_inst_tx_clk="">),     .hps_io_hps_io_emacl_inst_TXD0 (<connected-to-hps_io_hps_io_emacl_inst_txd0)), (<connected-to-hps_io_hps_io_emacl_inst_txd1="" .hps_io_hps_io_emacl_inst_txd1="">),     .hps_io_hps_io_emacl_inst_TXD2 (<connected-to-hps_io_hps_io_emacl_inst_txd2), (<connected-to-hps_io_hps_io_emacl_inst_txd2="" .hps_io_hps_io_emacl_inst_txd3="">),     .hps_io_hps_io_emacl_inst_RXD0 (<connected-to-hps_io_hps_io_emacl_inst_txd3), (<connected-to-hps_io_hps_io_emacl_inst_rxd3="" (<connected-to-hps_io_hps_io_emacl_inst_txd3),="" .hps_io_hps_io_emacl_inst_rxd0="">),     .hps_io_hps_io_emacl_inst_RXD0 (<connected-to-hps_io_hps_io_emacl_inst_rxd3>),     .hps_io_hps_io_emacl_inst_RXD0 (<connected-to-hps_io_hps_io_emacl_inst_rxd3>),     .hps_io_hps_io_emacl_inst_RXD0 (<connected-to-hps_io_hps_io_emacl_inst_rxd3>),     .hps_io_hps_io_emacl_inst_RXD0 (<connected-to-hps_io_hps_io_emacl_inst_rxd3>),     .hps_io_hps_io_emacl_inst_RXD0 (<connected-to-hps_io_hps_io_emacl_inst_rxd3>),     .hps_io_hps_io_emacl_inst_RXD0 (<connected-to-hps_io_hps_io_emacl_inst_rxd3>),     .hps_io_hps_io_emacl_inst_RXCTL (<connected-to-hps_io_hps_io_emacl_inst_rxctl>),     .hps_io_hps_io_emacl_inst_RX_CTL (<connected-to-hps_io_hps_io_emacl_inst_rxclk>),     .hps_io_hps_io_emacl_inst_RX_CTL (<connected-to-hps_io_hps_io_emacl_inst_rxclk>),     .hps_io_hps_io_emacl_inst_RX_CTL</connected-to-hps_io_hps_io_emacl_inst_rxclk></connected-to-hps_io_hps_io_emacl_inst_rxclk></connected-to-hps_io_hps_io_emacl_inst_rxctl></connected-to-hps_io_hps_io_emacl_inst_rxd3></connected-to-hps_io_hps_io_emacl_inst_rxd3></connected-to-hps_io_hps_io_emacl_inst_rxd3></connected-to-hps_io_hps_io_emacl_inst_rxd3></connected-to-hps_io_hps_io_emacl_inst_rxd3></connected-to-hps_io_hps_io_emacl_inst_rxd3></connected-to-hps_io_hps_io_emacl_inst_txd3),></connected-to-hps_io_hps_io_emacl_inst_txd2),></connected-to-hps_io_hps_io_emacl_inst_txd0)),></connected-to-dipsw_pio_external_connection_export),></connected-to-button_pio_external_connection_export),></pre>	
Copy Close	

図 3-48. 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[®] Prime に戻ります。

AI 🖸 🛆 🔍	
<ul> <li>Info: Reusing file C:/lab/soc_lab/cv_soc_lab/soc_system</li> <li>Info: avalon_st_adapter: "mm_interconnect_1" instantiat</li> <li>Info: border: "hps_io" instantiated altera_interface_gene</li> <li>Info: error_adapter_0: "avalon_st_adapter" instantiated e</li> <li>Info: error_adapter_0: "avalon_st_adapter" instantiated e</li> <li>Info: soc_system: Done "soc_system" with 35 modules, 93</li> <li>Info: qsys-generate succeeded.</li> <li>Info: Finished: Create HDL design files for synthesis</li> </ul>	n, ^ ə ra ** **
< >	
Generate: completed with warnings.	
Stop	e

図 3-49. Platform Designer システムの生成完了

____7. Quartus[®] 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 カテゴリーが選択された状態で表示されます)。



図 3-50. Add/Remove Files in Project を選択

____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 フォルダーを参照します。

🖉 Settings - atlas	
Category:	
General	Files
Files	Select the design files you want to include in the project. Click Add All to add all design
Libraries	project directory to the project.
✓ IP Settings IP Catalog Search Loc.	Eile name:

図 3-51. Settings ダイアログボックス



__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。

Select File								×
← → • ↑ 📜	« cv	_soc_lab > soc_system	synthesis	~	ซ	synthesisの検索	R	Q
整理 ▼ 新しいフォ	tルダー						•	?
PC	^	名前	更新日時			種類	サイズ	
📙 3D オブジェクト		📕 submodules	2020/01/16	13:06		ファイル フォルダー		
🛺 ダウンロード		soc_system.qip	2020/01/16	13:06		QIP ファイル	1,67	6 KB
<b>ニ</b> デスクトップ		soc_system.v	2020/01/16	13:05		V ファイル	6	3 KB
	771	を(N):			~	Dosign Filos (*	tdf * ubd * ubd	* ~
	, , , , , , , , , , , , , , , , , , , ,	aliav soc system dib	J		đ	開く(O)	キャンセ	JL

図 3-52. 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。

ategory:			C	Device/Boa
General	Files			
Files	Select the design files you	want to include in the project. Click Add All to	add all design f	files in the
Libraries	project directory to the project directory directory to the project directory dir	oject.		
IP Settings				
IP Catalog Search Loc	<u>F</u> ile name:			Add
Design Templates			×	
Operating Settings and C				Add A <u>l</u> l
	File Name	Туре	Library	Remove
Voltage				Mennove
Voltage Temperature	soc_system/synthesis/s	Soc system.did IP variation File (.did)		
Voltage Temperature ' Compilation Process Set	soc_system/synthesis/s	verilog HDL File	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	Up

## __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[®] 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 インターフェイス 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[®] Prime の *Processing* メニュー ⇒ *Start ⇒ Start Analysis & Synthesis* を選択します。

File Edit View Project A	ssignments Pro	cessing Tools Window Help			
	atlas 🔍	Stop Processing	Ctrl+Shift+C		
Project Navigator 🛛 🔥	Hierarchy 🕨	Start Compilation	Ctrl+L		
Entity:In	stance	Analyze Current File			
Cyclone V: 5CSEMA4U23C	6	Start	•	🐁 Start Hierarchy Elaboration	
ghrd_top ^h		Update Memory Initialization File		🔸 Start Analysis & Elaboration	
	$\Leftrightarrow$	Compilation Report	Ctrl+R	🍾 Start Analysis & Synthesis	Ctrl+K
	$\Theta$	Dynamic Synthesis Report		🐕 Start Partition Merge	
	😵 Quartus Prin File Edit View	e Standard Edition - C:/lab/soc_lab/cv_sc Project Assignments Processing T	oc_lab/soc_system Tools Window H	- atlas Ielp	

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

図 3-54. 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。



図 3-55. Start Analysis & Synthesis の正常終了確認



_3. Quartus[®] Prime の Tools メニュー ⇒ TCL scripts を選択します。



図 3-56. Tcl Scripts ウィンドウの起動

____4. soc_system  $\Rightarrow$  synthesis  $\Rightarrow$  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。

🛩 🏲 Project	^	Edit
✓ ► soc_system		Add to Droject
💙 📂 synthesis		Add to Project.
🗙 📂 submodules		
hps sdram p0 parameters.tcl		
hps_sdram_p0_pin_assignments.tcl		
hps_sdram_p0_pin_map.tcl	~	
review:		
# (C) 2001-2018 Intel Corporation. All rights reserved	d.	
# Your use of Intel Corporation's design tools, logic fu	inctions and oth	ier
# software and tools, and its AMPP partner logic func	tions, and any o	utput
# files from any of the foregoing (including device pro	ogramming or si	imulation
# files), and any associated documentation or informa	ation are expres	sly subject

図 3-57. Tcl Script の実行



_5. 完了したら [OK] ボタンをクリックします。



- 図 3-58. Tcl Script の完了
- ___6. Tcl Scripts ウィンドウを [*Close*] します。

Tcl Scripts		>
Libraries:		
🗸 🏲 Project	^	Edit
🗸 📂 soc_system		
🗙 📂 synthesis		Add to Project
🗙 📂 submodules		
hps_sdram_p0_parameters.tcl		
hps_sdram_p0_pin_assignments.tcl		
hps_sdram_p0_pin_map.tcl	~	
Preview:		
# (C) 2001-2018 Intel Corporation. All rights reserved.		^
# Your use of Intel Corporation's design tools, logic function	ns and oth	ner
# software and tools, and its AMPP partner logic functions,	and any o	utput
# files from any of the foregoing (including device program	iming or si	imulation
# files), and any associated documentation or information a	are expres	siy subject
<		>
Run	Close	Help

図 3-59. Tcl Scripts ウィンドウの Close



__7. Quartus[®] Prime の *Processing* メニュー  $\Rightarrow$  *Start Compilation* を選択(または、GUI 上の Start Compilation ボ タン  $\blacktriangleright$  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 3-60. Start Compilation の実行

__8. コンパイルの完了を確認します。

	tate view integer Assignments	Processing 1	oots wind
0 🗖	■ + ① ① ② ⑦ atlas	• 2	<b>* * *</b> @
Project	Navigator 🔥	Hierarchy	
	Entity:Instance	l.	
\lambda Cycl	one V: 5CSEMA4U23C6		
> 🔤 gl	hrd_top 🏝		
(			3
Tasks	Comp	ilation	▼ ≡ ₽ ₽
	Task		Time
-	✓ ► Compile Design		00:06:45
-	<ul> <li>Compile Design</li> <li>Analysis &amp; Synthesis</li> </ul>		00:06:45 00:03:42
1	<ul> <li>Compile Design</li> <li>Analysis &amp; Synthesis</li> <li>Fitter (Place &amp; Route)</li> </ul>		00:06:45 00:03:42 00:02:05
	<ul> <li>Compile Design</li> <li>Analysis &amp; Synthesis</li> <li>Fitter (Place &amp; Route)</li> <li>Assembler (Generate program)</li> </ul>	ramming files)	00:06:45 00:03:42 00:02:05 00:00:14
• • • • •	<ul> <li>Compile Design</li> <li>Analysis &amp; Synthesis</li> <li>Fitter (Place &amp; Route)</li> <li>Assembler (Generate programmer)</li> <li>Timing Analysis</li> </ul>	ramming files)	00:06:45 00:03:42 00:02:05 00:00:14 00:00:44
	<ul> <li>Compile Design</li> <li>Analysis &amp; Synthesis</li> <li>Fitter (Place &amp; Route)</li> <li>Assembler (Generate programe)</li> <li>Timing Analysis</li> <li>EDA Netlist Writer</li> </ul>	ramming files)	00:06:45 00:03:42 00:02:05 00:00:14 00:00:44
· · · · · · · · · · · · · · · · · · ·	<ul> <li>Compile Design</li> <li>Analysis &amp; Synthesis</li> <li>Fitter (Place &amp; Route)</li> <li>Assembler (Generate programe)</li> <li>Timing Analysis</li> <li>EDA Netlist Writer</li> <li>Edit Settings</li> </ul>	ramming files)	00:06:45 00:03:42 00:02:05 00:00:14 00:00:44

図 3-61. 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[®] Prime および Platform Designer で出力したファイルを確認します。

- 1. Windows[®] OS のエクスプローラーを使用して、出力ファイルのフォルダー(下記)に移動します。 C:¥lab¥soc_lab¥cv_soc_lab¥output_files
- 2. 上記フォルダーに .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[®] OS のエクスプローラーを使用して、ハードウェア/ソフトウェアのハンドオフ・ディレクトリーに移動し ます。

C:¥lab¥soc_lab¥cv_soc_lab¥hps_isw_handoff¥soc_system_hps_0

上記フォルダー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 インターフェイ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。

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® プロセッサー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[®] 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. 演習 2: ソフトウェア演習(1) Preloader の生成

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[®] SoC FPGA 向けにカスタ マイズが加えられたブートローダーです。 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[®] 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、ユーザー側で初期化用ソフトウェアの構築をすることなく Quartus[®] 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. 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® のスタートメニュー、または SoCEDS のインストール・フォルダー以下 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。



図 4-1. Embedded Command Shell の起動



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

__1. <mark>Embedded Command Shell</mark> から **bsp-editor** とタイプし、bsp-editor(Preloader Generator)の GUI を起動 します。

\$	bsp-editor 🗸
	₩. ~
	Intel FPGA Embedded Command Shell
١	Version 18.1 [Build 625]
	11149@HD11149B ~ \$ bsp-editor

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、フォルダーを指定します。

指定するフォルダーは、c:¥lab¥soc_lab¥cv_soc_lab¥hps_isw_handoff¥soc_system_hps_0 です。 別のプロジェクト・ディレクトリーを指定している場合は、プロジェクト・ディレクトリー以下の ¥hps_isw_handoff¥soc_system_hps_0 です。

🚵 New BSP	×
Hardware Preloader settings directory:	
Software Operating system: BSP target directory: BSP Settings File name:	U-Boot SPL Preloader (Cyclone V/ Version: default V Use default locations
Additional Td script:	
	OK Cancel
📫 Open	×
Look in: Soc_sys	tem_hps_0 🗸 🦻 📴 •
最近使… デスクトップ ドキュメント PC ネッドワーク Folder games Files of types	\soc_lab\cv_soc_lab\hps_isw_handoff\soc_system_hps_0       Open         Preloader settings directory       Cancel
New BSP	×
Hardware Preloader settings directory:	C:\Jab\soc_lab\cv_soc_lab\hps_isw_handoff\soc_system_hps_0
Software Operating system: BSP target directory: BSP Settings File name: Additional Td script:	U-Boot SPL Preloader (Cyclone V/ Version: default U-Boot SPL Preloader (Cyclone V/ Version: default U-Boot SPL Preloader (Cyclone V/ Version: default U-Boot SPL Preloader (Cyclone V/ Version: default Sectors (Cyclone V/ Version: default (Cyclone Versi
	OK





__3. Preloader のユーザーオプション(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。

le Edit Tools Help		
ain Software Packages   Drivers   Linker Script   Enab	le File Generation Target BSP Directory	
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone BSP target directory: .\	V/Arria Version: default 🔨	
Settings Common spl PRELOADER_TGZ PROSS_COMPILE boot BOOT_FROM_SOMMC BOOT_FROM_SOMMC BOOT_FROM_NAND BOOT_FROM_RAM QSPL_NEXT_BOOT_IMAGE SOMMC_NEXT_BOOT_IMAGE NAND_NEXT_BOOT_IMAGE FAT_SUPPORT FAT_BOOT_PARTITION FAT_BOOT_PARTITION FAT_DOT_PARTIDON_IMME	SPI PRELOADER_TGZ: CROSS_COMPILE: Spl.boot BOOT FROM QSPI BOOT FROM SDMMC BOOT FROM NAND BOOT FROM NAND BOOT FROM RAM QSPI_NEXT_BOOT_IMAGE: SDMMC_NEXT_BOOT_IMAGE:	preloader/uboot-socfpga.tar.gz arm-altera-eabi-
Advanced	NAND_NEXT_BOOT_IMAGE: FAT_SUPPORT FAT_BOOT_PARTITION: FAT_LOAD_PAYLOAD_NAME:	0xc0000

図 4-5. ユーザーオプションの確認

___4. Preloader のユーザーオプション(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<u>チェックを外します</u>。

(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)。

1ain Software Packages   Drivers   Linker Script   Enable F	File Generation   Target BSP Directory		
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone V/	Arria Version: default 🗸		
BSP target directory: .\			
AT_SUPPORT FAT_SOOT_PARTITION FAT_LOAD_PAYLOAD_NAME	SPLboot		
H warm reset_handshake	EXE ON FPGA FPGA_MAX_SIZE:	0x10000	Ĵ
WATCHDOG_ENABLE	FPGA_DATA_BASE:	0xffff0000	
EXE_ON_FPGA	FPGA_DATA_MAX_SIZE:	0x10000	 ]
FPGA_DATA_BASE	STATE REG ENABLE		
	BOOTROM HANDSHAKE CFGIO		
WARMEST SKIP CEGIO	WARMRST SKIP CFGIO		
SDRAM_SCRUBBING	SDRAM SCRUBBING		
SDRAM_SCRUB_BOOT_REGIC	SDRAM_SCRUB_BOOT_REGION_START:	0x1000000	
SDRAM_SCRUB_REMAIN_REG RAMBOOT_PLLRESET	SDRAM_SCRUB_BOOT_REGION_END:	0x2000000	 ]
ebug	SDRAM SCRUB REMAIN REGION		

図 4-6. Advanced 設定(1)



___5. Preloader のユーザーオプション(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。

BSP Editor - C:¥lab¥soc_lab	¥cv_soc	_lab¥software¥spl_bsp¥sett	ings.bsp —	
Main Software Packages Drivers Linker Script	Enable File (	Generation Target BSP Directory		
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (C BSP target directory: .\	yclone V/Arria	a Version: default 🗸		
FAT_BOOT_PARTITION FAT_LOAD_PAYLOAD_NAM Spl Freset_assert Warm_reset_handshake hoot Gebug DEBUG_MEMORY_WRITE DEBUG_MEMORY_ADDR DEBUG_MEMORY_ADDR DEBUG_MEMORY_SDZE SEMIHOSTING HARDWARE_DIAGNOSTIC SKIP_SDRAM @ performance	<ul> <li>spl.</li> <li>spl.</li> </ul>	debug         DEBUG_MEMORY_ADDR:         DEBUG_MEMORY_SIZE:         Image: Semihosting         HARDWARE DIAGNOSTIC         SKIP_SDRAM	0x###d00 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。

ile Edit Tools Help		File Edit Tools Help		
Main Software Packages Drivers Linker Script Ena	ble File Generation Target BSP Directory	Main Software Packages Drivers Linker Script B	nable File Generation Target BSP Directory	-
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclor BSP target directory: .\	re V/Ama Version: default V	SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cycl BSP target directory: .\	one V/Arria Version: default 🗸	
Settings Common PRELOADER, TGZ - CROSS, COMPLE - BOOT, FROM, SOMMC - BOOT, FROM, SOMMC - BOOT, FROM, SOMMC - BOOT, FROM, SOMMC - BOOT, FROM, MAND - BOOT, FROM, MAND - SOMMC, NEXT, BOOT, JMAGE - FAT, BOOT, DARTITION - FAT, BOOT,	spl.debug         DEBUG_MEMORY_ADDR:       Dxffffa00         DEBUG_MEMORY_SIZE:       0x200         Semi-osting       Semi-osting         HARDWARE       DAGNOSTIC         SKIP SDRAM       SI	Settings Common PRELOADER_TGZ 	SpLdebug	]
Information Problems Processing		Information Problems Processing		
<ol> <li>Searching for BSP components with category: driver Searching for BSP components with category: software Added operating system component "spil.1.0".</li> <li>Generated file "C: Vab/soc_Jab/sot_ware</li> </ol>	_element are_package_element spl_hap/settings.bsp"	Tcl message: "Generating file: C:/ab/soc_lab/cv_g     Tcl message: "Generating file: C:/ab/soc_lab/cv_g     Tcl message: "Reading file: C:/ab/soc_lab/cv_g     Tcl message: "Generating file: C:/ab/soc_lab/cv_g     Tcl message: "Generating file: C:/ab/soc_lab/cv_g     Tcl message: "Generating file: C:/ab/soc_lab/cv_g	oc_lab/software/spl_bos/generated Wokefie.template* oc_lab/software/spl_bos/generated/pli_config_h* Jab/spa_jow_handoffsoc_system_thos_0/soc_system_thos_0.hinf* oc_lab/software/spl_bos/generated/socs_config_cyclone.5.h* oc_lab/software/spl_bos/generated/socs_config_cyclone.5.c*	

図 4-8. bsp プロジェクトの生成

これにより Software フォルダー下の 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・ディレクトリーを bsp プロジェクト・ディレクトリーへ移動します。
  - \$ cd "C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp" 4



図 4-9. bsp プロジェクト・ディレクトリーへの移動

___2. make all コマンドを実行し Preloader を生成します。



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 preloader-mkpimage.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・メモリー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。



## 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™ 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 メモリー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[®] プロセッサーから 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。

## ▲ 注記:

- この演習を行う前に、Linux®(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[®] 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。
- ヘアメタル・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。

   <u>紐づけられているネットワーク・インターフェイスを PC に認識させておいてください。</u>

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 アダプターを接続してください。

- ____1. Quartus[®] 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 - [Chain1.cdf]	]		- 0	×
<u>File Edit V</u> iew P <u>r</u> ocessing	<u>T</u> ools <u>W</u> indow <u>H</u> elp		Search altera.com	۲
👗 Hardware Setup 🛛 No	👋 Hardware Setup			×
Enable real-time ISP to	Hardware Settings JTA	G Settings		
File	Select a programming hard hardware setup applies onl Currently selected hardware	ware setup to use when prog y to the current programmer e: No Hardware	ramming devices. This prog window.	gramming
	Hardware	DE-SoC [USB-1] Server Port	Add	Hardware
	DE-SoC	Local USB-1	Remov	ve Hardware
				Close

☑ 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。

<ul> <li>Programmer - [Chain2.cdf]</li> <li>File Edit View Processing Tools Window Help</li> </ul>	<ul> <li>Quartus Prime Programmer Standard Edition - [Chain1.cdf]</li> <li>File Edit View Processing Tools Window Help</li> </ul>
▲ Hardware Setup DE-SoC [USB-1] Mode: JTAG ・ Enable real-time ISP to allow background programming when available Start Start Stop Auto Dete X Delete Add File. Change Fil Save File	Hardware Setup DE-SoC [USB-1] Mode: JTAG      Hardware Setup DE-SoC [USB-1] Mode: JTAG      Fnable real-time ISP to allow background programming when available     Select Device     Select Device     SocsEBA66     SocsEBA6     SocsEBA6
	Add Devic 1 th Up 1 th Down

図 5-2. デバイスの選択



_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。



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</u> <u>atlas.sof</u> を、DE10-Nano ボードの場合は DE10-Nano.sof</u> を選択します。

🁋 Programm	er - [Chain2.cdf]*		×	Delete Select All	Del Ctrl+A
File Edit Vie	w Processing	Tools Windo	<b>M</b> n	Add File	
* Hardware	Setup DE Se		10	Change File	
	DE-30	C[03B-1]	ę	Save File	
Enable rea	al-time ISP to allo	w background		Add IPS File	
	File	Device		Change IPS File	
™ Start				Delete IPS File	
#h Stop	<none></none>	SOCVHPS		Add EKP File	
Auto Dete	<none></none>	5CSEMA4		Change EKP File	
Y Delate		4		Delete EKP File	
* Delete	く 右クリック	ל		Add PR Programming File	
Select New Look in: My Com 2 11149	w Programming F	ile lab\cv_soc_lab	\out	out_files • 🗘 🔾 🚺	*
File <u>n</u> ame:	atlas.sof	-11 /2 / 2			<u>O</u> pen
Files of type	Programming	-ites (*.sof *.pof	r.jar	п ^.jbc ^.eкр *.jic)  ▼	ancet

図 5-4. sof ファイルの選択



___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。

Programm <u>File</u> Edit Vie	er - [Chain2.cdf]* w P <u>r</u> ocessing <u>T</u>	ools <u>W</u> indow	<u>H</u> elp						Sea	— rch alter	a.com
📩 Hardware	Setup DE-SoC	[USB-1] / background pr	ogramming w	Mo hen available	ode: JTAG			▼ Pr	ogress:		
Start Stop	File <none> C:/lab/soc_lab/</none>	Device SOCVHPS 5CSEMA4U23	Checksum 00000000 01A7283D	Usercode <none> 01A7283D</none>	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP
<ul> <li>★ Delete</li> <li>Add File.</li> <li>Change Fil</li> <li>Save File</li> <li>Add Device</li> <li>Add Device</li> <li>1[™] Up</li> <li>1[™] Up</li> </ul>		SCSEMA4U2	3								
					Z						
👋 Programme	er - [Chain2.cdf]*									_	

e <u>E</u> dit <u>V</u> i	ew P <u>r</u> ocessing ]	<u>F</u> ools <u>W</u> indow	<u>H</u> elp						Sea	rch alter	ra.com
<mark>.</mark> Hardware	e Setup DE-So	C[USB-1]		Mo	ode: JTAG			- Pr	ogress:	100% (	Successfu
] Enable re	al-time ISP to allo	w background pr	ogramming w	hen available							
					1						
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP
Start Stop	File <none></none>	Device SOCVHPS	Checksum	Usercode <none></none>	Program/ Configure	Verify	Blan <mark>k-</mark> Check	Examine	Security Bit	Erase	ISP CLAMP

図 5-5. sof 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

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[®]のスタートメニュー、または SoC EDS のインストール・フォルダー以下 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。





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[®] SoC FPGA Edition 用の環境変数が設定されます。



#### 図 5-7. DS-5[™]の起動と起動画面



____3. Eclipse ツールを使用するワークスペース・フォルダーを設定します。

この演習では、「<u>3. 演習 1: ハードウェア演習</u>」の作業フォルダーに 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 (フォルダー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

🖨 Eclipse Launcher	×
Select a directory as workspace Eclipse プラットフォーム uses the workspace directory to store its preferences and development artifacts.	
ワ−クスペ−ス(W) C:¥lab¥soc_lab¥cv_soc_lab¥workspace ✓ 参照(B) □ 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(U) ・ Recent Workspaces	
ОК <i><b>キャンセル</b></i>	

図 5-8. DS-5™ 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。

0	workspace	- C/C++ - E	clipse プラットフォー <i>L</i>	A.							Х
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(R)	ウィンドウ(W	1) ヘル	プ <mark>(H</mark> )	
8	🛍 DS-5 🔨	こうこ 💌								(i) <>	
_	≡	閉じ	3	arm	Deve	loper		C	2	-	^
		W Dev	/elcor velop	ne t mer	to /	Arm Stud	® [ io:	)S Int	5 el [°]	R	Feedback

図 5-9. DS-5™の Welcome 画面

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。

) مرز	workspace - C/C++ - Eclipse プラ イル(F) 編集(E) ソース(S) リファクタ	ットフォーム 7リング(T) ナビゲート(N) 木
	新規(N) ファイルを開く(.) Open Projects from File System	Alt+シフト+N >
	閉じる(C) すべて閉じる(L)	Ctrl+W Ctrl+シフト+W
9 14 69	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
2	移動(V) 名前を変更(M) 更新(F) 行区切り文字の変換(D)	F2 F5 >
4	印刷(P) ワークスペースの切り替え(W) 再開	Ctrl+P
21 21	インポート(I) エクスホート(O)	

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。

(━ インポート		×
<b>選択</b> ア−カイブ・ファイルまたはディレクトリ−から新規プロジェクトを作成します。		Ľ
Select an import wizard:		
フィルタ入力		
<ul> <li>◇ ▷ 一般</li> <li>③ アーカイブ・ファイル</li> <li>◎ フォルダーまたはアーカイブ由来のプロジェクト</li> <li>◎ 既存プロジェクトをワークスペースへ</li> <li>□ 設定</li> <li>&gt; ▷ C/C++</li> <li>&gt; ▷ CVS</li> <li>&gt; ▷ DS-5</li> <li>&gt; ⊘ Git</li> </ul>		~
⑦ < 戻る(B) 次へ(N) > 終了(F)	キャンセ	JL





_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・フォルダーにあります。

C:¥intelFPGA¥18.1¥embedded¥examples¥software¥Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz

(<SoC EDS インストール・ディレクトリー>¥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ポート</b> 既存の Eclipse プロジェクトを検索	さ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(T):			$\sim$	参照(	R)
●アーカイブ・ファイルの選択(A):	$C: \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\$	CFPGA-HelloWorld-Baremetal-GNU.tar	.gz ~	参照(	R)
プロジェクト(P):					
Altera-SoCFPGA-HelloW	orld-Baremetal-GNU(Altera-SoCFPGA-HelloWorld-Baremetal-C	SNU/)		すべて選	択(S)
				選択をすべて	(解除(D)
				更新(	(E)
オプション ✓ ネストしたプロジェクトを検索(ト ✓ プロジェクトをワークスペースにコ □ ワークスペースに既に存在するご	) ピー(C) 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(T)			新規(W	)
ワーキング・セット(O):			×	選択(E)	)
0	< 戻る(B	) 次へ(N) > 終了(F	)	キャンセ	∑J↓

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





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[™] 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。



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。



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[™] 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[®] FPGA ダウンロード・ケーブル (USB-Blaster[™]) 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

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。



図 5-15. デバッグの構成



_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。



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		Ŕ	Š.
	名前(N): Altera-SoCFPGA-HelloWorld-Baremetal-Debug		
<ul> <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++ ボストモーテム・デパッガー</li> <li>C/C++ ポストモーテム・デパッガー</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>Altera-SoCFPGA-HelloWorld-Baremetal-Debug</li> <li>* IronPython Run</li> <li>* IronPython unittest</li> <li>Java アプリケーション</li> </ul>	<ul> <li>◆ 接続</li></ul>		^
<ul> <li>Java アブレット</li> <li>JUnit</li> <li>Jython run</li> <li>Jython unittest</li> <li>PyDev Django</li> <li>PyDev Google App Run</li> </ul>	<ul> <li>Bare Metal Debug</li> <li>Debug Cortex-A9_0</li> <li>Debug Cortex-A9_1</li> <li>Debug Cortex-A9x2 SMP</li> </ul>	~	
<ul> <li>● Python Run</li> <li>● Python unittest</li> <li>□, リモート Java アプリケーション</li> <li>▶ 起動グループ</li> </ul>	DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application. 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。"d	参照 lefault" コンフ・	~
フィルター一致: 19 / 19 項目	前回保管した状態に戻す(V)	適用(Y)	
0	デバッグ(D)	閉じる	

### 図 5-17. デバッグの実行



_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。



図 5-18. パースペクティブスイッチの確認

Windows Defender 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	×
でのアプリの機能のいくつかが Windows Defender 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 Defender ファイアウオールにより の機能のいくつかがブロックされています。 名前(1): 名前(1):	
ハベロ: このアプリは、インターネットから直接情報を受信しようとしています。ファイアウォールをパイパスしようとしている可能 性があります。 にこれらのネットワークトでの過信を許可する。	
□ 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)(B)	
⑦パブリックネットワーク(空港、喫茶店など)(非推進)(U) (このようなネットワークは多くの場合、セキュリティが低いかセキュリティが設定されていません)	
アブリ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(A) キャンセン	k

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・インターフェイス (例えば USB-Ethernet Interface アダプター)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)。



____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

듣 workspace - DS-5 デバッグ - Altera-SoCFPGA-HelloWo	rld-Baremetal-GNU/hello.c - Eclipse プラットフォーム		- 🗆 X
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(R) ウィンドウ(W) ヘルプ(H	)	
" ▼	· • • ◆ • • •		クイック・アクセス 🗄 🖻 👼 🗞
<ul> <li>※デパッグコン ※ 応プロジェクト 週リエート・シ [□] □</li> <li>□ ● ● ● ◎ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○</li></ul>	■コマンド ²⁰ ■履歴 ²⁶ スクリプト ²⁰ ²⁶ ²⁷ ²⁷ ²⁷ ²⁷ ²⁷ ²⁷ ²⁷ ²⁷	FF3	there 変数 S
ステータス: application exit: code 0	אעדב:	送信	変数の追加 参照
<pre>     Hello.c 33     2* * Copyright Altera 2013,2014     7     8 #include <stdio.h>     9     10 /* enable semihosting with gcc by def     11 intauto_semihosting;     12     13=int main(int argc, char** argv) {</stdio.h></pre>	ining anauto_semihosting symbol */		III 遊7t ☆ 日メモリ = スタック 日イベ ほアウ ● ロンク落み: Altera-SoCFPGA-HelloWorld-Raremetal-Debu ◎ ③ ▼ (<次の命令> 100 ● アドレス オペコード 逆7t977 リ S: 0xFFFF3430 E59F1004 LDR r1, [pc, ~ ◆ S: 0xFFFF3438 EAFFFFE B ● S: 0xFFFF3438 EAFFFFE B ■ App Console ☆ ■ ターゲットコン ⑨ Iラー・ログ ⁻ □ ■ App Console ☆ ■ ターゲットコン ⑨ Iラー・ログ ⁻ □ ■ App Console ☆ ■ ターゲットコン ⑨ Iラー・ログ ⁻ □ ■ App Console ☆ ■ ターゲットコン ⑨ Iラー・ログ ⁻ □ ■ App Console ☆ ■ ターゲットコン ⑨ Iラー・ログ ⁻ □

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。



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。



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。

(⇔ インポート	-		×
選択 アーカイブ・ファイルまたはディレクトリーから新規プロジェクトを作成します。			Ľ
Select an import wizard:			
フィルタ入力			
<ul> <li>◇ ○ 一般</li> <li>③ アーカイブ・ファイル</li> <li>○ ファイル・システム</li> <li>○ フォルダーまたはアーカイブ由来のプロジェクト</li> <li>○ 既存プロジェクトをワークスペースへ</li> <li>□ 設定</li> <li>&gt; ○ C/C++</li> <li>&gt; ○ CVS</li> <li>&gt; ○ PC 5</li> </ul>			^
> 🗠 DS-5 > 🔄 Git			~
⑦ < 戻る(B) 次へ(N) > 終了(F)		キャンセ	ル



_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・ディレクトリーではなく <u>演習データのディレクトリー以下である</u> 3	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					– 🗆 X
プロジェクトのインポート 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(T):					参照(R)
●アーカイブ・ファイルの選択(A):	rイルの選択(A): C:¥lab¥soc_lab¥cv_soc_lab¥software_example¥Atlas-Blinking-LED-Baremetal-GNU.tar.gz				~ 参照(R)
プロジェクト(P):					
Atlas-Blinking-LED-Baremetal-GNU(Atlas-Blinking-LED-Baremetal-GNU/)					すべて選択(S)
					選択をすべて解除(D)
					更新(E)
オプション					-
✓ ネストしたプロジェクトを検索(H)					
✓ フロジェクトをワークスペースにコヒー(C)					
□ ワーキング・セットにプロジェクトを追加(T)					新規(W)
ワーキング・セット(O):					選択(E)
0		< 戻る(B)	次へ(N) >	終了(F)	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・エクスプローラー 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。





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^M 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。

듣 デバッグ構成	>	<
構成の作成、管理、および実行	1	-
<ul> <li>○ ● ● ● ● ● ●</li> <li>○ / レタ入力</li> <li>○ / レタ入力</li> <li>○ / (レタ入力)</li> <li>○ / (レ + アプリケーションへのアタッチ)</li> <li>○ (ノ + + アプリケーションへのアタッチ)</li> <li>○ (ノ + + リモート・アプリケーション)</li> <li>◆ Attas-Blinking-LED-Baremetal-Debug</li> <li>◆ Attas-Blinking-LED-Baremetal-Debug</li> <li>◆ Attas-Blinking-LED-Baremetal-Debug</li> <li>◆ IronPython nuittest</li> <li>□ Java アプリケーション</li> <li>&gt;&gt; Java アプリケーション</li> <li>&gt;&gt; Java アプレット</li> <li>&gt;&gt; Unit</li> <li>● ' Jython run</li> <li>●' Jython run</li> <li>● Y Dev Django</li> <li>&gt;&gt; PyDev Google App Run</li> <li>● Dython Bun</li> </ul>	名前(N): Atlas-Blinking-LED-Baremetal-Debug ◆接続 ◎ ファイル ◆ デバッガ ◎ OS 認識機能 ◎ 引数 ■ 環境 ◎ エクスボート 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 ブラットフオームのフィルタ ◆ Altera > Arria 10 SoC > Cyclone V SoC (Dual Core) • Bare Metal Debug Debug Cortex-A9_0 Debug Cortex-A9_1 Debug Cortex-A9x2 SMP ダーグット接続 USB-Blaster ✓ DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application.	
<ul> <li>Python unitest</li> <li></li></ul>	接続 Bare Metal Debug Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1 参照 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" コンフィギュレ く	
フィルターー致: 20 / 20 項目	前回保管した状態に戻す(V) 適用(Y)	
0	デパッグ(D) 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。

(= 読み耳	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-Debug.launch' は読 み取り専用です。書き込み可能にしますか?	
	はい(Y) いいえ(N)	



- ___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。

듣 接続ブラ	ウザ	×
接続ブラウ	ቻ	
ターゲット接	続を選択します	
DE-SoC US	R-1	
DE-SoC	on localhost [USB-1]	
?	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。

(⇒ デバッグ構成			×
構成の作成、管理、および実行		Ť	5
			7
□ 圖 ¥  E 歩 ▼ フィルタ入力	名前(N): Atlas-Blinking-LED-Baremetal-Debug		
<ul> <li>C/C++ アブリケーション</li> <li>C/C++ アブリケーションへのアタッチ</li> <li>C/C++ ポストモーテム・デバッガー</li> <li>C/C++ ポストモーテム・デパッガー</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		^
◆ DS-5デバッガ ◆ Altera-SoCFPGA-HelloWorld-Baremetal-Debug 参 Attas-Blinking-LED-Baremetal-Debug	「ブラットフォームのフィルタ → Altera → Arria 10 SoC	^	
e [*] IronPython Run e [¥] IronPython unittest ₪ Java アプリケーション	<ul> <li>&gt; Arria V SoC</li> <li>~ Cyclone V SoC (Dual Core)</li> <li>~ Bare Metal Debug</li> </ul>		
◎ Java アブレット Je JUnit ● Jython run	Debug Cortex-A9_0 Debug Cortex-A9_1 Debug Cortex-A9x2 SMP	~	
Synon unittest     PyDev Django     PyDev Google App Run	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.		
<ul> <li>✓ Python unitest</li> <li>型, リモート Java アプリケーション</li> <li>▶ 起動グループ</li> </ul>	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。"default" コ	参照	~
<ul><li></li><li>フィルター一致: 20 / 20 項目</li></ul>	前回保管した状態に戻す(V)	適用(Y)	
0	デバッグ(D)	閉じる	

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。



図 5-29. パースペクティブスイッチの確認

Windows Defender 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。

Wind Wind	lows セキュリティの重要な警告	X
۲	このアプリの機能のいくつかが Windows Defender ファイアウォールでプロックされ ています	
<b>す</b> べての/	(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 Defender ファイアウォールにより の機能のいくつかがブロックされています。 名前(12):	
	条行元(P): /(ス(出):	
このアプリ 性がありま	は、インターネットから直接情報を受信しようとしています。ファイアウォールをバイバスしようとしている可能 ます。	
<b>D</b> 7	にこれらのネットワーク上での通信を許可する: 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)( <u>R</u> )	
	ブリックネットワーク(空港、粤茶店など)(非推奨)(U) このようなネットワークは多くの場合、セキュリティが低いかセキュリティが設定されていません)	
アプリにフ	アイアウォールの経由を許可することの危険性の詳細	1.14
	♥アクセスを許可する(A) キャンセル	

図 5-30. セキュリティの警告



(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。



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 ボード上のユーザー 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 ディレクトリーの下に、"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。

📙 🛛 🔁 📮 🛛 spl				- 🗆 🗙
ファイル ホーム 共有 表示	Ā			~ 📀
← → ∽ ↑ 📜 C:¥lab¥soc	lab¥cv_soc_lab¥software¥sp	l_bsp¥uboot-socfpga¥spl 🗠	<b>ひ</b> splの検索	م
lab ^	名前	更新日時	種類	サイズ
<pre>soc_lab   cv_soc_lab</pre>	<pre>arch board</pre>	2020/01/16 18:00 2020/01/16 18:01	ファイル フォルダー ファイル フォルダー	
.qsys_edit	common	2020/01/16 18:01	ファイル フォルダー	
db hps_isw_handof	drivers	2020/01/16 18:02 2020/01/16 18:02	ファイル フォルダー ファイル フォルダー	
incremental_db	spl	2020/01/16 18:00	ファイル フォルダー	0 KB
ip	gitignore	2018/09/14 00:22	GITIGNORE 77	1 KB
soc_system	Makefile	2018/09/14 00:22 2020/01/16 18:02	ファイル LST ファイル	6 KB 0 KB
software	📄 u-boot-spl	2020/01/16 18:02	ファイル	574 KB
📜 spl_bsp	u-boot-spl.bin	2020/01/16 18:02	BIN ファイル	36 KB
🤁 generated	u-boot-spl.lds	2020/0 <mark>1</mark> /16 18:02	LDS ファイル	1 KB
📜 uboot-socfp	📄 u-boot-spl.map	2020/01/16 18:02	MAP ファイル	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、ユーザー・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。

圖 参考: 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 ディレクトリーの下にある、"u-boot-spl" ファイ ルを、C:¥lab¥soc_lab¥cv_soc_lab¥workspace¥Atlas-Blinking-LED-Baremetal-GNU ディレクトリーにコピーします。

____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>73</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。</u>"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。

U-Boot SPL 2013.01.01 (Jan 16 2020 - 17:59:35) BOARD : Altera SOCFPGA Cyclone V Board CLOCK: EOSC1 clock 25000 KHz CLOCK: EOSC2 clock 25000 KHz CLOCK: F2S SDR REF clock 0 KHz CLOCK: F2S PER REF clock 0 KHz CLOCK: MPU clock 925 MHz CLOCK: DDR clock 400 MHz CLOCK: UART clock 100000 KHz CLOCK: MMC clock 50000 KHz CLOCK: QSPI clock 3613 KHz RESET: COLD SDRAM: Initializing MMR registers SDRAM: Calibrating PHY SEQ.C: Preparing to start memory calibration SEQ.C: CALIBRATION PASSED SDRAM: 1024 MiB

5-5. システム・ヘッダー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・ヘッダーファイル生成コマンド(sopc-create-header-files)を使用してシステム・ ヘッダー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. ディレクトリーの移動



3. Embedded Command Shell 上で、システム・ヘッダー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 4



図 5-35. システム・ヘッダー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ー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・ヘッダーファイルを 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 atlas_main.c 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。

le *atlas_main.c ⊠	
2⊕ * Copyright Altera 2013	72⊕/*
7	73 * Macros for device 'led pio', class 'altera avalon pio'
<pre>8 #include <stdio.h></stdio.h></pre>	74 * The macros are prefixed with 'LED_PIO_'.
9 #include "socal.h"	75 * The prefix is the slave descriptor.
10 #include "hps_0_arm_a9_0.h"	76 */
11	77 #define LED_PIO_COMPONENT_TYPE altera_avalon_pio
<pre>12 /*#define LED_BASE_ADDR (0xFF210040)*/</pre>	78 #define LED_PIO_COMPONENT_NAME led_pio
<pre>13 #define LED_BASE_ADDR LED_PIO_BASE</pre>	79 #define LED_PIO_BASE 0xff210040
14	80 #define LED_PIO_SPAN 32
150 int main(int argc, char** argv)	81 #define LED_PIO_END 0xff21005f
16 {	<pre>82 #define LED_PIO_BIT_CLEARING_EDGE_REGISTER 0</pre>
17 int i;	<pre>83 #define LED_PIO_BIT_MODIFYING_OUTPUT_REGISTER 1</pre>
18	84 #define LED_PIO_CAPTURE 0
<pre>19 printf("Hello from Atlas. \n");</pre>	85 #define LED_PIO_DATA_WIDTH 8
20	86 #define LED_PIO_DO_TEST_BENCH_WIRING 0
21 while(1)	87 #define LED_PIO_DRIVEN_SIM_VALUE 0
22 {	88 #define LED_PIO_EDGE_TYPE NONE
23 for(i=0; i < 16; i++){	89 #define LED_PIO_FREQ 50000000
<pre>24 alt_write_word(LED_BASE_ADDR,i);</pre>	90 #define LED_PIO_HAS_IN 1
<pre>25 printf("LED [%x] \n",i);</pre>	91 #define LED_PIO_HAS_OUT 1
26 }	92 #define LED_PIO_HAS_TRI 0
27 }	93 #define LED_PIO_IRQ_TYPE NONE
28	94 #define LED_PIO_RESET_VALUE 0
29 return 0;	95
30 }	

図 5-36. ソースコードの変更箇所とシステム・ヘッダーファイルの該当箇所

____6. 変更した atlas_main.c をセーブして、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[™] 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ーに解凍しておきます。解凍したフォルダー内に .img イメージファ イルがあることを確認します。
  - <u>Atlas-SoC ボード向け SD カード・イメージファイル</u>
  - <u>DE10-Nano ボード向け SD カード・イメージファイル</u>
- ____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 カードリーダー/ ライターを使用します)。 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)を確認します。

<b>S</b>	デバイスとプリンターを開く(O)
~	SDHC Card - SDHC (E:)の取り出し
8	
×	15:02 へ 15:02 本 日 代本 A 火曜日 2019/06/04

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。



	👒 Win32 Disk Imager - 1.0	- 🗆	×	
	Image File [4.14.73-Itsi_a/atlas_exte_sdimage_4.14.73-	-Itsi_a.img 🖻	Device [E:¥] ▼	
	Hash		1	
	Read Only Allocated Partitions Progress			
	Cancel Read Write	Verify Only	Exit	
🍤 Confirm ov	erwrite - 1.0	×		
Writi devi (Targ Are	ing to a physical device c in corrupt the ce. get Device: [E:¥] "") you sure you want to matinue?	(	Complete -	0 × e St tcessful.
	Yes No			ОК

図 6-2. Win32DiskImager

____5. PC から microSD カードを安全に取り外します。

## ⚠ 注記:

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:

アルティマ技術サポート「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」



6-2. Linux 起動とログイン

この演習では、以下のインターフェイスを使用します。

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ーフェイス

- ____1. ボードの 5V DC ジャック (J14) に電源アダプター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 コネクター(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ーを PC の USB コネクターへ接続します。
- ____3. ボードの HPS Ethernet コネクター(J10)へ、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 クターを PC のイーサーネット・コネクターへ接続します。
- ____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. 電源アダプター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 COM4)。 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<mark>ターミナルソフト</mark>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 COM4)。

Tera	a Term: シリアルポート 設定	2	×
	ボート(P):	COM4 ~	ОК
	スピード(E):	115200 ~	
	データ(D):	8 bit ~	キャンセル
	バリティ(A):	none ~	
	ストップビット(S):	1 bit ~	ヘルプ(H)
	フロー制御(F):	none ~	
	送信遅延 0 ミリ秒	/字(c) 0 :	ミリ秒/行(L)

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、 root 🚽 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. <mark>ターミナル</mark>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 斗

____4. 再度パスワードを入力します。



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[™] 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「*アダプターの設定の変更*」をクリックします。



### 図 6-9. アダプターの設定の変更

___2. 「イーサネット」をダブルクリックします。



図 6-10. 「イーサネット」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。

イーリネットの状態	X	ਡ イーサネットのプロパティ
般		ネットワーク 認証 共有
接続		接続の方法:
IPv4 接続:	ネットワーク アクセスなし	Intel(R) Ethernet Connection I219-LM
IPv6 接続:	ネットワーク アクセスなし	
メディアの状態:	有効	構成(C)
期間:	00:03:03	この接続は次の項目を使用します(O):
速度: 詳細(E)	1.0 Gbps	<ul> <li>□ ■ Microsoft ネットワーク用クライアント</li> <li>□ ■ Microsoft ネットワーク用ファイルとプリンター共有</li> <li>□ ■ VirtualBox NDIS6 Bridged Networking Driver</li> <li>□ ■ QoS パケット スケジューラ</li> <li>□ ■ Juniper Network Service</li> <li>□ ■ Bailable Multicast Protocol</li> </ul>
加华祆光	送信 — 👽 — 受信	
パケット:	228 0	1ノス(T-)ル(N) 部時(U) ノロバリ(K)
<b>়ি</b> 7০/ংন (P)	●無効にする(D) 診断(G) 閉じる(C)	はでサゴ 伝送制御プロトコル/インターネット プロトコル。相互接続されたさまざま なネットワーク間の通信を提供する、既定のワイド エリア ネットワーク プロトコルです。

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。

般	
ネットワークでこの機能がサポートされてい ができます。サポートされていない場合に 合わせてください。	∧る場合は、IP 設定を自動的に取得するこ す、ネットワーク管理者に適切な IP 設定を問
○ IP アドレスを自動的に取得する(	0)
⑦ たの IP アドレスを使う(S):	
IP アドレス(I):	192 . 168 . 1 . 31
サブネット マスク(U):	255 . 255 . 255 . 0
デフォルト ゲートウェイ(D):	4 × 14
○ DNS サーバーのアドレスを自動的	)に取得する(B)
● 次の DNS サーバーのアドレスを使	ะีวิ(E):
優先 DNS サーバー(P):	· · ·
代替 DNS サーパー(A):	x x 40
□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=0.655	ms
64 bytes from 192.168.1.31: seq=1 ttl=128 time=1.026	ms
64 bytes from 192.168.1.31: seq=2 ttl=128 time=0.953	ms
64 bytes from 192.168.1.31: seq=3 ttl=128 time=0.920	ms
^C	
192.168.1.31 ping statistics	
4 packets transmitted, 4 packets received, 0% packet	loss
round-trip min/avg/max = 0.655/0.888/1.026 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 Defender 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設定*」を確認し、Windows Defender 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。

Windows Detender 55 15 5% 10			- 0	3
→  *  *  *  *  *  *  *  *  *  *  *  *  *	ロール パネル項目 » Windows Defender ファイアウォール	~ U	コントロール パネルの検索	
ユートロール パネル ホーム	Windows Defender ファイアウォールによる P	この保護		
Wind ws Defender ファイアウォー ルを企っ アプリまたは機能を許可	Windows Defender ファイアウォールによって、ハッカーまたは# ークを経由したアクセスを防止できるようになります。	思意のあるソフ	トウェアによるインターネットまたはネ	ットワ
😲 通知設定 🛛 変更	1 セキュリティのため、設定のいくつかはシステム管理者に	よって管理され	ます。	
Windows Defender ファイアウォー ルの有効化または無効化	♥ ドメイン ネットワーク(M)		接続されていません	$\odot$
→ 昨年11 29	♥ プライベート ネットワーク(R)		接続されていません	$\odot$
ネ・ワークのトラブルシューティング			拉结这 2	
○ Windows Defender ファイアウォールを有 □ 許可されたアプリの一覧にあるアプリ+	「効にする 5含め、すべての着信接続をブロックする			
Windows Defender 77/70+-Wi	が新しいマプリをブロックしたときに通知を受け取る			
<ul> <li>○ Windows Defender ファイアウォールを無</li> <li>○ Windows Defender ファイアウォールを無</li> </ul>	が新しいアブリをブロックしたときに通知を受け取る 転効にする (推奨されません)			
<ul> <li>○ Windows Defender ファイアウォールを無</li> <li>○ Windows Defender ファイアウォールを無</li> </ul>	が新しいアプリをブロックしたときに通知を受け取る 転効にする (推奨されません) のK	キャンセル	,	
<ul> <li>□ I State アメン 足にののアプレ ✓ Windows Defender ファイアウォールを無         </li> <li></li></ul>	が新しいアプリをブロックしたときに通知を受け取る 時効にする (推奨されません) のK	キャンセル	,	
<ul> <li>○ Windows Defender ファイアウォールを無</li> <li>○ Windows Defender ファイアウォールを無</li> <li>○ ドメイン、ットワーク(M)</li> </ul>	が新しいアプリをブロックしたときに通知を受け取る 転効にする (推奨されません) のK 接続されていません 〇	キャンセル	<b>,</b>	
<ul> <li>○ Windows Defender ファイアウォールを無</li> <li>○ Windows Defender ファイアウォールを無</li> <li>○ ドメイン、ットワーク(M)</li> <li>○ プライベート、ミットワーク(R)</li> </ul>	が新しいアプリをブロックしたときに通知を受け取る 転効にする (推奨されません) のK 接続されていません ② 接続されていません ③	キャンセル		

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・フォルダー (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。





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・フォルダーを設定します。

この演習では、「<u>3. 演習 1: ハードウェア演習</u>」の作業フォルダーに 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(フォルダー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

🖨 Eclipse Launcher	×
Select a directory as workspace Eclipse プラットフォーム uses the workspace directory to store its preferences and development artifacts.	
ワ−クスペ−ス(W) C:¥lab¥soc_lab¥cv_soc_lab¥workspace	
OK キャンセル	

図 6-17. workspace の作成

____4. DS-5™ 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。

e	worksp	ace - C/	C++ - Ec	:lipse プラットフォー.	4							×
ファ	イル(F) #	編 <mark>集(E</mark> )	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(R)	ウィンドウ(	(W) ヘル	レプ(H)	
8	🗰 DS-5	へようこ									(i) (i)	¢ 🗖 🗗
×			閉じ	3	arm	Deve	eloper		)	Q	-	^
	_		W )ev	elco elop	met mer	to A nt S	Arm Stud	® [ io:	)S- Int	-5 -el	R	Feedback



_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・フォルダーにあります。

(<SoC EDS インストール・ディレクトリー>¥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

● インポート プロジェクトのインポート ボタットは、「アンドクリンドやますスティーク」は、たきおります。	– – ×
ホイチ Eclipse ノロシェクトを使来するブイレクトリーを選択します ○ ルート・ディレクトリーの選択(T): ③ アーカイブ・ファイルの選択(A): ● インポートするプロジェクトを含むアーカイブの選択	。 参照(R) 参照(R)
← → × ↑ 📜 C:¥intelFPGA¥18.1¥embedded¥examples¥softv	vare ~ U softwareの検索 🔎
整理 ▼ 新しいフォルダー	III • 🔟 🕜
intelFPGA 名前 intelFPGA 名前 intelFPGA 名前 intelFPGA 名社era-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Hardware intera-SoCFPGA-Har	更新日時 ・ しは-SPI-CV-GNU.tar.gz 2018/09/14 00:43 しは-Timer-A10-ARMCC.tar.gz 2018/09/14 00:43 しは-Timer-AU-GNU.tar.gz 2018/09/14 00:43 しは-Timer-AV-GNU.tar.gz 2018/09/14 00:43 しは-Timer-CV-ARMCC.tar.gz 2018/09/14 00:43 d-Baremetal-ARMCC.tar.gz 2018/09/14 00:43 d-Baremetal-ARMCC.tar
- 12x-F	×
イ <b>ロシェクトのインホート</b> 既存の Eclipse ブ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の・	★昭/₽)
<ul> <li>アーカイブ・ファイルの選択(A):</li> <li>C:¥intelFPGA¥18.1¥embedded¥examples¥software</li> </ul>	≰Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz ∨ 参照(R)
ロジェクト(P):	
Altera-SoCFPGA-HelloWorld-Linux-GNU(Altera-SoCFPGA-HelloWorld-Linux-GNU)	GNU/) すべて選択(S)
	選択をすべて解除(D)
	更新(E)
オプション ○ ネストしたプロジェクトを検索(H) ○ プロジェクトをワークスペースにコピー(C) □ 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(T)	新規(W)
ワーキング・セット(O):	~ 選択(E)
?	る(B) 次へ(N) > <b>終了(F)</b> キャンセル



_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™ 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。



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。

暄C/C++	(デフォルト)			
GCVS リ	ポジトリー・エク	スプローラー		
🥏 DS-5 🗆	コンフィギュレージ	ション		
🏶 DS-5 🤊	デバッグ			
Git				
🕸 Java				
່ະ ^J Java ທ	型階層			
影 Java 参	照			
LDRAli	te for ARM D	S-5 softw	/are9.5.6	
🗬 PyDev				
<b>≝⁰チ−ム</b> 同	期化			
<b>☆デバック</b>	r			
トリソース				
℡リモート	・システム・エク	スプローラー		

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。

⋹ workspace - リモート・システム・エクスプロ−ラ− - Eclipse プラットフォ-		들 workspace - リモート・システム・エクスプローラー - Eclipse プラットフォ-
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)		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)
• ▼ • * * 8 • • • • • • • • • • • • • • • •	=++++	[☆ - 8] 6] 5 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0
<b>過</b> リモート・システム 🛚 � チーム 🕒 🏷 ↔ 🍭 🖻 🕸 ▽ 🖻 🗖	まには	週リモート・システム ※ 受う 空白部分を右クリック
リモート・システムの接続を定義		▲ 新規接続(A) 接続をインポート(B)

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。

Jモート・ リモート・	<b>システム・タイプの選択</b> システムへ SSH アクセスするための接続
27751.4	マイゴ・
フィルタ入	лу. """
<b>~ ⊵</b> —舟	Ř
₿ F	IP OF
[₽] s F	SH のみ
S E I⊐ S	IP のみ SSH のみ
S F	IP ФА SSH ФЪ

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。

듣 新規接続		-		×
<b>リモート 1 システム</b> 接続情報の定義	ム接続(SSH のみ)			
親プロファイル:	HD11149B			*
ホスト名: 接続名: 記述/説明: マホスト名を検証	192.168.1.30       Atlas SoC       Atlas SoC	ホスト名: 接続名: 記述/説明:	192.168 Atlas SoC Atlas Soc	5.1.30 または DE10 Nano C または DE10 Nano
?  < 戻	る(B) 次へ(N) >	終了(F)	キャンセル	1

図 6-27. 接続設定

- ___6. 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、ユーザー ID とパスワードを入力するウィンドウが表示されます。
- ____7. 「**ユーザー 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。

🖨 workspace - リモート・システム・	듣 パスワードの入力	×
ファイル(F) 編集(E) ナビゲート(N) □ マ 国 国 国 里 □ □ □ ■ № ■ リモート・システム ※ ☞ チーム マ □ Atlas SoC マ □ SSH ターミナル	システム・タイプ: ホスト名: 接続名: ユーザー ID: パスワード(任意)(B):	SSH のみ 192.168.1.30 Atlas SoC root ****** ✓ ユーザー ID の保管 □ パスワードの保管(C)
	ОК	キャンセル(A)

図 6-28. ユーザー ID とパスワードを入力

___8. 下図の警告が出た場合は [*はい*] をクリックします。

듣 警告		$\times$
?	WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! 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 50:cd:28:c5:5b:85:b1:b1:d7:d2:3f:02:b7:78:76:42. 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 delete the old key and insert the new key? (\$U\V\$\vec{L}(Y) U\V\$\vec{L}(N)	

#### 図 6-29. 警告表示



_9. 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

- ____10. 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。
- ____ 11. "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

___12. 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 💀 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・エクスプローラー・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。

● workspace - C/C++ - Eclipse ブラットフォーム ファイル(F) 編集(E) ソース(S) リファクタリング(T)	新規(N) 次ヘジャンプ(I)	>
□ ▼	新規ウィンドウで開く(N) Show In	>
<ul> <li>※ Altera-SoCFPGA-HelloWorld-Linux-GNU</li> <li>※ パイナリー</li> <li>》 Includes</li> <li>&gt; @ hello.c</li> <li>&gt; 参 hello - [arm/le]</li> </ul>	■ Copy ■ たりしック ■ 名前を変更(M)	
>	<ul> <li></li></ul>	
	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	> > >
	リモートシステムビューで表示 実行(R)	
L	デバック(D) プロファイル(P)	> □ 1 □−カル C/C++ アブリケーション > 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・コンフィグレーションを作成します。

構成	成の作成、管F	理、および実行
DS	-5 デバッグセッシ	ションを開始するための
1	₩   ⊟ ≱ ▼	
フィ	レタ入力	
D		ケーション
D	C/C++ アプリ	ケーションへのアタッチ
D		モーティニカレック
Ē	C/C++ IJŦ-	·· 7
ä	▶ DS-5デバッガ	
é	[*] IronPython	] 新規(W)
e	^P IronPython	□ 複製(D)
		₩ ¥117Δ/T



- __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 ⇒ Cyclone V SoC (Dual Core) ⇒ Linux Application Debug ⇒ 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 ⁶ ファイル ⁶ デバッガ ⁶ OS 認識機能 ⁶⁹ 引数 ³ 環境 ⁴ エクスポート	● 開く	-		×
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: ファイルシステム ワークスペース	ファイルを選択します:	sTempFiles	inux-GNU 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 圖 ファイル ◆ デバッガ ◎ 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-SoCFPGA-HelloWorld-Linux-GNU/hello}
ファイルシステム ワークスペース 🗹 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): HelloWorld	
☞ 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	
実行制御 〇 接続のみ 〇 エントリポイントからデバッグします  のシンボルからデバッグします main	

## 図 6-37. デバッグ構成の設定(4)



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。

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。

30	int	<pre>main(int argc, char** argv) -</pre>
[₽] 31		<pre>printf("Hello SoC FPGA!\n");</pre>
32		return 0;
33	}	
34	188	

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。

□         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □         □	R/W R/W R/W
KO     K     KO     KO	R/W R/W R/W
• R0         0x00000010         32           • R1         0x00000000         32           • R2         0x00000001         32           • R3         0x000083CD         32           • R4         0x8EEEEB8         32	R/W R/W R/W
• R1         0x00000000         32           • R2         0x00000001         32           • R3         0x000083CD         32           • R4         0x8EFEEB88         32	R/W R/W
- R2 0x0000001 32 - R3 0x00083CD 32	R/W
- R3 0x000083CD 32	
- PA OVBEEEEBBS 22	R/W
VIA UNDERTIDUO 32	R/W
• R5 0x0000000 32	R/W
• R6 0x0000000 32	R/W
• R7 ØxBEFFFB98 32	R/W
• R8 0x0000000 32	R/W
• R9 0x0000000 32	R/W
• R10 0xB6FFEFA4 32	R/W
• R11 0x0000000 32	R/W
• R12 0x0000004 32	R/W
• SP 0xBEFFFB98 32	R/W
• LR 0xB6F6F363 32	R/W
• PC 0x000083E2 32	R/W
* CPSR 0x600F0030 32	R/W
+ CALL 65/65 L 37 内	

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。

名前	値	型	カウント	サイス	場所	アクセス	
[⊋] 😂 ローカル	変数: 2						
- ┛ argc	1	int		32	0xBEFFFB9C	R/W	
🗄 🗣 argv	ØxBEFFFCF4	char**	1	32	0xBEFFFB98	R/W	
- 🗁 ファイルスタティック変数	変数: 0/0						
🕒 グローバル	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 ライブラリー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 < 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 10 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[®] 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 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 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。

## 📄 参考:

• SoC はじめてガイド

<u>インテル の SoC FPGA</u> <u>SoC はじめてガイド - HPS-FPGA 間のアクセス方法</u> <u>SoC はじめてガイド - Preloader Generator の使用方法</u> <u>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</u>

● SoC 関連情報

マクニカ・ホームページ技術情報: SoC 関連記事/ドキュメント マクニカ・ホームページ技術情報: SoC 関連 FAQ アルティマ技術サポート: SoC 関連記事/ドキュメント アルティマ技術サポート: 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・ホームページ技術情報
   アルティマ技術サポート



# 改版履歴

Revision	年月	概要
1	2020年1月	初版
2	2020 年 6 月	(5 ページ)
		「インテル [®] SoC Embedded Development Suite」を「インテル [®] SoC FPGA エンベデッド開発スイート」に変更
		(6ページ)
		・「Quartus® Prime Standard Edition v18.1」を「インテル® Quartus® Prime Standard Edition v18.1」に変更
		・「SoC Embedded Development Suite v18.1」を「インテル [®] SoC FPGA エ ンベデッド開発スイート Standard Edition v18.1」に変更

#### 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 半導体事業 お問い合わせフォーム
- 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。