

# Nios<sup>®</sup> II 入門編トライアル・コース 【演習マニュアル】

Ver.18.1



# Nios® II 入門編トライアル・コース【演習マニュアル】

<u>目次</u>

1. はじめに	4
2. 演習で使用する評価ボード	6
3. 開発ツールとデザイン開発フロー	7
3-1. 組み込みシステム・ハードウェアおよび FPGA デザインの開発フロー	7
3-1-1. 組み込みシステム・ソフトウェアの開発フロー	
4. ハードウェア・プロジェクトの作成	9
4-1. Quartus® Prime の起動	9
4-2. プロジェクトを作成	9
5. Platform Designer システム・モジュールの作成	12
5-1. Platform Designer の起動	
5-2. 使用するコンポーネント	
5-3. Nios®Ⅱ プロセッサ・コアの追加	
5-4. JTAG UART シリアル・インタフェースの追加	
5-5. オンチップ・メモリ・インタフェースの追加	
5-6. 外部 SRAM インタフェースの追加	
5-7. トライステート・コンポーネント	
5-8. 外部 EPCQ インタフェースの追加	
5-9. リセット・ベクタ と 例外ベクタの設定	20
5-10. PIO ペリフェラルの追加	
5-11. PLL(フェース・ロックド・ループ)の追加	
5-12. クロックとリセットの接続	
5-13. ベース・アドレスの設定	
5-14. Platform Designer システム・モジュールの生成	
6. ハードウェア・デザインの作成	
6-1. Platform Designer システム・モジュールの生成	
6-2. コンフィグレーション・モードの設定	
6-3. 外部端子のピン・アサイン	
6-4. タイミング制約ファイルの設定	
6-5. ハードウェア・デザインのコンパイル	



# Nios® II 入門編トライアル・コース 【演習マニュアル】

6-6. ダウンロード・ケーブルの接続
6-7. プログラミング・ツールの設定
7. ソフトウェアの実行41
7-1. Nios®ⅡSBT の起動
7-2. ソフトウェア・プロジェクトの作成42
7-3. C ソース・コードの作成
7-4. ソフトウェア・プロジェクトのビルドとプログラムの実行46
8. Memory Test の実行(オプション)51
8-1. ソフトウェア・プロジェクトの作成51
8-2. ソフトウェア・プロジェクトのビルドとプログラムの実行52
9. オンチップ・メモリからのブート(オプション)54
9-1. Nios®ⅡSBT での HEX ファイルの生成54
9-2. Quartus <sup>®</sup> Prime での設定とコンパイルから実行まで56
9-3. HEX ファイル更新時の Quartus® Prime プロジェクトへの反映方法
10. EPCQ からのブート (オプション)
10-1. リセット・ベクタと例外ベクタの設定59
10-2. ソフトウェア・プロジェクトのビルド60
10-3. Convert Programming File を使用した書き込みファイルの生成の生成
10-4. Quartus® Prime Programmer での JIC ファイルの書き込み64
改版履歴65

### 1. <u>はじめに</u>

この資料は、Nios<sup>®</sup> II のシンプルなデザインを作成する演習を通じて、開発ボード上で簡単な動作確認が行えるまでの手順を示した資料です。

また、一般的な組み込みシステムと、Nios<sup>®</sup> Ⅱ システムを比較しながら、デザイン開発フローとの関連性を持たせることで、組み込み系エンジニアとロジック設計エンジニアの双方が互いに理解できることを目的としています。

今回の演習では、次のようなシステム構成を題材にしています。



図1 演習題材のシステム構成(一般的な組み込みシステムから見た場合)

一般的な組み込みシステムのソフトウェア開発では、ターゲットの CPU ボードに対して、コンソール用ホスト・マシンの実行ファイルを RAM 上にダウンロードしてソフトウェアを実行します。ダウンロードや簡単なデバッグ・ コマンドは、シリアル・ポートやイーサーネットを経由して行われます。また、デバッグ用ホスト・マシンからデバッ ガを介して、CPU デバッグ・モジュール経由での高度なデバッグを行うこともできます。

図 1 では、逓倍した入力クロックを CPU の動作クロックやシステム・クロックに適用し、 ROM をソフトウェ ア・プログラムの格納場所に使用します。そしてワーク・メモリには RAM を使用します。

今回の演習では、Parallel I/O (PIO) を経由して、外部の LED を点滅させるソフトウェア・プログラムを作成しますが、これらの周辺ペリフェラルは CPU 専用内部バスで接続されます。

ROM や RAM と接続するメモリ・インタフェースは、汎用メモリ I/F ブリッジを経由して、 CPU 内部バスと接続されます。

前ページのシステム構成をインテル® FPGA の開発ツールで実現した場合、図2 のようなブロック構成になります。



図 2 演習題材のシステム構成 (Nios'll システムから見た場合)

この場合、JTAG\_UART というインテル<sup>®</sup> FPGA 特有のシリアル・ポートを利用することで、ホストとのシリアル 通信をインテル<sup>®</sup> FPGA ダウンロード・ケーブル(USB-Blaster<sup>™</sup> 等)を使用して、JTAG ポート経由で行うことが できます。これによりソフトウェア・プログラムは、JTAG 経由で RAM 上にダウンロード後、実行されます。 JTAG デバッグ・モジュール経由でのデバッグについても、ダウンロード・ケーブルを介して行われます。

入力クロックは、図 2 のように、FPGA 内蔵の PLL で逓倍することもできます。 EPCQ デバイス (インテル® FPGA シリアル・コンフィグレーション・デバイス) をプログラムの格納場所に使用して、ワーク・メモリには SRAM を使用します。

今回の演習では PIO を経由して、外部の LED を点滅させるソフトウェア・プログラムを作成しますが、これらの周辺ペリフェラルは Nios® II プロセッサ用内部バスである Platform Designer インタコネクトで接続されます。

EPCQ デバイスや SRAM と接続するメモリ・インタフェースは、それぞれコントローラを経由して、Platform Designer インタコネクトと接続されます。

図 2 の組み込みシステムは、システム名 nios2\_system というブロックとして FPGA 内に実装されます。

### 2. 演習で使用する評価ボード

今回の演習では、Terasic 社の Cyclone® VGX スタータ開発キットを使用します。

このボードの構成は、4 個のプッシュ・ボタンと 18 個の汎用 LED、10 個のスライド・スイッチ、4 個の 7 セ グ LED を実装しており、シンプルかつ高機能な動作が確認できます。 USB-Blaster™ の本体部分が基板に実装 (Embedded USB-Blaster) されておりますので、付属の USB ケーブルのみでホスト PC と、基板上 FPGA や Nios® II プロセッサ間の通信が可能です。また、EPCQ デバイスと揮発性 RAM として LPDDR2-SDRAM, SRAM を実装しております。クロック・システムは本演習では、 50MHz の水晶発振器を利用し、 SRAM や EPCQ デ バイスに対しては、 FPGA 内蔵の PLL で逓倍したクロックを利用しています。その他、詳細については、以下 の説明ページを参照ください。

Terasic 社の Cyclone V GX スタータ開発キット

https://www.intel.co.jp/content/www/jp/ja/programmable/products/boards\_and\_kits/dev-kits/altera/kit-terasic-cyclone-v-gx-starter.html



#### 3. 開発ツールとデザイン開発フロー

開発フローと開発ツールとの関係を示します。

3-1. 組み込みシステム・ハードウェアおよび FPGA デザインの開発フロー

① デザイン・エントリ

🛆 ALTIMA

- ハードウェア記述言語(Verilog HDL / VHDL)プロジェクトに登録します。回路図や EDIF ネットリストでもプロジェクトに登録できます
- ・ 組み込みシステム部分のハードウェア記述言語は、Platform Designer が生成します
- ・ タイミング制約は、SDC ファイルとして生成しプロジェクトに登録します
- ② 論理合成
  - ハードウェア記述言語から論理回路を合成します
  - ・ 3<sup>rd</sup> パーティーの論理合成ツールや Quartus<sup>®</sup> Prime (Synthesis) を実行します
  - ・ 結果は、専用のネットリストに反映されます
- ③ 配置配線
  - ・ 合成した論理回路に対して、対象の FPGA の配線経路の情報を加味します
  - Quartus<sup>®</sup> Prime(Fitter)を実行します
  - ・ 結果は、専用のネットリストに反映されます
- ④ タイミング解析
  - TimeQuest Timing Analyzer を使用して、登録した SDC ファイルの内容に基づきタイミング解析を行います
- ⑤ アセンブル
  - ・ Quartus<sup>®</sup> Prime(Assembler)を実行します
  - ・ 配置配線結果をビットストリーム(プログラミング・データ)に展開します
  - このビットストリームが、ターゲットのデバイスに実装されます
  - ・ インテル® FPGA のビットストリームには、sof ファイルと、pof ファイルとが存在します
  - ・ ターゲットのデバイスが揮発性素子 (FPGA) の場合、 sof が適用されます
- ⑥ プログラミング
  - ターゲットのデバイスに対して、ビットストリームを書き込みます
  - Quartus<sup>®</sup> Prime Programmer を使用します
- ⑦ 実機動作

3-1-1. 組み込みシステム・ソフトウェアの開発フロー

- ① ソフトウェア・プログラムの作成
  - ・ ソフトウェア開発環境やテキスト・エディタを使用します
  - ・ 今回の演習では、Nios® II SBT のエディタを使用して、プログラミング言語を記述します
- ② C/C++ コンパイル

- ・ C/C++ コ ンパイラが、プログラミング言語をアセンブリ言語に変換します
- ・ 今回の演習は、Nios<sup>®</sup> Ⅱ SBT に組み込まれている GNU ベースの C/C++ クロス・コンパイラを使用します
- ③ アセンブル
- ④ リンク
  - リンカが、ライブラリ情報をオブジェクト・ファイルにリンクします
  - ・ 様々な中間ファイルを経て、最終的に実行ファイル(ELF ファイル)を生成します
  - Nios® II SBT では、ELF ファイルは、拡張子 .elf で表記されます
- ⑤ ダウンロード・ファイル (ROM 化ファイル)の生成
  - ・ バイナリ・ユーティリティが、ELF ファイルをモトローラ S レコード・フォーマットに変換します
  - ・ Nios® II SBT では、S レコード・ファイルを、拡張子 .flash で表記します
    - 🛕 上記のフローの ① ~ ④ は、makefile の記述内容に従い、一括処理されます
    - ▲ この一括処理は、Nios® II SBT で「ビルド」を行うことにより実現されます
- ⑥ ダウンロード、および実機動作
  - Nios<sup>®</sup> II SBT が、JTAG ポートや JTAG\_UART ペリフェラルを経由して ELF ファイルを RAM 上に格 納した後、プログラムを実行します

### 4. ハードウェア・プロジェクトの作成

Quartus<sup>®</sup> Prime を用いてハードウェア・プロジェクトを作成します。Nios<sup>®</sup> II の開発を行う際は、必ず、ハードウェア・プロジェクトを作成する必要があります。

今回の演習では、nios2\_basic\_lab という名称のハードウェア・プロジェクトを作成します。また、開発環境は、 Quartus® Prime v18.1 + Nios® II SBT v18.1 を使用します。

#### ▲ 演習に使用するファイルはすでに解凍されておりすぐに演習が開始できる環境になっています。講師の指示に 従い演習を進めてください。もしファイルが未解凍である場合は、講師の指示に従いファイルの解凍を行ってく ださい。

4-1. Quartus<sup>®</sup> Prime の起動

🛆 ALTIMA

デスクトップのショートカット、もしくはスタートメニューより Quartus® Prime を起動します。

#### 4-2. プロジェクトを作成

1. Quartus<sup>®</sup> Prime のメニュー・バーから File ⇒ New Project Wizard を選択します。 New Project Wizard が開きます。

Introduction ページが表示された場合は [Next] ボタンをクリックします。

2. プロジェクト・フォルダを指定します。最上段のボックスに以下のパスを入力します。

C:¥Lab¥nios2\_lab¥nios2\_basic\_prj

- ▲ Quartus<sup>®</sup> Prime プロジェクトのパス名には、スペースや日本語文字は入れないでください。Nios<sup>®</sup> Ⅱ SBT をインストールする際のインストール・パスについても同様です。
- 3. プロジェクト名を指定します。二段目のボックスには以下のプロジェクト名を入力します。

nios2\_basic\_lab

- トップ階層を指定します。三段目のボックスに以下のプロジェクト名が反映されていることを確認します。
   nios2\_basic\_lab
- 5. 右下の [Next] ボタンをクリックします。

Directory frame, rop Level Linky	
What is the <u>w</u> orking directory for this project?	
C:\Lab\nios2_lab\nios2_basic_prj	
What is the name of this project?	
nios2_basic_lab	
What is the name of the top-level design entity for this project? This name is ca design file.	se sensitive and must exactly match the entity name in the
nios2 basic lab	

- 6. Project Type が開くので、Empty Project が選択されていることを確認して [Next] ボタンをクリックしま す。
- 7. Add Files が開くので、何もせず [Next] ボタンをクリックします。
- 8. Family & Device Setting が開くので、デバイス・ファミリを選択します。

Cyclone V (E/GX/GT/SX/SE/ST)

9. 下記の使用するデバイスの型番を Name Filter に入力します。

#### 5CGXFC5C6F27C7

10. Available devices から該当の型番を選択して、 [Next] ボタンをクリックします。

elect the family and c	device you want to ta	rget for comp	pilation.			
'ou can install additio 'o determine the versi	nal device support w ion of the Quartus Pri	ith the Install ime software	Devices comma in which your ta	and on the Tools m arget device is supp	enu. ported, refer to the <u>Device</u>	Support List webpage
Device family				Show in 'Available	devices' list	
Eamily: Cyclone V (	(E/GX/GT/SX/SE/ST)		-	Pac <u>k</u> age:	Any	•
Dev <u>i</u> ce: All			•	Pin <u>c</u> ount:	Any	Ŧ
Target device				Core sp <u>e</u> ed grade:	Any	+
Auto device selection	cted by the Fitter			Name filter:	5CGXFC5C6F27C7	
Specific device so           O         Other:         n/a	elected in 'Available c	levices' list		Show advance	d devices	
vailable devices:						
Name	Core Voltage	ALMs	Total I/Os	GPIOs	GXB Channel PMA	GXB Channel PCS
5CGXFC5C6F27C7	1.1V	29080	364	336	5	6
	m					

11. EDA Tools Settings が開くので、何もせず [Next] ボタンをクリックします。

12. Summary が開くので、以下のようになっていることを確認して [Finish] ボタンをクリックします。

new Project Wizard		
Summary		
When you click Finish, the project will be created with	the following settings:	
Project directory:	C:\Lab\nios2_lab\nios2_basic_prj	
Project name:	nios2_basic_lab	
Top-level design entity:	nios2_basic_lab	
Number of files added:	0	
Number of user libraries added:	0	
Device assignments:		
Design template:	n/a	
Family name:	Cyclone V (E/GX/GT/SX/SE/ST)	
Device:	5CGXFC5C6F27C7	
Board:	n/a	
EDA tools:		
Design entry/synthesis:	<none> (<none>)</none></none>	
Simulation:	<none> (<none>)</none></none>	
Timing analysis:	0	
Operating conditions:		
Core voltage:	1.1V	
Junction temperature range:	0-85 °C	

13. Project Navigator ウインドウの Hierarchy タブに nios2\_basic\_lab が設定されていることを確認してください。

Quartus Prime Standard Edition -	C:/Lab/nios2_lab/r	nios2_basic_prj/nio	s2_basic_lat	o - nios2	_basic_la	b
<u>Eile Edit View Project Assignn</u>	nents Processing	<u>T</u> ools <u>W</u> indow <u>H</u>	<u>i</u> elp			
	nios2_basic_lab	• 2	<b>6</b> 6 4		× ₹ ·	0
Project Navigator	Hierarchy	▼Q₽₽×				
Enti	ty:Instance					
À Cyclone V: 5CGXFC5C6F27C7						
nios2_basic_lab <sup>1</sup> / <sub>1</sub>						

### 5. Platform Designer システム・モジュールの作成

Nios® II プロセッサ・コアを含む Platform Designer システム・モジュールを作成します。

#### 5-1. Platform Designer の起動

- 1. Quartus<sup>®</sup> Prime の Tools メニュー ⇒ Platform Designer を選択します。
- Platform Designer が起動したら、File メニュー ⇒ Save as... を選択して先にファイルにセーブします。
   今回は nios2\_system.qsys というファイル名で保存します。



5-2. 使用するコンポーネント

Platform Designer システム・モジュールに、Nios® II プロセッサ・コア、周辺 I/O、ペリフェラルを追加します。

追加した Nios<sup>®</sup> II プロセッサ・コアやペリフェラル等を接続するシステムバス (Avalon バス) は、 Platform Designer により自動的に最適化され生成されます。今回の演習で追加するコンポーネントは以下になります。

- Nios<sup>®</sup> II プロセッサ・コア
- ・ JTAG UART シリアル・インタフェース
- オンチップ・メモリ・インタフェース
- ・ 外部 SRAM インタフェース
- トライステート・コンポーネント
- ・ 外部 EPCQ インタフェース
- ・ PIO ペリフェラル
- PLL

#### 5-3. Nios® II プロセッサ・コアの追加

- IP Catalog ウインドウの Library の項目から、Processors and Peripherals ⇒ Embedded Processors 下の Nios II Processor を選択し、[Add] ボタンをクリックします。
  - ▲ IP Catalog ウインドウの検索入力(虫メガネのボックス部分)に該当するワード(ここでは、Nios II)を入 力すると簡単に該当のコンポーネントが見つかります。

🗂 IP Catalog 🛛	
🔍 Nios II	× 🔯
Project	
New Compo	ment
Library	
🗎 Basic Function:	3
😑 Simulation;	Debug and Verification
🖻 Simulati	on
	Nios II Custom Instruction Master BFM Intel FPGA IP
	Nios II Custom Instruction Slave BFM Intel FPGA IP
-Processors and	Peripherals
Co-Process	ors
🖻 Nios II (	Justom Instructions
	Bitswap
	Custom Instruction Interconnect
	Custom Instruction Master Translator
	Custom Instruction Slave Translator
	Floating Point Hardware
L 🙍	Floating Point Hardware 2
Embedded F	rocessors
- Nice	II (Classic) Processor
<ul> <li>Nios</li> </ul>	I Processor
2 2	
New Edit	🚺 💠 Add



2. Nios<sup>®</sup> Ⅱ 設定ウインドウが表示されますので、Nios Ⅱ/f が選択されていることを確認して [Finish] ボタ ンをクリックします。

▲ この時点で、Message ウインドウにエラーが出ますが気にせず先に進んでください。

Show signals nics2_gen2_0 k data_master;	Main Vectors	Caches and Memory Interfaces   Arithmetic mplementation Nios II/e Nios II/f	Instructions   MMU and MPU Settings   JTAG Debu	Advanced Features
set reset avaion instruction_master		Nios II/e	Nios II/f	
nterrupt reset debug_reset_request	Summary	Resource-optimized 32-bit RISC	Performance-optimized 32-bit RISC	
bug_mem_slave avalon nies_oustom_instruction_master_ alters_nios2_pen2	Features	JTAG Debug ECC RAM Protection	JTAG Debug Hardware Multiply/Divide Instruction/Data Gaches Tightly-Coupled Masters ECC RAM Protection External Interrupt Controller Shadow Register Sets MPU MMU	
	RAM Usage	2 + Options	2 + Options	
m ),	   • [		m	j
rror: nios2_gen2_0. Instruction Cache is larger than the Instruction Address. Please rror: nios2_gen2_0. Reset slave is not specified. Please select the reset slave nios2_gen2_0. Flease slave is not specified. Please select the reset slave	reduce the Instruction	on Cache Size. Current Tag Size is 0		

- 3. System Contents タブの Name 項目の nios2\_gen2\_0 をハイライト後、マウスで右クリックして Rename を選択します。
- 4. nios2\_gen2\_0 から nios2\_cpu に名前を変更します。



5-4. JTAG UART シリアル・インタフェースの追加

\Lambda ALTIMA

Nios® II とシリアル接続を行うためのインタフェースとして、JTAG UART を追加します。

- 1. IP Catalog ウインドウの Library の項目から、Interface Protocols ⇒ Serial より JTAG UART Intel FPGA IP を選択し、[Add] ボタンをクリックします。
- 2. JTAG UART ウインドウが表示されるので、デフォルトのまま [Finish] ボタンをクリックし、 Platform Designer のメイン・ウインドウに戻ります。



- 3. Name 項目の jtag\_uart\_0 をハイライト後、マウスで右クリックして Rename を選択します。
- 4. jtag\_uart\_0 から jtag\_uart に名前を変更します。

	🗉 jtag_uart	ITAG UART Intel FPGA IP
$  \diamond   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow $	clk	Clock Input
$  \rangle \rangle \langle + + \rangle \rangle$	reset	Reset Input
$  \qquad \diamond \rightarrow \rightarrow$	avalon_jtag_slave	Avalon Memory Mapped Slave
│	irq	Interrupt Sender

5-5. オンチップ・メモリ・インタフェースの追加

1. IP Catalog ウインドウの Library の項目から、Basic Functions の On Chip Memory より On-Chip Memory (RAM or ROM) を選択し、[Add] ボタンをクリックします。

营 IP Catalog 🛛 🗌	
🔍 onchip	× 🕸
Project	
Library	
Basic Functions	
🖮 On Chip Memory	
On-Chip Elash Intel EF	for ROM) Intel FPGA IP

- 2. On -Chip Memory (RAM or ROM) ウインドウが表示されるので、 Total Memory Size に 131072 と入力し、 [Finish] ボタンをクリックします。
  - ▲ 128K と入力し、Tab キーを押すと自動的に 131072 に変換されます。

. Diagram			
v signals	Type:	Pom (Wettable)	
onchip_memory2_0			
	Single clock operation		
clock	Block type:	DONI_CAR +	
avalon	Block ope.	HUI	
reset1 reset			
altera_avalon_onchip_memory2	T Size		
	Enable different width for Dual-po	rt access	
	Slave S1 Data width	22	
	Total memory size:	181072 bytes	
	minimize memory block usage (ma	y mpact maxy	
	Read latency		
	Slave s1 Latency		
	Slave s2 Latency		
	Reset Request:	Evolution	
	reset request	Enabled 👻	
	* ECC Parameter		
	Extend the data width to support EUC	bits: Disabl 👻	
	Memory initialization		
	Initialize memory content		
	Enable non-default initialization fi	e	
	Type the filename (e.g. my ram	hex) or select the hex file using the file browser button.	
	User created initialization file	onchin member	
	Eachie Partial Passation to a	princip_mentation	
	E Enable Fartial Reconfiguration init	anzation mode	
	Lable In-System Memory Conten	Editor feature	
	Instance ID.	NONE	
	M		
	wemory will be initialized from	niosz_system_onchip_memory2_U hex	

- 3. Name 項目の onchip\_memory2\_0 をハイライト後、マウスで右クリックして Rename を選択します。
- 4. onchip\_memory2\_0 から onchip\_memory に名前を変更します。

	onchip_memory	)n-Chip Memory (RAM or ROM) Inte
$\diamond \rightarrow \rightarrow$	clk1	Clock Input
	s1	Avalon Memory Mapped Slave
$ \qquad \qquad$	reset1	Reset Input

5-6. 外部 SRAM インタフェースの追加

1. IP Catalog ウインドウの Library の項目から、Qsys Interconnect の Tri-State Components の Generic Tri-State Controller を選択し、 [Add] ボタンをクリックします。

📩 IP Catalog 🛛	- d' 🗆
🔍 generic	×
Basic Functions	
Generic Serial Flash Interface	Intel FPGA IP
<ul> <li>Flash</li> <li>Generic QUAD SPI controller I</li> <li>Generic QUAD SPI Controller I</li> </ul>	I Intel FPGA IP Intel FPGA IP
Qsys Interconnect     Orri-State Components     Generic Tri-State Controller	-

- 2. Presets 一覧より ISSI\_IS61WV25616EDBLL を選択し、 [Apply] ボタンをクリックします。次に [Finish] ボタ ンをクリックし画面を閉じます。
  - ▲ 下図では、ISSI\_IS61WV25616EDBLL は、Library フォルダ内に一覧されていますが、Project フォル ダ内に表示されている場合もあります。

Block Diagram     Show signals     generic_tristate_controller_0     clk clock tristate_conduit tcm     reset reset     wavaon     atera_generic_tristate_controller	Sirral Selection       Sirral Relation         Address width:       16         Data width:       16         Bytes per word:       2         Enable the following signals:         Refer to the Avalon Interface Specifications for definitions of these signals: http://www         V readdata         V writedata         Polycenable         V write         begintransfer         V byteenable         V owite         writebyteenable         V outputenable         V outputenable	Presets     Project     Offick. Newto create a preset.     Ubrary     Offick. Newto create a preset.     Offickto create a preset.
Warning: generic_tristate_controller_0 Pro	perties (isMemoryDevice) have been set on interface <b>uas</b> – in composed mode these are i <b>n</b>	Cancel

- 3. Name 項目の generic\_tristate\_controller\_0 をハイライト後、マウスで右クリックして Rename を選択し ます。
- 4. generic\_tristate\_controller\_0 から sram に名前を変更します。

V		🖻 🛄 sram	ieneric Tri-State Controller
	$\diamond$ $\rightarrow$ $\rightarrow$	clk	Clock Input
	$\diamond \rightarrow \rightarrow$	reset	Reset Input
	$\rightarrow \rightarrow $	uas	Avalon Memory Mapped Slave
	×<	tom	Tristate Conduit Master

5-7. トライステート・コンポーネント

1. IP Catalog ウインドウの Library の項目から、Qsys Interconnect の Tri-State Components の Tri-State Conduit Bridge を選択し、[Add] ボタンをクリックします。



2. 設定はデフォルトのまま、[Finish] ボタンをクリックして画面を閉じます。

Tri-State Conduit Bridge - tristate_conduit_bridge_0	×
Tri-State Conduit Bridge altera_tristate_conduit_bridge	Documentation
Block Diagram	
tristate_conduit_bridge_0	
altera_tristate_conduit_bridge	
	Cancel

- 3. Name 項目の tristate\_conduit\_bridge\_0 をハイライト後、マウスで右クリックして Rename を選択しま す。
- 4. tristate\_conduit\_bridge\_0 から ext\_sram\_bus に名前を変更します。

	ext_sram_bus	Fri-State Conduit Bridge
$  \diamond   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow $	clk	Clock Input
$  \qquad \diamond \longrightarrow $	reset	Reset Input
$\diamond \longrightarrow$	tcs	Tristate Conduit Slave
φ	out	Conduit

- 5. Connections 項目の sram の tcm ポートと、ext\_sram\_bus の tcs ポートの交点の白丸をクリックして 黒丸に変更します。この操作により、sram と ext\_sram\_bus の 2 つのペリフェラルが接続されます。
- 6. ext\_sram\_bus の out の Export 項目をダブル・クリックすると自動的に ext\_sram\_bus\_out という文字 列が挿入されます。この操作で、sram のバス出力が Platform Designer システムの外部へエクスポートされるようになります。

<b>V</b>		日 喧 sram	Generic Tri-State Controller		
	$\diamond$ $\rightarrow$ $\rightarrow$	clk	Clock Input	Double-click to export	unconnected
	$   \diamond + + \diamond \longrightarrow$	reset	Reset Input	Double-click to export	[clk]
	$     \diamond \diamond \longrightarrow$	uas	Avalon Memory Mapped Slave	Double-click to export	[clk]
		tom	Tristate Conduit Master	Double-click to export	[clk]
<b>V</b>		🖯 ext_sram_bus	Tri-State Conduit Bridge		
	$\diamond$ $\rightarrow$	clk	Clock Input	Double-click to export	unconnected
	$  \rightarrow \rightarrow$	reset	Reset Input	Double-click to export	[clk]
	│         ( +)→	tes	Tristate Conduit Slave	Duality circle in export	[clk]
		out	Conduit	ext_sram_bus_out	

5-8. 外部 EPCQ インタフェースの追加

1. IP Catalog ウインドウの Library の項目から、Basic Function の Configuration and Programming の Serial Flash Controller II Intel FPGA IP を選択し、[Add] ボタンをクリックします。

📩 IP Catalog 🛛 🔤	đ 🗆
serial 🔰	۵
Project	^
Library	
Basic Functions	
Configuration and Programming	
Generic Serial Elash Interface Intel EPGA 1	P
Serial Flash Controller II Intel FPGA IP	Ħ

2. Serial Flash Controller II Intel FPGA IP ウインドウが起動するので、Configuration device type に EPCQ256 を選択し、[Finish] ボタンをクリックします。

Shock Diagram Show signals epcq_controller2_0 avl_csr avalon interrupt interrupt_sender avl_mem avalon	Configuration device type: Choose I/O mode: Number of Chip Selects used:
clock_sink_clock reset	



- 3. Name 項目 epcq\_controller\_2\_0 をハイライト後、マウスで右クリックして Rename を選択します。
- 4. epcq\_controller\_2\_0 から epcq に名前を変更します。

	🗆 🛄 epcq	Serial Flash Controller II Intel FPGA
	avl_csr	Avalon Memory Mapped Slave
	avl_mem	Avalon Memory Mapped Slave
	interrupt_sender	Interrupt Sender
$  \diamond   \longrightarrow$	clock_sink	Clock Input
$\diamond \longrightarrow \diamond$	reset	Reset Input

- 5-9. リセット・ベクタ と 例外ベクタの設定
  - 1. Nios<sup>®</sup> II と外部メモリを接続します。

Connection 欄で nios2\_cpu の data\_master と instruction\_master を onchip\_memory の s1 ポート と sram の uas ポートと、epcq の avl\_csr ポートと avl\_mem ポートに図のように接続します。 さらに jtag\_uart の avalon\_jtag\_slave も nios2\_cpu の data\_master と図のように接続します。

Use	Connections	Name	Description	Export	Clock
<b>V</b>		🗆 clk_0	Clock Source		
	D	clk_in	Clock Input	clk	exported
	→ → →	clk_in_reset	Reset Input	reset	
		clk	Clock Output	Double-click to export	clk_0
		clk_reset	Reset Output	Double-click to export	
<b>V</b>		🗏 🛄 nios2_cpu	Nios II Processor		
	$  \diamond   \rightarrow$	clk	Clock Input	Double-click to export	unconnected
	$  \qquad \diamond \qquad \rightarrow$	reset	Reset Input	Double-click to export	[clk]
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]
		irq	Interrupt Receiver	Double-click to export	[clk]
		debug_reset_request	Reset Output	Double-click to export	[clk]
	$     \phi \phi   \rightarrow$	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]
		custom_instruction_m	Custom Instruction Master	Double-click to export	
$\checkmark$		🗆 jtag_uart	JTAG UART Intel FPGA IP		
	$ \diamond$	clk	Clock Input	Double-click to export	unconnected
		reset	Reset Input	Double-click to export	[clk]
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]
		irq	Interrupt Sender	Double-click to export	[clk]
V		onchip_memory	On-Chip Memory (RAM or ROM) Inte		
		clk1	Clock Input	Double-click to export	unconnected
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]
		reset1	Reset Input	Double-click to export	[clk1]
<b>V</b>		日 喧 sram	Generic Tri-State Controller		
	$ \diamond$ $	clk	Clock Input	Double-click to export	unconnected
		reset	Reset Input	Double-click to export	[clk]
		uas	Avalon Memory Mapped Slave	Double-click to export	[clk]
		tom	Tristate Conduit Master	Double-click to export	[clk]
1		🗉 ext_sram_bus	Tri-State Conduit Bridge		
	$ \diamond $	clk	Clock Input	Double-click to export	unconnected
	$   \diamond   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow   \rightarrow$	reset	Reset Input	Double-click to export	[clk]
	$ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $	tcs	Tristate Conduit Slave	Double-click to export	[clk]
		out	Conduit	ext_sram_bus_out	
<b>V</b>		日 喧 epcq	Serial Flash Controller II Intel FPGA		
	$   \uparrow \downarrow \downarrow \rightarrow$	avl_csr	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
	$   \downarrow \downarrow \downarrow \downarrow \rightarrow \downarrow$	avl_mem	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
		interrupt_sender	Interrupt Sender	Double-click to export	[clock_sink]
	$\diamond$ $\rightarrow$	clock_sink	Clock Input	Double-click to export	unconnected
	$\diamond \longrightarrow \diamond$	reset	Reset Input	Double-click to export	[clock_sink]



2. nios2\_cpu をハイライト後、マウスで右クリックし Edit を選択します。再び、Nios® II プロセッサ・コアの GUI 画面が起動します。

Vectors タブの Reset Vector の項目のプルダウン・メニューより、onchip\_memory.s1 を選択します。

また、同様にプルダウン・メニューより、Exception Vector の項目も onchip\_memory.s1 を選択します。 設定が終わったら、[Finish] ボタンをクリックします。

▲ 本演習では、まず Nios® II をオンチップ・メモリからブートする手法を説明します。後の章(オプション)では、Reset Vector の設定を EPCQ に指定して、EPCQ からのブートの方法について説明します。

Nios II Processor - Nios 2_cp Nios II Processor altera_nios2_gen2	Docu	mentation
clk cłock	Reset Vector         onchip_memorys1           Reset vector offset         0x0000000           Reset vector:         0x0000000	
irg interrupt debug_mem_slave avalon	Exception Vector Exception vector memory: onchip memorys1 Exception vector offset: 0x0000020 Exception vector: 0x0000020	E
	Fast TLB Miss Exception Vector         Fast TLB Miss Exception vector offset         0x00000000         Fast TLB Miss Exception vector:         0x00000000         0x00000000	
m	>	-
	Cancel	Finish

#### 5-10. PIO ペリフェラルの追加

1. IP Catalog ウインドウの Library の項目から、Processors and Peripherals ⇒ Peripherals の項目より PIO (Parallel I/O) Intel FPGA IP を選択し、 [Add] ボタンをクリックします。

TP Catalog 🕺	- 6 0
🔍 pio	× 💱
Project	
New Component	
Library	
🛱 Interface Protocols	
B-PCI Express	
🖮 QSYS Example Designs	
645 or 1285 PIO AVST	
Processors and Peripherals	
Peripherals	
PIO (Parallel I/O) Intel FPGA IP	

- 2. Basic Settings タブ内の設定内容が以下になっていることを確認し、[Finish] ボタンをクリックします。
  - Width : 8 bits
  - Direction : Output

PIO (Parallel I/O) Intel FPGA IP - pio_0 PIO (Parallel I/O) Intel FPGA IF	
Block Diagram     Show signals     pio_0     ck     reset     reset     s1     avaion	Documentation
external_connection	
4 [] Þ	Test bench wiring           Hardwire PD inputs in test bench           Drive inputs to field;         0x00000000000000000000000000000000000
	Cancel Finish

- 3. Name 項目の pio\_0 をハイライト後、マウスで右クリックして Rename を選択します。
- 4. pio\_0 から led\_pio に名前を変更します。

	🗉 led_pio	10 (Parallel I/O) Intel FPGA IP
	clk	Clock Input
$  \rangle \langle + \rangle \langle - \rangle \rangle$	reset	Reset Input
$  \qquad \diamond \rightarrow \rightarrow$	s1	Avalon Memory Mapped Slave
	external_connection	Conduit





- 5. Connections 項目の led\_pio の s1 を nios2\_cpu の data\_master に接続します。
- external\_connection の Export 項目をダブル・クリックすることで、LED 出力を Export します。信号名 led\_pio\_external\_connection が自動的に入力されます。この設定を行うと、Connections 項目にこの信号 が Export であることが表示されます。

🗆 led_pio	PIO (Parallel I/O) Intel FPGA IP	
→ clk	Clock Input	Double-click to export
→ reset	Reset Input	Double-click to export
→ s1	Avalon Memory Mapped Slave	Duble click to copart
<ul> <li>external_connection</li> </ul>	Conduit	led_pio_external_connection

5-11. PLL (フェース・ロックド・ループ) の追加

50MHz のクロックを逓倍して 100MHz のクロック 1 系統と 25MHz のクロック 1 系統を生成します。生成した クロックは、 Platform Designer のシステム・クロック(プロセッサの動作クロックも含む)として使用します。

1. IP Catalog ウインドウの Library の項目から、Basic Function ⇒ Clocks; PLLs and Resets ⇒ PLL の項目 より PLL Intel FPGA IP を選択し、 [Add] ボタンをクリックします。

📩 IP Catalog 🛛 🕅		
🔍 pll		× 🐼
Project		-
Library	onent	
Basic Function	IS	
Glocks; PL	Ls and Resets	
E-PLL .	ALTON LIVER FOR A TO	<u>111</u>
	ALTELL FILE FEGALE ALTELL RECONERS INTELEPOA IP	
	IOPLL Intel FPGA IP	
	IOPLL Reconfig Intel FPGA IP	
•	PLL Intel FPGA IP	
	PLL Reconfig Intel FPGA IP	

- 2. PLL の設定を行うための GUI 画面が起動しますので、下記のように設定し、 [Finish] ボタンをクリックしま す。
  - Reference Clock Frequency : 50MHz
  - Number Of Clocks : 2
  - outclk\_0.Desired Frequency : 100MHz
  - outclk\_1.Desired Frequency : 25MHz



nam aignala	GIOCK SWITCHOW	er   Cascading   MIF Streaming   Settings   Advanced Parameters	
IOW SIGNOIS	Device Speed Grade:		
pll 0	PLL Mode:	Integer-N PLL	
	Reference Clock Frequence	100% 50.0 MHz	
efclk clock clock outclk0	Operation Mode.	direct 👻	
eset reset clock outclk1	Enable locked output	port	
conduit locked	Enable physical outpu	ut clock parameters	
atera el	Clocks		
artera_for	Number Of Clocks:		
	outciku		
	Desired Frequency.	100.0 MHz	
	Actual Frequency	100.000000 M 👻	
	Phase Shift units:	ps 💌	
	Phase Shift:	0 ps	
	Actual Phase Shift:	0 ps 💌	
	Duty Cycle:	50 %	
	T outclk1		
	Desired Frequency	25.0 MHz	
	Actual Frequency:	25.000000 M 👻	
	Phase Shift units:	ps 👻	
	Phase Shift:	0 ps	
	Actual Phase Shift:	0 os 👻	
	Duty Cycle:	50 %	
	Conv		

- 3. Name 項目の pll\_0 をハイライト後、マウスで右クリックして Rename を選択します。
- 4. pll\_0 から pll に名前を変更します。



#### 5-12. クロックとリセットの接続

1. Platform Designer の System Contents タブに戻り、クロックとリセットの接続を行います。画面を見やすくするために pll を選択し、 [Move Up] ボタン ( ^ )を 7 回クリックし、 clk\_0 の下に pll を移動します。

öystem Contents 🐰 Address M	ap &   Interconnect Requirement n <b>Path</b> :nll	s 23		
Use Connections	Name	Description	Export	Clock Base
	□ clk_0 □ clk_in □ clk_in_reset clk □ clk_reset	Clock Source Clock Input Reset Input Clock Output Reset Output	clk reset Double-click to export	exported clk_0
	🗆 pli	PLL Intel FPGA IP		
	refclk reset outolk0 outolk1 ccked	Clock Input Reset Input Clock Output Clock Output Conduit	Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export	unconnected pll_outclk0 pll_outclk1
	E I I nios2_cpu clk reset data_master instruction_master irq debug_reset_request debug_mem_slave	Nios Il Processor Clock Input Reset Input Avalon Memory Mapped Master Avalon Memory Mapped Master Interrupt Receiver Reset Output Avalon Memory Mapped Slave	Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export	unconnected [cik] [cik] [cik] [cik] [cik] [cik] <b>0</b>

 次に pll の入力として、外部 50MHz のクロック (clk\_0.clk) を接続します。併せて、リセット (clk\_0.clk\_reset) も接続します。図のように接続してください。

Use Connections	Name	Description	Export	Clock Ba
	⊟ cik_0	Clock Source		
· · · · · · · · · · · · · · · · · · ·	D clk_in	Clock Input	clk	exported
	o cikjinjin	eset Reset Input	reset	
	clk	Clock Output	Double-click to export	clk_0
	clk_rese	et Reset Output	Double-click to export	
	🖯 pli	PLL Intel FPGA IP		
	→ refclk	Clock Input	Double-click to export	clk_0
	reset	Reset Input	Double-click to export	
	outclk0	Clock Output	Double-click to export	pll_outclk0
	outclk 1	Clock Output	Double-click to export	pll_outclk1
	ocked	Conduit	Double-click to export	
	日 喧 nios2	Cpu Nios II Processor		
	clk	Clock Input	Double-click to export	unconnected
¢	reset	Reset Input	Double-click to export	[clk]
	data_m	aster Avalon Memory Mapped Ma	ster Double-click to export	[clk]
	instruct	tion_master Avalon Memory Mapped Ma	ster Double-click to export	[clk]
	irq	Interrupt Receiver	Double-click to export	[clk]
	debug_r	eset_request Reset Output	Double-click to export	[clk]
	debug_r	nem_slave Avalon Memory Mapped Sla	ve Double-click to export	[clk] 🛋
	custom	instruction m., Custom Instruction Master	Double-click to export	

 Clock の項目(点線部分)で、clk\_0 と pll を除いた全てのペリフェラルに対して、プルダウン・メニューより Unconnected から pll\_outclk0/1 に変更します。 Epcq のみ pll\_outclk1 を接続します。その他は、 pll\_outclk0 を接続します。これは、Connections の項目から接続で行うことも可能です。



- 4. 次に Connections 項目にてリセット信号を接続します。 clk\_0 の clk\_reset を各ペリフェラルのリセット信号に接続します。
- 5. 同様に nios2\_cpu の debug\_reset\_request も clk\_0、 pll 以外の各ペリフェラルのリセット信号に接続します。これは、デバッグ用途で使用されるリセット信号となります。

Use	Connections	Name	Description	Export	Clock	Base
V	I	🗆 clk_0	Clock Source		[	Ī
	0-0-D	⊢ clkin	Clock Input	clk	exported	
	D	clk in reset	Reset Input	reset	-	
		< clk	Clock Output	Double click to export	clk 0	
		< elk reset	Reset Output	Double-click to export	-	
		🗆 oli	Altera PLL	Construction Sector Sector Sector Sector		1
		> refolk	Clock Input	Double-click to eroort	clk 0	
		+ recet	Beset Input	Double-click to expert		
	$\mathbf{V}_{\mathbf{v}}$	< outelk0	Glock Output	Double-click to export	oll outelkil	
		< outcilk1	Clock Output	Double-click to export	pll_outclk1	
	0-	locked	Conduit	Double-click in export	phootener	
			Nice II Processor	Double citex to export		
141			Clock Input	Dentstanne fick der erstende	nll outolk 0	
			Poost Input	Double circle to expert	Interior Contraction	
		/ reset	Auster Menery Manad Master	Pouble click to export	[CIK]	
		<ul> <li>uata_master</li> <li>isst_status</li> </ul>	A day Manager Manager	Double click to export	E-IL-1	
		instruction_master	Avaion Memory Mapped Master	Double Chick to export	LCIK]	100.0
6/		Irq	Interrupt Receiver	Double-click to export	[CIK]	IRU U
1 i		CepreTexerTextest	Reserroutput	Double-click to export	LOIKI	
		debug_mem_slave	Avaion Memory Mapped Slave	Double-click to export	[CIK]	= 0×0000_0800 0
and a		custom_instruction_master	Custom Instruction Master	Double-click to export		
4	12 12 12 12 12 12 12 12 12 12 12 12 12 1	🖂 jtag_uart	JTAG UART	10. 200 200	5 10 E025	
		→ clk	Clock Input	Double-click to export	pll_outclk0	
		→ reset	Reset Input	Double-click to export	[clk]	
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	■ 0×0000_0000 0
	¢	irq	Interrupt Sender	Double-click to export	[clk]	
V		onchip_memory	On-Chip Memory (RAM or ROM)			
		→ clk1	Clock Input	Double-click to export	pll_outclk0	
		⇒ s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	⇒ 0×0000_0000 0
		→ reset1	Reset Input	Double-click to expert	[clk1]	
V	$ \Psi      \Psi $	🗆 🖳 sram	Generic Tri-State Controller			
1		→ clk	Clock Input	Double-click to export	pll_outclk0	
		→ reset	Reset Input	Double-click to export	[clk]	
	$ \Psi  \leftrightarrow \Psi$	+ uas	Avalon Memory Mapped Slave	Double-click to export	[clk]	■ 0×0000_0000 0
		< tcm	Tristate Conduit Master	Double-click to export	[clk]	
1		🖯 ext_sram_bus	Tri-State Conduit Bridge		-	
		→ clk	Clock Input	Double click to export	pll_outclk0	
		→ reset	Reset Input	Double-click to export	[clk]	
	$ \Psi      \Psi +$	→ tcs	Tristate Conduit Slave	Double-click to export	[clk]	
	•0	> out	Conduit	ext sram bus out		
1		日 啦 epcq	Altera Serial Flash Controller			
575-25s		→ clock sink	Clock Input	Double-click to export	pll outclk1	
		→ reset	Reset Input	Double-click to export	[clock sink]	
	$\Psi \rightarrow \Psi$	→ avicsr	Avalon Memory Mapped Slave	Double-click to export	[clock sink]	₩ 0×0000 0000 00
		→ avImem	Avaion Memory Mapped Slave	Double-click to export	[clock sink]	# 0x0000 0000
		interrupt sender	Interrupt Sender	Double-click to export	[clock sink]	i i contracti di c
			PIO (Parallel I/O)	- Annual Annual and an angrout	Conservation and	
			Clock Input	Double-stick to second	oll outel+0	
		+ recet	Reset Input	Duchla click to accord	[clk]	
		el	Avalon Memory Manned Slave	Doublesslick to export	folk1	
		external connection	Conduit	led nin external connection	LO IN	- 0.0000_0000
		external_connection	oondan	leg his external connection		

6. 次に jtag\_uart、epcq から Nios<sup>®</sup> II コアへの割り込み接続を設定します。 IRQ 列にある jtag\_uart の白丸部分をクリックします。次に epcq の IRQ 列にある白丸をクリックします。 jtag\_uart の割り込み番号が
 0 に、epcq の割り込み番号が 1 に設定されました。0 が優先順位の高い設定となります。

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
<b>V</b>		⊟ clk_0	Clock Source					1
		clkin	Clock Input	clk	exported			
	· · · · · · · · · · · · · · · · · · ·	clk_in_reset	Reset Input	reset				
		clk	Clock Output	Double-click to export	clk_0			
		clk_reset	Reset Output	Double-click to export				
<b>V</b>		🖯 pll	PLL Intel FPGA IP					
_	$  \bullet   \bullet \bullet \longrightarrow$	refclk	Clock Input	Double-click to export	clk_0			
	$   \bullet \longrightarrow$	reset	Reset Input	Double-click to export				
		outclk0	Clock Output	Double-click to export	pll_outclk0			
		outclk1	Clock Output	Double-click to export	pll_outclk1			
		locked	Conduit	Double-click to export				
<b>V</b>		🗆 🖳 nios2_cpu	Nios II Processor					
	$  \diamond   \bullet \bullet \rightarrow$	clk	Clock Input	Double-click to export	pll_outclk0			
	$   \bullet   \to \to$	reset	Reset Input	Double-click to export	[clk]			
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		irq	Interrupt Receiver	Double-click to export	[clk]	IRQ	0 IRQ 31	5
		debug_reset_request	Reset Output	Double-click to export	[clk]			
	│	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	© 0x0000_0800	0×0000_0fff	
	×	custom_instruction_m	Custom Instruction Master	Double-click to export				
1		🖯 jtag_uart	JTAG UART Intel FPGA IP					
		clk	Clock Input	Double-click to export	pll_outclk0			
	$   \bullet   + + + + \bullet \longrightarrow$	reset	Reset Input	Double-click to export	[clk]			
	│	avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	₽° 0x0000_0000	0x0000_0007	
		irq	Interrupt Sender	Double-click to export	[clk]			) 🏳
V		onchip_memory	On-Chip Memory (RAM or ROM) Inte.					
	Ŷ │ <b>♦ १ │ │ ┨ │ ── →</b>	clk1	Clock Input	Double-click to export	pll_outclk0			
	│	s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	■ 0×0000_0000	0×0001_ffff	
	$   \bullet   +   +   \bullet \rightarrow$	reset1	Reset Input	Double-click to export	[clk1]			
<b>V</b>		曰 🖳 sram	Generic Tri-State Controller					
		clk	Clock Input	Double-click to export	pll_outclk0			
	$   \bullet   +   +   +   \bullet \longrightarrow$	reset	Reset Input	Double-click to export	[clk]			
	│	uas	Avalon Memory Mapped Slave	Double-click to export	[clk]	■ 0×0000_0000	0×0007_ffff	
		tom	Tristate Conduit Master	Double-click to export	[clk]			
V		🗆 ext_sram_bus	Tri-State Conduit Bridge					
	$\bigcirc   \bullet \bullet   + \bullet   + \bullet   + \bullet \rightarrow$	clk	Clock Input	Double-click to export	pll_outclk0			
	$\bullet + + + + \bullet \bullet \bullet \to \bullet$	reset	Reset Input	Double-click to export	[clk]			
		tcs	Tristate Conduit Slave	Double-click to export	[clk]			
	$\sim$	out	Conduit	ext_sram_bus_out				
		년 년 epcq	Serial Flash Controller II Intel FPGA _					
		avl_csr	Avaion Memory Mapped Slave	Double-click to export	[clock_sink]	" Ux0000_0000	0×0000_003f	
		avl_mem	Avaion Memory Mapped Slave	Double-click to export	[clock_sink]	= 0x0000_0000	0x01ff_fff	
		interrupt_sender	Interrupt Sender	Double-click to export	[clock_sink]			1
		clock_sink	Clock Input	Double-click to export	pil_outcik1			$\sim$
	$  $ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$	reset	Reset Input	Double-click to export	[clock_sink]			
V		led_pio	PIO (Parallel I/O) Intel FPGA IP					
		Clk .	Clock Input	Double-click to export	pll_outclk0			
	• • • • • • • • • • • • • • • • • • • •	reset	Reset Input	Double-click to export	[clk]			
	••• •• •• ••	sl	Avalon Memory Mapped Slave	Double-click to export	[clk]	0×0000_0000	0×0000_000t	
		external_connection	Conduit	led_pio_external_connection				

5-13. ベース・アドレスの設定

- ペリフェラルの追加が行われると、Platform Designer は自動的に各ペリフェラルに対して、ベース・アドレ スを設定します。 System メニュー ⇒ Assign Base Address を選択すると、ベース・アドレスが自動的に変 更されます。
  - ▲ Base 項目のベース・アドレスをダブル・クリックすることで、直接ベース・アドレスを入力することもでき ます。
  - ▲ Address Map タブをクリックすると、設定されたベース・アドレスの一覧を確認できます。



System Contents 🛛 🚥 Addres:	s Map 🛛 Interconnect Requirements 🕮					
System: nios2_system Path: led_pio						
	nios2_cpu.data_master	nios2_cpu.instruction_master				
epcq.avl_csr	0x0414_1000 - 0x0414_103f					
epcq.avl_mem	0x0200_0000 - 0x03ff_ffff 0x0200_0000 - 0x03ff_fff					
jtag_uart.avalon_jtag_slave 0x0414_1050 - 0x0414_1057						
led_pio.s1 0x0414_1040 - 0x0414_104f						
nios2_cpu.debug_mem_slave	0x0414_0800 - 0x0414_0fff	0x0414_0800 - 0x0414_0fff				
onchip_memorys1	0x0412_0000 - 0x0413_ffff	0x0412_0000 - 0x0413_ffff				
sramuas	0x0408_0000 - 0x040f_fff	0x0408_0000 - 0x040f_ffff				

- ▲ 表示されるアドレス値は、上図と一致するとは限りません。異なっていても問題はないので、そのまま 進めてください。
- 2. 最後に Messages タブの内容を確認します。下図のようにいくつかのワーニングやメッセージが表示されていますが、問題ないのでそのまま進めます。

o≞ Messag	Xo≣ Messages ∞					
Туре	Path	Message				
	2 Warnings					
	nios2_system.sram	Properties (isMemoryDevice) have been set on interface uas - in composed mode these are ignored				
	nios2_system.pll	pll.locked must be exported, or connected to a matching conduit.				
	3 Info Messages					
	nios2_system.jtag_uart	JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board				
	nios2_system.pll	The legal reference clock frequency is 5.0 MHz700.0 MHz				
	nios2_system.pll	Able to implement PLL with user settings				

3. Platform Designer 設定ファイルの保存をするため、File メニュー ⇒ Save を選択し、nios2\_system.qsys ファイルを保存します。



5-14. Platform Designer システム・モジュールの生成

- 1. Platform Designer のメイン・ウインドウから Generate メニュー ⇒ Generate HDL を選択します。
- 2. Generate ウインドウが表示されるので、[Generate] ボタンをクリックしてシステムを生成します。

Synthesis	
Synthesis files are used to com	pile the system in a Quartus project.
Create HDL design files for syn	nthesis: Veril
Create timing and resource	estimates for third-party EDA synthesis tools.
🔽 Create block symbol file (b	st)
* Simulation	
The simulation model contains	generated HDL files for the simulator, and may include simulation-only features.
Simulation scripts for this com	ponent will be generated in a vendor-specific sub-directory in the specified output directory.
Follow the guidance in the gene	arated cimulation covints about how to structure your design's simulation covints and how to use the in-return-rimulation
and ip-make-simscript commar	ind-line utilities to compile all of the files needed for simulating all of the IP in your design.
Create simulation model:	None 👻
• Output Directory	
Path:	C/Lab/nios2 lab/nios2 basic pri/nios2 system

3. Platform Designer システム・モジュールの生成が正常に終了した場合、メッセージ・ボックス内に図のよう なメッセージが表示されます。 [Close] ボタンをクリックして、メッセージ・ボックス を閉じてください。

Senerate Completed	X
AII 🛛 🛆 🕕	
<ul> <li>Info: multiplexer: "asmi2_inst_epcq_ctrl" instantiated altera</li> <li>Info: Reusing file C:/Lab/nios2_lab/nios2_basic_prj/nios2_</li> <li>Info: asmi2_cmd_generator_0: "asmi2_inst_epcq_ctrl" instantiated altera</li> <li>Info: asmi2_qspi_interface_0: "asmi2_inst_epcq_ctrl" instantiated erro</li> <li>Info: error_adapter_0: "avalon_st_adapter" instantiated erro</li> <li>Info: error_adapter_0: "avalon_st_adapter_006" instantiated</li> <li>Info: avst_fifo: "xip_controller" instantiated altera_asmi2_x</li> <li>Info: nios2_system: Done "nios2_system" with 53 modules, 8</li> <li>Info: Finished: Create HDL design files for synthesis</li> </ul>	n_merlin_multiplexer " system/synthesis/su ntiated altera_asmi2_cr ntiated altera_asmi2_qs r_adapter "error_adap error_adapter "error_a ip_controller "avst_fif 7 files
🛆 Generate: completed with warnings.	
	Stop Close



- 4. Platform Designer のメイン・ウインドウからから File メニュー ⇒ Exit で Platform Designer を閉じてください。
- 5. 以下のようなウインドウが表示されたら [OK] ボタンで閉じてください。



### 6. <u>ハードウェア・デザインの作成</u>

Platform Designer システム・モジュールをトップ階層に配置して、ハードウェア・デザインを作成します。

- 6-1. Platform Designer システム・モジュールの生成
  - 1. Quartus<sup>®</sup> Prime で File メニュー ⇒ Open を選択します。
  - トップ階層のファイルは、nios2\_basic\_lab.bdf というファイル名で予め用意されています。
     Project Navigator ウインドウの nios2\_basic\_lab をダブル・クリックするとファイルが開きます。

Quartus Prime Standard Edition - C:/La ile Edit View Project Assignments	b/nios2_lab/nios Processing To	s2_basic_prj/nic	os2_basic_lab - nios2_basic_lab -
<u>] た 日 イ D 的 つ の [nios2</u>		• 2	( \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$
oject Navigator	A Hierarchy	- <	nios2_basic_lab.bdf
Entity:Inst	ance		▋ ╋ + D D つぐ 署 <b>N 4 ∜ A D ╚ - D つ つ つ へ \ \ D O \ \ \ B</b> A 4 A
C Lone V: 5CGXPCEC6E27C7			
nios2 basic lab			1
		1 .	1
			1
1	1.1	- <b>m</b> - x	
SKS CC	ompliation	·	
Task		Time	
			aram byte addressfig. 01
Compile Design			<b>*</b>
Applyric & Synthesis			SRAM_BE_n[1.0]
Anatysis & Synthesis			SRAM_OE_NX
Fitter (Place & Route)			SRAW WE n CUTPUT
			SRAW DI15.01 BDR
Assembler (Generate progra	amming files)		
No. 1 Statement of the state			
🖻 🖻 Timing Analysis			1
EDA Netlist Writer			
Calls Callings			2
Edit Settings			pt_otex_n
Program Device (Open Program	nmer)		
· ····································			1
			SRAM A/17 / 01OUTPUTSram_byte_address[181]

- 3. 回路図の空白部分にマウス・ポインタを置いてダブル・クリックすると、Symbol ウインドウが起動します。
- 4. ブラウズ・ボタンをクリックし、C:¥Lab¥nios2\_lab¥nios2\_basic\_prj¥nios2\_system フォルダ内の nios2\_system.bsf ファイルを選択し [Open] をクリックします。

	Cook in:	e Domputer 73	C:\Lab\nios2_lab\ni ar inios2_syste	os2_basic_prj	i\nios2_sys	stem	×	G	0	0	2	] [
Insert symbol as block	My C	Computer 73	er 🔛 synthesis	em.bsf								
	File name:	nios2	2 system bsf							(	•	pen



5. 次のように、Library ウインドウに nios2\_system のブロック図が表示されるので、[OK] ボタンをクリックします。

Libraries:	and the second se	
C:/intelfpga/18.1/quartus/libraries/	cik_cik_cik_cik_cik_cik_cik_cik_cik_cik_	
Name:	Lied_pio_external_connection_export[7_0] export reset	
<u>Repeat-insert mode</u> Insert symbol as block         Insert symbol         In	reset_n reset_n	

- 6. マウス・ポインタが Platform Designer システム・モジュールの枠型に変わります。
- nios2\_basic\_lab.bdf ファイルの回路図上の空白に、シンボルを配置します。問題が無ければ、下図のように、ぴったりと配置できるはずです。できない場合には、何かが間違っている可能性があるので、 Platform Designer システムを見直す必要があるかもしれません。



- 8. File メニュー  $\Rightarrow$  Save を選択して、回路図を保存します。
- Assignments メニュー ⇒ Settings ⇒ Files にて nios2\_system.qip、nios2\_basic\_lab.bdf を設定します。 ブラウズ・ボタンで nios2\_system/synthesis フォルダから nios2\_system.qip を選択して、 [Add] ボタン で追加します。次に、同様の操作で nios2\_basic\_lab.bdf を設定します。設定できたら [OK] ボタンをクリ ックして画面を閉じます。

Select the design files you want to include in the pr	oject. Click Add All to add all design	files in the	project directory to the	e project.
File name:				Add
٩			×	Add All
File Name	Туре	Library	Design Entry/Syntl	Remove
nios2_system/synthesis/nios2_system.qip nios2 basic lab.bdf	IP Variation File (.qip) Block Diagram/Schematic File		<none> <none></none></none>	Up
			10000000000000000000000000000000000000	Down

6-2. コンフィグレーション・モードの設定

CPLD とは異なり、FPGA は SRAM で構成されている揮発性素子であり、ボードの電源を切るとユーザが作成した回路情報は全て消えてしまいますので、専用の ROM に回路情報を格納する必要があります。ボードの 電源を入れると、ROM から FPGA に回路データを転送することで FPGA 内にユーザ回路が形成されます。

この動作は、コンフィグレーションと呼ばれています。ソフトウェア・エンジニアがボードに実装されている IC に対して初期設定やモード設定を行うときのコンフィグレーションとは意味合いが異なりますので注意してください。

インテル FPGA では様々なコンフィグレーション・モードが存在しますが、ハードウェア・デザインを作成する 場合、どのコンフィグレーション・モードを使用するかをユーザ側で設定する必要があります。

- 1. Quartus<sup>®</sup> Prime の Assignments メニュー  $\Rightarrow$  Device を選択します。
- 2. 起動した Device ウインドウ中央の Device and Pin Options ボタンをクリックします。

Select the family and	device you want to ta	rget for comp	Devices common	d on the Tools n	manu .		
iou can instatt additit	mat device support w	itin the mistali	Devices comman	d on the roots i	nenu.		
Fo determine the vers	ion of the Quartus Pr	ime software	in which your targ	get device is sup	ported, refer to the <u>Devic</u>	<u>e Support List</u> webpage.	
Device family				Show in 'Avail	lable devices' list		
Family: Cyclone V	(E/GX/GT/SX/SE/ST	10	•	Package:	Anv		•
Pannty. Cyclone v	(2/07/01/37/32/31)			rac <u>n</u> age.	Ally		
Dev <u>i</u> ce: All			•	Pin <u>c</u> ount:	Any		STA
Target device				Core speed g	rade: Any		
				Name filter:			_
Auto device sele	cted by the Fitter				li.		
Specific device s	elected in 'Available o	devices' list		IMI S <u>n</u> ow adva	anced devices		
🔘 Other: n/a			(	Device and Pin	Ontions		
				Device and 1 m	opuons		
Available devices:							
Name	Core Voltage	ALMs	Total I/Os	GPIOs	GXB Channel PMA	GXB Channel PCS	
5CGXFC5C6F23A7	1.1V	29080	268	240	6	6	2
5CGXFC5C6F23C6	1.1V	29080	268	240	6	6	2
5CGXFC5C6F23C7	1.1V	29080	268	240	6	6	2
5CGXFC5C6F23I7	1.1V	29080	268	240	6	6	2
5CGXFC5C6F27C6	1.1V	29080	364	336	6	6	2
FCCVFCFCCCF37C7	1.1V	29080	364	336	6	6	2
DUGXFUDU0F2/U/	1.1V	29080	364	336	6	6	2
5CGXFC5C6F27C7		29080	203	175	6	6	2
5CGXFC5C6F27C7 5CGXFC5C6F27I7 5CGXFC5C6M13C6	1.1V			4.77.0	-	e	2
5CGXFC5C6F27C7 5CGXFC5C6F27I7 5CGXFC5C6M13C6 5CGXFC5C6M13C7	1.1V 1.1V	29080	203	1/5	6	0	-
5CGXFC5C6F27C7 5CGXFC5C6F27I7 5CGXFC5C6M13C6 5CGXFC5C6M13C7 4	1.1V 1.1V III	29080	203	175	6	0	Þ

- 3. Device and Pin Options ウインドウが起動します。
- 4. Category 欄より Configuration を選択します。
- 5. プルダウン・メニューより、Configuration Scheme の項目に対して、Active Serial x1 を選択します。 [OK] ボタンをクリックして画面を閉じます。

General	Configuration						
Configuration	Specify the device configuration scheme and the configuration device.						
Programming Files Unused Pins	Configuration <u>s</u> cheme: Active Serial x	(can use Configuration Device)					
Dual-Purpose Pins Capacitive Loading Board Trace Model I/O Timing Voltage Pin Placement Error Detection CRC CvP Settings Partial Reconfiguration	Configuration <u>m</u> ode: Standard						
I/O Timing Voltage	Use configuration device:	to					
Pin Placement Error Detection CRC CvP Settings Partial Reconfiguration	Configuration device I/O voltage: Au	Configuration Device Options to					
	VID Operation mode						
	Configuration pin:	Configuration Pin Options					
	Active serial clock source: 100 MHz In	iernal Oscillator					
	Enable input tri-state on active confi Description:	guration pins in user mode					
	The method used to configure a device with a design. Available configuration schemes depend on selected device family: Passive Serial (PS), Passive Parallel x8 (PPx8), Passive Parallel x16 (PPx16), Passive Parallel x32 (PPx32), Active Serial x1 (ASx1), Active Serial x4 (ASx4) and AVST x8, x16 and x32.						

6. Device ウインドウに戻るので、この画面も [OK] ボタンをクリックして閉じます。

6-3. 外部端子のピン・アサイン

FPGA 内のユーザ回路の信号に、デバイス毎で規定されている外部端子のピン番号をアサインします。

- ▲ 本演習では時間節約のため、予め用意されたピン設定用のファイルを読み込むことで、ピン配置を行い ます。
- 1. Processing メニュー ⇒ Start ⇒ Start Analysis & Elaboration を選択して、デザインの文法チェックを行 います。Task 画面の Analysis & Elaboration に緑のチェックが付いたら正常終了です。



- 2. Quartus<sup>®</sup> Prime の Assignments メニュー ⇒ Import Assignments を選択します。
- 3. Import Assignments ウインドウが起動します。
- 4. **Copy existing assignments into nios2\_basic\_lab.qsf.bak before importing** のチェックを**外し**ます(QSF ファ イルのバックアップを残す場合はそのままで OK)。File name 横のブラウザ・ボタンをクリックします。

specify the source and ca	tegories of assignments to import	ES 🦲	
ile name:	1999 (1997) (1997) (1997)		Categories
Dopy existing assignm	nents into nios2_basic_lab.qsf.bak	before im	Advanced
		Cancol	Hele

5. プロジェクト・フォルダから Pin\_assign\_for\_lab.csv を選択し [開く] ボタンをクリックし、 File name 欄にファ イル名が表示されたら [OK] ボタンをクリックします。





6. Quartus<sup>®</sup> Prime の Assignments メニュー ⇒ Pin Planner を選択します。次の図のようにピン情報が表示されれば、ピン配置が正常に反映されたことになります。



#### 6-4. タイミング制約ファイルの設定

\Lambda ALTIMA

FPGA のハードウェア・デザインのタイミング制約を追加します。タイミング制約は、Fmax やデバイス内部の レジスタのセットアップ/ホールド時間、デバイスピンからの入出力のタイミングなどの制約を指します。設計した デザインが、指定したタイミング制約を正しく満たすことで、ハードウェア・デザインが電気的に正しく動作するこ とが保証されます。

ここでは、予め本演習用に作成した制約ファイルを読み込むことでタイミング制約を行います。制約の記述方 法などについては、詳細になり過ぎるので、ここでは、割愛させて頂きます。

- 1. Quartus<sup>®</sup> Prime の Assignments メニュー  $\Rightarrow$  Settings を選択します。
- 2. 左欄の Category 枠より Timing Analyzer をハイライトします。
- ブラウズ・ボタンをクリックして、プロジェクト・フォルダ内の my\_original\_constrain.sdc を選択後、 [Add] ボタンをクリックして、制約ファイルを登録します。

▲ my\_original\_constrain.sdc ファイルは、トップ回路図へのタイミング制約ファイルとなります。

4. ファイルの登録が完了したら、[OK] ボタンをクリックして画面を閉じます。

General	Timing Analyzer						
Files	Energify Timing Analyzer entions						
Libraries	Specify Liming Analyzer options.						
IP Settings IP Catalog Search Locations	SDC files to include in the project						
Design Templates	<u>Eile name:</u>		Add				
Operating Settings and Conditions Voltage	•	×	Remove				
Temperature	File Name	Туре	<u>U</u> р				
Compilation Process Settings Incremental Compilation	my_original_constrain.sdc nios2_system/synthesis/nios2_system.qip	Synopsys Design Constraints File IP Variation File (.qip)	Down				
EDA tool Settings Design Entry/Synthesis Simulation Board-Level Compiler Settings							
Verilog HDL Input	☑ <u>E</u> nable Advanced I/O Timing	Report worst-case paths during compilation					
Default Parameters	Tcl Script File for customizing reports during compilatio	n					
Timing Analyzer	Tcl Scrint Eile name:						
Assembler Design Assistant Signal Tao Logic Analyzer	Run default timing analysis before running custom s	cript	()				
Logic Analyzer Interface	Metastability analysis						
Power Analyzer Settings SSN Analyzer	Synchronizer identification: Auto						
	Description:						
	Associates a Synopsys Design Constraint File (.sdc) with	this project.					

6-5. ハードウェア・デザインのコンパイル

- 1. Quartus<sup>®</sup> Prime の Processing メニュー  $\Rightarrow$  Start Compilation を選択します。
- 2. コンパイルには多少の時間を要します。終了し Tasks ウインドウが図のように Compile Design に緑の ☑ マークがついていれば正常終了です。



3. Timing Analyzer フォルダ内の各種レポートに、赤字になっているフォルダが無いことを確認してください。

12	nios2_basic_lab.bdf			Compilatio	on Report - nios2_basic_lab
Fable c	of Contents (	P 8	Flow Sum	mary	
	🖥 Flow Elapsed Time	*	< <filte< td=""><td>st&gt;&gt;</td><td></td></filte<>	st>>	
=	Flow OS Summary		Flow State	us	Successful - Thu Jan 24 14:49:03 2019
E	Flow Log		Quartus P	rime Version	18.1.0 Build 625 09/12/2018 SJ Standard Edition
Þ. 💼	Analysis & Synthesis	1	Revision N	Name	nios2_basic_lab
Þ 💼	Fitter		Top-level	Entity Name	nios2_basic_lab
0	Flow Messages		Family		Cyclone V
0	Flow Suppressed Messages		Device		5CGXFC5C6F27C7
Þ 📕	Assembler		Timing M	odels	Final
-	Timing Analyzer		Logic utili	zation (in ALMs)	2,287 / 29,080 ( 8 % )
	🖽 Summary		Total regi	sters	3934
	Parallel Compilation		Total pins		49 / 364 ( 13 % )
	📰 SDC File List		Total virtu	ial pins	0
	I Clocks		Total bloc	k memory bits	1,113,920 / 4,567,040 ( 24 % )
$\triangleright$	🖻 📒 Slow 1100mV 85C Model	Ξ	Total DSP	Blocks	3/150(2%)
$\triangleright$	🟱 📒 Slow 1100mV 0C Model		Total HSS	I RX PCSs	0/6(0%)
Þ	Fast 1100mV 85C Model		Total HSS	I PMA RX Deserializers	0/6(0%)
Þ	Fast 1100mV 0C Model		Total HSS	I TX PCSs	0/6(0%)
	📅 Multicorner Timing Analysis Summary		Total HSS	I PMA TX Serializers	0/6(0%)
Þ	Advanced I/O Timing		Total PLL	5	1/12(8%)
₽	Clock Transfers		Total DLL	s	0/4(0%)
	Report TCCS				
	Report RSKM				
⊳	🖓 📒 Unconstrained Paths				
	Messages				

6-6. ダウンロード・ケーブルの接続

ALTIMA

- 1. 今回演習で使用する評価ボード Cyclone<sup>®</sup> V GX Starter Kit の USB BLASTER ポート (J10) とホスト PC とを 付属の USB ケーブルで接続します。
- 2. 評価用ボードに電源ケーブルを接続し、赤いボタンを押し電源を入れます。
- 6-7. プログラミング・ツールの設定
  - 1. Quartus<sup>®</sup> Prime の Tools メニュー ⇒ Programmer 選択します。 Programmer ウインドウが表示されま す。
  - 2. Hardware Setup の項目に USB-Blaster が設定されていることを確認します。

▲ 設定されていない場合には、講師に確認してください。



- 3. Auto Detect ボタンを押して、ボード上のデバイスをサーチします。
- 4. Select Device が表示されたら、5CGXFC5C6 をチェックして [OK] ボタンをクリックします。
- 5. File 項目の <none> となっている箇所をダブル・クリックして、SOF ファイルを選択します。 output\_files/nios2\_basic\_lab.sof
- 6. Program / Configure のチェック・ボックスにチェックを入れます。
- 7. Start ボタンをクリックすると、FPGA にハードウェア・デザインの書き込みが開始されます。
- 8. Programmer ウインドウの Progress の項目が 100% になれば書き込み完了です。

Edit View	Processing Tools Window Help						Search alter	a.com
Hardware Setup. Enable real-time I	USB-Blaster [USB-0]	nen available		Mode: JTAG	•	Progress:		
Ma Start	File	Device	Checksum	Usercode	Program/ Verify Configure	Blank- Examin Check	ne Security Bit	Erase
W <sup>th</sup> Stop	output_files/nios2_basic_lab.sof	5CGXFC5C6F27	03ADE38C	03ADE38C				
X Delete								
Add File				m				
Save File								
Add Device 1 <sup>1</sup> / <sub>b</sub> Up 1 <sup>1</sup> / <sub>b</sub> Down	TDI 5CGXFC5C6F27							

### 7. ソフトウェアの実行

Nios<sup>®</sup> II Software Build Tool (Nios<sup>®</sup> II SBT) は、Nios<sup>®</sup> II 向けの統合開発環境になります。今回の演習では、 Nios<sup>®</sup> II SBT を使用して、C ソース・コードの作成やビルド、およびターゲットの評価ボードへのプログラム・ダウ ンロードまでを実施します。

作成する C ソース・コードは Nios<sup>®</sup> II の PIO を経由して、評価ボードの LED を点滅させる簡単なプログラ ムになります。

- 7-1. Nios® II SBT の起動
  - Quartus<sup>®</sup> Prime の現在のハードウェア・プロジェクト・フォルダ(推奨)に、Nios<sup>®</sup> II SBT 用のワークスペース を作成します。例えば、今回の演習で使用している Quartus<sup>®</sup> Prime の現在のハードウェア・プロジェクトは、 C:¥Lab¥nios2\_lab¥nios2\_basic\_prj フォルダになりますが、このフォルダの下にワークスペース名 software というフォルダを手動で作成してください。

C:¥Lab¥ nios2\_lab¥nios2\_basic\_prj¥software

整理 🔹 ライブラリに油	追加 ▼ 共有 ▼ 新しいフォルダー			
🚖 お気に入り	名前	更新日時	種類	サイズ
	🔒 .qsys_edit	2018/11/15 14:22	ファイル フォル	
🖥 ライブラリ	🍶 db	2018/12/03 10:34	ファイル フォル…	
	nios2_system	2018/12/03 10:17	ファイル フォル	
	🌗 software	2018/12/03 14:03	ファイル フォル	
-9-19E1-9-	li solution	2018/11/15 13:57	ファイル フォル	
	🌽 unsaved	2018/11/15 14:21	ファイル フォル	
🏟 ネットワーク	ISSI_IS61WV25616EDBLL.qprs	2014/07/29 16:40	QPRS ファイル	3
	led_output.c	2011/05/20 16:14	Cファイル	1
	my_original_constrain.sdc	2014/07/01 07:58	SDC ファイル	2
	🔁 nios2_basic_lab.bdf	2016/02/24 09:59	Quartus II Bloc	7
	🛐 nios2_basic_lab.qpf	2018/11/15 14:15	Quartus II Proje	2
	nios2_basic_lab.qsf	2018/11/15 14:15	QSF ファイル	3
	nios2_basic_lab.qws	2018/12/03 10:34	QWS ファイル	1
	nios2_system.qsys	2018/12/03 10:15	QSYS ファイル	65
	nios2_system.sopcinfo	2018/12/03 10:15	SOPCINFO ファ	412
	Pin_assign_for_lab.csv	2014/07/01 13:27	Microsoft Excel	5

- 2. ショートカットや Windows® のスタートメニューから、Nios II SBT を起動します。
- Workspace Launcher が起動した場合、前述のワークスペース (C:¥Lab¥ nios2\_lab¥nios2\_basic\_prj¥software)を指定して、[OK] ボタンをクリックします。

Workspace Launcher	
Select a workspace Eclipse stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: C:VLabYnics2_labVnics2_basic_prjVsoftware	Browse
Use this as the default and do not ask again	
	OK Cancel



4. Nios<sup>®</sup> II SBT が起動します。



7-2. ソフトウェア・プロジェクトの作成

Nios<sup>®</sup> II SBT の New Project ウィザードを用いて、ソフトウェア・プロジェクトを作成します。

以下の手順に従ってください。

1. Nios<sup>®</sup> || SBT のメイン画面の File メニュー ⇒ New ⇒ Nios || Application and BSP from Template を選 択します。

File Edit Naviga	e Search Project Run Nios	II V	Vindow Help
New	Alt+Shift+N ▸	C++	Nios II Application and BSP from Template
Open File		C++	Nios II Application
Close Close All	Ctrl+W Ctrl+Shift+W	10 10 10 10 10 10 10 10 10 10 10 10 10 1	Nios II Board Support Package Nios II Library Project
Save	Ctrl+S		Other Ctrl+

2. Nios<sup>®</sup> II Applocation and BSP from Template ウインドウが起動します。

- ソフトウェア・プロジェクトを作成する際に、組み込みシステムのハードウェア情報が記述されたシステム 定義ファイルを指定します。Target hardware information 欄の SOPC Information File name の右側に あるボタンをクリックして、ハードウェア・デザインを作成したフォルダにある Platform Designer が生成し た nios2\_system.sopcinfo を選択します。
- 4. Application project 欄の Project name には soft\_test をタイプします。
- 5. Templates 欄からリストの一番上の Blank Project を選択して、[Finish] ボタンをクリックします。

Nos II Application and bor non	n Template	and the second second	
ios II Software Examples	ard support package based on a software example templa	te	
Target hardware information			
SOPC Information File name:	C:¥Lab¥nios2_lab¥nios2_basic_prj¥nios2_system.sopcinfo		
CPU name:	nios2_cpu 🔹		
Application project			
Project name: soft_test			
Project template Template Blank Project Count Binary Float2 Functionality Eloat2 GCC	Template description Blank Project creates an empty project to which you can add your code. For details, click Finish to create the project and refer to the readme but file in the project directory.	E	
Hoat2 Performance Hello Freestanding Hello MicroC/OS-II	The BSP for this template is based on the Altera HAL operating system. To use a BSP based on a different operating system, click Next and select the BSP from	•	

6. 新しいプロジェクトが作成されて、 Project Explorer の項目に soft\_test と soft\_test\_bsp が追加されま す。

▲ プロジェクトの本体は、C:¥Lab¥nios2\_lab¥nios2\_basic\_prj¥software 以下に作成されます。

<mark>ြဲ Project Explore</mark> r 🛛	- 6
-	<u>⊈</u> > ⊽
Soft_test	
Soft test bsp Ini	ios2 system]

#### 7-3. C ソース・コードの作成

Nios® II SBT の New Source File ウィザードを用いて、ソース・コードを作成します。

1. soft\_test フォルダをハイライトして、右クリックから New  $\Rightarrow$  Source File を選択します。

		New	•		Project
isoft t		Go Into		19	File
∠ soft_t		Open in New Window		Ê	File from Template
	Ð	Сору	Ctrl+C		Folder
	圈	Paste	Ctrl+V	<b>C</b> + <b>1</b>	Nios II Application
	×	Delete	Delete	CH	Nios II Application and BSP from Template
	80,	Remove from Context	Ctrl+Alt+Shift+Down	C+4	Nios II Board Support Package
		Source	•	C	Nios II Library
		Move		G	Class
		Rename	F2	h	Header File
	Ès	Import		C	Source File
	è.	Export		63	Source Folder

 New Source File ウインドウの Source folder 欄へ、Browse ボタンよりソフトウェア・プロジェクトのアプリ ケーション・プロジェクト(\_bsp がついていない方のフォルダ)を選択します。 Source file 欄へ、 led\_output.c のソフトウェアのファイル名を拡張子 .c をつけて入力し、[Finish] ボタンをクリックしま す。

New Source	File	
Source File Create a new	v source file.	C
Source folder:	soft_test	Browse
Source file:	led_output.c	
Template:	Default C++ source template	Configure
?	Finish	Cancel



- 3. 次の図のように、led\_output.c をハイライトして、図に掲載されている内容と同じ内容の C ソース・コードを、右側のメイン・エディタ内に直接記述してください。
  - ▲ 時間短縮のため、ハードウェア・プロジェクト・フォルダ内に完成した led\_ouput.c ファイルがありますので、それをテキスト・エディタで開いて、コピー&ペーストしても構いません。

```
c led_output.c 🖾
  2⊕ * led_output.c.
  7
  8
  9 #include <stdio.h>
?10 #include <io.h>
 11 #include <unistd.h>
?12 #include "system.h"
 13
 140 int main(void)
 15 {
         printf("Hello from Nios II, start!!");
 16
 17
 18
         while(1)
 19
         {
             IOWR(LED_PIO_BASE, 0, 0x55);
 20
             usleep(500000);
 21
 22
             IOWR(LED_PIO_BASE, 0, 0xAA);
             usleep(500000);
 23
 24
             IOWR(LED PIO BASE, 0, 0x00);
 25
             usleep(500000);
 26
             IOWR(LED PIO BASE, 0, 0xFF);
 27
             usleep(500000);
 28
         ł
 29
         return 0;
 30
    }
 31
 32
 33
```

4. 記述が終了したら、File メニュー ⇒ Save を選択して、ファイルを保存します。



Flash Programme

7-4. ソフトウェア・プロジェクトのビルドとプログラムの実行

1. システムの設定を確認します。 soft\_test\_bsp フォルダを右クリックして、 Nios II ⇒ BSP Editor を選択 します。

🏠 Project Explorer 🔅		E 😫 🗟 🗸 🖓 E	c led	Loutput.c 🛛
⊿ 😂 soft_test			20	* led_output.c[]
👂 🗊 Includes			7	
Ied_output.e	5		9	<pre>#include <stdio.h></stdio.h></pre>
create-this-	app		210	<pre>#include <io.h></io.h></pre>
Makefile			11	<pre>#include <unistd.h> #include "system b"</unistd.h></pre>
readme.txt			13	AINCINGE SARCENTU
soft test bsn [	nincī	customl	140	int main(void)
		New	•	printf("Hello from Nics II st
		Go Into		princi ( nello nom gaog il, sa
		Ones in New Window		while(1)
		Open in New Window		I TOWR (LED PTO BASE 0 0x55
		Сору	Ctrl+C	usleep(500000);
	175	Paste	Ctrl+V	IOWR(LED_PIO_BASE, 0, 0xAA
	×	Delete	Delete	IOWR(LED PIO BASE, 0, 0x00
		Romana from Contaut Ctd - Alt - Ch	ft i Dawa	usleep(500000):
~~~~	~~	~~~~~≪中略≫~	~~	~~~~~~
		Debug As		ld Console [soft_test_bsp]
		Profile As Restore from Local History	+	<pre>&gt;8 **** Clean-only build of config lean</pre>
		Nios II	۶.	Nios II Command Shell
	*	Run C/C++ Code Analysis		Generate BSP

- 2. BSP Editor が起動します。
- Linker Script タブを選択して、各リンカ・セクション(.bss, .text 等)の設定を下図のように全て、 onchip\_memory に設定(Linker Region Name をクリック ⇒ ブルダウンから選択)したら、 [Generate] ボタンをクリックし、[Exit] ボタンで画面を閉じます。

Compare With

ain Software Packages Drivers Linker Script Enable File Generation Target BSP Directory inker Section Mappings Linker Section Name Linker Region Name Linker Section Mappings Linker Section Mappings Linker Section Mappings Linker Region Name Linker Section Mappings Linker Section Name Linker Section Sectio	ve efaults
Inker Section Mappings Linker Section Name Linker Region Name Linker R	efaults
Linker Region Name Linker Region	i efaults
biss onchip_memory onchip_memory exceptions, onchip_memory onchip_memory onchip_memory exceptions, onchip_memory onchip_memory onchip_memory onchip_memory onchip_memory onchip_memory onchip_memory onchip_memory onchip_memory exceptions, onchip_memory onc	ve efaults
	efaults
exceptions: onchip_memory onch	
heap onchip_memory onchip_memory onchip_memory onchip_memory onchip_memory onchip_memory stack onchip_memory onchip_memory onchip_memory text onchip_memory	
rodata onch ip_memory onch ip_memory onch ip_memory onch ip_memory stack onch ip_memory onch ip_memory onch ip_memory onch ip_memory text onch ip_memory onc	
nvdata onchip_memory onchip_memory stack onchip_memory onchip_memory text onchip_memory on chip_memory onchip_memory onchip_memo	
stack onchip_memory onchip_memory text onchip_memory crichip_memory crichip_memor	
text onchip sessory onchip sessory	
Inker Memory Regions	
Jinker Memory Regions	
Intel Perinty regions Intel Perinty regions	
and region rance read as range r	ve
Incomp internet of a second se	ofoulto
Sec 0.04/12/000 0.04/12/011 0.001/9 02 0 0 CSUC	crauits
nam 000400000 004001111 Stam 024200 0	
Add Memory	/ Device
Remove Memory	bry Device
Memory	leane
	Jaugenn
Memory	Map

- ▲ この設定によって本プログラムは、ワーク・メモリとして onchip\_memory のみを使用する設定となりま す。現時点では、外部 SRAM 等のハードウェアの動作が保証されないため、オンチップ・メモリを使用 して Nios®Ⅱ の初期動作の確認を行います。
- soft\_test\_bsp フォルダを右クリックして、Properties を選択します。起動画面内のカテゴリで Nios II BSP Properties をハイライトして設定項目を確認します。ここでは、BSP プロジェクトのデバッグ・レベルの設定 や最適化レベルの設定などを行います。

本演習では、デフォルトの状態で進めていきますので、[OK] ボタンをクリックします。

Properties for soft_test_bs	P	
type filter text	Nios II BSP Properties	⇔ • ⇔ • •
<ul> <li>Resource Builders</li> <li>C/C++ Build</li> <li>C/C++ General Linux Tools Path</li> <li>Nios II BSP Properties</li> <li>Project References</li> <li>Run/Debug Settings</li> <li>Task Repository</li> <li>WikiText</li> </ul>	SopcInfo:	BSP Editor
?	ОК	Cancel

 次に soft\_test フォルダを右クリックして、 Properties を選択します。起動画面内のカテゴリで Nios II Application Properties をハイライトして設定項目を確認します。ここでは、アプリケーション・プロジェクト のデバッグ・レベルの設定や最適化レベルの設定などを行います。

本演習では、デフォルトの状態で進めていきますので、[OK] ボタンをクリックします。

<ul> <li>Resource</li> <li>Builders</li> <li>C/C++ Build</li> </ul>		
<ul> <li>C/C++ General Linux Tools Path</li> <li>Nios II Application Prope Project References Run/Debug Settings</li> <li>Task Repository WikiText</li> </ul>	Configuration: default [ active ]  Flags  ELF name: soft_test.elf Defined symbols: Undefined symbols: Assembler flags: Warning flags: Undefined symbols: Linker flags User flags User flags: Linker flags User flags User flags: SP project location: ./soft_test_bsp/	Manage Configurations



 ソフトウェアをビルド(Build)します。Nios<sup>®</sup> II SBT の左枠のアプリケーション・プロジェクトのフォルダ (\_bsp がついていない方)を選択し、右クリック ⇒ Build Project をクリックします。

Console にエラーメッセージが出なければビルド完了です。

😂 Nios II - Eclipse	A REAL PROPERTY AND A REAL PROPERTY A REAL PROPERTY A REAL PROPERTY A REAL PROPERTY A REAL PROPERTY A REAL PROPERT		
File Edit Source Refactor Navigate Sea	rch Project Run Nios II Window		
□ • □ □   □   □ = □ • □ • □ • □ • □ • □ • □ • □ • □ •	\$ • 0 • <b>%</b> • 😕 🖯 🔗 •		
Project Explorer 🕱			
⊿ 😂 soft_test			
▶ 🗊 Incluc New	•		
▶ 🛃 led_ol 🛛 🛛 Go Into			
Create	Build Project		
Maket	Building project		
📄 readn 📳 Copy			
▲ Soft_test Paste			
🖻 🔊 Incluc 💥 Delete			
🕨 👝 driver 🧶 Remove from Context	Always run in background		
HAL Source			
D le alt_sy Move		Run in Background Cancel	Details >>
Iinker Rename	F2		
b j system			
create import			
📄 linker 🎦 Export			
🗋 Makel 🛛 Build Project	N		
mem Clean Project	No. 6		

- 7. ビルドしたソフトウェアをターゲットのシステムで実行します。Nios<sup>®</sup> II SBT の左枠のアプリケーション・プロジェクトのフォルダ(\_bsp がついていない方)を選択し、右クリック  $\Rightarrow$  Run As  $\Rightarrow$  Nios II Hardware をクリックします。
- 8. Run Configurations 画面が起動したら、 Nios II Hardware フォルダの soft\_test Nios II Hardware configuration がハイライトされていることを確認します。
- 9. Target Connection タブを選択して、右上の [Refresh Connections] ボタンをクリックします。図のように USB-Blaster が検出して、ターゲット・ボードとの接続が確認できます。
- 10. 右下にある [Apply] ボタンをクリックして、その下の [Run] ボタンをクリックします。

eate, manage, and run configuration The expected Stdout device name do	ons les not match the selected	target byte stream devi	ce name.				
🗎 🗶 📄 🌦 🔹	Name: soft_test Nios I	I Hardware configuratio	n				
pe filter text	Project 🛄 Target (	Connection 🔅 Debug	ger 🖏 Source	Common			
C/C++ Application C/C++ Remote Application	Connections Processors:						
Launch Group	Cable	Device	Device ID	Instance ID	Name	Architecture	Refresh Connect
🛤 Nios II Hardware	ISB-Blaster on localho	st [USB-0] 5CGTFD5(C5		0	nios2-0	Nios253	Resolve Name
Soft_test Nios II Hardware of Nios II Hardware of Nios II Hardware v2 (beta)	Byte Stream Devices:						System ID Propert
📕 Nios II ModelSim	Cable	Device	Device ID	Instance ID	Name	Version	
Nios II ModelSim v2 (beta)	JSB-Blaster on localho	st [USB-0] 500TFD5(05.		0	jtaguart 0		
	Disable 'Nios Il Console Quartus Project File name	view Using default .sopcinfo & .jc	li files extracted fr	rom ELF >			
	System ID checks						
	Ignore mismatched sys	tem ID					
	Ignore mismatched sys	tem timestamp					
	Pownload		m				•
ter matched 8 of 8 items					Re <u>v</u> ert		Apply



11. Console ウインドウに、C 言語ソース上に記載した printf() 関数の出力キャラクタ (例:Hello from Nios II, start!!) が確認できます。同時に、開発ボード上の緑色 LED の点滅も確認してください。確認したら、赤い ボタンをクリックしてプログラムを終了します。



#### ~ 以上で演習本編はすべて終了です。~

~ 時間が余った方は、次のオプション演習へ進んでください ~

【 補足 】 システム・ヘッダ・ファイルの自動生成 とアルテラ専用マクロ

Nios® II ソフトウェアでは、system.h というシステム・ヘッダ・ファイルをツールが自動生成します。

Nios® II や Platform Designer のハードウェア情報 (各ペリフェラルのベース・アドレス等のアドレス・マップ情報や IRQ の優先順位など) が記述されており、アプリケーション・コードでインクルードするだけでペリフェラル にアクセスができるようになります。

インテル FPGA では専用マクロが幾つか用意されており、今回の題材では、その中の一つとして IOWR マクロを使用しました。これらのマクロは、ヘッダ・ファイル io.h をインクルードすることで利用できます。

IOWR(LED\_PIO\_BASE, 0, 0x55)

ベース・アドレス 0x02201020 (本演習では、別のアドレス値にアサインされていますが、このアドレスは構成 の仕方によって異なり、ほとんどのケースでユーザが任意に指定可能です) にマッピングされている led\_pio ペリフェラル内のレジスタ・アドレス 0 に 0x55 をライトします。 led\_pio のベース・アドレスは、 system.h 内で LED\_PIO\_BASE として定義されています。



usleep(500000) は、500000 us つまり 0.5 秒だけプログラムの実行を停止する為の関数(HAL API) です。

### 8. <u>Memory Test の実行 (オプション)</u>

これまでの手順で Nios<sup>®</sup> II のプログラムを JTAG 経由でオンチップ・メモリにダウンロードし動作させるところ まで成功しました。ここでは、外部メモリの動作が正しいことを確認するために予め用意されている Memory Test プログラムを動作させて、外部 SRAM の動作について確認を行います。

#### 8-1. ソフトウェア・プロジェクトの作成

Nios<sup>®</sup> II SBT の New Project ウィザードを用いて、ソフトウェア・プロジェクトを作成します。 以下の手順に従ってください。

- 1. Nios<sup>®</sup> II SBT のメイン画面の File メニュー ⇒ New ⇒ Nios II Applocation and BSP from Template を選択 します。
- 2. Nios II Applocation and BSP from Template ウインドウが起動します。
- ソフトウェア・プロジェクトを作成する際に、組み込みシステムのハードウェア情報が記述されたシステム 定義ファイルを指定します。 Target hardware information 欄の SOPC Information File name の右側に あるブラウザ・ボタンをクリックして、ハードウェア・デザインを作成したフォルダにある Platform Designer が生成した nios2\_system.sopcinfo を選択します。
- 4. Application project 欄の Project name には memory\_test とタイプします。
- 5. Templates 欄から Memory Test を選択して、 [Finish] ボタンをクリックします。

Aise I Software Examples       Create a new application and board support package based on a software example template       Target hardware information       SOPC Information File name:     Ci¥Lab¥nios2_lab¥nios2_basic_prj¥nios2_system.sopcinfi       OPU name:     nios2_cpu       Application project       Project name:     memory_test       V Use default location       Project location:     Ci¥Lab¥nios2_lab¥nios2_basic_prj¥software¥memory_test       Immory fest     Template description       Memory Test allows you to test the RAM and flash       Helio MicroC/OS-II     Memory Test allows you to test.       Beckue the RAM test is destructive, do not run the RAM test on any memory best.       Simple Socket Server (RC)       For details, click Finish to create the project and refer	II Application and BSP from	1 Template		
Target hardware information         SOPC Information File name:       C:¥Lab¥nios2_lab¥nios2_system.sopcinf(         CPU name:       nios2_cpu         Application project       Image: Imag	II Software Examples Ite a new application and boa	ird support package based on a software example templa	te	
SOPC Information File name:       C:¥Lab¥nios2_lab¥nios2_basic_prj¥nios2_system.sopcinfr         CPU name:       nios2_cpu         Application project         Project name:       memory_test         I Use default location       Project location:         C:¥Lab¥nios2_lab¥nios2_basic_prj¥software¥memory_test          Project template       Template description         Hello MicroC/OS-II       Memory Test allows you to test the RAM and flash memory on your board. The application presents a menu to choose which memory to test.         Because the RAM test is destructive, do not run the RAM test on any memory being used by this program, including code, data, and exception locations.          For details, click Finish to create the project and refer           @        Enish       Cancel	get hardware information			
CPU name:       nios2_cpu         Application project         Project name:       memory_test         If Use default location         Project location:       C:\#Lab\#nios2_lab\#nios2_basic_prj\#software\#memory_test         Project location:       C:\#Lab\#nios2_lab\#nios2_basic_prj\#software\#memory_test         Project template       Template description         Hello Freestanding       Memory Test allows you to test the RAM and flash memory on your board. The application presents a menu to choose which memory to test.         Because the RAM test is destructive, do not run the RAM test on any memory being used by this program, including code, data, and exception locations.         Simple Socket Server (RC         ************************************	PC Information File name:	C:¥Lab¥nios2_lab¥nios2_basic_prj¥nios2_system.sopcinf		
Application project       Project name:     memory_test       If Use default location     Project location:       C:¥Lab¥nios2_lab¥nios2_labsic_prj¥software¥memory_test        Project template     Template description       Hello Freestanding     Memory Test allows you to test the RAM and flash memory on your board. The application presents a menu to choose which memory to test.       Hello World Small     Because the RAM test is destructive, do not run the RAM test on any memory being used by this program, including code, data, and exception locations.       Simple Socket Server (RC, memory Test).     For details, click Finish to create the project and refer       ************************************	U name:	nios2_cpu 👻		
Project name:       memory_test         Image: Second S	lication project			
Image: Classic location       Classic location:       Classic location:       Classic location:       Classic location:       Classic location:       Classic location:       Image: Classic loc	ject name: memory_test			
Image: Control of the second secon				
Project location:       C:¥Lab¥nios2_lab¥nios2_basic_prj¥software¥memory_test         Project template         Templates         Hello Freestanding         Hello MicroC/OS-II         Hello World         Hello World         Hello World         Hello World Small         Memory Test         Memory Test         Because the RAM test is destructive, do not run the RAM test on any memory being used by this program, including code, data, and exception locations.         Simple Socket Server         Simple Socket Server         Simple Socket Server         Simple Socket Server         Hello Socket Server         Hello Socket Server         Hello Socket Server	Use default location			
Project template         Templates         Hello Freestanding         Hello MicroC/OS-II         Hello World         Hello World Small         Memory Test allows you to test the RAM and flash         Memory Test         Memory Test single Socket Server         Simple Socket Server         Simple Socket Server         Simple Socket Server         Simple Socket Server         Hello Socket Server         Hello Socket Server         Hello Socket Server         Socket Server         Socket Server      <	Project location: C:¥Lab¥r	ijos7 Jah¥nios2 hasic pri¥software¥memory test		
Project template         Templates         Hello Freestanding         Hello MicroC/OS-II         Hello World         Hello World Small         Memory Test         Memory Test sidestructive, do not run the         RAM test on any memory being used by this program,         including code, data, and exception locations.         Simple Socket Server         Simple Socket Server (RC)         Image: Concerce minime         Image: Concerce minime         Reads         List Concerce minime         Reads         Next >         Einish         Cancel		inter_dominate_base_prj contrar er mentary_teat		
Templates       Template description         Hello Freestanding       Memory Test allows you to test the RAM and flash memory on your board. The application presents a menu to choose which memory to test.         Hello World       memory Test         Hello World Small       Because the RAM test is destructive, do not run the RAM test on any memory being used by this program, including code, data, and exception locations.         Simple Socket Server (Rtwisser and the rest of the test is destructive and refer test and the rest of the test is destructive and refer test and refer t	roject template			
Templates       Template description         Hello Freestanding       Memory Test allows you to test the RAM and flash memory on your board. The application presents a menu to choose which memory to test.         Hello World Small       memory Test         Memory Test       Because the RAM test is destructive, do not run the RAM test on any memory being used by this program, including code, data, and exception locations.         Simple Socket Server (R(       For details, click Finish to create the project and refer         *       III				
Hello Freestanding     Memory Test allows you to test the RAM and flash       Hello MicroC/OS-II     memory on your board. The application presents a       Hello World     menu to choose which memory to test.       Hello World Small     Because the RAM test is destructive, do not run the       Memory Test     Because the RAM test is destructive, do not run the       RAM test on any memory being used by this program,     including code, data, and exception locations.       Simple Socket Server (R(     For details, click Finish to create the project and refer       III     Memory	emplates	Template description		
Image: Second	Hello Freestanding	Memory Test allows you to test the RAM and flash		
Image: Socket Server (RC, Simple Socket Server (RC, S	Helio MicroC/OS-II	memory on your board. The application presents a		
Image: Work of the second s	Hello World Cmall	menu to choose which memory to test.	and the second se	
Image: Socket Server     Simple Socket Server       Simple Socket Server     RAM test on any memory being used by this program, including code, data, and exception locations.       Simple Socket Server     For details, click Finish to create the project and refer       The socket Server     For details, click Finish to create the project and refer       Image: Socket Server     Simple Socket Server       Image: Socket Server     For details, click Finish to create the project and refer       Image: Socket Server     Simple Socket Server	Memory Test	Bacques the BAM test is destructive, do not run the		
Simple Socket Server     including code, data, and exception locations.       Simple Socket Server (R(	Memory Lest Small	RAM test on any memory being used by this program		
Simple Socket Server (R(     For details, click Finish to create the project and refer       ?     < Back	Simple Socket Server	including code, data, and exception locations.		
With Concert     For details, click Finish to create the project and refer       III     +       For details, click Finish to create the project and refer       III       IIII       IIII       IIIIII       IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Simple Socket Server (R(	including code, data, and exception locations.		
?     < Back	Woh Convor	For details, click Finish to create the project and refer		
? < Back Next > Einish Cancel	• •			
? < Back Next > Einish Cancel				
? < Back Next > Einish Cancel				
? < Back Next > Einish Cancel				
Back         Next >         Einish         Cancel				
Concertainer	1	< Back Next >	Finish	Cancel
		- Frank	Linian	ounice:

6. 新しいプロジェクトが作成されて、Project Explorer の項目に memory\_test と memory\_test\_bsp が追加 されます。

※プロジェクトの本体は、C:¥Lab¥nios2\_lab¥nios2\_basic\_prj¥software 以下に作成されます。



- 8-2. ソフトウェア・プロジェクトのビルドとプログラムの実行
  - 1. システムの設定を確認します。 memory\_test\_bsp フォルダを右クリックして、 Nios II ⇒ BSP Editor を選 択します。
  - 2. BSP Editor が起動したら、 Main タブにて enable\_rediced\_device\_driver の項目をチェックします。この設 定で、ビルドするソフトウェアのサイズを小さくすることができます (ただし、一部機能が制限されます)。

ile Edit Tools Help		
ain Software Packages Drivers Linker Script Enable File G	eneration Target BSP Directory	
SOPC Information file:\\pios2_system.sopcinfo CPU name: nios2_cpu Operating system: Altera HAL BSP target directory: C:\Lab\nios2_lab\nios2_basic_prj\soft	Version: default → ware\memory_test_bsp	
-Settings -Common	hal sys_clk_timer: timestamp_timer: stdin: stdout: stdout: stderr: enable_small_c_library enable_sprof enable_reduced_device_driver enable_sim_optimize hal.linker	none none jtag_uart jtag_uart jtag_uart

 次に Linker Script タブを選択して、各リンカ・セクション (.bss, .text 等)の設定を下図のように全て、 onchip\_memory に設定 (Linker Region Name をクリックで選択) したら、右下の [Generate] ボタンをクリ ックし、 [Exit] ボタンで画面を閉じます。

Main Software Packages Drivers Linker Script	Enable File Generation Target BSP Directo	жА	
Linker Section Mappings			
Linker Section Name	Linker Region Name	Memory Device Name	
bes	onchip memory	onchip_memory	
.entry	reset	onchip_memory	
.exceptions	onchip_memory	onchip_memory	
.heap	onchip_memory	onchip_memory	
.rodata	onchip_memory	onchip_memory	
.rwdata	onchip_memory	onchip_memory	
.stack	onchip_memory	onchip_memory	
.text	onchip memory	onchip memory	

- 4. ソフトウェアをビルドします。Nios<sup>®</sup> II SBT の左枠のアプリケーション・プロジェクトのフォルダ(memory\_test) を選択し、右クリック ⇒ Build Project をクリックします。
- 5. ビルドが完了したら、ビルドしたソフトウェアをターゲットのシステムで実行します。 Nios<sup>®</sup> II SBT の左枠の アプリケーション・プロジェクトのフォルダ (memory\_test) を再度選択し、右クリック ⇒ Run As ⇒ Nios II Hardware をクリックします。
- 6. Memory Test プログラムが実行され Nios® II のコンソール画面に以下のように表示されます。

🖹 Problems 🧔 Tasks 📮 Console 🛗 Nios II Console 🕱 🔲 Properties
memory_test Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtag_uart.jtag
<pre>&lt;&gt; Nios II Memory Test. &lt;&gt; This software example tests the memory in your system to assure it is working properly. This test is destructive to the contents of the memory it tests. Assure the memory being tested does not contain</pre>
the executable or data sections of this code or the exception address of the system.
Memory Test Main Menu
a: Test RAM
b: Test Flash q: Exit
Select Choice (a-b): [Followed by <enter>]</enter>

7. まずは、外部 SRAM インタフェースの動作を確認します。以下のように入力します。正常終了すると以下 のように表示されます。全て passed となっていれば問題ありません。

📳 Problems 🧔 Tasks 📃 Console 🛗 Nios II Cons	sole 🛛 🔲 Properties
memory_test Nios II Hardware configuration - cable: USB-Blaster	on localhost [USB-0] device ID: 1 instance ID: 0 name: jtag_uart.jtag
Select Choice (a-b): [Followed by <enter>]</enter>	
Base address to start memory test: (i.e. 0x800000)	- a: lest RAIM を選択
> 0×04080000 -	- sram のベース・アドレス
End Address:	
D×D40fffff ◀	- sram のエンド・アドレス
Testing RAM from 0x4080000 to 0x40FFFFF -Data bus test passed -Address bus test passed -Byte and half-word access test passed -Testing each bit in memory device passed Memory at 0x4080000 Okay	正常に終了すると passed と表示される
Press enter to continue	

8. SRAM の動作を確認することができたので、プログラムを終了 🔳 します。

# 9. オンチップ・メモリからのブート(オプション)

使用する文字は、以下のルールに従ってください。これまでの手順で、Nios® II のソフトウェア・プログラムを JTAG 経由のダウンロードで動作させることに成功しました。また、Memory Test にて外部 SRAM が正常にア クセスできることを確認しました。次に Nios II のソフトウェア・プログラムを FPGA デバイス内のオンチップ・メモ リからブートする方法について進めていきます。

まず前提として、下図のように Platform Designer 上にある Nios® II のリセット・ベクタの設定がオンチップ・メ モリに設定されていることを確認してください。この設定により、 FPGA 起動後に Nios® II がリセット・ベクタで指 定されたアドレスからソフトウェア・プログラムをロードしブートが開始します。

stem:nios2_system Path:nios2_cpu			_	
ios II Processor era_nios2_gen2				
Main Vectors Caches and Memory Interf	aces Arithmetic Instructions	MMU and MPU Settings	JTAG Debug	Advanced Features
Reset Vector				
Reset vector memory	onchip_memorys1 -			
Reset vector offset:	0×00000000			
Reset vector:	0×04120000			
* Exception Vector				
Exception vector memory	onchip_memorys1 -	1		
Exception vector offset:	0×00000020			
Exception vector:	0×04120020			
* Fast TLB Miss Exception Vector				
Fast TLB Miss Exception vector memory	None			
Fast TLB Miss Exception vector offset:	0×00000000			
	here and the second			

- 9-1. Nios® II SBT での HEX ファイルの生成
  - 1. Nios<sup>®</sup> II SBT にてビルド済のアプリケーション・ソフトウェア・プロジェクト(本例では先の演習で動作確認した soft\_test を使用します)を右クリックし、プルダウン・メニューから Make Targets ⇒ Build を選択します。

Make Targets		Create	
Index	×.	Build	Shift+F9
Build Configurations	•	Rebuild Last Target	F9

2. Make Targets 画面が起動したら、mem\_init\_generate をハイライトして、[Build] ボタンをクリックします。

Target	Location	Add
mem_init_install		Remove
@ mem_init_generate		
leip 🛞 help		Edit



3. この操作によって、mem\_init フォルダがアプリケーション・プロジェクトの中に生成されます。



- 4. mem\_init フォルダ内には、meminit.qip ファイルと nios2\_system\_onchip\_memory.hex ファイル (このフ ァイル名は、プロジェクトごとに異なります)を確認することができます。
- 5. nios2\_system\_onchip\_memory.hex ファイルをテキスト・エディタで開くと、Hex フォーマットでの記述となり ますが、ソフトウェア・コードがデータとして反映されていることが確認できます(下図は一例となります)。



- 9-2. Quartus® Prime での設定とコンパイルから実行まで
  - 1. Quartus<sup>®</sup> Prime で Assignments メニュー ⇒ Settings を選択します。

Ass	ignments	Processing	Tools	Window	Help
٠	Device				
1	<u>S</u> ettings	20		Ctrl+Shi	ift+E
_				145	

Settings ウインドウが起動します。画面左側の Category 欄から Files を選択して、右側の File name 欄に、前述の meminit.qip ファイルを指定します。File name 枠内に追加されていることが確認できたら、右下の [OK] ボタンをクリックして Settings ウインドウを閉じます。

整理 ▼ 新しいフォル	レダー		•	
E ピクチャ ·	* 名前 *	更新日時	種類	サイズ
ビデオ	hdl_sim	2019/01/24 16:10	ファイル フォル	
👌 ミュージック	meminit.qip	2019/01/24 16:10	QIP ファイル	11
■ コンピューター	1			
■ コンピューター				

Settings - nios2_basic_lab		and the second se		
ategory:				Device/Boa
General	Files			
Files	Select the design files you want to include in the proj	ect. Click Add All to add all design files i	n the project directory to	the project
Libraries				
IP Settings	File name:			Add
IP Catalog Search Locations Design Templates			×	Add Al
<ul> <li>Operating Settings and Conditions</li> </ul>	File Name	Туре	Library Design Entry/S	Remov
Voltage	software/soft test/mem init/meminit.gip	IP Variation File (.qip)	<none></none>	
Temperature	my_original_constrain.sdc	Synopsys Design Constraints File	<none></none>	Up
Compilation Process Settings	nios2_system/synthesis/nios2_system.qip nios2_basic_lab.bdf	IP Variation File (.qip) Block Diagram/Schematic File	<none></none>	Down
EDA Tool Settings Design Entry/Synthesis		9		Propertie
Simulation				

3. Processing メニュー ⇒ Start Compilation を選択して、フル・コンパイルを行います。





4. コンパイルが完了すると作成した HEX ファイルの内容を含んだ SOF/POF ファイルが生成されるので Quartus<sup>®</sup> Prime Programmer で SOF ファイルを書き込みます。

Eait View	Processing Tools Window Help								Search alter	a.com	
ardware Setu able real-time	p USB-Blaster [USB-0] = ISP to allow background programming w	hen available		Mode: JTAG		Ŧ	Prog	ress:	399		
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAM
Stop	output_files/nios2_basic_lab.sof	5CGXFC5C6F27	03ADE38C	03ADE38C	<b>V</b>						
C Delete Add File hange File Save File	· · · · · · · · · · · · · · · · · · ·										Þ

- 5. 正常であれば、FPGA への書き込みが完了した時点で LED の点滅が確認できます。
- また soft\_test では、標準出力に JTAG\_UART を使用しているため、Nios® II Command Shell から標準出 カへの出力を確認してください。すべてのプログラム ⇒ Intel FPGA 18.1.0.625 Standard Edition ⇒ Nios II EDS 18.1.0.625 ⇒ Nios II Command Shell (Quartus Prime 18.1) を選択すると、Nios® II Command Shell が 起動しますので、コンソールから nios2-terminal と入力します。

「Hello from Nios II, start!!」のメッセージが表示されます。



9-3. HEX ファイル更新時の Quartus® Prime プロジェクトへの反映方法

\Lambda ALTIMA

Nios® II の実行ファイルである ELF ファイルが変更された際には、mem\_init\_generate を再度実行すること で HEX ファイルを更新しますが、そのたびに Quartus® Prime でフル・コンパイルするのは効率的ではありません。ここでは HEX ファイルに対応するオンチップ・メモリの内容のみ更新する方法を実行します。

- Nios<sup>®</sup> II SBT で soft\_test プロジェクトのプログラム (led\_output.c) を編集して、LED の点滅速度や点滅 パターンを変えてみましょう。編集したらファイルを保存し、soft\_test フォルダを右クリック ⇒ Build Project を実行します。
- 2. soft\_test フォルダを右クリック ⇒ Make Targets ⇒ Build を選択し、mem\_init\_generate を実行します。
- 3. Quartus<sup>®</sup> Prime の Processing メニューから Update Memory Initialization File を選択します。この処理で 更新された HEX ファイルを解析してファイルに問題があるかどうかをチェックします。

この時点では、HEX ファイルの内容は SOF/POF ファイルには反映されていません。



 次に、Processing メニューから Start ⇒ Start Assembler を選択します。この処理で SOF/POF ファイル に新しい HEX ファイルの内容が反映されます。フル・コンパイルに比べて、FPGA 内部のオンチップ・メモ リのみ更新するので処理が短時間で済みます。

Pro	cessing <u>T</u> ools <u>W</u> indow <u>H</u> elp			
	Stop Processing	Ctrl+Shift+C	• • • • •	
	Start <u>C</u> ompilation	Ctrl+L	i2_basic_lab.bdf 🛛 🔍 😓	Cor
1	Analyze Current <u>File</u>		Flow Summary	- 11
	Start	ŀ.	Start Hierarchy Elaboration	
•	<u>U</u> pdate Memory Initialization File Compilation <u>R</u> eport Dynamic Synthesis Report	Ctrl+R	Start Analysis & Elaboration         Start Analysis & Synthesis         Ctrl+K         Start Partition Merge	
1	Power Analyzer Tool		Start <u>F</u> itter	
	SS <u>N</u> Analyzer Tool		Start Assembler	
	Peceive Compilation Status Natification	une .	Start Timing Analyzer Ctrl+Shift+	т

5. この処理が完了したら、SOF/POF ファイルのタイムスタンプが更新されていることを確認してください。また、改めて FPGA に書き込んで実機での動作を確認してください。

### 10. EPCQ からのブート (オプション)

🛆 ALTIMA

前章ではオンチップ・メモリからのブート方法を実施しましたが、この章ではボードに実装されている EPCQ デ バイスからのブートを実施します。合わせて、今までの内容では Nios® II のソフトウェアのワーク・メモリとして オンチップ・メモリを使用しましたが、この章では SRAM デバイスをワーク・メモリとして設定し、動作を見ていき ます。これにより、オンチップ・メモリの使用量を少なくすることが可能になります。 Platform Designer 上のオン チップ・メモリの設定でサイズの設定を小さくすることで、オンチップ・メモリの使用リソースが削減されますので、 他のロジックでさらにオンチップ・メモリを有効に使用できるようになります。

#### 10-1.リセット・ベクタと例外ベクタの設定

使用する文字は、以下のルールに従ってください。

- 1. Platform Designer を起動し、nios2\_cpu をハイライト後、マウスで右クリックし [Edit] を選択します。 Nios® II プロセッサ・コアの GUI 画面が起動します。
- 2. Vectors タブの Reset Vector の項目のプルダウン・メニューより、epcq.avl\_mem を選択し、Reset vector offset には、0x00420000 を設定します。
- 3. Exception Vector のプルダウン・メニューより sram.uas を選択します。

Main Vectors Caches and Memory In	terfaces Arithmetic Instructions MMU and
Reset Vector	
Reset vector memory	epcq.avl_mem 👻
Reset vector offset:	0×00420000
Reset vector:	0×02420000
Exception Vector	
Exception vector memory	sramuas 👻
Exception vector offset:	0×00000020
Exception vector:	0×04080020

▲ 0x00420000 というアドレス値は、後述の JIC ファイルの生成時に生成されるマップ・ファイル(\*.map) の値から参照した値となります。EPCS/EPCQ デバイスから FPGA のハードウェアデータ(SOF ファイ ル)をコンフィグレーションする場合には 0x0000000 番地からと決まっており、Nios® II のソフトウェ アデータ(ELF ファイル)は、そのあとの番地に置かれます。どこの番地に置かれるかはデバイスの 規模により変わってくるので、本来は JIC ファイルを先に生成して、マップ・ファイルからソフトウェアデ ータのスタート・アドレスを確認しておく必要があります。

BLOCK	START ADDRESS	END ADDRESS
epcq.hex Page_0	0×00420000 0×00000000	0x0043187B x0041F013
Configurat Configurat Quad-Seria	ion device: 5CGXFC5 ion mode: Active Se I configuration dev	C6 rial ice dummy clock cycle: 4
Notes:		
- Data che	cksum for this conv	ersion is 0xC03B1793
- All the :	addresses in this f	ile are byte addresses

- 4. Nios® II プロセッサ・コアの画面を [Finish] で閉じ、File メニュー ⇒ Save より変更を保存します。
- 5. Platform Designer のメイン・ウインドウから Generate メニュー ⇒ Generate HDL を選択します。
- 6. Generation ウインドウが表示されるので、[Generate] ボタンをクリックしてシステムを生成します。
- 7. Platform Designer の生成が終わったら、Quartus<sup>®</sup> Prime の **Processing** メニュー ⇒ **Start Compilation** を選択し、ハードウェア・デザインのコンパイルを実行します。ここでは、コンパイルの完了を待たずに、次 にステップに進んでください。

10-2.ソフトウェア・プロジェクトのビルド

🛆 ALTIMA

- 1. ハードウェアを変更したので、ソフトウェア・プロジェクトをクリーンします。 Nios<sup>®</sup> II SBT にて soft\_test フォ ルダを右クリックして、 Nios II ⇒ Clean Project を選択します。
- 2. 次に、Nios<sup>®</sup> II SBT にて soft\_test\_bsp フォルダを右クリックして、Nios II ⇒ BSP Editor を選択します。 BSP Editor が起動します。
- 3. Main タブの Setting ⇒ Advanced ⇒ hal ⇒ linker を表示します。図のように全てのチェックを外してく ださい。

ain Software Packages Drivers Linker Script Enable File C	Seneration Target BSP Directory
SOPC Information file:\\nios2_system.sopcinfo CPU name: nios2_cpu Operating system: Altera HAL BSP target directory: C:\Lab\nios2_lab\nios2_basic_prj\soft	Version: default →
<ul> <li>Securings</li> <li>Common</li> <li>Advanced</li> <li>hal</li> <li>max_file_descriptors</li> <li>enable_instruction_related_exceptions_ap</li> <li>log_port</li> <li>enable_exit</li> <li>enable_clean_exit</li> <li>enable_clean_exit&lt;</li></ul>	allow_code_at_reset alload allow_code_at_reset alload alload_copy_rodata alload_copy_rwdata alload_copy_exceptions alload_copy_exceptions



 Linker Script タブを選択して、各リンカ・セクション(.bss,.text 等)の設定を下図のように entry、exceptions 以外を全て sram に設定したら、[Generate] ボタンをクリックし、[Exit] ボタンで画面 を閉じます。

ile Edit Tools Help					
Main   Software Packages   Drivers	Linker Script Enable File Generation	Target BSP Directory			
Linker Section Mappings					
Linker Section Name	Linker Region Name	r i	Memory Device Name		Add
.bss	sram		sram		Remove
entry	reset		epcq_avl_mem		Restore Defaults
.exceptions	sram		sram		
.heap	sram		sram		
.rodata sram			sram		
.rwoata	sram		sralli sram		
text	sram		sran		
Linker Memory Regions					
Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)	Add
onchip_memory	0×04120000 - 0×0413FFFF	onchip_memory	131072	0	Remove
sram	0×04080020 - 0×040FFFF	sram sram	524256	32	Restore Defaults
sram_BEFORE_EXCEPTION	0×04080000 - 0×0408001F	= sram	32	0	
spcq_avl_mem	0×02420020 - 0×03FFFFF	epcq_avl_mem	29229024	4325408	Add Memory Device
reset	0×02420000 - 0×0242001	epcq_avi_mem	32	4325376	Remove Memory Device.
Spcq avi mem before reset	0x02000000 - 0x0241FFF	· jepcq avi mem	4320376		Momoru Llanco
					Memory Usage
2					Memory Map
Graved out entries are automatic	ally created at generate time. They	are not editable or persisted i	n the BSP settings file		
			in the ber county inte.		
nformation Problems Processing					
Finished loading drivers from ense	mble report.				
Loading BSP settings from setting	s file.				
Finished loading SOPC Builder sys	tem info file ", ,  , \nios2_system.sopcir	fo [relative to settings file]"			
Changed mapped section ".bss" fi	rom memory region "onchip_memory" t	o memory region "sram".			
Changed mapped section ".heap"	from memory region "onchip_memory"	"to memory region "sram".			
Changed mapped section ".rodata	a" from memory region "onchip_memor	y" to memory region "sram".			
A second seco	a" from memory region "onchip_memor	y" to memory region "sram".			
U Changed mapped section ".rwdat					
Changed mapped section ".rwdat Changed mapped section ".stack"	from memory region "onchip_memory"	to memory region "sram".			

- 5. soft\_test を右クリック ⇒ Build Project をクリックしソフトウェアをビルドします。
- 6. soft\_test を右クリックして、プルダウン・メニューから Make Targets ⇒ Build を選択します。 Make Targets 画面が起動したら、 mem\_init\_generate をハイライトして、 [Build] ボタンをクリックします。
- 7. この処理で mem\_init フォルダ内には、新たに epcq.hex というファイルが生成されていることが確認できます。





10-3.Convert Programming File を使用した書き込みファイルの生成

- 1. Quartus<sup>®</sup> Prime のコンパイルが完了していることを確認し、File メニュー ⇒ Convert Programming File を選択します。
- Output programming file の設定を行います。ここでは JIC 形式でファイルを生成するので、Programming File Type には JTAG Indirect Configuration File (.jic) を選択します。 Mode は Active Serial を選択します。 Configuration device には EPCQ256 を選択します。

le <u>Tools W</u> indow				Search altera.com
pecify the input files to	convert and the type of p	programming file to generate.		
ou can also import inpu	t file information from o	ther files and save the convers	ion setup information created here fo	r
iture use.				
Conversion setup files				
	Open Conversion Setup	Data	Save Co	nversion Setup
<b>5</b>				
Output programming fi	le			
	1	ration Eile ( iic)		
Programming file type:	JTAG Indirect Configu	ration File Giej		
Programming file type: Options/Boot info	JTAG Indirect Configu Configuration device:	EPCQ256	Mode:	Active Serial
Programming file type: Options/Boot info File name:	JTAG Indirect Configu Configuration device: output_files/output_fi	EPCQ256 lejic	Mode:	Active Serial

3. 次に Input files to convert の設定を行います。Flash Loader の行をハイライトした状態で、[Add Device] ボタンをクリックして、Selected Devices 画面を起動します。

File/Data area	Properties	Start Address	Add Hex D
Flash Loader			Add Sof Pa
SOF Data	Page_0	<auto></auto>	

4. Selected Devices 画面で、 **Device family** 欄から **Cyclone V** をハイライトし、**Device name** 欄の **5CGXFC5C6** を選択して、 **[OK]** ボタンをクリックして設定を反映させます。

Device family	Device name	
Arria V GZ	▲ SCGXBC9A7	A New
Cyclone	5CGXBC9C6	
Cyclone 10 LP	5CGXBC9C7	Import
Cyclone II	5CGXBC9D6	Export
Cyclone III	5CGXBC9D7	
Cyclone III LS	5CGXBC9E6	Edit
Cyclone IV E	5CGXBC9E7	Remove
Cyclone IV GX	E 5CGXFC3B6	E Merrore
🗹 Cyclone V	5CGXFC3B7	Uncheck Al
HardCopy II	5CGXFC4C6	
HardCopy III	5CGXFC4C7	
HardCopy IV	5CGXFC4F6	
🗌 MAX 10	5CGXFC4F7	
MAX II	SCGXFC5C6	
MAX V	☐ 5CGXFC5C7	
C CALL		Ť

5. SOF Data の行をハイライトして、[Add File] ボタンをクリックし、SOF ファイルを選択します。

File/Data area	Properties	Start Address	Add Hex Data
Flash Loader			Add Sof Page
5CGXFC5C6			Contraction of the sector

6. [Add Hex Data] ボタンをクリックして、Nios<sup>®</sup> II SBT で作成した epcq.hex ファイルを指定します。Absolute addressing と Big endian が選択されていることも確認し、[OK] ボタンで設定を反映させます。

ි Add Hex Data	X
Addressing mode	
Absolute addressing	
© Relative addressing	
Set start address: 0x0	
Bit-level endianness	
🔘 Little endian	
Big endian	
Hex file	
c_prj/software/soft_test/mem_	init/epcq.hex
01	Cancel

7. 最後に右下の [Generate] ボタンをクリックすると、output\_file.jic ファイルが生成されます。

File/Data area	Properties	Start Address	Add He <u>x</u> Dat
<ul> <li>Flash Loader</li> <li>5CGXFC5C6</li> </ul>			Add <u>S</u> of Pag
SOF Data	Page_0	<auto></auto>	Add Eile
nios2_basic_lab.sof	5CGXFC5C6F27		Remove
Hex Data	Absolute addressing	0x00420000	Un
epcq.hex			
			Down
			Properties
		(	<u>G</u> enerate Close Help



\Lambda ALTIMA

10-4.Quartus<sup>®</sup> Prime Programmer での JIC ファイルの書き込み

- 1. Quartus<sup>®</sup> Prime の Tools メニュー ⇒ Programmer を選択します。Programmer ウインドウが表示されま す。
- 2. Hardware Setup の項目に USB-Blaster が設定されていることを確認します。
- 3. [Auto Detect] ボタンを押して、ボード上のデバイスをサーチします。
- 4. Select Device が表示されたら、5CGXFC5C6 をチェックして [OK] ボタンをクリックします。
- 5. File 項目の <none> となっている箇所をダブル・クリックして、生成した output\_file.jic ファイルを指定しま す。

output\_files/output\_file.jic

- output\_files/output\_file.jic 行の Program / Configure のチェック・ボックスにチェックを入れます。
   Serial Flash Loader (SFL) 用のファイルが自動的に設定されて、以下の画面のようになります。
- 7. [Start] ボタンをクリックして書き込みを開始します。



- 8. SOF ファイルの書き込みがはじめに行われ、次に HEX (ELF) ファイルの書き込みが行われます。処理に は、1~2 分程度かかります。
- 9. EPCQ デバイスへの書き込みが終わったら、ボードの電源を入れ直してプログラムが正常に動作すること を LED の点滅にて確認してください。

ボードの電源を入れた後、EPCQ デバイスから AS モードで SOF をロードし FPGA が起動します。次に、 FPGA 上の Nios® II がリセット・ベクタとして設定した EPCQ デバイスのアドレス位置からソフトウェア・プロ グラムをロードして動作を開始します。

10. また、Nios<sup>®</sup> II Command Shell を使用して標準出力への出力を確認してください。すべてのプログラム ⇒ Intel FPGA 18.1.0.625 Standard Edition ⇒ Nios II EDS 18.1.0.625 ⇒ Nios II Command Shell (Quartus Prime 18.1) と選択し、Nios<sup>®</sup> II Command Shell を起動します。コンソールから nios2-terminal と入力します。



### 改版履歴

Revision	年月	概要
1	2017 年 7 月	初版
2	2018 年 4 月	ALTIMA テンプレートに変更
3	2019年1月	ver18.1 対応
4	2019 年 7 月	Web 公開用に体裁修正(p26, p27 の図を微修正)

#### 免責およびご利用上の注意

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

- 1. 本資料は非売品です。許可無く転売することや無断複製することを禁じます。
- 2. 本資料は予告なく変更することがあります。
- 本資料の作成には万全を期していますが、万一ご不明な点や誤り、記載漏れなどお気づきの点がありましたら、本資料を入手されました下記代理店までご一報いただければ幸いです。
   株式会社マクニカ アルティマ カンパニー <a href="https://www.alt.macnica.co.jp/">https://www.alt.macnica.co.jp/</a> 技術情報サイト アルティマ技術データベース <a href="https://www.alt.macnica.co.jp/">https://www.alt.macnica.co.jp/</a>
- 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。