

カスタムマイコン設計トライアル

~MAX[®] 10 FPGA を使用した組み込み設計を体感~ **演習マニュアル**

(MAX® 10 開発キット編)



カスタムマイコン設計トライアル 演習マニュアル

(MAX 10 開発キット編)

<u>目次</u>

本書をお読みになる前に	3
はじめに	4
演習で使用する開発環境	4
演習 1. FPGA 設計フローを体験	6
演習 2. FPGA を使用したカスタムマイコンの作成	19
演習 3. 内蔵 ADC を使用して温度測定	42

本書をお読みになる前に

この資料の内容は 2020 年 10 月現在のものです。

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

文書中の記号

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

文中の表記

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

はじめに

この資料は、「カスタムマイコン設計トライアル」受講者向けの演習マニュアルです。

すべての演習を進めるにあたり、各演習でそれぞれのステップの作業を漏れなく実行する必要があります。そのため、進行経過が確認できるように各ステップの先頭に(____)を設けています。そのステップの作業が完了したら、そこへチェック・マークをつけるようにしてください。

演習で使用する開発環境

この演習を実施するためには、以下の環境が必要です。

No.	環境	概要
1	パソコン	Quartus® Prime が動作するスペックを保有するパソコンをご用意ください。
		・サポート OS 情報は、 <u>Table 1: Operating System Support</u> をご覧ください。
		・搭載するメモリ容量の情報は、使用するバージョンの <u>Table 2: Release</u>
		<u>Notes and readme.txt Files</u> をご覧ください。
2	Quartus Prime Standard Edition	開発ソフトウェアをインストールしてください。
	または	各種ソフトウェアのダウンロードおよびインストール方法は、以下の Web
	Quartus Prime Lite Edition	ページをご参照ください。
3	Nios [®] II Software Build Tools for	・ <u>インテル Quartus Prime 開発ソフトウェアおよび ModelSim - Intel FPGA</u>
	Eclipse	Edition のダウンロード方法
		・ <u>インテル Quartus Prime 開発ソフトウェアおよび ModelSim - Intel FPGA</u>
		Edition のインストール方法
4	開発ボード	このチュートリアルに対応した開発 ボードは以下のとおりです。
		・ <u>MAX® 10 FPGA 開発キット: 10M50DAF484C6GES</u>

【表 1-1】 演習実施に必要な開発環境



No.	環境	概要
5	インテル FPGA ダウンロード・ ケーブル II 用ドライバー	MAX 10 開発キットは、オンボード インテル ダウンロード・ケーブル II 対応 です。 ダウンロード・ケーブルをはじめて利用するパソコン をご利用の際は、事前にドライバ ー をインストールしてください。 インストール方法は、以下の Web ページをご参照ください。 ・ <u>USB-Blaster™ II のドライバをインストールしてみよう</u>
6	演習データ	<u>このマニュアルを入手した Web ページ</u> からダウンロードしてください。フ ァイルの解凍先は任意です。
		▲ 注記: 解凍先は、全角やスペースの含まれないフォルダーパスを指定 してください。
		なお このマニュアルは、演習データの展開先を下記フォルダーとして作成 しています。
		C:/intelFPGA_prj/

演習 1. FPGA 設計フローを体験

インテル Quartus Prime 開発ソフトウェア (以下、Quartus Prime) を使用し、MAX 10 FPGA の開発フローを体験 しましょう。

● 演習で使用する回路の概要

この演習では、VHDL もしくは Verilog HDL で すでに記述されたデザイン (デジタル論理回路) を使用します。 デザインのブロックイメージは、以下のとおりです。



● 論理回路の動作

ボード上の5 個の LED が1 秒間隔で点滅します。

● 主な作業内容

- プロジェクト作成
- デザイン確認
- ピン配置設定の確認
- コンパイル実行
- ・ FPGA ヘダウンロード

1. プロジェクト作成

\Lambda ALTIMA

以下の手順に沿って Quartus Prime のプロジェクトを作成します。

____1. Quartus Prime を起動します。

Windows OS の場合は、[スタート] > Intel FPGA <version_build> Standard Edition > Quartus (Quartus Prime <version>) をクリック、もしくはデスクトップに生成した Quartus Prime のショートカット・アイコンをダブルクリックしてください。

Linux OS の場合は、ターミナル上で quartus コマンドを実行し、起動させます。

____2. Quartus Prime のメニューから File ➤ New Project Wizard を選択してウィンドウを開き、Introduction ページ で [Next] ボタンをクリックします。

S c	Quartus F	rime Sta	ndard Ed	ition		
File	Edit	View	Project	Assignments	s Processing Tools Window Help	
	New			Ctrl+N	New Project Wizard	×
	Open Close			Ctrl+O Ctrl+F4	Introduction	
A	New Pr	oject Wi	zard		The New Project Wizard helps you create a new project and preliminary project settings, including the following: Project name and directory	
×	Open F Save Pr	roject oject		Ctrl+J	 Name of the top-level design entity Project files and libraries Target device family and device EDA tool settings 	
					(Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.	
					Don't show me this introduction again	
					< <u>B</u> ack <u>Next></u> Einish Cancel <u>H</u> el	p

【図 1-1-1】 New Project Wizard / Introduction ページ

_ 3. New Project Wizard: Directory、Name、Top-Level Entity ページ において、作業フォルダー、プロジェクト名、 最上位階層のエンティティ名を入力します。【図 1-1-2】を参考に、以下の手順に従ってください。

① 上段に、作業フォルダーを指定します。

右端のボタン (ブラウズボタン) をクリックし、演習1用の作業フォルダーを指定します。

/cm_lab_dev/lab1 フォルダーには、VHDL 用 (vhdl フォルダー) と Verilog HDL 用 (Verilog フォルダー) の 作業フォルダーを用意しています。いずれかを選択し、[フォルダーの選択] ボタンをクリックします。



② <u>中段</u>に、プロジェクト名を入力します。この演習では LED_Flash と入力します。

- ③ <u>下段</u>に、最上位階層のエンティティ名を入力します。この演習では LED_Flash と入力ます。
- ④ [Next] ボタンをクリックします。

New Project Wizard	×
Directory, Name, Top-Level Entity	
What is the <u>w</u> orking directory for this project?	
C:/intelFPGA_prj/cm_lab_dev/lab1/vhdl	
What is the name of this project?	
LED_Flash	
What is the name of the <u>t</u> op-level design entity for this project? This name is case sensitive a natch the entity name in the design file.	nd must exactly
LED Flash	

【図 1-1-2】 New Project Wizard: Directory、Name、Top-Level Entity ページ (VHDL フォルダーを選択した場合)

4. New Project Wizard: Project Type ページ において、Empty project を選択後、[Next] ボタンをクリックします。



___5. New Project Wizard: Add Files ページ において、回路を構成する既存のデザインファイルを選択します。

File name 欄右端のボタン (ブラウズボタン) をクリックし、【表 1-1-1】を参考に、作業フォルダー内にあるデ ザインファイルを選択して [**開**() ボタンをクリックします。



【表 1-1-1】 プロジェクトに登録するデザインファイル

選択した言語	ファイル名
VHDL	LED_Flash.vhd
Verilog HDL	LED_Flash.v

ウィンドウ内の一覧に、LED_Flash.vhd もしくは LED_Flash.v が追加されたことを確認し、[Next] ボタンをクリックします。

- __6. New Project Wizard: Family, Device & Board Settings ページ において、開発ボードに搭載されているターゲ ット・デバイスの型番を選択します。
 - ターゲット・デバイスの型番: 10M50DAF484C6GES
 - 【図 1-1-4】を参考に、以下の手順に従ってください。
 - ① Family 欄において、MAX 10 FPGA (DA/DF/DC/SA/SF/SC) を選択します。
 - ② Show in 'Available devi ces' list セクションで、型番に応じた条件を指定します。
 - ③ Available devices 欄から 10M50DAF484C6GES をハイライト選択します。

🕞 New Project Wizard						2
Family, Device & Board Settin	igs					
	•					
Device Board						
Select the family and device you want to	target for compilat	tion. wices com	mand on the Tools m	e011		
To determine the service of the Oscietaria	Dimension (here)	wices com		enu.		
To determine the version of the Quartus I	Prime sottware in t	which you	evice is supp	orted, refer to the <u>De</u>	evice Support List webpage	e. —
Device family			Show in 'Available (levices' list		
Eamily: MAX 10 (DA/DF/DC/SA/SC)		-	Pac <u>k</u> age:	FBGA	•	
Dev <u>i</u> ce: All		•	Pin <u>c</u> ount:	484	-	
Target device			Core sp <u>e</u> ed grade:	6	-	-
Auto device selected by the Fitter			Name filter:			
Specific device selected in 'Available	e devices' list		Show advanced	devices		
O <u>O</u> ther: n/a						
Available devices:						
Name Core Vol	tago I.Ec	Total		Memory Ritz	Emboddod multiplio	
10M50DAF484C6GES 1.2V	49760	360	360	1677312	288	ה
						יי
4					>	
٢		- 1	ダイアログボック	ウスの右隅をドラ	ッグ & ドロップす	,
٢			ダイアログボック ると、ウィンドウ	ウスの右隅をドラ のサイズを変更 ⁻	ッグ & ドロップす できます。	•

【図 1-1-4】 Family, Device & Board Settings ページ



- ④ [Next] ボタンをクリックします。
- ____7. New Project Wizard: EDA Tool Settings ページ は、この演習では使用しないため [Next] ボタンで次へ進み ます。
- ____8. New Project Wizard: Summary ページ で、設定した内容を再確認します。
 - Project directory (作業フォルダーのパス)
 - Project name (プロジェクト名)
 - Top-level design entity (最上位階層のエンティティ名)
 - ・ Device (Device assignments 内) (デバイス型番)

Summary When you click Finish, the project will be o	created with the following settings:
Project directory:	C:/intelFPGA_prj/cm_lab_dev/lab1/vhdl
Project name:	LED_Flash
Top-level design entity:	LED_Flash
Number of files added:	1
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	MAX 10 (DA/DF/DC/SA/SC)
Device:	10M50DAF484C6GES
Board:	n/a
EDA tools:	
Design entry/synthesis:	<none> (<none>)</none></none>
Simulation:	<none> (<none>)</none></none>
Timing analysis:	0
Operating conditions:	
Core voltage:	1.2V

【図 1-1-5】 Summary ページ (例: VHDL)

問題が無ければ [Finish] ボタンをクリックします。



以上で、プロジェクト作成が完了しました。

Eile Edit View Project Assignments Processing Tools Window Help 	S Quartus Prime Standard Edition - C:/intelFPGA_p	rj/cm_lab_dev/lab1/vhdl/LED_Flash - LED_Flash
□ 下 □ ケ □ つ C LED_Flash プロジェクトがセットされると、"タイトルバーに作業フォルダーのパス - プロジェクト名 (リビジョン名)"が表示されます。 ▲ ▲ ●	<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>P</u> roject <u>A</u> ssignments P <u>r</u> oc	essing <u>T</u> ools <u>W</u> indow <u>H</u> elp
Project Navigator	□ 〒 日 ← □ 10 で LED_Flash	プロジェクトがセットされると、"タイトルバーに作業フォルダー のパス - プロジェクト名 (リビジョン名)"が表示されます。
EntityInstance	Project Navigator	2. ₽ ₽ ×
	Entity:Instance	"指定した FPGA の型番"および "最上位階層のエンティティ名" が表示されます。

【図 1-1-6】 プロジェクトをセットした後の Quartus Prime 画面 (例: VHDL)

2. <u>デザイン確認</u>

プロジェクトに追加した LED_Flash.vhd もしくは LED_Flash.v ファイルのコードを確認します。ファイルを開いてみましょう。

このファイルは、本紙6ページ "演習で使用する回路の概要"を VHDL/Verilog HDL で記述したものです。

今回 デザインを編集する必要はありません。

___1. *Project Navigator ウィンドウ* (Quartus Prime 画面左上) の LED_Flash をダブルクリックしてください。最上位 階層のデザインファイルが開きます。



【図 1-2-1】 デザインファイルを開く (例: VHDL)

Quartus Prime のメニューから File ➤ Open を選択し、ファイルを指定して開く方法もあります。

① Note:



コードを確認したら、ワーキングシートのタブ右端にある [X] ボタンをクリックし、ファイルを閉じます。



____ 2. デザインを初期チェックするため、Quartus Prime のメニューから *Processing* ➤ Start ➤ Start Analysis & Elaboration を実行します。

今回は完成したデザインを使用しているため、エラーが発生することなく Analysis & Elaboration のステップ が終了するはずです。

nents	Proc	cessing Tools Window Help		_
LED_	STOP	Stop Processing	Ctrl+Shift+C	🗢 😂 🔺 🐎 🙀
<u> </u>		Start Compilation	Ctrl+L	
y	$\mathbf{\Sigma}$	Analyze Current File		
nce		Start	•	Start Hierarchy Elaboration
		Update Memory Initialization File		✓ Start Analysis & Elaboration
	•	Compilation Report	Ctrl+R	🖌 Start Analysis & Synthesis Ctrl+K

【図 1-2-3】 Start Analysis & Elaboration

以上で、デザインの確認は終了です。

3. <u>ピン配置設定の確認</u>

デザインが完成したら、コンパイルを実行します。

実際の設計では、Quartus Prime でコンパイルを実行する前にいくつかの設定を行う必要がありますが、今回の 演習ではあらかじめ設定された環境が提供されています。

ここでは、ユーザー設定のうち ピン配置設定がどのようになっているのかを確認してみましょう。

_1. Quartus Prime のメニューから *Assignments ➤ Pin Planner* を選択し、Pin Planer を起動します。

File Edit View Project	Ass	ignments F	Processing	Tools	Window	Help																					
🗋 🔂 🖬 🗲 🖸 💼	2	Device					4																				
🚸 🔁 🖬 😵 🗛 🕖	∎ -′	Settings			Ctrl+Shi	ft+E																					
Project Navigator	4	Assignment	t Editor		Ctrl+Shi	ft+A																					
	4	Pin Planner			Ctrl+Shi	ft+N		【図 1-3-1】		F	Pi	Pin l	Pin Plaı	Pin Plann	Pin Planner	Pin Planner	Pin Planner	Pin Planner									
		D			-								1 11 1 101											i in i ianitei	i in i ianitei	i in i ianitei	



_ 2. Pin Planner の All Pins リスト (ウィンドウ下部) に、設定済みのピンの割り当て (ピンアサイン) が表示され ています。

今回使用している FPGA のピンは、ボード上の水晶発振器からのクロック信号を入力する CLK と、ボード上の LED をドライブするための出力 LED 5本のみです。

各ピンに対して、Location 項にピン番号、I/O Standard 項に I/O 規格が、開発ボードに合わせた内容であら かじめ入力されていることが確認できます。

×	Named: * 🗸 🐇	Edit: 🗡 🗹							
ъ Д	Node Name	Direction	Location	I/O Standard	I/O Bank				
	💾 CLK	Input	PIN_M9	2.5 V	2				
	LED1	Output	PIN_T20	1.5 V	5				
	LED2	Output	PIN_U22	1.5 V	5				
	LED3	Output	PIN_U21	1.5 V	5				
	LED4	Output	PIN_AA21	1.5 V	5				
	LED5	Output	PIN_AA22	1.5 V	5				
s S	< <new node="">></new>								
E I									
R	<								

【図 1-3-2】 All Pins リスト (Pin Planner)

____3. ピンアサイン設定が確認できたら、Pin Planner 上のメニューから *File ➤ Close* を選択し、Pin Planner を閉 じます。

4. <u>コンパイル実行</u>

コンパイルを実行し、デザインの論理合成と配置配線などを行います。

____1. Quartus Prime のメニューから Processing ➤ Start Compilation をクリックし、コンパイルを実行します。

Project Assignments	Pro	cessing Tools Window Help					
🗅 🗈 🔿 ୯ 🔤 🛍	STOP	Stop Processing	Ctrl+Shift+C	/ 🧳 🤇	🎽 🐟 🛛 🎟 🚺) ► K 🗘	
🍹 🗲 🖽 😂 💺 😫		Start Compilation	Ctrl+L				
À Hierarchy	Ð	Analyze Current File		Ĩ I	Start Com	pilation の	
Entity:Inst		Start	+		ショートカ	ットボタン	



____2. Messages ウィンドウに、Quartus Prime Full Compilation was successful. のインフォメーション・メッセージが 表示されたら、エラーが発生することなくコンパイルが終了しています

ges)) () ()	3321 2930	.01 Design is fo Quartus Prij 000 Quartus Prij	Illy constrained for hold requirements Timing Analyzer was successful. O en The Full Compilation was successful. O		
Messa	Syste	em (12)	Processing (138)		【図 1-4-2】	Messages ウィンドウ



この演習はエラーが発生することなくコンパイルが完了するはずです。もしエラーが発生した場合は、解消しないと次のステップには進めません。回避するためにエラーの原因を追究し修正する必要があります。

また、エラー・メッセージの他、ワーニング・メッセージが発生する場合もあります。ワーニングは解消しなくて も次のステップへ進めますが、必ず内容を確認し、その内容を回避すべきか無視できるのかをユーザーが判 断してください。

各メッセージ内容の詳細を確認するには、ヘルプ機能の活用が有効です (【図 1-4-3】参照)。 ヘルプには、 メッセージが発生した要因や回避するためのヒントが掲載されています。その情報からエラーやワーニングを 回避してください。

× 5 4	A11 (3 🕰 🔺 🔽	アイコンをクリックして各 右上にメッセージ数を表	·メッセー: 示。	ジをフィルター	ges	i kepurt	
≡	Туре	ID Message	accors has not been s		Hide Previo	us Compilation Messages	5	
		20030 Parallel compt	ilation is enabled and	d wi	Locate Node		•	
		2021 Found 2 design	i units, including 1 e	ent i	Search the	web		
		2021 Found 2 design	units, including i e		Help		F1	
s		.0327 VHDL error at Quartus Prime	counter_bus_mux.v/d(1 Analysis & Elaboratio	t7):	Jeave Feedb ッセージを選	ack 訳して右クリック ➤ Help	o メニュー	c
ssage	<							
Mes	System	Processing (11)						
			Ļ					
	Content List	of Messages					1 Parent topic	
	ID:1032 <numbe< td=""><td>7 VHDL error at </td><td>cation>: can't determi ions</td><td>ine defi</td><td>nition of o</td><td>perator "<name>"</name></td><td>found</td><td></td></numbe<>	7 VHDL error at	cation>: can't determi ions	ine defi	nition of o	perator " <name>"</name>	found	

CAUSE: In a <u>VHDL Design File (.vhd</u>) at the specified location, you used the specified operator. However, Quartus Prime Integrated Synthesis cannot determine the definition of the operator because no definition exists or Quartus Prime Integrated Synthesis found the specified number of different possible definitions for the operator.

ACTION: Make sure the design clearly specifies the definition of the operator.

【図 1-4-3】 Quartus Prime Help の活用

以上で コンパイルは完了です。

5. <u>FPGA ヘダウンロード</u>

コンパイルにより生成されたデザインのデータを、ボード上の FPGA にダウンロードし、動作を確認します。

<u>∧</u>注記:

実際の開発では、コンパイル後に Quartus Prime の Timing Analyzer によるタイミング検証を行い、期待 どおりの動作が実現できるかを検証します。期待するタイミングを満足できることが確認できたら、ボード 上のデバイスへデータを書き込みます。

今回の演習ではタイミング検証を省略していますが、自身の開発時は必ずタイミング検証を行った上でデバイスへの書き込みを行ってください。

① Note:

ALTIMA

MAX 10 開発キットは、FPGA ヘデータをダウンロードする際 ボードに組み込まれたインテル FPGA ダウ ンロード・ケーブル II (旧称 USB Blaster[™] II) 回路を使用します。パソコンとボードの接続は、キットに付 属している mini-USB ケーブルで行います。パソコンとボードが通信するためには、インテル FPGA ダウ ンロード・ケーブル II 用のドライバーをインストールする必要があります。インストール方法は、【表 1-1】 No.5 をご覧ください。

_ 1. キット付属の電源コネクタを接続し、ボードの電源が OFF であることを確認してください。その後、付属の mini-USB ケーブルを用いてボードとパソコンを接続します。



【図 1-5-1】 パソコンととボードを接続

- ___2. 開発ボードへ電源を供給します。
- ____3. Quartus Prime のメニューから *Tools ➤ Programmer* を選択し、Programmer を起動します。
- ___4. Programmer の [Hardware Setup] ボタン 右横の欄が、USB -Blaster II と表示されていることを確認します。

Ardware Setup USB-BlasterII [USB-1]	Mode:	JTAG	•
Enable real-time ISP to allow background programming when available	-		

【図 1-5-2】 Hardware Setup

もし No Hardware 表示の場合には、[Hardware Setup] ボタン をクリックし、Hardware Settings タブ の Currently selected hardware 項のプルダウン・リストから、USB-Blaster II を選択し、[Close] ボタンをクリック してください。



5. Programmer の *Mode* プルダウン・リストから *JTAG* を選択します。

🚖 Hardware Setup	USB-BlasterII [USB-1]	Mode:	JTAG	•
Enable real-time ISP	to allow background programming when available			
1				

- 【図 1-5-3】Mode 選択
- ___6. Programmer の Files 欄に、ダウンロードするファイル LED_Flash.sof が選択されていることを確認します。

<none> となっている場合には、[Add File] ボタン をクリックし、起動したファイルブラウザから LED_Flash.sof を選択します。(作業フォルダー内 output_files フォルダー に生成されています。)

▶ [™] Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Bl C
Stop	output_files/LED_Flash.sof	10M50DAF484C6GES	00274514	00274514	\checkmark		
-100							

【図 1-5-4】 ダウンロードする sof ファイルを指定

____7. プログラミング・オプションを選択します。

今回の演習では、FPGA の SRAM 領域にデータを転送するため、Program/Configure にチェックを入れてください。

Mu Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Bl
Stop	output_files/LED_Flash.sof	10M50DAF484C6GES	00274514	00274514			

【図 1-5-5】 プログラミング・オプションを選択

____8. Programmer の *[Start]* ボタン をクリックし、データの書き込みを開始します。

Progress バーが 100% になったら書き込み完了です。Messages ウィンドウには Successful のインフォメーションが表示されます

📥 Hardware Setup.	USB-Blasterii [USB-1]		Mode:	JTAG		• Pr	ogress:	100% (Successful)	
Enable real-time I	SP to allow background programm	ning when available								
► [₩] Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase
Stop	output_files/LED_Flash.sof	10M50DAF484C6GES	00274514	00274514						





開発ボード上の LED (5 個) が 1 秒間隔で点滅していれば、正しく回路データがダウンロードされています。

【図 1-5-7】開発ボード上の LED の位置

__ 9. FPGA の動作を確認後、ボードの電源を OFF にします。その後、再び接続し、ボードの電源を投入してく ださい。先ほど書き込んだ MAX 10 の動作は消去され、データを書き込む前の状態に戻っていることが確 認できます。

① Note:

MAX 10 内部の SRAM 領域にデータを書き込んだ場合は 電源 OFF 時にデータを保持することができま せん。MAX 10 内部の不揮発性メモリ領域 (CRAM) にデータを書き込む際は、pof ファイルを使用します。 (この演習では実施しません。)

以上で、演習1の作業は終了です。

※ 時間に余裕がある場合には、以下のオプション演習を実施してみましょう。

実施しない場合には、Quartus Prime のメニューから *File ➤ Close Project* を選択し、演習 1 のプロジェクト を終了してください。その後、開発ボードの電源を OFF にしてください。

<u><オプション演習></u>

演習1 で作成したプロジェクトのデザインを編集し、LED の点滅間隔を半分にしてみましょう。 作業内容は以下のとおりです。

- ・デザインを編集
- ・コンパイル実行
- ・ FPGA ヘダウンロード



もし Quartus Prime の GUI を閉じてしまった場合は、再度 Quartus Prime を起動し、Quartus Prime のメニュー から *File ➤ Open Project* を選択し、作業フォルダーから *LED_Flash.qpf* を開いて 演習 1 で作成したプロジェク ト環境を開きます。

- 1. *Project Navigator ウィンドウ* (Quartus Prime 画面左上) において *LED_Flash* をダブルクリックし、デザイン ファイルを開きます。
- ____2. 下記のようにコードを編集してください。
 - ① VHDL の場合: 32 行目
 - if (div_cntr2 = X"2FA") then \rightarrow if (div_cntr2 = X"17E") then

② Verilog HDL の場合: 26 行目

if (div_cntr2 == 762) \rightarrow if (div_cntr2 == 382)

- ____3. Quartus Prime のメニューから *File ➤ Save* で編集したファイルを保存します。
 - __4. Quartus Prime のメニューから Processing ➤ Start Compilation を選択し、コンパイルを実行します。
 - ※ Programmer ウィンドウを開いたままコンパイルを実行すると、下記のメッセージ・ウィンドウが表示される場合があります。 ここでは No を選択してメッセージ・ウィンドウを閉じます。

🕥 Qua	rtus Prime			×
	'LED_Flash.cdf' has been r	modified. Do you	I want to save y	our changes?
		<u>Y</u> es	No	Cancel
		.	•	

[図	1-5-8】	メッセージ	
----	--------	-------	--

___5. 再コンパイルにより生成された LED_Flash.sof を MAX 10 FPGA にダウンロードします。

操作手順は、<u>14ページの 5.FPGA ヘダウンロード</u> を参考にしてください。

LED の点滅間隔は短くなりましたか?

すべての作業が終了したら、Quartus Prime のメニューから *File ➤ Close Project* を選択し、演習 1 のプロジェク トを終了してください。その後、開発ボードの電源を OFF にしてください。

以上で、演習1の作業はすべて終了です。

演習 2. FPGA を使用したカスタムマイコンの作成

MAX 10 FPGA に Nios® II プロセッサー、JTAG UART、PIO、On-Chip メモリ を実装し、Nios II にサンプルの C 言語ソースコードを実行させ LED を制御させる簡単なシステムを構築しましょう。

● 演習で構築するシステムの概要

🛆 ALTIMA



JTAG UART というインテル FPGA 特有のシリアル・ポートを利用し、ホストとのシリアル通信をインテル FPGA ダ ウンロード・ケーブル II 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 ムで printf() 関数を使用したコンソール出力を JTAG 経由で行うことが可能となります。JTAG デバッグ・モジュー ル経由でのデバッグについても、インテル FPGA ダウンロード・ケーブル II を介して行われます。

今回の演習では、PIO (Parallel I/O) 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 プロセッサー用内部バスである プラットフォーム・デザイナー・インタコ ネクトで接続されます。

上図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。

- 主な作業内容
 - プラットフォーム・デザイナーでシステム・モジュールを作成
 - ハードウェア・デザイン作成
 - ・ ハードウェア・デザインを FPGA ヘダウンロード
 - ソフトウェアを実行

1. プロジェクトをセット

ALTIMA

以下の手順に沿って Quartus Prime でシステム・モジュールを作成します。

この演習では、すでにベースとなる Quartus Prime プロジェクト (nios2_basic_lab) が作成されています。既存の プロジェクトにデザインを追加することで、ハードウェア用のプロジェクトを完成させます。Nios II プロセッサーの 開発を行う際は、必ずハードウェア用のプロジェクトを作成する必要があります。

演習1 終了後に Quartus Prime の GUI を閉じてしまった場合は、Quartus Prime を起動してください。

____1. Quartus Prime のメニューから File ➤ Open Project を選択し、以下の既存プロジェクト・ファイルを選択して [**開**() ボタンをクリックします。

C:/intelFPGA_prj/cm_lab_dev/lab2/nios2_basic_lab.qpf

Open Project		×
\leftarrow \rightarrow \checkmark \uparrow \bigcirc \checkmark \land intelFPGA_prj \rightarrow cm_lab_dev \rightarrow lab2 \rightarrow \checkmark \circlearrowright	lab2の検索	Q
整理 ▼ 新しいフォルダー		?
<pre>software software lab1 lab2 lab3 v</pre>		
ファイル名(<u>N</u>): nios2_basic_lab.qpf ~	Quartus Prime Project File (*. 開く(<u>Q</u>) キャンセ	qр ∼ ル

【図 2-1-1】 Open Project

- ___2. Quartus Prime の画面左上に位置する Project Navigator ウィンドウの *Hierarchy ビュー* において、以下2 点 を確認してください。
 - nios2_basic_lab が登録されている
 - ターゲットの FPGA 型番が 10M50DAF484C6GES である



【図 2-1-2】 Hierarchy ビュー (Project Navigator ウィンドウ)

2. プラットフォーム・デザイナーでシステム・モジュールを作成

Quartus Prime に付属するシステム統合ツール プラットフォーム・デザイナー (Platform Designer) を使用して、 Nios II プロセッサーを含むシステム・モジュールを作成します。

2-1. プラットフォーム・デザイナーを起動

🛆 ALTIMA

_1. Quartus Prime のメニューから Tools ➤ Platform Designer を選択します。

____2. プラットフォーム・デザイナーが起動し、ファイルの選択画面が表示されます。

この演習には、途中まで作成したシステム・モジュールがあらかじめ用意されています。作業フォルダーから nios2_system.qsys を選択し、[開く] ボタンをクリックします。



【図 2-2-1】 Platform Designer で .qsys を選択

作成途中のシステム・モジュールが復元されます。





2-2. Nios II プロセッサーを追加

システム・モジュールに Nios II プロセッサーを追加します。

1. プラットフォーム・デザイナーの IP Catalog ウィンドウ内 Library から、Processors and Peripherals ➤ Embedded Processors ➤ Nios II Processor を選択し、[Add] ボタンをクリックします。

Nios II プロセッサーのパラメーター設定画面が表示されます。*Main* タブにおいて Nios II/e を選択し、 [Finish] ボタンをクリックします。

Platform Designer - nios2_system.qsys* (C:	¥intelFPGA_prj¥cm_lab_dev¥lab2¥	nios2_syst		
<u>File Edit System Generate View Tools H</u> elp)			
📫 IP Catalog 🛛 🗕 🗗 🗖	System Contents 🛛	Address M		
🔍 🗙 🕅	Nios II Processor - nios2_gen2_0			×
DSP Interface Protocols Dower Memory Interfaces and Controllers Processors and Peripherals	Nios II Processor Megetere Block Diagram Show signals	Main rectors	Caches and Memory Interfaces Arithmetic In	Documentation
	nios2	Select an I Nios II Core	Nios II/e Nios II/e	
I Hard Processor Components I Hard Processor Systems	reset		Nios II/e	Nios II/f
	debug_mem_slaveavalen nios_c	Features	Nesource-optimized 32-bit NISC JTAG Debug ECC RAM Protection	Performance-optimized 32-bit KISC JTAG Debug Hardware Multiply/Divide Instruction/Data Caches Tightly-Coupled Masters EGC RAM Protection External Interrupt Controller
New Edit	この時点で ここでは先	? Messages に進んでく	s ウィンドウにエラーが表示 ださい。	*されますが、 、 、
	S Error: nios2_gen2_0: Reset slave is not Error: nios2_gen2_0: Exception slave is	specified. Please s not specified. Plea:	elect the reset slave se select the exception slave	
				Cancel

【図 2-2-3】 Nios II プロセッサーを追加

___2. プラットフォーム・デザイナーの System Contents タブにおいて、Nios II Processor の nios2_gen2_0 (Name 項目) をマウスで右クリック ➤ Rename を選択し、nios2_cpu に変更します。



【図 2-2-4】 nios2_cpu ヘリネーム

2-3. Nios II プロセッサーのリセットベクタと例外ベクタを設定

____ 1. Nios II プロセッサーのリセットベクタおよび例外ベクタに FPGA 内部のオンチップ・メモリを指定するため、 Nios II プロセッサーとオンチップ・メモリを接続します。

プラットフォーム・デザイナーの *System Contents* タブにおいて、【表 2-2-1】および【図 2-2-5】を参考に、 Nios II プロセッサーのデータ・マスターとインストラクション・マスターを各モジュールに接続します。

Name	ポート名	接続先
nios2_cpu	data_master	jtag_uart.avalon_jtag_slave
		onchip_memory.s1
		led_pio.s1
	instraction_master	onchip_memory.s1

【表 2-2-1】 Nios II とオンチップ・メモリの接続



_ 2. nios2_cpu をマウスで右クリック ➤ Edit を選択し、再び Nios II プロセッサーの GUI 画面を起動します。 Vectors タブに切り替え、Reset vector memory および Exception vector memory を下図のとおり変更し、 [Finish] ボタンをクリックします。

Main Vectors Caches and Memory Interfa	aces Arithmetic Instructions MMU and MPU Settings JTAG
Reset Vector	
Reset vector memory.	onchip_memorys1 v
Reset vector offset:	0×0000000
Reset vector:	0×0000000
Exception Vector	
Exception vector memory:	onchip_memorys1 v
Exception vector offset:	0×0000020
Exception vector:	0×0000020

【図 2-2-6】 Nios II プロセッサーの Vectors タブ



2-4. クロックおよびリセットを接続

1. プラットフォーム・デザイナーの System Contents タブにおいて nios2_cpu を選択し、見やすい位置に移動 するため、ツールバーの [Move Up] ボタンを3回クリックして、clk モジュールの下へ配置します。

	System Contents 🛛 Address N	1ap 🛛 Interconnect Req	juirements 🛛	
	🛎 🔺 🗮 System:nios2_syste	m Path: nios2_cpu		
+	Use Connections	Name	Description	Export
		🗆 clk	Clock Source	
×	□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	clk_in	Clock Input	clk
	Move Up ボタン P	clk_in_reset	Reset Input	reset
1°_		clk	Clock Output	Double-click to export
C		<u>clk_reset</u>	Reset Output	Double-click to export
		🗖 🖳 nios2_cpu	Nios II Processor	
-		clk	Clock Input	Double-click to export
≖		reset	Reset Input	Double-click to export
		data_master	Avalon Memory Mapped Master	Double-click to export
		instruction_master	Avalon Memory Mapped Master	Double-click to export
		irq	Interrupt Receiver	Double-click to export
		debug_reset_request	Reset Output	Double-click to export
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export
		custom_instruction_m	Custom Instruction Master	Double-click to expor
		⊟ jtag_uart	JIAG UARI Intel FPGA IP	
		l clk	lf Jock Input	Double-click to export

【図 2-2-7】 Nios II プロセッサーの配置変更

____ 2. Connections 項目において、【表 2-2-2】 および 【図 2-2-8】 を参考に各モジュールのクロックとリセット接続 します。

Name	ポート名	接続先
clk	clk	nios2_cpu.clk
	clk_reset	nios2_cpu.reset
nios2_cpu	debug_reset_request	jtag_uart.reset
		led_pio.reset
		nios2_cpu.reset
		onchip_memory.reset1

【表 2-2-1】 クロックとリセットの接続

___3. jtag_uart から Nios II プロセッサー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。同時に Connections 項目の jtag_uart の irq が nios2_cpu の irq に接続されることを確認してください。(【図 2-2-8】参照)



13	System	Contents 🛛 Address N	Nap 🖾 Interconnect Requ	uirements 🛛					
		🛛 📗 System: nios2_syste	m Path:jtag_uart.irq						
+	Use	Connections	Name	Description	Export	Clock	Base	End	IRQ T
1			🗆 clk	Clock Source					
×		D-	clk_in	Clock Input	olk	exported			
		→ →	clk_in_reset	Reset Input	reset				
			clk	Clock Output	Double-click	clk			
			clk_reset	Reset Output	Double-click				
	\checkmark		曰 🛄 nios2_cpu	Nios II Processor					
•	:	$ \bullet \longrightarrow$	clk	Clock Input	Double-click	clk			
T			reset	Reset Input	Double-click	[clk]			
			data_master	Avalon Memory Mapped Master	Double-click	[clk]			
			instruction_master	Avalon Memory Mapped Master	Double-click	[clk]			
		│	irq	Interrupt Receiver	Double-click	[clk]	IRQ 0	IRQ 31	
			debug_reset_request	Reset Output	Double-click	[clk]			
		$ \phi \phi \downarrow \longrightarrow$	debug_mem_slave	Avalon Memory Mapped Slave	Double-click	[clk]		0×0ff	
		×	custom_instruction_m	Custom Instruction Master	Double-click				
			🗆 jtag_uart	JTAG UART Intel FPGA IP					
		$ \bullet + \bullet \bullet \bullet \to$	clk	Clock Input	Double-click	clk			
			reset	Reset Input	Double-click	[clk]			
			avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click	[clk]	─ 0x0000	0×000'	
			irq	Interrupt Sender	Double-click	[clk]			
	\checkmark		onchip_memory	On-Chip Memory (RAM or RO					<u> </u>
		$ \bullet + + + \rightarrow$	clk1	Clock Input	Double-click	clk			
		$ \phi \phi \phi \phi \phi$	sl	Avalon Memory Mapped Slave	Double-click	[clk1]	■ 0x0000	0×7fff	
		$ \bullet \bullet \bullet \to$	reset1	Reset Input	Double-click	[clk1]			
			🗆 led_pio	PIO (Parallel I/O) Intel FPGA					
			clk	Clock Input	Double-click	clk			
			reset	Reset Input	Double-click	[clk]			
		$\bullet \bullet \longrightarrow$	s1	Avalon Memory Mapped Slave	Double-click	[clk]	= 0x0000	0x000f	
		<u>^</u> ~	external_connection	Conduit	led_pio				

【図 2-2-8】 クロックとリセットの接続 / 割り込みの接続

2-5. ベースアドレスを設定

__1. プラットフォーム・デザイナーのメニューから System ➤ Assign Base Address を選択し、ベースアドレスを自動 で適切な値に設定します。

File Edit	System Generate View Tools Help						
r IP (Upgrade IP Cores	 - 🗗 🗆	茸 System C	iontents 🖾	Address Ma	ap 🖾	Intercon
	Assign Base Addresses	× 🕸		🕷 System	:nios2_system	n Path	:jtag_uart
Proie	, solgi interrupt numbers		T Use (Connections		Name	

【図 2-2-9】 Assign Base Address

____2. プラットフォーム・デザイナーの Address Map タブに切り替え、ベースアドレスの一覧を確認します。

表示される順序やアドレスが【図 2-2-10】と異なっていても、特に問題ありません。

System Contents 🛛 💷 Addres	s Map 🛛	Interconnect Requirements 🛛 🖾		
System:nios2_system 🏾 🖓 ath:jtag_u	arttirg			
	nios2_cpu	data_master		nios2_cpu.instruction_master
jtag_uart.avalon_jtag_slave 0x0001_101		0 - 0×0001_1017	1	
led_pio.s1	0x0001_100	10 - 0×0001_100f		
onchip_memorys1	0×0000_800	10 - 0×0000_ffff	0	0x0000_8000 - 0x0000_ffff
nios2_cpu.debug_mem_slave	0x0001_080	10 - 0x0001_0fff	0	0x0001_0800 - 0x0001_0fff

【図 2-2-10】 Address Map タブ



___3. プラットフォーム・デザイナーのメニューから *File ➤ Save* を選択し、nios2_system.qsys ファイルを保存します。 保存が完了したら、Save System ウィンドウを *[Close]* ボタンで閉じます。

2-6. システム・モジュールを生成

____1. プラットフォーム・デザイナーのメニューから Generate ➤ Generate HDL を選択すると、Generation ウィンド ウが表示されます。右下にある [Generate] ボタンをクリックし、システムを生成します。

Generation	X
▼ Synthesis	ここの設定は、この演習ではデフォルトのまま Generate を実行します。
Synthesis files are used Create HDL design files Create timing and re Create block symbol	to compile the system in a Quartus project. for synthesis: Veril v source estimates for third-party EDA synthesis tools. file (bsf)
Simulation The simulation model con Simulation scripts for thi Follow the guidance in th and <i>ip-make-simscript</i> c Create simulation model:	ntains generated HDL files for the simulator, and may include simulation-only features. s component will be generated in a vendor-specific sub-directory in the specified output directory. he generated simulation scripts about how to structure your design's simulation scripts and how to use the <i>ip-setup-simulation</i> ommand-line utilities to compile all of the files needed for simulating all of the IP in your design.
Output Directory	
Path:	C:/intelFPGA_prj/cm_lab_dev/lab2/nios2_system



- ____2. Generate ウィンドウのメッセージ・ボックスに "Generate: completed successfully." と表示されれば、システ ム・モジュールが正常に生成されました。Generate ウィンドウを **[Close]** ボタンをクリックして閉じます。
- ____3. プラットフォーム・デザイナーのメニューから File ➤ Exit を選択し、プラットフォーム・デザイナーを閉じます。 以下のウィンドウが表示されたら [OK] ボタンをクリックして閉じます。

🕥 Qua	rtus Prime X
1	You have created an IP Variation in the file C:/intelFPGA_prj/cm_lab_dev/lab2/nios2_system.qsys.
	To add this IP to your Quartus project, you must manually add the .qip and .sip files after generating the IP core.
	The .qip will be located in <generation_directory>/synthesis/nios2_system.qip</generation_directory>
	The .sip will be located in <generation_directory>/simulation/nios2_system.sip</generation_directory>
	ок



3. ハードウェア・デザインの作成

プラットフォーム・デザイナーで生成したシステム・モジュールを最上位階層に配置して、ハードウェア・デザイン を作成します。

3-1. システム・モジュールをインスタンスする

____ 1. Quartus Prime のメニューから File ➤ Open を選択し、あらかじめ用意された最上位階層のデザインファイ ルを開きます。

フォーマットは、以下 2 種類の言語が用意されていますので、自分の設計手法である言語ファイルいずれか を選択し、開いてください。

言語	選択するファイル名
VHDL	nios2_basic_lab.vhd
Verilog HDL	nios2_basic_lab.v

【表 2-3-1】 クロックとリセットの接続

__2. nios2_basic_lab.vhd (.v) は、あらかじめ nios2_system モジュールをインスタンスする記述が途中まで行われ ています。ポートを接続する箇所が未完成のため、【表 2-3-2】 従い 点線枠部分を編集してください。

【表	2-3-2】	編集箇所と編集内容
----	--------	-----------

VHDL Ø	D場合 (28 行目)
編	u0 : nios2_system
集	<pre>PORT MAP (clk_clk => CONNECTED_TO_clk_clk, clk.clk</pre>
前	<pre>led_pio_export => CONNECTED_TO_led_pio_export, led_pio.export</pre>
	reset_reset_n => CONNECTED_TO_reset_reset_n reset.reset_n
);
編	u0 : nios2_system
集	<pre>PORT MAP (clk_clk => osc_clk,</pre>
後	<pre>led_pio_export => tmp,</pre>
	reset_reset_n => reset_n
);
Verilog I	HDL の場合 (15 行目)
編	nios2_system u0 (
集	.clk_clk (<connected -="" clk_clk="" to="">), // clk.clk</connected>
前	.led_pio_export (<connected -="" led_pio_export="" to="">), // led_pio.export</connected>
	.reset_reset_n (<connected -="" reset_reset_n="" to="">) // reset.reset_n</connected>
);
編	nios2_system u0 (
集	.clk_clk (osc_clk),
後	.led_pio_export (tmp),
	.reset_reset_n (reset_n)
);



- _3. 編集終了後、Quartus Prime のメニューから *File ➤ Save* を選択し、デザインファイルを保存します。
- ____ 4. nios2_basic_lab.vhd (.v) が開いている状態で、Quartus Prime のメニューから Project ➤ Set as Top -Level Entity を実行します。この操作で、nios2_basic_lab.vhd (.v) はこのプロジェクトの最上位デザインとして定義され、プロジェクトに登録されます。



【図 2-3-1】 Set as Top-Level Entity

_ 5. *Project Navigator ウィンドウ* (Quartus Prime 画面左上) のプルダウンを *Files ビュー* に切り替え、下記ファ イルが登録されていることを確認してください。



【図 2-3-2】 Project Navigator > Files (VHDL)

【図 2-3-3】 Project Navigator > Files (Verilog HDL)

確認後は、Project Navigator ウィンドウのビューを Hierarchy に戻してください。

3-2. FPGA の外部端子をレイアウト (ピンアサイン) する

作成したデザイン上の信号を FPGA を外部端子へアサインするため、ピン番号を指定します。

なお、この演習ではすでにピンアサインが実施されています。どのように設定されているかを確認しましょう。

____1. Quartus Prime のメニューから Assignments ➤ Pin Planer を起動し、All Pins リスト の枠内が【図 2-3-4】 のとおり設定されていることを確認してください。

₽ ₽	Node Name	Direction	Location	I/O Standard	I/O Bank
	💠 leds[0]	Unknown	PIN_T20	1.5 V	5
	🔷 leds[1]	Unknown	PIN_U22	1.5 V	5
	📀 leds[2]	Unknown	PIN_U21	1.5 V	5
	🔷 leds[3]	Unknown	PIN_AA21	1.5 V	5
	📀 leds[4]	Unknown	PIN_AA22	1.5 V	5
	🔷 reset_n	Unknown	PIN_D9	3.3-V LVTTL	8
	🔷 clk	Unknown	PIN_M9	2.5 V	2
2	< <new node="">></new>				
II Pi	1				
4	s				

【図 2-3-4】 All Pins リスト (Pin Planner)

___2. 問題がなければ、Pin Planner のメニューから File ➤ Close により Pin Planner を閉じます。

3-3. ハードウェア・デザインをコンパイル

完成したハードウェア・デザインをコンパイルします。

___1. Quartus Prime のメニューから Processing ➤ Start Compilation をクリックし、コンパイルを開始します。

Project Assignments	Pro	cessing Tools Window Help					
🗅 🗈 つ 🤉 🔣 🕯	STOP	Stop Processing	Ctrl+Shift+C	/ 🧳 🤇	🌣 🚸 🎟 🚺	• × ∢	
🍹 🗲 🖽 🚫 💺 😫		Start Compilation	Ctrl+L				
À Hierarchy		Analyze Current File			Start Com	pilation ${\cal O}$	
Entity:Inst		Start	*		ショートカ	ットボタン	

【図 2-3-5】 コンパイル実行

コンパイルには多少の時間を要します。



2. Messages ウィンドウに、Quartus Prime Full Compilation was successful. のインフォメーション・メッセージが表 示されたら、エラーが発生することなくコンパイルが終了しています



【図 2-3-6】 Messages ウィンドウ

この演習はエラーが発生することなくコンパイルが完了するはずです。もしエラーが発生した場合は、解消しないと次のステップには進めません。回避するためにエラーの原因を追究し修正する必要があります。

また、エラー・メッセージの他、ワーニング・メッセージが発生する場合もあります。ワーニングは解消しなくて も次のステップへ進めますが、必ず内容を確認し、その内容を回避すべきか無視できるのかをユーザーが判 断してください。

各メッセージ内容の詳細を確認するには、ヘルプ機能を活用してください。(14ページ【図 1-4-3】参照)

4. ハードウェア・デザインを FPGA ヘダウンロード

作成したハードウェア・デザインのデータを MAX 10 FPGA にダウンロードします。

▲ 注記:

実際の開発では、コンパイル後に Quartus Prime の Timing Analyzer によるタイミング検証を行い、期待 どおりの動作が実現できるかを検証します。期待するタイミングを満足できることが確認できたら、ボード 上のデバイスへデータを書き込みます。

今回の演習ではタイミング検証を省略していますが、自身の開発時は必ずタイミング検証を行った上でデ バイスへの書き込みを行ってください。

① Note:

MAX 10 開発キットは、FPGA ヘデータをダウンロードする際 ボードに組み込まれたインテル FPGA ダウ ンロード・ケーブル II (旧称 USB Blaster[™] II) 回路を使用します。パソコンとボードの接続は、キットに付属 している mini-USB ケーブルで行います。パソコンとボードが通信するためには、インテル FPGA ダウンロ ード・ケーブル II 用のドライバーをインストールする必要があります。インストール方法は、【表 1-1】 No. 5 をご覧ください。

___1. キット付属の電源コネクタを接続し、ボードの電源が OFF であることを確認してください。その後、付属の mini-USB ケーブルを用いてボードとパソコンを接続します。 (【図 2-4-1】参照)

____2. 開発ボードへ電源を供給します。





【図 2-4-1】 パソコンととボードを接続

- __3. Quartus Prime のメニューから *Tools ➤ Programmer* を選択し、Programmer を起動します。
- ____4. Programmer の [Hardware Setup] ボタン 右横の欄が、USB -Blaster II と表示されていることを確認します。

Hardware Setup USB-BlasterII [USB-1]	Mode:	JTAG	•
Enable real-time ISP to allow background programming when available			

もし No Hardware 表示の場合には、[Hardware Setup] ボタン をクリックし、Hardware Settings タブ の Currently selected hardware 項のプルダウン・リストから、USB-Blaster II を選択し、[Close] ボタンをクリック してください。

___5. Programmer の *Mode* プルダウン・リストから JTAG を選択します。

Aardware Setup USB-BlasterII [USB-1]	Mode: JTAG 🔻
Enable real-time ISP to allow background programming when available	

【図 2-4-3】 Mode

____6. Programmer の Files 欄に、ダウンロードするファイル nios2_basic_lab.sof が選択されていることを確認し ます。(【図 2-4-4】参照)

<none> となっている場合には、[Add File] ボタン をクリックし、起動したファイルブラウザから nios2_basic_lab.sof を選択します。(作業フォルダー内 output_files フォルダー に生成されています。)

____7. プログラミング・オプションを選択します。

今回の演習では、FPGA の SRAM 領域にデータを転送するため、Program/Configure にチェックを入れてください。

▶ [™] Start	File	Device	Checksum	Usercode	Program/ Configure	Verify
■ [™] Stop	output_files/nios2_basic_lab.sof	10M50DAF484C6GES	003914B2	003914B2		
Auto Detect			•			

【図 2-4-4】 ダウンロードする sof ファイルを指定

___8. Programmer の **[Start]** ボタン をクリックし、データの書き込みを開始します。

Progress バーが 100% になったら書き込み完了です。Messages ウィンドウには Successful のインフォメー ションが表示されます

🚖 Hardware Setup	. USB-BlasterII [USB-1]	Mode:	JTAG	•	Pro	gress:	100% (5	iuccessful)			
Enable real-time ISP to allow background programming when available											
Mu Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	
Stop	output_files/nios2_basic_lab.sof	10M50DAF484C6GES	003914B2	003914B2	\checkmark						

【図 2-4-5】書き込み完了

ハードウェア・デザインのデータは FPGA に転送されましたが、FPGA 内部の Nios II プロセッサーが実行する ためのプログラムがまだありません。そのため、開発ボードの LED は消灯したままです。

続いて、Nios II プロセッサーが実行するソフトウェアの環境を構築し、実行しましょう。

5. ソフトウェアを実行

Nios II Software Build Tools (以下 Nios II SBT) は、Nios II プロセッサー向けの統合開発環境です。

今回の演習では、Nios II SBT を使用して、C ソースコードのビルド、およびターゲット・プログラムの開発ボードへのダウンロードまでを実施します。使用する C ソースコードは、Nios II プロセッサーから PIO を経由して、開発ボードの LED を点滅させる簡単なプログラムを用います。

5-1. Nios II SBT を起動

___1. Quartus Prime のメニューから Tools ➤ Nios II Software Build Tools for Eclipse をクリックして起動します。

① Note:

もし この起動方法で、GUI が表示されない、あるいは ソフトウェア・プロジェクト作成時にエラー (Failed to execute: wsl./create-this-app -no-make) が発生した場合は、以下のフローにより Nios II SBT を起動し てください。

<Windows 10 の場合>

- Windows 10 の [スタート] ➤ Intel FPGA <version_build> Standard Edition ➤ Nios II Command Shell (Quartus Prime <version>) を右クリック選択し、ショートカット・アイコンのファイルの場所を開きま す。
- 2. Nios II Command Shell (Quartus Prime <version>) を [管理者として実行] し、Nios II Command Shell を起動します。

3. eclipse-nios2.exe コマンドを実行して起動します。

Image: /mnt/c/intelFPGA/20.1

Altera Nios2 Command Shell

Version 20.1, Build 711

root@ :/mnt/c/intelFPGA/20.1# eclipse-nios2.exe

<Linux OS の場合>

1. ターミナル上で下記コマンドにより Nios II Command Shell を起動します。

<Nios II EDS install path>/nios2 command shell.sh

2. eclipse-nios2 をタイプして起動します。

_ 2. Workspace Launcher が起動します。 *[Browse] ボタン* をクリックし Workspace に演習 2 用の Quartus Prime プロジェクト・フォルダー下の *software* を指定して *[OK] ボタン* をクリックします。

Workspace Launcher			Х
Select a w orkspace			
Eclipse stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.			
Workspace: C:¥intelFPGA_prj¥cm_lab_dev¥lab2¥software	~	<u>B</u> rowse	
Use this as the default and do not ask again			
	ОК	Cancel	

【図 2-5-1】 Workspace Launcher

Nios II SBT が起動します。

【図 2-5-2】 Nios II SBT

5-2. ソフトウェア・プロジェクトを作成

____1. Nios SBT 上のメニューから File ➤ New ➤ Nios II Application and BSP from Template をクリックし起動します。

		Nios II ·	- Eclipse							
	File	Edit	Navigate	Search	Project	Run Nic	sll	Window Help		
Ī		New			Alt	+Shift+N >	C+	Nios II Application and BSP from Template		1
		Open	File				C#	Nios II Application		s
		Close				Ctrl+W	C+	Nios II Board Support Package		
		Close	All		Ctrl+	+Shift+W	C.	Nios II Library		
		Save				Ctrl+S		Project		A
		Save A	ls				Ľ	Other	Ctrl+N	a١
							_			1

【図 2-5-3】 Nios II Application and BSP from Template を起動

- ____ 2. プラットフォーム・デザイナーで作成した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 エリアの SOPC Information File name 欄に、ハードウェア・デザインを作成した lab2 フォルダー内の nios2_system.sopcinfo を選択します。
 - ・ Application project エリアの Project name 欄に、soft_test と入力します。
 - ・ Templates 枠内 から Blank Project を選択します。

Nios II Application ar	nd BSP fro	m Tem	nplate —		×
Nios II Software Exe	mples				
Create a new applicatior template	n and boar	rd sup	port package based on a software example		
Target hardware inform	nation				
SOPC Information File	e name: [C:¥int	elFPGA_prj¥cm_lab_dev¥lab2¥nios2_system.sopcinfo		
CPU name:		nios2_	_cpu ~		
Application project					
Project name: soft_	test				
Use default location Project location: Project template	on C:¥intelF	PGA_I	prj¥cm_lab_dev¥lab2¥software¥soft_test]	
Femplates Bank Project Board Diagnostics Count Binary Float2 Functionali Float2 GCC Float2 Performanc Hello Freestanding Hello MicroC/OS- Hello World Hello World Hello World Small Memory Test Memory Test Small	ty e g II	*	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.txt file in the project directory. 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 the BSP projects list. For information about how this software example relates to	^	

【図 2-5-4】 Nios II Application and BSP from Template

設定後、[Finish] ボタン をクリックします。

____ 3. 新しいソフトウェア・プロジェクトが作成され、Nios II SBT の Project Explorer タブ内に *soft_test* と *soft_test_bsp* が追加されます。

	vios II	- Eclipse					
<u>F</u> ile	<u>E</u> dit	<u>N</u> avigate	Se <u>a</u> rch	<u>P</u> roject	<u>R</u> un	Ni <u>o</u> s II	Wi
i 🖻 י	- 11		💣 🔻 🕯	🗳 🔻 숱	• 양	• 🎋	- (
P	roject	Explorer 🔀	3			T	
			E	1 🔁 🕯	, v		
> 6	🤔 sof	t_test					
> 6	🎒 sof	t_test_bsp [r	nios2_syst	em]			
	ľ	図 2-5-5 】、	/フトウェフ	ァ・プロジー	┏クト作	БŮ	

5-3. C ソースコードのインポート

あらかじめ用意された C 言語のソースファイルをソフトウェア・プロジェクトにインポートします。

___1. Windows エクスプローラを開き、この演習2 フォルダー下の *software* フォルダーにある *led_output.c* ファ イルを、Nios II SBT の Project Explorer タブ内にある *soft_test* フォルダーにドラッグ&ドロップします。

【図 2-5-6】C ソースコードをプロジェクトヘコピー

____2. Nios II SBT の Project Explorer タブ内にある soft_test フォルダーを展開し、led_output.c が soft_test フォ ルダーにインポートされていることを確認してください。led_output.c をダブルクリックすると、C ソースコード が表示されます。

Nios II - soft_test/led_output.c - Eclipse						
<u>File Edit Source Refactor Navigate Search Pr</u>	roject <u>R</u> un Ni <u>o</u> s II <u>W</u> indow <u>H</u> elp					
i ➡ ▼ = @	× • O • • • • / ≥ ⊝					
🎦 Project Explorer 💥 📄 🔄 🐨 🗖 🗖	led_output.c ☆					
✓ ²⁰ soft_test	1 #include <stdio.h></stdio.h>					
) jockudes	2 #include <io.h></io.h>					
Ied_output.c	<pre>3 #include <unistd.h></unistd.h></pre>					
create-this-app	4 #include "system.h"					
Makefile	5					
readme.txt	6 #define PERIOD 500000					
> 🖉 soft test bsp [nios2 system]	7					
<pre>/ /</pre>	8⊖ int main(void)					
	0 I					
【図 2-5-7】 led_output.c						

5-4. ソフトウェア・プロジェクトのビルド

1. Nios II SBT の Project Explorer タブ内にある soft_test_bsp フォルダーを右クリックで選択 ➤ Nios II ➤ BSP Editor を選択し、システムの設定を確認します。

📑 🗝 🔒 🕼 📄 🗃 🕶 😂	• 🖻 • 🞯 • 🔅 •	• 🜔 • 💁 • 🙋 🗁 🖋 •	• 🍠 🕸 ▼ 🖗 ▼ 🗢 ← ▼ →	•
Project Explorer 🛛 🖃 🔄		<pre>led_output.c ☆ </pre>	h>	
) includes) includes		2 #include <io.h> 3 #include <io.h> 3 #include <unistd 4 #include "system 5 6 #define PERIOD 5</unistd </io.h></io.h>	h> h" 00000	
> Soft_test_bsp [nios2_	New Go Into		>	
	Open in New Window		m Nios II, start!!\n");	
•	Сору	Ctrl	I+C	
	Delete	Ctn D-1	BASE, 0, 0x55);	
	Move	ton		
	Nios II		Nios II Command Shell	
**	Run C/C++ Code Analy	ysis	Generate BSP	
	Team Compare With		BSP Editor	
	Desperties	Alt. E.	Flash Programmer	
	Properties		+⊐€L	

- 【図 2-5-8】BSP Editor を起動
- ___2. BSP Editor ウィンドウの *Main タブ* において、下記オプションを有効 (√) にします。その後、BSP Editor ウィ ンドウ右下に位置する *[Generate] ボタン* をクリックし、 *[Exit] ボタン* で BSP Editor ウィンドウを閉じます。
 - enable_small_c_library = On
 - enable_reduced_device_drivers = On

🗄 BSP Editor - settings.bsp 🦳 🗆 🗸						Х	
File Edit Tools Help							
Main Software Packages Drivers Linker Script Enable	File	Generation Target BSP Directory					
SOPC Information file: C:\intelFPGA_prj\cm_lab_dev\ab	o2\n	os2_system.sopcinfo					
CPU name: nios2_cpu							
Operating system: Altera HAL		Version: default 🗸					
BSP target directory: C:\intelFPGA_prj\cm_lab_dev\ab	2\sc	ftware\soft_test_bsp					
E-Settings	^	hai				^	
		sys_clk_timer:	none 🗸				
		timestamp timer:					
timestamp_timer		unestamp_unet.	none 🗸				
stdin	stdin: jtag uart v						
stdout	stout						
enable small c library	stdout jtag_uart v						
enable_gprof		stderr:	itaq uart 🗸				
enable_reduced_device_drivers	enable_reduced_device_drivers						
enable_sim_optimize I enable_small_c_library							
⊢enable avcention stack							
exception_stack_memory_region_n							
enable_interrupt_stack		enable_sim_optimize					

【図 2-5-9】 BSP Editor

__3. ソフトウェアをビルド (Build) します。

Nios II SBT の Project Explorer タブ内にあるアプリケーション・プロジェクトのフォルダー (soft_test) を選択し、 右クリック ➤ Build Project をクリックします。

Nios II - soft_test/led_output.c - Eclipse					
File Edit Source	Refactor Navigate Search Project Run Nios II Windo	ow Help			
Project Explore	New Solution State	? - <i>J</i> ≫ ± -			
✓ ²⁹ soft_test	Open in New Window	o.h>			
> 🔐 Include > 🛃 Ied_ou	Copy Ctrl+C	-≳ td.h>			
📄 create- 👘	Paste Ctrl+V	em.h"			
💧 Makefi 🗙	Delete Delete	500000			
eadmi	Remove from Context Ctrl+Alt+Shift+Down				
y w son_test_t	Source >				
	Move				
	Rename F2	llo from <u>Nios</u>			
<u>P-1</u>	Import				
2	Export	ED PIO BASE,			
	Build Project	(PERIOD);			
	Clean Breject	ED PIO BASE.			
	clean Project				

【図 2-5-10】 プロジェクトをビルド

____4. エラーなくビルドが完了することを確認します。

🖹 Problems 🧔 Tasks 📮 Console 🐹 🔲 Properties					
CDT Build Console [soft_test]					
Info: 24 KBytes free for stack + heap.					
Info: Creating soft_test.objdump					
<pre>nios2-elf-objdump.exedisassemblesymsall-headersource soft_test.elf >soft_te:</pre>					
[soft_test build complete]					
10:22:21 Build Finished (took 17s.785ms)					

【図 2-5-11】 Console ウィンドウ

5-5. ソフトウェア・プログラムを実行

____1. Nios II SBT の Project Explorer タブ内の soft_test フォルダー を右クリック ➤ Run As ➤ Run Configurations をクリックします。

Nios II - soft_test/led File Edit Source Re	_output.c - Eclipse efactor Navigate Search Project Run	Nios II Window Help	
📑 🖛 📙 💼 🛙	🖻 ▼ 🔂 ▼ 📴 ▼ 🕝 ▼ 🕸 ▼ 🗘 ▼ 🎙	• • 🙋 🖨 🔗 • 🍠 📚 🕯	b - P - + + + + + + + + + + + + + + + + +
Project Explorer 🛛			Ied_output.c 器
> soft_test > sin Binarie > m Includ	New Go Into	>	<pre>1 #include <stdio.h> 2 #include <io.h> 3 #include <unistd.h></unistd.h></io.h></stdio.h></pre>
	show in remote systems view		
_	Profiling Tools	>	
	Run As	> 🐥 1 Lauterbae	ch ISS
	Debug As	> C 2 Local C/C	C++ Application
	Profile As	> 🛗 3 Nios II Ha	ardware
	Restore from Local History	📴 4 Nios II M	odelSim
	Nios II	> Run Confic	jurations
	Undate Linked Resources	L	

【図 2-5-12】Run Configurations

____2. Run Configurations ウィンドウのアイテムから Nios II Hardware を選択してダブルクリックします。

Run Configurations	×
Create, manage, and r Nios II Hardware Tab Group	un configurations
Image: Specific action type filter text c C C C Nios II Hardware Nios II ModelSim Image: Nios II ModelSim Image: Nios II ModelSim Filter matched 7 of 7 items	Configure launch settings from this dialog: Configure launch settings from this dialog: Press the 'New' button to create a configuration of the selected type. Configure 'Duplicate' button to copy the selected configuration. Press the 'Delete' button to remove the selected configuration. Press the 'Delete' button to configure filtering options. - Edit or view an existing configuration by selecting it. Configure launch perspective settings from the <u>'Perspectives'</u> preference page.
?	<u>R</u> un Close

【図 2-5-13】 Run Configurations

____ 3. Run Configurations ウィンドウの *Name 欄* において、デフォルトの New_configuration から任意の名称に 変更します。この演習では *soft_test* と入力してください。

Run Configurations		×
Create, manage, and run config Nios II Hardware Tab Group	gurations	
	Name: soft_test	
type filter text C C/C++ Application C C/C++ Remote Application Launch Group √ Nios II Hardware Nios II Hardware v2 (beta)	Project Targe Project name: Project ELF file name: Enable browse for f	t Connection I Debugger I Source] Common soft_test C:\intelFPGA_pri\cm_lab_dev\lab2\software\soft_test\soft_test.elf file system ELF file

_4. Run Configurations ウィンドウの Target Connection タブ を選択し、右上の [Refresh Connections] ボタン を クリックします。【図 2-5-15】 のように USB-Blaster II が検出されるのを確認します。

Run Configurations							>
Create, manage, and run conf The expected Stdout device name do	igurations es not match the selected target by	yte stream device name	e.				
Image: Second secon	Name: soft_test	tion 🗱 Debugger	🍫 Source 🔳	<u>C</u> ommon			
Launch Group	Cable	Device	Device ID	Instance ID	Name	Architecture	Refresh Connections
Kios II Hardware New_configuration Nios II Hardware v2 (beta) Nios II ModelSim	USB-BlasterII on localhost	[USB-1] 10M08SA(. .	1	0	nios2 O	Nios2:3	Resolve Names System ID Properties
Nios II ModelSim v2 (beta)	Cable USB-BlasterII on localhost	Device [USB-1] 10M08SA(. .	Device ID	Instance ID 0	Name jtaguart O	Version	

【図 2-5-15】 接続を確認

System ID checks 欄の2つのオプションを有効にし、[Run] ボタン をクリックします。

System ID checks Ignore mismatched system ID Ignore mismatched system timestamp		
Download ☑ Download ELF to selected target system ☑ Start processor ☐ Reset the selected target system		
	Re <u>v</u> ert	Appl <u>y</u>
	Run	Close

【図 2-5-16】 System ID checks のオプション

___ 5. Nios II SBT の Nios II Console ウィンドウに、C 言語ソースコード上に記載した printf 関数の出力キャラクタ "Hello from Nios II, start!!" が確認できます。

【図 2-5-17】 Nios II Console ウィンドウ

同時に、開発ボード上の LED の点滅を確認してください。これにより、Nios II が正しくソフトウェア・プログラ ムを実行したことがわかります。

LED の点滅を確認したら、Nios II Console ウィンドウ右上の赤いボタンをクリックしてターミナルを終了します。

5-6. ソフトウェア・プログラムを変更して実行

C 言語のコードを編集して、プログラムを変更してみましょう。

___1. soft_test フォルダー下の led_output.c をダブルクリックし、C 言語ソースコードを開きます。

__2. led_output.c の 6 行目に PERIOD として定義されている値を 500000 から 1000000 に変更します。

Nios II SBT のメニューから File ➤ Save を選択し、C 言語ソースコードを保存します。

Nios II - soft_test/led_output.c - Eclipse						
<u>File Edit Source Refactor Navigate Search E</u>	roject <u>R</u> un Ni <u>o</u> s II <u>W</u> indow <u>H</u> elp					
📑 • 🗉 🕞 📠 🙋 • 🎯 • Ĉ • 🧭 • 🕯	× • O • • • • @ ⊕ <i>A</i> • []					
🎦 Project Explorer 👷 🖃 🔄 🖘 🗢 🗖	€ led_output.c 🖾					
✓ [™] soft_test	1 #include <stdio.h></stdio.h>					
> 🕍 Binaries	2 #include <io.h></io.h>					
> 🔊 Includes	<pre>3 #include <unistd.h></unistd.h></pre>					
> 🔁 obj	4 #include "system.h"					
> 📝 led_output.c						
> 🐝 soft_test.elf - [alteranios2/le]	6 #define PERIOD 1000000					
create-this-app	8 int main (woid)					
【図 2-5-18】 led output.c を編集						

- ____3. アプリケーション・プロジェクトのフォルダー (soft_test) を右クリックで選択 ➤ Build Project をクリックし、プ ロジェクトをビルドします。
- ____4. ビルドが完了したら、soft_test プロジェクトのフォルダーを右クリック ➤ Run As ➤ Nios II Hardware をクリック クし、プログラムを実行します。
- ____5. Nios II SBT の Nios II Console ウィンドウに、 "Hello from Nios II, start!!" の文字が確認でき、同時に開発ボー ド上の LED の点滅する速度が変化していることを確認してください。

6. LED の点滅を確認したら、Nios II Console ウィンドウ右上の赤いボタンをクリックしてターミナルを終了します。

____7. Nios II SBT のメニューから File ➤ Exit で、Nios II SBT を閉じます。

すべての作業が終了したら、Quartus Prime のメニューから *File ➤ Close Project* により、演習 2 のプロジェクト を終了してください。

また、開発ボードの電源を OFF にしてください。

① Note:

この演習では、ハードウェア・イメージおよび Nios II のブート・イメージを 共に MAX 10 FPGA の SRAM 領域にダウンロードしているので、開発ボードの電源を切ると FPGA 内のデータは消去されます。その ため製品として成立させるためには、ハードウェア・イメージは MAX 10 の Flash メモリ領域 (CFM) に プログラムし、Nios II のブート・イメージは MAX 10 の内部あるいは外部に用意した不揮発性メモリに格 納する必要があります。

※ これらの作業は今回の演習では行いません。別コースのセミナを受講されるか、メーカーのドキュメントや弊社の Web コンテンツをご覧ください。

以上で、演習2の作業はすべて終了です。

<u>演習 3. 内蔵 ADC を使用して温度測定</u>

この演習では、MAX 10 FPGA に Nios II プロセッサー、JTAG UART、PIO、On-Chip メモリ、ADC を実装し、サンプ ルの C 言語ソースコードの動作にしたがって MAX 10 FPGA 内蔵の温度センサからアナログ信号を ADC でデ ジタル信号に変換してプロセッサー内で演算処理し、結果を表示するデザインを作成します。

● 演習で構築するシステムの概要

これらの周辺ペリフェラルは Nios II プロセッサー用内部バスである プラットフォーム・デザイナーのインタコネ クトで接続されます。

上図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。

- 主な作業内容
 - プラットフォーム・デザイナーでシステム・モジュールを作成
 - ハードウェア・デザイン作成
 - ・ ハードウェア・デザインを FPGA ヘダウンロード
 - ソフトウェアを実行

1. プロジェクトをセット

\Lambda ALTIMA

以下の手順に沿って Quartus Prime でシステム・モジュールを作成します。

この演習では、すでにベースとなる Quartus Prime プロジェクト (nios2_adc_lab) が作成されています。既存の プロジェクトにデザインを追加することで、ハードウェア用のプロジェクトを完成させます。Nios II プロセッサーの 開発を行う際は、必ずハードウェア用のプロジェクトを作成する必要があります。

演習 2 終了後に Quartus Prime の GUI を閉じてしまった場合は、Quartus Prime を起動してください。

____1. Quartus Prime のメニューから File ➤ Open Project を選択し、以下の既存プロジェクト・ファイルを選択して [開ぐ] ボタンをクリックします。

C:/intelFPGA_prj/cm_lab_dev/lab3/nios2_adc_lab.qpf

Open Project	×
\leftarrow \rightarrow \checkmark \uparrow \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark	lab3の検索 ク
整理 ▼ 新しいフォルダー	l: - II ?
<pre>> intelFPGA_prj</pre>	
ファイル名(<u>N</u>):	 Quartus Prime Project File (*.qp 〜 開く(<u>O</u>) キャンセル :

【図 3-1-1】 Open Project

- ____2. Quartus Prime の画面左上に位置する Project Navigator ウィンドウの Hierarchy ビュー において、以下2 点を確認してください。
 - nios2_adc_lab が登録されている
 - ターゲットの FPGA 型番が 10M50DAF484C6GES である

🕥 Quartus Prime Standa	ard Edition - C:/intelFPGA_prj/cm_lab_dev/lab3/nios2_adc_lab - nios2_adc_lab
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>P</u> ro	ject <u>A</u> ssignments P <u>r</u> ocessing <u>T</u> ools <u>W</u> indow <u>H</u> elp
🗋 🗖 🖬 🤟 🖒 🖡	🗈 🔿 🤆 İ nios2_adc_lab 💿 🗸 🍼 🎸 💿 🕨 🚩 🌿 🐥 🌍 🛔
i 🗇 🔁 🗟 🦃 🚑 i 🦻	🚈 🧐 💁 👷 🚧 🗞 🛷 🗨 ୟ 💐 🕸 📼 📾 🔖 👗
Project Navigator	→ Hierarchy ▼ Q II B ×
	Entity:Instance
MAX 10: 10M50DAF4	484C6GES
nios2_adc_lab	

【図 3-1-2】 Hierarchy ビュー (Project Navigator ウィンドウ)

2. プラットフォーム・デザイナーでシステム・モジュールを作成

Quartus Prime に付属するシステム統合ツール プラットフォーム・デザイナー (Platform Designer) を使用して、 Nios II プロセッサーを含むシステム・モジュールを作成します。

2-1. プラットフォーム・デザイナーを起動

🛆 ALTIMA

_1. Quartus Prime のメニューから Tools ➤ Platform Designer を選択します。

____2. プラットフォーム・デザイナーが起動し、ファイルの選択画面が表示されます。

この演習には、途中まで作成したシステム・モジュールがあらかじめ用意されています。作業フォルダーから nios2_system.qsys を選択し、[開く] ボタンをクリックします。

Dpen System - Platfe File Edit System Gener	orm Designer - unsav rate View Tools He	ved.qsys* (C elp	¥intelFPG/	_prj¥cr	m_lab_dev¥lab3¥unsa	ved.qsys)		
📂 IP Catalog 🛛 🛛	- 🗗 🗆	📜 Syste	m Contents	: 23	Address Map ⊠	Interconnect Requirer	ments 🖾	
Project	₩ ₩	+ Use	Conn	Name		Description		Export
System System Library Basic Functions DSP Interface Protocol Low Power Memory Interfaces Processors and P Ogys Interconnec State Common	ファイルの場所(0): 最近使った項 デフクレップ	lab3 db incremen software nios2_sys	tal_db tem.qsys]			•	< set ouble-olic
New Edit	۲: ۲: ۲: ۲: ۲: ۲: ۲:	ァイル名(N): ァイルのタイプ(T): Platf	orm De	signer System Files (*.qsys) v	■ 取消	

【図 3-2-1】Platform Designer で .qsys を選択

作成途中のシステム・モジュールが復元されます。

【図 3-2-2】 Platform Designer で .qsys を復元

2-2. ADC コアを追加

1. プラットフォーム・デザイナーの IP Catalog ウィンドウにある検索バーに、*adc* とキーワードを入力します。 フィルター検索された Library から、*Modular ADC core Intel FPGA IP* を選択し、[Add] ボタンをクリックします。

<u>File Edit System Generate View Tools H</u>elp

📑 IP Catalog 🛛 🗕 🗖	5 🗖	13	System	Contents 🛛	Address N	Map ⊠	Inte
🔍 adc 🛛 🗙			Z	📲 System:	nios2_syste	m Patł	n: clk_
Project	-	+	Use	Connections		Name	
New Component						🗆 clk_(0
Library Processors and Peripherals Modular ADC core Intel FPGA IP Modular Dual ADC core Intel FPGA IP		× 🕅 州				ck ck ck ck	in in_res reset os2 (

【図 3-2-3】 IP Catalog から ADC IP モジュールを検索

___2. Modular ADC core Intel FPGA IP の General タブ において、【表 3-2-1】 および 【表 3-2-2】 のとおりにパラ メーターを設定します。

【表 3-2-1】 Modular ADC core Intel FPGA IP のパラメーター設定 (1)

項目	設定内容
Reference Voltage > Reference Voltage Source	Internal
Channels タブ >TSD タブ	Use on-chip TSD オプション =√ (On)

Modular ADC core Intel FPGA IP - modular_adc_0		
Modular ADC core Intel FPGA IP		
T Block Diagram	General	
Show signals	Core Configuration	
modular ada 0	Core Variant:	Standard sequencer with Avalon-
modular_auc_o	Debug Path:	Disabl 🗸
clock	▼ Clocks	
reset_sink reset	ADC Sample Rate:	1 Mhz 🗸
adc_pll_clock	ADC Input Clock:	10 M 🗸
adc_pll_locked	Reference Voltage	
sequencer_csr avalon	Reference Voltage Source:	Internal 🤝
sample_store_csr avaion	Internal Reference Voltage:	3 🗸 V
attera modular ado	* Logic Simulation	
atera_noural_au	Enable user created expected output file	e: Disabl 🗸
	Channels Sequencer CH0 CH1 CH2 CH3 CH4 CH5 C	H6 CH7 CH6 TSD
	▼ On-chip Temperature Sensing D	liode
	☑ Use on-chip TSD	

【図 3-2-4】 Modular ADC core Intel FPGA IP のパラメーター設定 (1)

【表 3-2-2】 Modular ADC core Intel FPGA IP のパラメーター設定 (2)

項目	設定内容
Sequencer タブ > Conversion Sequence Length > Number of slot used	64
Slot 1 ~ Slot 64	TSD

👗 Modular ADC core Intel FPGA IP - modula	r_adc_0
Modular ADC core Intel altera_modular_adc	FPGA IP
T Block Diagram	General
Show signals	Fore Configuration
modular_adc_0	▶ Clocks
clock clock interrupt	▶ Reference Voltage
reset_sink	Logic Simulation
adc_pll_clock	Channels Sequencer
adc_pll_lockedconduit	Conversion Sequence Length
sequencer_csravalon	Number of slot used: 64 🗸
sample_store_csravalon	Conversion Sequence Channels
	Slot 1: TSD 🗸
	Slot 2 : TSD 🗸
	Slot 3 : TSD 🗸
	Slot 4 : TSD

【図 3-2-5】 Modular ADC core Intel FPGA IP のパラメーター設定 (2)

___3. Modular ADC core Intel FPGA IP 画面の右下に位置する *[Finish] ボタン* をクリックし、ADC IP のパラメーター 設定ウィンドウを閉じます。

プラットフォーム・デザイナーの System Console 内に Modular ADC core Intel FPGA IP が追加されました。

【図 3-2-6】 nios2_system に Modular ADC core Intel FPGA IP が追加

2-3. ADC コアを Nios II プロセッサーに接続

____1. プラットフォーム・デザイナーの *System Contents* タブにおいて、modular_adc_0 を見やすい位置にするため、 ツールバーの *[Move Up] ボタン* を5回クリックし、nios2_cpu の下に移動します。

※ 画面を見やすくするための操作です。必須ではありません。

【図 3-2-7】 modular_adc_0 の配置変更

_ 2. Connections の項目において、【表 3-2-3】 および【図 3-2-8】 を参考に ADC コアにクロックおよびリセット を接続し、Nios II プロセッサーと接続します。

【表	3-2-3	クロックとリセットの接続
----	-------	--------------

Name	ポート名	接続先
modular_adc_0	clock	clk_0.clk
	reset_sink	clk_0.clk_reset
	sequencer_csr	nios2_cpu.data_master
	sample_store_csr	nios2_cpu.data_master
	sample_store_irq	nios2_cpu.irq

13	System	Contents 🛛	Address Map 🛛	Interconnect Requ	uirements 🖾	
	X A	💐 System: n	ios2_system Patl	h:clk_0		
+	Use	Connections	Name		Description	Export
1			🗆 olk_	0	Clock Source	
×			D clk	_in	Clock Input	clk
		<u>٩</u>	clk,	_in_reset	Reset Input	reset
· _			clk		Clock Output	Double-click to expa
▲			clk	reset	Reset Output	Double-click to expa
 ▲	\checkmark		曰 喧 喧 ni	ios2_cpu	Nios II Processor	
•		🛉 📔 🚽	──→ clk		Clock Input	Double-click to expa
T		🔶 🔶 🛉	res	et	Reset Input	Double-click to expa
			dat	a_master	Avalon Memory Mapped Master	Double-click to expa
			ins	truction_master	Avalon Memory Mapped Master	Double-click to expa
			→ irq		Interrupt Receiver	Double-click to expa
			det det	oug_reset_request	Reset Output	Double-click to expa
		🛉 🛉 -	──→ deb	oug_mem_slave	Avalon Memory Mapped Slave	Double-click to expa
			×—— cus	tom_instruction_m	Custom Instruction Master	Double-click to expe
			⊡⊡∎	odular_adc_0	Modular ADC core Intel FPGA IP	
				ck	Clock Input	Double-click to expe
			res	et_sink	Reset Input	Double-click to expa
				pll_clock	Clock Input	Double-click to expa
			ado	pll_locked:	Conduit	Double-click to expa
			sec	luencer_csr	Avalon Memory Mapped Slave	Double-click to expe
			→ sar	nple_store_csr	Avalon Memory Mapped Slave	Double-click to expe
			sar	nple_store_irq	Interrupt Sender	Double-click to expe
	\checkmark		🗆 oncl	nip_memory	On-Chip Memory (RAM or ROM)	Inte
		╇┼┼╂┼┼	──── clk	1	Clock Input	Double-click to expa

【図 3-2-8】 クロック、リセット、Nios II と接続

__3. ADC コアに対して外部からクロックを供給するため、adc_pll_clock と adc_pll_locked ポートの Export カラ ムをダブルクリックし、ポート名を表示させます。

						custom_instruction_m	Oustom instruction Master	Double-click to export	
						🗆 🛄 modular_adc_0	Modular ADC core Intel FPGA IP		
♦┤		+	-		\rightarrow	clock	Clock Input	Double-click to export	clk_0
+		+	-÷	_	- >	reset sink	Reset Input		[etoek]
		$\left \right $	_	-	D-	adc_pll_clock	Clock Input	modular_adc_0_adc_pll_clock	exported
					\sim	adc_pll_locked	Conduit	modular_adc_0_adc_pll_locked	
	•	+ +	-	_ -			Avalon Memory Mapped Slave	Bicaićie Aicaica	
	+	•	-		\rightarrow	sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]
			+		\vdash	sample_store_irq	Interrupt Sender	Double-click to export	[clock]
		1 1					··· /=··· =-···		

【図 3-2-9】 PLL と接続するためのポートを作成

__4. jtag_uart から Nios II プロセッサーへの割り込み接続を設定します。

IRQ カラムの接続の交点において、jtag_uart に [0]、timer に [1]、modular_adc_0 に [2] と設定されてい ることを確認します。(【図 3-2-10】参照)

	System: nios2_:	system Path :clk_0						
Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
		🗖 clk_0	Clock Source					
		⊏– clk_in	Clock Input	clk	exported			
	<u>о</u>	─⊃ clk_in_reset	Reset Input	reset				
		— clk	Clock Output	Double-click to export	clk_0			
		─ ⊂ clk_reset	Reset Output	Double-click to export				
\checkmark		曰 🛄 nios2_cpu	Nios II Processor					
	•	→ clk	Clock Input	Double-click to export	clk_0			
		→ reset	Reset Input	Double-click to export	[clk]			
		── data_master	Avalon Memory Mapped Ma.	Double-click to export	[clk]			
		instruction_master	Avalon Memory Mapped Ma.	Double-click to export	[clk]			
		→ irq	Interrupt Receiver	Double-click to export	[clk]	IRQ (IRQ 31	\leftarrow
	$ \vdash$	debug_reset_reque	st Reset Output	Double-click to export	[clk]			
	+ +	→ debug_mem_slave	Avalon Memory Mapped Sla	Double-click to export	[clk]	= 0x0800	0×0fff	
	×—	custom_instruction	_m Custom Instruction Master	Double-click to export				
		曰 🗓 modular_adc_0	Modular ADC core Intel FP					
	♦		Clock Input	Double-click to export	clk_0			
	🛉 🔶 —	→ reset_sink	Reset Input	Double-click to export	[clock]			
		□ adc_pll_clock	Clock Input	modular_adc_0_adc_pll_clock	exported			
		→ adc_pll_locked	Conduit	modular_adc_0_adc_pll_locked				
	• •	→ sequencer_csr	Avalon Memory Mapped Sla	Double-click to export	[clock]	■ 0x0000	0×0007	
	• • +	→ sample_store_csr	Avalon Memory Mapped Sla	Double-click to export	[clock]	= 0x0000	0×01ff	
	+	── sample_store_irq	Interrupt Sender	Double-click to export	[clock]			≻ - 2
\checkmark	* * * * 	→ 🗄 onchip_memory	On-Chip Memory (RAM or		clk_0	■ 0x0000_8000	0×0000_ffff	Y
	🛉 🛉 🛉 🔶	→ ⊞ sw_io	PIO (Parallel I/O) Intel FP		clk_0	■ 0x0000_1030	0×0000_103f	
\checkmark	+ + + + +	→ ⊞ led_io	PIO (Parallel I/O) Intel FP		clk_0	■ 0x0000_1020	0×0000_102f	
		🗆 jtag_uart	JTAG UART Intel FPGA IP					
	♦	→ clk	Clock Input	Double-click to export	clk_0			
	+ + + + +	→ reset	Reset Input	Double-click to export	[clk]			
	• •	→ avalon_jtag_slave	Avalon Memory Mapped Sla	Double-click to export	[clk]	■ 0x1048	0×104f	
	+	— irg	Interrupt Sender	Double-click to export	[clk]			戸向
\checkmark		🗆 timer	Interval Timer Intel FPGA IF	5				Y
		→ clk	Clock Input	Double-click to export	clk_0			
		→ reset	Reset Input	Double-click to export	[clk]			
	• • •	→ s1	Avalon Memory Mapped Sla	Double-click to export	[clk]	= 0x1000	0×101f	
	I –	ira	Interrupt Sender	Double-click to expert	low1			ित्ती

【図 3-2-10】割り込みの設定

2-4. ベースアドレスを設定

__1. プラットフォーム・デザイナーのメニューから System ➤ Assign Base Address を選択し、ベースアドレスを自動 で適切な値に設定します。

File E	dit	System (ienerate View Tools Help			
1	IP C	Up	grade IP Cores	-	1	System
	ado	Ass	ign Base Addresses	× 🔯		∞ ▲
Due		Ass	ign Interrupt Numbers		+	Use

___2. プラットフォーム・デザイナーのメニューから File ➤ Save を選択し、nios2_system.qsys ファイルを保存します。 保存が完了したら、Save System ウィンドウを [Close] ボタンで閉じます。

2-5. システム・モジュールを生成

- 1. プラットフォーム・デザイナーのメニューから Generate ➤ Generate HDL を選択すると、Generation ウィンド ウが表示されます。設定項目がありますが、この演習ではデフォルトのまま、右下にある [Generate] ボタン をクリックし、システムを生成します。
- ____2. Generate ウィンドウのメッセージ・ボックスに "Generate: completed successfully." と表示されれば、システ ム・モジュールが正常に生成されました。Generate ウィンドウを *[Close]* ボタンをクリックして閉じます。
- ____3. プラットフォーム・デザイナーのメニューから File ➤ Exit を選択し、プラットフォーム・デザイナーを閉じます。 以下のウィンドウが表示されたら [OK] ボタンをクリックして閉じます。

🕥 Qua	rtus Prime	×
i	You have created an IP Variation in the file C:/intelFPGA_prj/cm_lab_dev/lab3/nios2_system.qsys.	
	To add this IP to your Quartus project, you must manually add the .qip and .sip files after generating the IP core.)
	The .qip will be located in <generation_directory>/synthesis/nios2_system.qip</generation_directory>)
	The .sip will be located in <generation_directory>/simulation/nios2_system.si</generation_directory>	р
	ОК	

【図 3-2-12】 Generation ウィンドウ

3. ハードウェア・デザインの作成

プラットフォーム・デザイナーで生成したシステム・モジュールを最上位階層に配置して、ハードウェア・デザイン を作成します。

この演習では、最上位階層デザインファイルがすでに用意され、システム・モジュールのインスタンスも完了して います。

3-1. ファイルの確認

____1. Quartus Prime の画面左上に位置する Project Navigator ウィンドウの Hierarchy ビュー内に表示されている nios2_adc_lab をダブルクリックし、nios2_adc_lab.v が表示されることを確認してください。(【図 3-1-1】参照)

コードを確認後、nios2_adc_lab.v タブ右端にある [X] ボタン をクリックし、ファイルを閉じます。

<u>File Edit View Project Assignments Processing Transformer</u>	pols <u>W</u> indow <u>H</u> elp
🗋 🔂 🖶 🤟 🛍 🔊 で 🛛 nios2_adc_lab	 ✓ ॐ ∅ ♦ ► ¥ K ♀ ◎ ♣ ≫ №
🔄 🗇 🔁 🗟 🦃 👫 🕖 🖽 😂 💁 😒 💔 🕵 🔍	ै थर् थर् 👽 🛲 🗟 🔖 歳
Project Navigator 💦 Hierarchy 🔻 🔍 📮 🗗 🗙	nios2_adc_lab.v
Entity:Instance	1 1 1 1 1 1 1 1 1 1
MAX 10: 10M085AE144C8G	<pre>1 ⊟module nios2_adc_lab (2 input clk, 3 input reset_n, 4 input [2:0] SW, 5 output [2:0] LED 6); 7 9 wire adc_10M_clk; 10 wire nios_S0M_clk; 11 wire pll_locked; 12 wire reset; 13 14 assign reset = !reset_n; 15 16 17 ⊟ all_pll alt_pll_inst (18 .areset (reset), 19 .inclk0 (clk), 20 clk inv clk)</pre>
Tasks Compilation 🔻 🗏 📮 🗗 🗙	21 .c1 (nio_50M_clk), 22 .locked (pll_locked)
Task ^	23 24 55 インスタンス済み
✓ ► Compile Design	26 ⊟ nios2_system u0 (27 .clk_clk (nios_50M_clk), // clk.
> > Analysis & Synthesis	28 .reset_reset_n (reset_n), // reset_29 .led_io_external_connection_export (LED), // led_
> Fitter (Place & Route)	30 .sw_io_external_connection_export (SW), // swi 31 .modular_adc_0_adc_pll_clock_clk (adc_10M_clk), // modu 22
> Assembler (Generate programming files)	<pre>32 .modular_adc_0_adc_pil_locked_export(pll_locked) // modu 33);</pre>
> Timing Analysis	34 35 26 andrody 1a
EDA Netlist Writer	30 enamodu i e

【図 3-3-1】 nios2_adc_lab.v

____2. 先ほど作成したシステム・モジュール nios2_system を現在のプロジェクトに登録します。

Quartus Prime のメニューから Project ➤ Add/Remove Files in Project を選択します。File name 欄の右端にあ るブラウズボタンをクリックし、Select File ウィンドウにおいて ¥lab3¥nios2_system¥synthesis 内にある nios2_system.qip を選択します。(【図 3-3-2】参照)

このプロジェクトに必要なその他のデザインファイルは、登録済みです。

nios2_system.qip が登録されたことを確認後、Settings ダイアログボックスを [OK] ボタンで 閉じます。

og Search plates	Locations	Eile name:			<u>A</u> dd
ettings ar	nd Conditions	File Name	Туре	Library Design Entry	Remove
ature		> nios2_system/synthesis/nios2_system.qip	IP Variation File (.qip)	<none></none>	_
1 Process	Settings	top.sdc	Synopsys Design Constraints File	<none></none>	<u>U</u> p
ntal Com	pilation	nios2_adc_lab.v	Verilog HDL File	<none></none>	Down
ettings		> all_pll.qip	IP Variation File (.qip)	<none></none>	<u>_</u>
Entry/Syn	thesis				Properties
on	Select File			×	
evel :ttings	\leftrightarrow \rightarrow \checkmark \uparrow	v nios2_system > synthesis >	✓ ひ synthesisの検索	٩	
put	整理 ▼ 新	iしいフォルダー			
HDL Inpu Paramete lyzer	✓ 🔒 lab3 .qs db	sys_edit submodules p nios2_system.qip nios2_system.v			
stant	> 📙 nio	os2_system			
.ogic Ana zer Interf.	sof	ftware ¥¥cm_lab_dev¥lab	o3¥nios2_system¥synthesis	フォルダー内	
/zer Setti er		ファイル名(N): nios2_system.qip	∽ Design Files (*.tdf *.vhc 開く(O) =	l*.vhdl* ~ キャンセル :	

【図 3-3-2】 nios2_systemqip を選択

3-2. FPGA の外部端子をレイアウト (ピンアサイン) する

作成したデザイン上の信号を FPGA を外部端子へアサインするため、ピン番号を指定します。 なお、この演習ではすでにピンアサインが実施されています。どのように設定されているかを確認しましょう。

____1. Quartus Prime のメニューから Assignments ➤ Pin Planer を起動し、All Pins リスト の枠内が【図 3-3-3】 のとおり設定されていることを確認してください。

ъ Д	Node Name	Direction	Location	I/O Standard	I/O Bank
	🔷 leds[0]	Unknown	PIN_T20	1.5 V	5
	🔷 leds[1]	Unknown	PIN_U22	1.5 V	5
	🔷 leds[2]	Unknown	PIN_U21	1.5 V	5
	🔷 leds[3]	Unknown	PIN_AA21	1.5 V	5
	🔶 leds[4]	Unknown	PIN_AA22	1.5 V	5
	💎 reset_n	Unknown	PIN_D9	3.3-V LVTTL	8
	📀 clk	Unknown	PIN_M9	2.5 V	2
s S	< <new node="">></new>				
E I					
A	<				

【図 3-3-3】 All Pins リスト (Pin Planner)

___2. 問題がなければ、Pin Planner のメニューから File ➤ Close により Pin Planner を閉じます。

3-3. ハードウェア・デザインをコンパイル

____1. Quartus Prime のメニューから Processing ➤ Start Compilation をクリックし、コンパイルを開始します。

Project Assignments	Proc	essing Tools Window Help			
DDD70	STOP	Stop Processing	Ctrl+Shift+C	/ 🗳 🤄	5 🎨 🚥 🕨 🖌 🕹 😂
🍹 🗲 🖽 😂 💺 😫		Start Compilation	Ctrl+L		
A Hierarchy		Analyze Current File		Ĩ I	Start Compilation $ \sigma $
Entity:Inst		Start	•		ショートカットボタン

[【]図 2-3-5】 コンパイル実行

コンパイルには多少の時間を要します。

____2. Messages ウィンドウに、Quartus Prime Full Compilation was successful. のインフォメーション・メッセージが表示されたら、エラーが発生することなくコンパイルが終了しています

この演習はエラーが発生することなくコンパイルが完了するはずです。もしエラーが発生した場合は、解消しないと次のステップには進めません。回避するためにエラーの原因を追究し修正する必要があります。

また、エラー・メッセージの他、ワーニング・メッセージが発生する場合もあります。ワーニングは解消しなくて も次のステップへ進めますが、必ず内容を確認し、その内容を回避すべきか無視できるのかをユーザーが判 断してください。

各メッセージ内容の詳細を確認するには、ヘルプ機能を活用してください。(14ページ【図 1-4-3】参照)

4. ハードウェア・デザインを FPGA ヘダウンロード

作成したハードウェア・デザインのデータを MAX 10 FPGA にダウンロードします。

▲ 注記:

実際の開発では、コンパイル後に Quartus Prime の Timing Analyzer によるタイミング検証を行い、期待 どおりの動作が実現できるかを検証します。期待するタイミングを満足できることが確認できたら、ボード 上のデバイスへデータを書き込みます。

今回の演習ではタイミング検証を省略していますが、自身の開発時は必ずタイミング検証を行った上でデ バイスへの書き込みを行ってください。

Note:

MAX 10 開発キットは、FPGA ヘデータをダウンロードする際 ボードに組み込まれたインテル FPGA ダウ ンロード・ケーブル II (旧称 USB Blaster[™] II) 回路を使用します。パソコンとボードの接続は、キットに付属 している mini-USB ケーブルで行います。パソコンとボードが通信するためには、インテル FPGA ダウンロ ード・ケーブル II 用のドライバーをインストールする必要があります。インストール方法は、【表 1-1】 No. 5 をご覧ください。

_ 1. キット付属の電源コネクタを接続し、ボードの電源が OFF であることを確認してください。その後、付属の mini-USB ケーブルを用いてボードとパソコンを接続します。 (【図 2-4-1】 参照)

【図 3-4-1】 パソコンととボードを接続

____2. 開発ボードへ電源を供給します。

__3. Quartus Prime のメニューから *Tools ➤ Programmer* を選択し、Programmer を起動します。

4. Programmer の [Hardware Setup] ボタン 右横の欄が、USB -Blaster II と表示されていることを確認します。

Aardware Setup USB-BlasterII [USB-1]	Mode:	JTAG	•
Enable real-time ISP to allow background programming when available			

【図 3-4-2】 Hardware Setup

もし No Hardware 表示の場合には、[Hardware Setup] ボタン をクリックし、Hardware Settings タブ の Currently selected hardware 項のプルダウン・リストから、USB-Blaster II を選択し、[Close] ボタンをクリック してください。

____5. Programmer の *Mode* プルダウン・リストから *JTAG* を選択します。

📥 Hardware Setup	USB-Blasterii [USB-1]	Mode:	JTAG	
Enable real-time ISP	to allow background programming when available			

	3-4-3	Mode
--	-------	------

6. Programmer の Files 欄に、ダウンロードするファイル nios2_adc_lab.sof が選択されていることを確認します。
<none> となっている場合には、[Add File] ボタン をクリックし、起動したファイルブラウザから
nios2_adc_lab.sof を選択します。(作業フォルダー内 output_files フォルダー に生成されています。)

____7. プログラミング・オプションの Program/Configure にチェックを入れます。

____8. Programmer の [Start] ボタン をクリックし、データの書き込みを開始します。

Progress バーが 100% になったら書き込み完了です。Messages ウィンドウには Successful のインフォメー ションが表示されます

Ardware Setup USB-Blasteril [USB-1]				Mode: JTAG			Progress: 100% (Successful)			
Enable real-time !	SP to allow background programmir	ng when available								
▶ [™] Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase
Stop	output_files/nios2_adc_lab.sof	10M50DAF484C6GES	003F59DF	003F59DF	\checkmark					

ハードウェア・デザインのデータは FPGA に転送されましたが、FPGA 内部の Nios II プロセッサーが実行する ためのプログラムがまだありません。

続いて、Nios II プロセッサーが実行するソフトウェアの環境を構築し、実行しましょう。

5. ソフトウェアを実行

この演習では、Nios II SBT を使用して、C ソースコードのビルド、およびターゲット・プログラムの開発ボードへの ダウンロードまでを実施します。

5-1. Nios II SBT を起動

_1. Quartus Prime のメニューから Tools ➤ Nios II Software Build Tools for Eclipse をクリックして起動します。

① Note:

もし この起動方法で、GUI が表示されない、あるいは ソフトウェア・プロジェクト作成時にエラー (Failed to execute: wsl ./create-this-app -no-make) が発生した場合は、以下のフローにより Nios II SBT を起動し てください。

<Windows 10 の場合>

- Windows 10 の [スタート] ➤ Intel FPGA <version_build> Standard Edition ➤ Nios II Command Shell (Quartus Prime <version>) を右クリック選択し、ショートカット・アイコンのファイルの場所を開きま す。
- 2. Nios II Command Shell (Quartus Prime <version>) を [管理者として実行] し、Nios II Command Shell を起動します。
- 3. eclipse-nios2.exe コマンドを実行して起動します。

manufic first intelFPGA/20.1

Altera Nios2 Command Shell

Version 20.1, Build 711

root@ CCUBS :/mnt/c/intelFPGA/20.1# eclipse-nios2.exe

<Linux OS の場合>

1. ターミナル上で下記コマンドにより Nios II Command Shell を起動します。

<Nios II EDS install path>/nios2_command_shell.sh

2. eclipse-nios2 をタイプして起動します。

__ 2. Workspace Launcher が起動します。 *[Browse] ボタン* をクリックし Workspace に演習 3 用の Quartus Prime プロジェクト・フォルダー下の *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:¥intelFPGA_prj¥cm_lab_dev¥lab3¥software	∼ <u>B</u> rowse
Use this as the default and do not ask again	OK Cancel

【図 3-5-1】 Workspace Launcher

Nios II SBT が起動します。

5-2. ソフトウェア・プロジェクトを作成

____1. Nios SBT 上のメニューから File ➤ New ➤ Nios II Application and BSP from Template をクリックし起動します。

File Edit Navigate Search Project Run Nios II Window Help New Alt+Shift+N >	
New Alt+Shift+N > 🔂 Nios II Application and BSP from Template	
Open File Nios II Application	s
Close Ctrl+W Ctrl+W Nios II Board Support Package	
Close All Ctrl+Shift+W 🕅 Nios II Library	6
Save Ctrl+S	
Save As	trl+N a

【図 3-5-2】 Nios II Application and BSP from Template を起動

- ____ 2. プラットフォーム・デザイナーで作成した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。(【図 3-5-3】 参照)
 - Target hardware information エリアの SOPC Information File name 欄に、ハードウェア・デザインを作成した lab3 フォルダー内の nios2_system.sopcinfo を選択します。
 - ・ Application project エリアの Project name 欄に、adc_test と入力します。
 - ・ Templates 枠内 から Blank Project を選択します。

設定後、[Finish] ボタン をクリックします。

State a new application and board support package based on a software example table state a new application and board support package based on a software example table state hardware information Softer Information File name: C#IntelFPGA_prj*cm_lab_dev#lab3*nios2_system.sopcinfo CPU name: nios2_cpu Application project Project name: C#IntelFPGA_prj*cm_lab_dev#lab3*software#dac_test Project tocation: C#IntelFPGA_prj*cm_lab_dev#lab3*software#dac_test Implate Template Nor diagnostics Soft to station Binary Boat2 Applicationality Hab World Snall Hab World Snall Memory Test Memory Test Snall	Nios II Application and BSP fr	om Template —		×
Create anew application and board support package based on a software example template Target hardware information SOPC Information File name: C#intelFPGA_prj#cm_lab_dev#lab3¥nios2_system.sopcinfo CPU name: nios2_cpu Application project Image: C#intelFPGA_prj#cm_lab_dev#lab3¥software¥adc_test Project name: citetif Image: C#intelFPGA_prj#cm_lab_dev#lab3*software¥adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3*software¥adc_test Project template Template description Image: Count Binary Black Project creates an empty project to which you can add for create file in the project directory. Hoat2 Functionality Float2 Performance Hello Micro2/05-II The SP for this template is based on a different operating system. Citck Next and select the BSP from the BSP project is list. Memory Test Small	Nios II Software Examples			
Target hardware information SPC Information File name: C#intelFPGA_prj#cm_lab_dev#lab3¥nios2_system.sopcinfo CPU name: nios2_cpu Application project	Create a new application and bo template	ard support package based on a software example		
SOPC Information File name: C#intelFPGA_prj#cm_lab_dev#lab3¥nios2_system.sopcinfo CPU name: nios2_cpu Application project Project name: dc_test Image: Use default location Project location: C#intelFPGA_prj#cm_lab_dev#lab3¥software¥adc_test Project template Image: Project template Image: Project template Image: Image:	Target hardware information			
CPU name: nios2_cpu Application project Project name: adc_test Image: Comparison of the state of the s	SOPC Information File name:	C:¥intelFPGA_prj¥cm_lab_dev¥lab3¥nios2_system.sopcinfo)	
Application project Project name: dc_test Image: Image: Image: Image: Project location: C#intelFPGA_prj*cm_lab_dev#lab3*software*adc_test Image: Project template Image: Image: Image: Project template Image: Image: Image: Image: Image: Image	CPU name:	nios2_cpu v		
Project name: dc_test Image: State of the	Application project			
Use default location: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Project template Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#software#adc_test Image: C#intelFPGA_prj#cm_lab_dev#lab3#	Project name: adc_test			
(?) < Back Next > Einish Cancel	✓ Use default location Project location: Cremplate Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 GCC Float2 GCC Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test Memory Test Small	FPGA_prj¥cm_lab_dev¥lab3¥software¥adc_test 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.txt file in the project directory. 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 the BSP projects list. For information about how this software example relates to	····	
	?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel	_

【図 3-5-3】 Nios II Application and BSP from Template

____ 3. 新しいソフトウェア・プロジェクトが作成され、Nios II SBT の Project Explorer タブ内に *adc_test* と *adc_test_bsp* が追加されます。

🖨 Nios II - Eclipse		
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject	<u>R</u> un	Ni
📬 ▼ 📙 🕼 🔝 🔂 ▼ 😂 ▼ 💕 ▼	• 🞯	•
Project Explorer 🕱 📃 🖻 E	3	
 Adc_test Adc_test_bsp [nios2_system] 		
【図 3-5-4】 ソフトウェア・プロジェクト作	■成	

5-3. C ソースコードのインポート

あらかじめ用意された C 言語のソースファイルをソフトウェア・プロジェクトにインポートします。

____1. Windows エクスプローラを開き、この演習 3 用フォルダー下の *software* フォルダーにある *ondie_temp.c* ファイルを、Nios II SBT の Project Explorer タブ内にある *adc_test* フォルダーにドラッグ&ドロップします。

🖨 Nios II - Eclipse	
File Edit Navigate Sear	:h Project Run Nios II Window Help
: 📬 🗸 📙 🐚 📄 🔂 🗸	
Project Explorer	↓ ♪ J マーム 共有 表示
adc test	← → ~ ↑ 📙 « intelFPGA_prj > cm_lab_dev > lab3 > software
> adc_test_bsp [nios2_	Johen] V lab3 ^metadata
	.dc_test
	> db adc_test_bsp
	> incremental_db
	> nios2_system
	output_files
	✓ software
🖨 File	Operation ×
Select	how files should be imported into the project:
	Copy files を選択し、OK ボタンをクリック
Config	ure Drag and Drop Settings
?	OK Cancel

【図 3-5-5】C ソースコードをプロジェクトヘコピー

____2. Nios II SBT の Project Explorer タブ内にある adc_test フォルダーを展開し、ondie_temp.c が adc_test フォ ルダーにインポートされていることを確認してください。ondie_temp.c をダブルクリックすると、C ソースコー ドが表示されます。

5-4. ソフトウェア・プロジェクトのビルド

1. Nios II SBT の Project Explorer タブ内にある adc_test_bsp フォルダーを右クリックで選択 ➤ Nios II ➤ BSP Editor を選択し、システムの設定を確認します。(【図 3-5-6】参照)

Nios II - adc_test/ondie_temp.c - E File Edit Source Refactor Nav	clipse igate Search Proj	ject Run Nios II	Window Help	
	C • C • 🔅	• 🜔 • 💁 • 🍅	6 🔗 🕶 🌛 💝	2 × 2 × ← ← • → •
🍋 Project Explorer 🛛 🖃 🔄	• • • • •	💼 ondie_temp.c 🔀		
✓ [™] adc_test		2⊕ * main.c[
> 🔊 Includes		7		
> & ondie_temp.c		8		
create-this-app		9 //#includ	e "stdio.h"	
Makefile		10 #include	"system.n" "sys/alt irg b"	
Contraction for a sector 1		12 #include	"altera avalon	timer regs.h"
> adc_test_bsp [niost_system]	New		······	io regs.h"
	Galata			adc_sequencer_regs.h"
	do into			-
	Open in New Win	dow		_
	Сору		Ctrl+C	adc_avg_in);
1	Paste		Ctrl+V	
~	Delete		Delete	
	Debug As		>	
	Profile As		>	
	Restore from Loca	al History		
	Nios II		>	Nios II Command Shell
X	Run C/C++ Code	Analysis		Generate BSP
	Team	2	>	BSP Editor
	Compare With		>	Flash Programmer

【図 3-5-6】BSP Editor を起動

- ____2. BSP Editor ウィンドウの *Main タブ*において、下記オプションを有効 (√) にします。
 - enable_small_c_library = On
 - enable_reduced_device_drivers = On

🔅 BSP Editor - settings.bsp			_	×
File Edit Tools Help				
Main Software Packages Drivers Linker Script Enable File	Generation Target BSP Directory			
SOPC Information file: C:\intelFPGA_prj\cm_lab_dev\ab3\ni	ios2_system.sopcinfo			
CPU name: nios2_cpu				
Operating system: Altera HAL	Version: default 🗸			
BSP target directory: C:\intelFPGA_prj\cm_lab_dev\lab3\so	ftware\adc_test_bsp			
- Settings	hal			^
l ⊡hal	sys_clk_timer:	timer 🗸		
	timestamp_timer:	none \vee		
stdin stdout	stdin:	jtag_uart 🗸		
	stdout:	jtag_uart \vee		
	stderr:	jtag_uart ∨		
enable_sim_optimize	enable_small_c_library			
enable_exception_stack	enable_gprof			
exception_stack_size	enable_reduced_device_drivers			
exception_stack_memory_region_n enable_interrupt_stack	enable_sim_optimize			

その後、BSP Editor ウィンドウ右下に位置する [Generate] ボタン をクリックし、 [Exit] ボタン で BSP Editor ウィンドウを閉じます。

__3. ソフトウェアをビルド (Build) します。

Nios II SBT の Project Explorer タブ内にあるアプリケーション・プロジェクトのフォルダー (adc_test) を選択し、 右クリック ➤ Build Project をクリックします。

エラーなくビルドが完了することを確認します。

5-5. ソフトウェア・プログラムを実行

- ____1. Nios II SBT の Project Explorer タブ内の adc_test フォルダー を右クリック ➤ Run As ➤ Run Configurations をクリックします。
- ____2. Run Configurations ウィンドウのアイテムから Nios II Hardware を選択してダブルクリックします。

Run Configurations		×
Create, manage, and run Nios II Hardware Tab Group	configurations	
Image: Second system Image: Second system	Configure launch settings from this dialog: • Press the 'New' button to create a configuration of the selected type. • Press the 'Duplicate' button to copy the selected configuration. • Press the 'Delete' button to remove the selected configuration. • Press the 'Delete' button to remove the selected configuration. • Press the 'Elete' button to configure filtering options. • Edit or view an existing configuration by selecting it. Configure launch perspective settings from the 'Perspectives' preference page.	
?	<u>R</u> un	Close

【図 3-5-8】 Run Configurations

___3. Run Configurations ウィンドウの Name 欄 において、デフォルトの New_configuration から任意の名称に 変更します。この演習では adc_test と入力してください。

Run Configurations		×
Create, manage, and run c Nios II Hardware Tab Group	onfigurations	
	Name: adc_test	
type filter text	📔 Project 🔪 🎩 Target Connection 🕸 Debugger 🦆 Source 🔲 Common	
C/C++ Application C/C++ Remote Application	Project name: adc_test ~	
 Launch Group Mios II Hardware 	Project ELF file name: C:\intelFPGA_prj\cm_lab_dev\lab3\software\adc_test\adc_test.elf	~
New_configuration	Enable browse for file system ELF file	

__4. Run Configurations ウィンドウの Target Connection タブ を選択し、右上の [Refresh Connections] ボタン を クリックします。【図 2-5-15】 のように USB-Blaster II が検出されるのを確認します。

System ID checks 欄の2つのオプションを有効にし、[Run] ボタンをクリックします。

Run Configurations Create, manage, and run configurations The expected Stdout device name does not match the selected target byte stream device name. Image: I
Create, manage, and run configurations The expected Stdout device name does not match the selected target byte stream device name. Image: Ima
The expected Stdout device name does not match the selected target byte stream device name.
Image: Second
type filter text Image: Construction Image: Construction<
E C/C++ Application E C/C++ Remote Appli Processors: Processors: Cable Device Device ID Instance ID Name Architecture Refresh Connections Cable Device Device ID Instance ID Name Architecture Refresh Connections Miss II Hardware ISB-BlasterII on localhost (USB-1) 10M08SA(. [E 1 0 nios2 0 Nios2:3 Resolve Names Nios II Hardware V2 (Image: Set Image
Image: Construction of Constructing Construction of Construction of Constructi
▶ Launch Group Cable Device Device ID Instance ID Name Architecture Refresh Connections
INos II Hardware ISB-BlasterII on localhost [USB-1] 10M08SA(.[E1 0 nios2 0 Nios2:3 Resolve Names Nios II Hardware V2 (ISB-BlasterII on localhost [USB-1] 10M08SA(.[E1 0 nios2 0 Nios2:3 Resolve Names Nios II Hardware V2 (ISB-BlasterII on localhost [USB-1] 10M08SA(.[E1 0 nios2 0 Nios2:3 Resolve Names Image: Nios II Hardware V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II ModelSim V2 (Image: Nios II Nios
System ID Properties Byte Stream Devices: System ID Properties Image: Colspan="2">Colspan="2"Colspan=
Byte Stream Devices: Cable Device Device Device Device Device Name Version
Nios II ModelSim v2 (Cable Device Device ID Instance ID Name Version
ISB-BlasterII on localbost [USB-1] 10M08SA(.IE] 0 (tasuart 0 1
Disable Nice & Concells view
Quartus Project File name: < Using default.sopcinto & .jdi files extracted from ELF >
System ID checks
☑ Ignore mismatched system ID
☑ Ignore mismatched system timestamp
Download
Download ELF to selected taroet system
Start processor
Reset the selected target system
< >> Parinet Apply
Filter matched 8 of 8 items
(?) Close

【図 3-5-10】 Run Configurations ダイアログボックス

_ 5. Nios II SBT の Nios II Console ウィンドウに、C 言語ソースコード上に記載した printf 関数の出力キャラクタ が確認できます。同時に、開発ボード上の MAX 10 内蔵の ADC が取得した温度が表示されています。

確認したら、Nios II Console ウィンドウ右上の赤いボタンをクリックしてターミナルを終了します。

__6. Nios II SBT のメニューから *File ➤ Exit* で、Nios II SBT を閉じます。

すべての作業が終了したら、Quartus Prime のメニューから File ➤ Close Project により、演習 3 のプロジェクト を終了してください。

また、開発ボードの電源を OFF にしてください。

① Note:

この演習では、ハードウェア・イメージおよび Nios II のブート・イメージを 共に MAX 10 FPGA の SRAM 領域にダウンロードしているので、開発ボードの電源を切ると FPGA 内のデータは消去されます。その ため製品として成立させるためには、ハードウェア・イメージは MAX 10 の Flash メモリ領域 (CFM) に プログラムし、Nios II のブート・イメージは MAX 10 の内部あるいは外部に用意した不揮発性メモリに格 納する必要があります。

※ これらの作業は今回の演習では行いません。別コースのセミナを受講されるか、メーカーのドキュメントや弊社の Web コンテンツをご覧ください。

以上で、演習3の作業はすべて終了です。

演習はすべて終了しました。お疲れ様でした。

免責およびご利用上の注意

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

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