

# 独習インテル<sup>®</sup> FPGA OpenCL<sup>™</sup> 基礎演習 (Sodia ボード編)

Ver.15

# 独習インテル® FPGA OpenCL™ 基礎演習 (Sodia ボード編)

## 目次

|   |    |
|---|----|
| 1. はじめに .....                                     | 3  |
| 1-1. 演習環境 .....                                   | 3  |
| 1-2. Sodia ボード .....                              | 4  |
| 1-3. 付属ファイルの内容 .....                              | 6  |
| 2. デザイン・フロー .....                                 | 7  |
| 2-1. プログラミング・モデル .....                            | 7  |
| 2-1-1. 一般的なプログラミング・モデル .....                      | 7  |
| 2-1-2. FPGA SDK for OpenCL™ での実装モデル .....         | 7  |
| 2-2. 開発フロー .....                                  | 8  |
| 3. 事前準備 .....                                     | 9  |
| 3-1. Quartus® Prime のインストール .....                 | 9  |
| 3-2. SoC EDS のインストール .....                        | 9  |
| 3-3. FPGA SDK for OpenCL™ のインストール .....           | 9  |
| 3-4. v15.1 用 Sodia ボード・パッケージ(ボード・テンプレート)の追加 ..... | 10 |
| 3-5. Windows® 環境変数の確認・設定 .....                    | 10 |
| 3-6. 演習ファイルの抽出 .....                              | 11 |
| 3-7. その他ツールのインストール .....                          | 11 |
| 3-8. Sodia と PC の接続 .....                         | 11 |
| 3-9. PC 側ネットワークの設定 .....                          | 12 |
| 3-10. UART の接続確認と Linux の起動 .....                 | 13 |
| 3-11. Ethernet の接続確認 .....                        | 14 |
| 4. 演習 .....                                       | 15 |
| 4-1. 演習の概要 .....                                  | 15 |
| 4-2. エンベデッド・コマンド・シェルの起動 .....                     | 15 |
| 4-3. カーネルのコンパイル .....                             | 16 |
| 4-4. ホスト・アプリケーションのコンパイル .....                     | 17 |
| 4-5. ファイルの転送 .....                                | 18 |
| 4-6. OpenCL™ の実行 .....                            | 19 |
| 4-7. カーネルのコンパイル結果の確認 .....                        | 20 |
| 改版履歴 .....  | 21 |

## 1. はじめに

この資料は、インテル® SoC FPGA を利用して OpenCL™ による FPGA ロジックの高位合成を行う方法を解説します。

インテル® FPGA SDK for OpenCL™ を使用した演習を体験することで、迅速かつ簡単にハードウェア・アクセラレータを構築する方法を理解することができます。演習では、単純な配列加算の処理を OpenCL™ で実装します。

### 1-1. 演習環境

本演習では下記の開発環境を使用します。

【表 1-1.1】 この資料の説明で使用している主な開発環境

| 項番 | 項目   | 内容  |
|----|--|---|
| 1  | ホスト PC   | Microsoft® Windows® 7 Professional sp1 (64 bit) 搭載の 64 bit マシン  |
| 2  | インテル® Quartus® Prime 開発ソフトウェア  | SoC FPGA のハードウェアを開発するためのツールです。<br>ソフトウェア開発に必要なハンド・オフ・ファイルの生成も行います。<br>本説明書では、Quartus® Prime のライト・エディション v15.1 を使用しています。   |
| 3  | インテル® SoC FPGA エンベデッド・デザイン・スイート (SoC EDS) & ARM® DS-5™ Intel® SoC FPGA Edition | SoC FPGA のソフトウェアを開発するためのツールです。<br>ホスト・アプリケーション・ソフトウェアをコンパイルすることができます。<br>本説明書では、SoC EDS v15.1 を使用しています。   |
| 4  | FPGA SDK for OpenCL™   | FPGA で OpenCL™ ソフトウェアを開発するためのツールです。<br>OpenCL™ カーネルをコンパイル・実行することができます。<br>本説明書では、FPGA SDK for OpenCL™ v15.1 を使用しています。   |
| 5  | Sodia ボード  | 本資料の説明でターゲット・ボードとして使用する Cyclone® V SoC FPGA を搭載した Mpression Sodia ボードです。  |
| 6  | 演習デザイン・ファイル  | 本資料の演習で使用する Sodia ボード上で動作する、単純な配列加算の処理を OpenCL™ で実装します。<br>ホスト・アプリケーションでは入力テーブルの確保と乱数の格納を行い、カーネル (FPGA ロジック) で演算された加算値が正しいかを比較します。<br>付属ファイルの <b>Lab フォルダ</b> の下に、Altera-SoCFPGA-OpenCL-vectorAdd.tar.gz があります。 |
| 7  | UART ターミナル・ソフト   | 本演習では Tera Term を使用しています。ダウンロードやインストールについては、別途、作成者または関連サイトの情報を参考にしてください。  |

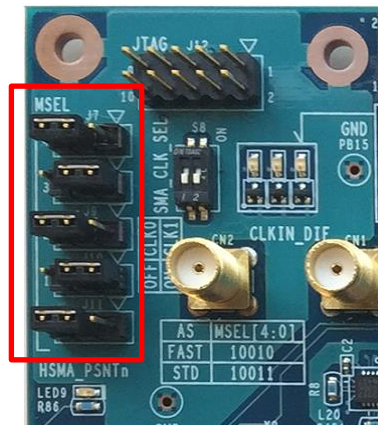
## 1-2. Sodia ボード

このボードは、Cyclone® V SoC FPGA が搭載されています。



【図 1-2.1】 Sodia ボード

(1) Sodia ボードの MSEL(J7 ~ J11)の設定を確認します。

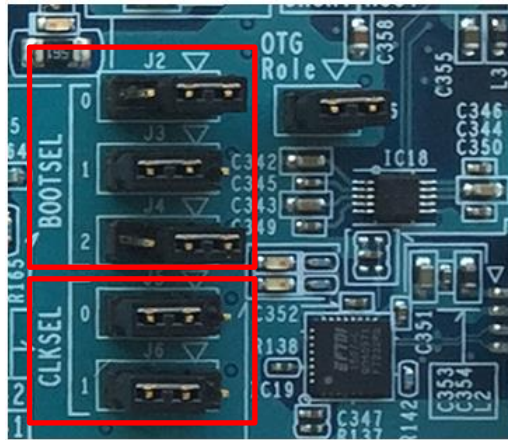


【図 1-2.2】 MSEL(J7 ~ J11)の設定

【表 1-2.1】 MSEL(J7 ~ J11)設定表

| ボード・リファレンス | 信号名    | 内容   | 設定  |
|------------|--------|--|-----|
| J7         | MSEL 4 | OpenCL™ 向け SD カード・イメージは、Hard Processor System (HPS) からの FPGA のコンフィギュレーションを実行します。<br>このため、MSEL は、Fast Passive Parallel (FPP) モードに設定する必要があります。 | 2-3 |
| J8         | MSEL 3 |  | 1-2 |
| J9         | MSEL 2 |  | 2-3 |
| J10        | MSEL 1 |  | 1-2 |
| J11        | MSEL 0 |  | 2-3 |

(2) Sodia ボードの BOOTSEL (J2,J3,J4)と CLKSEL (J5,J6)の設定を確認します。



【図 1-2.3】 BOOTSEL (J2,J3,J4)と CLKSEL (J5,J6)の設定

【表 1-2.2】 BOOTSEL (J2,J3,J4) 設定表

| ボード・リファレンス | 信号名       | 内容                         | 設定  |
|------------|-----------|----------------------------|-----|
| J2         | BOOTSEL 0 | BOOTSEL は SD カード・ブートを指定します | 1-2 |
| J3         | BOOTSEL 1 |                            | 2-3 |
| J4         | BOOTSEL 2 |                            | 1-2 |

【表 1-2.3】 CLKSEL (J5,J6) 設定表

| ボード・リファレンス | 信号名      | 内容                                     | 設定  |
|------------|----------|--|-----|
| J5         | CLKSEL 0 | CLKSEL は任意の設定で問題ありませんが、ここでは 00b を選択します | 2-3 |
| J6         | CLKSEL 1 |  | 2-3 |

### 1-3. 付属ファイルの内容

独習インテル® FPGA OpenCL™ 基礎演習 (Sodia ボード版) の付属ファイルを解凍すると、以下のものが含まれています。

■ **BSP\_Sodia フォルダ** (ボード・テンプレート)

- ・ board

  - c5soc\_sodia

    - arm32 : ホスト・アプリケーション用ライブラリ等

    - c5soc\_sodia\_sharedonly : ボード・テンプレート・プロジェクト

    - driver

    - board\_env.xml

- ・ share

■ **Lab フォルダ** (演習ファイル)

- ・ Altera-SoCFPGA-OpenCL-vectorAdd.tar.gz

■ **SD\_Image フォルダ** (OpenCL™ 用 SD カード・イメージ)

- ・ sodia\_opencl\_sdimage\_v1511\_r3.tar.gz

■ README.txt (インテル® FPGA 開発ツール 入手先 URL)

## 2. デザイン・フロー

FPGA SDK for OpenCL™ は、エンベデッド ARM® Cortex®-A9 プロセッサ・コアをホストとして、ANSI C ベースの言語である OpenCL™ C で記述したソフトウェア関数(カーネル)を、FPGA デザイン(ハードウェア・アクセラレータ)に合成し、実行することができます。

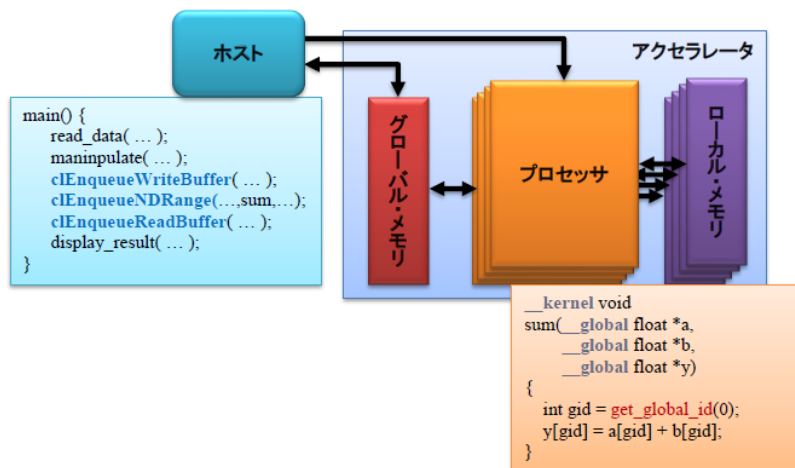
この章では、OpenCL™ 開発における、ホストとカーネル、FPGA デザインへの合成などデザイン・フローを理解することができます。

### 2-1. プログラミング・モデル

#### 2-1-1. 一般的なプログラミング・モデル

OpenCL™ の一般的なプログラミング・モデルを下図に示します。

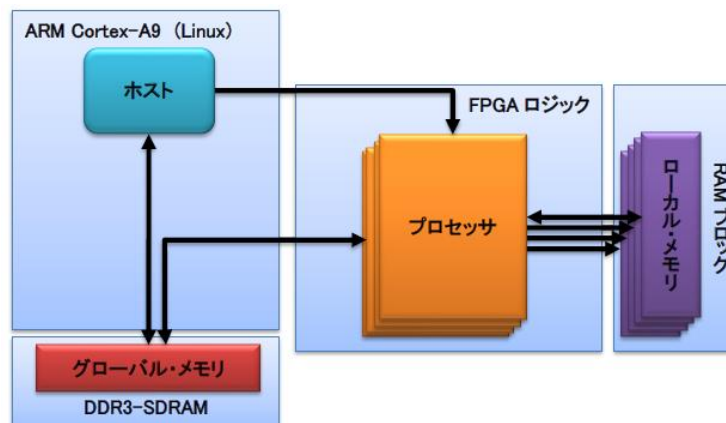
ホストでは、グローバル・メモリを確保し、プロセッサで処理するデータを準備し、プロセッサに制御を移します。プロセッサはそのデータを処理した後、再びホストに制御を戻し、処理が完了したことを通知します。



【図 2-1-1.1】 一般的なプログラミング・モデル

#### 2-1-2. FPGA SDK for OpenCL™ での実装モデル

FPGA SDK for OpenCL™ では、ホストを ARM® Cortex®-A9 プロセッサ・コアとして、グローバル・メモリを DDR3 SDRAM、プロセッサを FPGA ロジック、ローカル・メモリは FPGA 内の RAM ブロックに実装されます。



【図 2-1-2.1】 FPGA SDK for OpenCL™ での実装モデル

## 2-2. 開発フロー

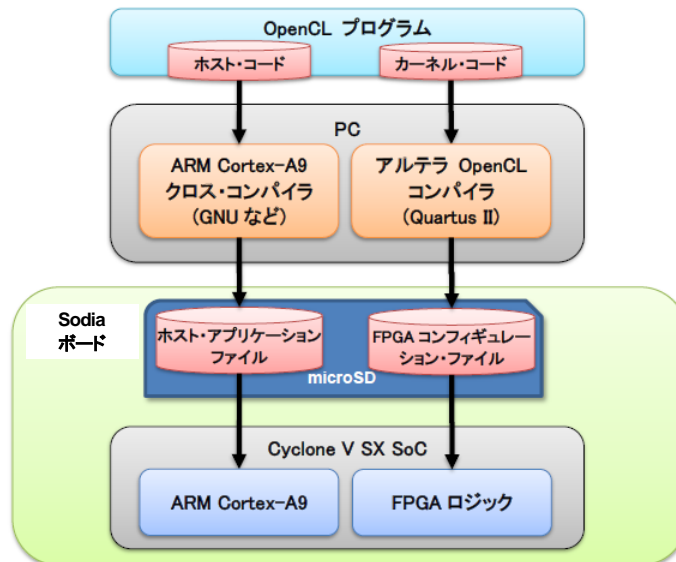
まず、OpenCL™ プログラムを記述します。OpenCL™ プログラムには FPGA に実装する部分を記述したカーネル・コードと、そのカーネルを制御するホスト・コードを記述します。

カーネル・コードは OpenCL™ C で記述するので、そのままではプロセッサで動作させることはできません。最初にカーネル・コードを標準の C 言語で記述し、正しく動作することを確認した上で OpenCL™ C に変換することを推奨します。

続いて、カーネル・コードを FPGA デザインに合成、コンパイルし、.aocx (Altera Offline Compiler eXecutable file) を生成し、ホスト (Linux) に転送します。

ホスト・コードも、PC のクロス・コンパイラ環境で実行可能なホスト・アプリケーションにコンパイルし、ホストに転送します。

ホスト・アプリケーションを実行すると、生成された .aocx ファイルで FPGA をコンフィギュレーションし、カーネル・コードに記述された機能と同等の処理が FPGA ロジックを使用して実行されます。



【図 2-2.1】FPGA SDK for OpenCL™ で開発フロー



### 3. 事前準備

この章では、演習の実施に必要な機材を確認し、ツールをインストールします。

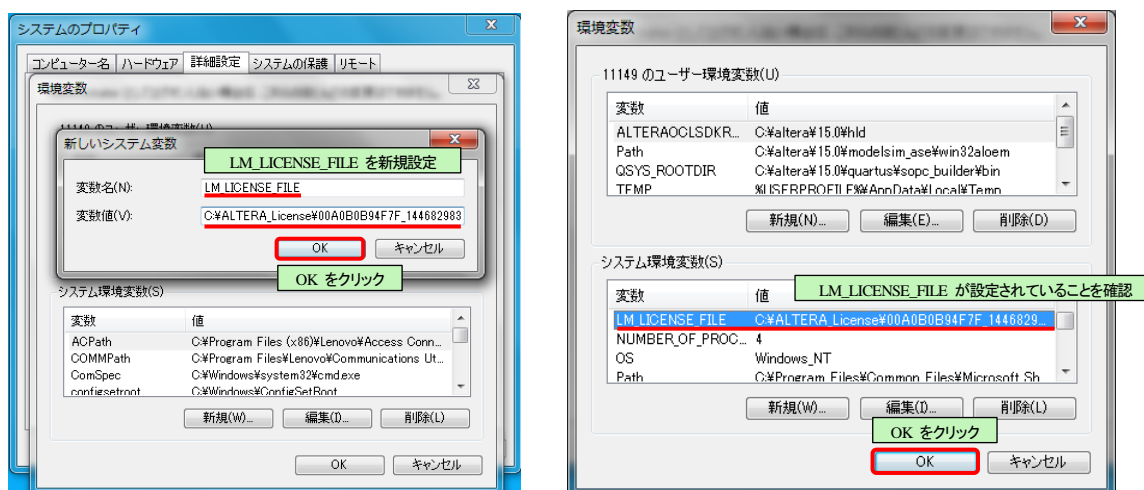
なお、後述のツールを既にインストール済みの場合は新たにインストールする必要はありませんので、読み飛ばしてください。

#### 3-1. Quartus® Prime のインストール

- (1) Quartus® Prime v15.1 (Windows® 版) をダウンロードしインストールします。本説明書では、Quartus® Prime のライト・エディション v15.1 を使用しています。
- (2) インストール後、ライセンス設定を行ってください。  
Quartus® Prime を起動し、Tools メニュー ⇒ License Setup を選択し、License Setup ダイアログが表示したら、License file に **OpenCL™ 評価用ライセンス・ファイル** のパスとファイル名を設定します。

#### [注意事項]

- ✓ **OpenCL™ 評価用ライセンス・ファイル** は、別途お客様担当の代理店にお問い合わせください。
- (3) Windows® の環境変数を開いて、LM\_LICENSE\_FILE を追加し、ライセンス・ファイルのパスとファイル名を設定します。



【図 3-1.1】 LM\_LICENSE\_FILE 環境変数の設定

#### 3-2. SoC EDS のインストール

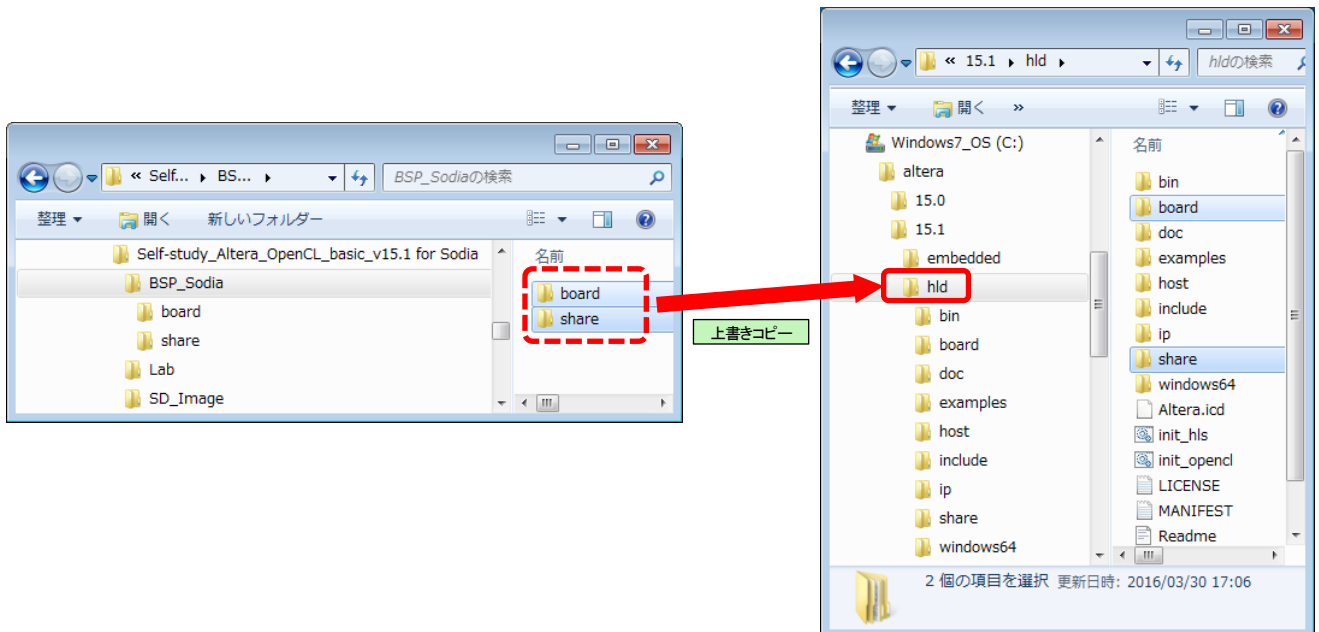
SoC EDS v15.1 (Windows® 版) をダウンロードしインストールします。

#### 3-3. FPGA SDK for OpenCL™ のインストール

FPGA SDK for OpenCL™ v15.1 (Windows® 版) をダウンロードしインストールします。

### 3-4. v15.1 用 Sodia ボード・パッケージ(ボード・テンプレート)の追加

付属ファイルの **BSP\_Sodia フォルダ** の中身(board と share)を、FPGA SDK for OpenCL™ v15.1 インストール・フォルダ(例: C:\altera\15.1\hld )の下に上書きコピーしてください。



【図 3-4.1】 Sodia ボードパッケージ(ボード・テンプレート)の追加

これにより board フォルダの下に、c5soc\_sodia フォルダが追加され、share\models\dm フォルダの下に、5csx6d6f31c8es\_dm.xml ファイルが追加されます。

### 3-5. Windows® 環境変数の確認・設定

Windows® の環境変数を開いて下記の変数を追加してください。

(1) 以下の環境変数が存在し正しい値が設定されていることを確認します。

- 変数: **ALTERAOCLSDKROOT**  
値:(例: C:\altera\15.1\hld )
- 変数: **QUARTUS\_ROOTDIR**  
値:(例: C:\altera\15.1\quartus )

(2) 以下の環境変数を追加します。

- 変数: **AOCL\_BOARD\_PACKAGE\_ROOT**  
値:%ALTERAOCLSDKROOT%\board%c5soc\_sodia
- 変数: **PATH**  
値:%QUARTUS\_ROOTDIR%\bin64;%ALTERAOCLSDKROOT%\windows64\bin

### 3-6. 演習ファイルの抽出

付属ファイル内の **Lab** フォルダ 中にある **Altera-SoCFPGA-OpenCL-vectorAdd.tar.gz** ファイルを PC の任意のワーク・フォルダ (この例では、C:¥Work) にコピーして解凍します。

解凍は、**エンベデッド・コマンド・シェル** からコマンド・ラインで実行できます。

Windows® エクスプローラで、SoC FPGA エンベデッド・デザイン・スイートのインストール・フォルダ (この例では、C:¥altera¥15.1¥embedded) を開き、その下にある **Embedded\_Command\_Shell.bat** をダブルクリックしてエンベデッド・コマンド・シェルを起動し、以下のコマンドで **Altera-SoCFPGA-OpenCL-vectorAdd.tar.gz** ファイルを解凍します。

```
$ cd "C:¥Work"
$ tar -xzf Altera-SoCFPGA-OpenCL-vectorAdd.tar.gz
```

#### [注意事項]

- ✓ コピー & 解凍先のフォルダパス名にスペースや日本語(全角文字)がないよう注意してください。

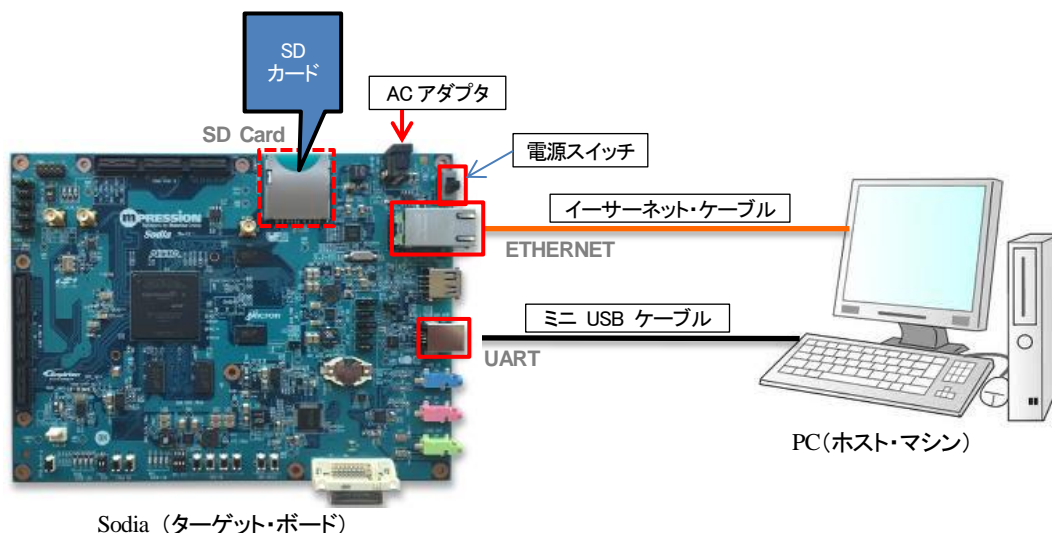
### 3-7. その他ツールのインストール

UART ターミナル・ソフトをインストールします。本資料では Tera Term を使用していますが、機能的に同等であれば別のツールを使っても構いません。

なお、ツールのダウンロードやインストールにつきましては、別途、作成者または関連サイトの情報を参考にしてください。

### 3-8. Sodia と PC の接続

PC と Sodia を接続し、SD カードをカード・スロットに挿入し、電源を入れてください。



【図 3-8.1】 Sodia と PC の接続

#### [注意事項]

- ✓ 付属ファイル内の **SD\_Image** フォルダ の下に

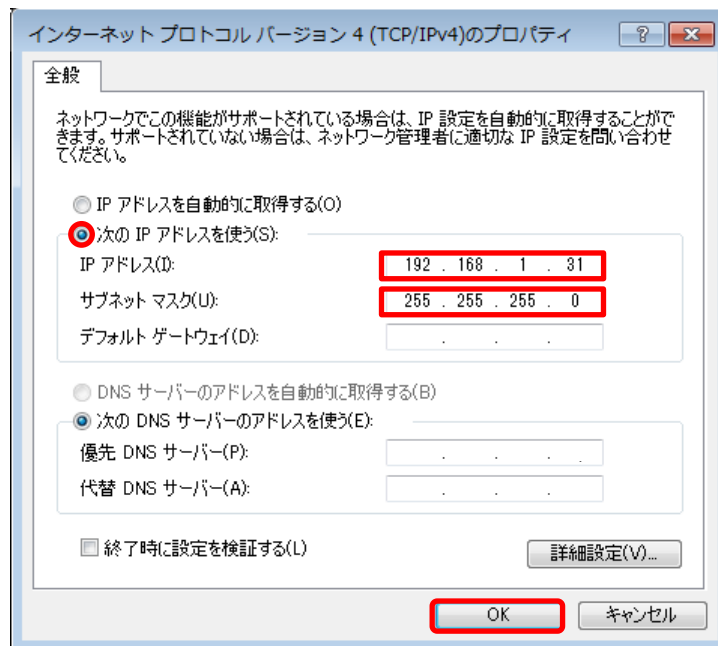
sodia\_openc1\_sdimage\_v1511\_r3.tar.gz

ファイルがありますので、これを **エンベデッド・コマンド・シェル** から **tar -xzf** コマンドで解凍し、フリーソフト Win32DiskImager を使用して SD カードに書き込みます。

### 3-9. PC 側ネットワークの設定

Sodia と PC を Ethernet で接続するための設定を行います。

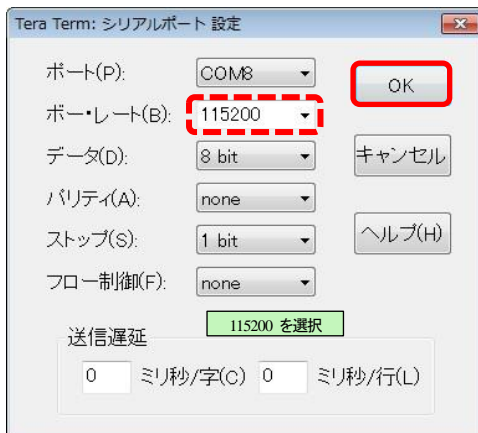
- (1) Windows® のスタート・メニューから「コントロールパネル」を選択し、「ネットワークと共有センター」をクリックします。
- (2) 「アダプタの設定の変更」を開いて、「ローカルエリア接続」を選択に右クリックして「プロパティ」を選択します。
- (3) プロパティ・ダイアログが表示されたら、「インターネットプロトコルバージョン 4 (TCP/IPv4)」を選択し、「プロパティ」ボタンをクリックします。
- (4) 「次の IP アドレスを使う」にチェックを入れ、IP アドレス と サブネット マスク に任意のアドレスを入力します。この例では、IP アドレス:192.168.1.31、サブネットマスク:255.255.255.0 に設定します。



【図 3-9.1】 PC 側ネットワークの設定

### 3-10.UART の接続確認と Linux の起動

- (1) Sodia と PC が UART ケーブルで接続されていることを確認します。
- (2) Sodia に接続されている COM ポートを選択し「ボー・レート」を 115200 に設定します。



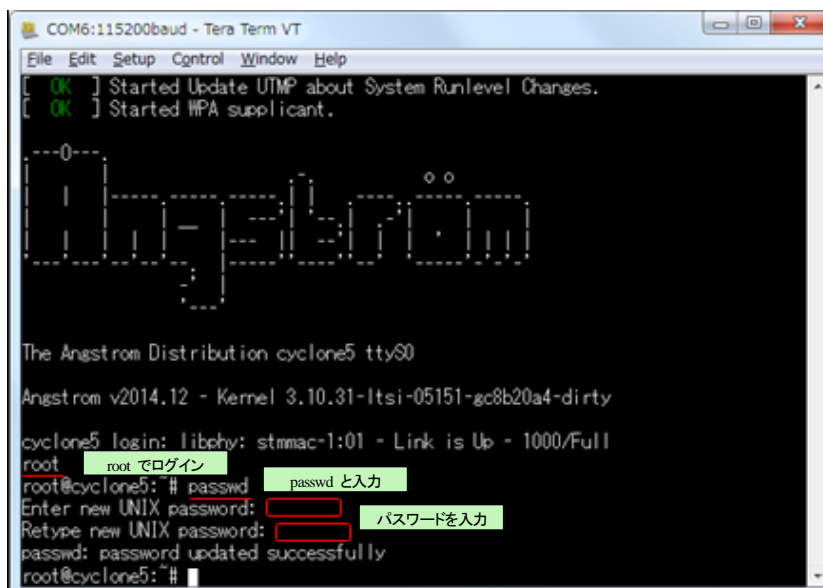
【図 3-10.1】 Tera Term でのシリアル・ポート設定

- (3) Sodia の WARM RESET ボタンを押下します。



【図 3-10.2】 Sodia の WARM RESET ボタンを押下

- (4) Linux コンソールに Linux の起動ログが表示されます。ログインするには、login プロンプトに root と入力します。
- (5) passwd と入力して root のパスワードを設定します。
- (6) 新規パスワードが要求された場合は、この例では altera.123 を入力した後、もう一度 altera.123 を入力します(入力の際、パスワードは表示されません)。



【図 3-10.3】 ログインとパスワードの設定

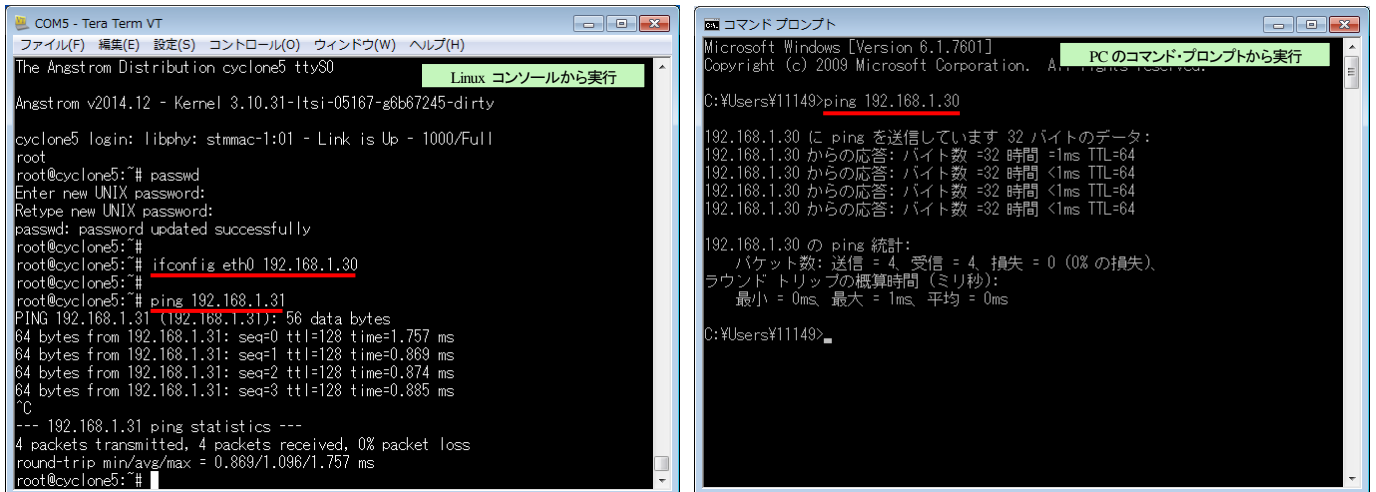
### 3-11.Ethernet の接続確認

Sodia と PC が Ethernet ケーブルで接続されていることを確認します。

- (1) Linux コンソールから 任意の IP アドレスを設定します。この例では、192.168.1.30 を設定しますので、以下のように入力してください。

```
# ifconfig eth0 192.168.1.30 ↓
```

- (2) 続いて、Linux および PC 双方から Ping を送り、正しく接続できていることを確認します。その際、どちらかの Ping が通らない場合は Windows® のファイアウォール設定を解除するなどしてください。



【図 3-11.1】 Ethernet の接続確認

## 4. 演習

### 4.1. 演習の概要

演習には、前述「3-6. 演習ファイルの抽出」で解凍した Altera-SoCFPGA-OpenCL-vectorAdd を使用します。単純な配列加算の処理を OpenCL™ で実装します。

ホスト・アプリケーションでは入力テーブルの確保と乱数の格納を行い、カーネル (FPGA ロジック) で演算された加算値が正しいかを比較します。

### 4.2. エンベデッド・コマンド・シェルの起動

後ほど実行する、「カーネルのコンパイル」と「ホスト・アプリケーションのコンパイル」は、このエンベデッド・コマンド・シェルからコマンド・ラインで実行します。

- (1) Windows® エクスプローラで、SoC FPGA エンベデッド・デザイン・スイートのインストール・フォルダ (この例では、C:\altera\15.1\embedded) を開き、その下にある **Embedded\_Command\_Shell.bat** をダブルクリックしてエンベデッド・コマンド・シェルを起動します。
- (2) エンベデッド・コマンド・シェルから 以下を実行します。

```
$ export ALTERAOCLSDKROOT="C:\altera\15.1\hld"
$ export PATH=${ALTERAOCLSDKROOT}/bin:$PATH
$ export AOCL_BOARD_PACKAGE_ROOT=${ALTERAOCLSDKROOT}/board/c5soc_sodia
```

- (3) 現在認識されているボードを確認します。

```
$ cd "C:\altera\15.1\hld\bin"
$ aoc --list-boards
Board list:
c5soc_sodia_sharedonly
```

### 4.3. カーネルのコンパイル

カーネルのコンパイルでは、バックグラウンドで Quartus® Prime が実行されますので、Quartus® Prime が起動中であれば終了させてください。

- (1) エンベデッド・コマンド・シェルから 以下のコマンドを入力します。

```
$ cd "C:\Work\Altera-SoCFPGA-OpenCL-vectorAdd\device"
$ aoc -v --report vectorAdd.cl -o vectorAdd.aocx --board c5soc_sodia_sharedonly
aoc: Environment checks are completed successfully.
You are now compiling the full flow!!
aoc: Selected target board c5soc_sodia_sharedonly
aoc: Running OpenCL parser...
c:/Work/Altera-SoCFPGA-OpenCL-vectorAdd/device/vectorAdd.cl:23:47: warning: declaring kernel argument with no 'restrict' may lead to low kernel performance
    __kernel void vectorAdd(__global const float *x,
                           ^
c:/Work/Altera-SoCFPGA-OpenCL-vectorAdd/device/vectorAdd.cl:24:47: warning: declaring kernel argument with no 'restrict' may lead to low kernel performance
    __global const float *y,
                           ^
2 warnings generated.
aoc: OpenCL parser completed successfully.
aoc: Compiling...
aoc: Linking with IP library ...

+-----+
; Estimated Resource Usage Summary
+-----+
; Resource                + Usage
+-----+
; Logic utilization        ; 19%
; Dedicated logic registers ; 7%
; Memory blocks            ; 10%
; DSP blocks                ; 0%
+-----+
aoc: First stage compilation completed successfully.
aoc: Hardware generation completed successfully.
```

- (2) ご使用の PC スペックにもよりますが、カーネルのコンパイルには 13 分程度かかります。エンベデッド・コマンド・シェルのプロンプトが表示されるまで待ちます。

- (3) コンパイルが成功すると、以下のように **vectorAdd.aocx** ファイルが生成されます。

```
$ ls -l
合計 2276
drwxrwxr-x+ 1 elsfae None      0 11月 15 15:57 vectorAdd
-rwxrwxr-x+ 1 elsfae None 141648 11月 15 15:41 vectorAdd.aoco
-rwxrwxr-x+ 1 elsfae None 2594656 11月 15 15:57 vectorAdd.aocx
-rw-rwxr--+ 1 elsfae None   1703 6月 30 2014 vectorAdd.cl
```



#### 4.4. ホスト・アプリケーションのコンパイル

- (1) エンベデッド・コマンド・シェルから 以下のコマンドを入力します。

```
$ cd "C:\Work\Altera-SocFPGA-OpenCL-vectorAdd"
$ make all
arm-linux-gnueabi-g++ host/src/main.cpp common/src/AOCL_Utils.cpp -o vector_add -IC:/altera/15.1/hld/host/include -Icommon/inc -LC:/altera/15.1/hld/board/c5soc_sodia/arm32/lib -LC:/altera/15.1/hld/host/arm32/lib -Wl,--no-as-needed -lalteracli -lalterahalmm -lalterammdpcie -lelf -lrt -ldl -lstdc++
```

- (2) コンパイルが成功すると、以下のように **vector\_add** ファイルが生成されます。

```
$ ls -l
合計 52
drwxrwxr-x+ 1 elsfae None    0 11月 15 10:18 common
drwxrwxr-x+ 1 elsfae None    0 11月 15 15:57 device
drwxrwxr-x+ 1 elsfae None    0 11月 15 10:18 host
-rw-rwxr--+ 1 elsfae None 1406 12月 10 2014 Makefile
-rwxrwxr-x+ 1 elsfae None 43296 11月 15 16:04 vector_add
```

#### 4.5. ファイルの転送

ホスト・アプリケーションとコンパイル済みのカーネルを Sodia の SD カードに転送します。

- (1) SCP(セキュア・コピー)を使用してファイルを転送することができます。

SCP を経由してホスト PC から Sodia の SD カードにファイルを転送するには、ホスト PC から

```
scp <source_filename> root@<board_ip_address>:<target_filename>
```

コマンドを使用します

- (2) エンベデッド・コマンド・シェルから 次のように入力し vector\_add ホスト・アプリケーションを Sodia に転送します。

接続を続けるかどうかを聞かれた場合は、**yes**  を入力します。この例ではパスワードは **altera.123**  を使用します。

```
$ scp vector_add root@192.168.1.30:/home/root 
Could not create directory '/home/11149/.ssh'.
The authenticity of host '192.168.1.30 (192.168.1.30)' can't be established.
ECDSA key fingerprint is SHA256:SzhkriFhnzX/arvfUwpBVVAG57AUhFXGLCSwRCnHAOE.
Are you sure you want to continue connecting (yes/no)? yes  
Failed to add the host to the list of known hosts (/home/11149/.ssh/known_hosts)
.
Password: _____ 
vector_add                               100% 42KB 42.3KB/s 00:00
```

- (3) 次のように入力し vectorAdd.aocx ファイルを Sodia に転送します。再度、パスワード **altera.123**  を使用します。

```
$ scp ./device/vectorAdd.aocx root@192.168.1.30:/home/root 
Could not create directory '/home/11149/.ssh'.
The authenticity of host '192.168.1.30 (192.168.1.30)' can't be established.
ECDSA key fingerprint is SHA256:SzhkriFhnzX/arvfUwpBVVAG57AUhFXGLCSwRCnHAOE.
Are you sure you want to continue connecting (yes/no)? yes  
Failed to add the host to the list of known hosts (/home/11149/.ssh/known_hosts)
.
Password: _____ 
vectorAdd.aocx                           100% 2534KB 2.5MB/s 00:01
```

これで、Sodia の SD カードにファイルが転送できました。

#### 4-6. OpenCL™ の実行

転送されたカーネルで FPGA をコンフィギュレーションし、ホスト・アプリケーションを実行することで OpenCL™ の動作を確認することができます。

- (1) Linux コンソールから 以下のように入力し、ターゲットの Linux 上で ドライバ・モジュールをロードします。

```
# source ./init_openc1.sh
```

- (2) 次のように入力して、ボードが正常に動作していることを確認します。

```
# aocl diagnose
aocl diagnose: Running diagnostic from /home/root/aocl-rte-15.1.0-1.arm32/board/c5soc/arm32/bin

Verified that the kernel mode driver is installed on the host machine.

Using platform: Altera SDK for OpenCL
Board vendor name: Altera Corporation
Board name: sodia_sharedonly : Cyclone V SoC Development Kit

Buffer read/write test passed.

DIAGNOSTIC_PASSED
```

- (3) 次のように入力して、転送したファイルを実行可能にします。

```
# chmod 777 vector_add
# chmod 777 vectorAdd.aocx
```

- (4) ターゲットの Linux 上で OpenCL™ カーネル・プログラムのプログラミングを行います。リコンフィギュレーションが成功すると、LED の点滅パターンが変化します。

```
# aocl program /dev/ac10 vectorAdd.aocx
aocl program: Running reprogram from /home/root/aocl-rte-15.1.0-1.arm32/board/c5soc/arm32/bin
Reprogramming was successful!
```

- (5) ターゲットの Linux 上で ホスト(ARM®)プログラムを実行します。

```
# ./vector_add
Initializing OpenCL
Platform: Altera SDK for OpenCL
Using 1 device(s)
  c5soc_sodia_sharedonly : Cyclone V SoC Development Kit
Using AOCX: vectorAdd.aocx
Reprogramming device with handle 1
Launching for device 0 (1000000 elements)

Time: 159.887 ms
Kernel time (device 0): 7.243 ms

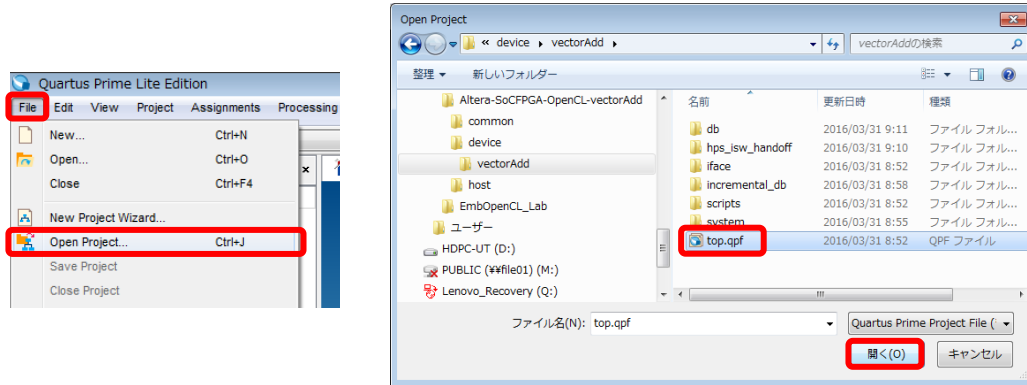
Verification: PASS
```

#### 4.7. カーネルのコンパイル結果の確認

コンパイルされたカーネルのデザインを確認します。

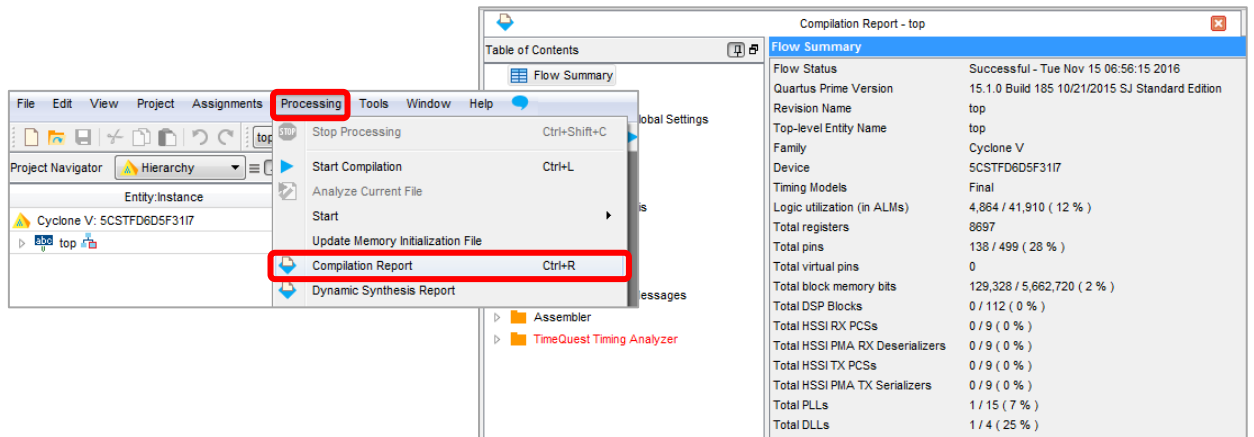
(1) Quartus® Prime を起動して、File メニュー ⇒ Open Project から、

C:\Work\Altera-SoCFPGA-OpenCL-vectorAdd\device\vectorAdd\top.qpf を開きます。



【図 4-7.1】 Quartus® Prime を起動して、File メニュー ⇒ Open Project

(2) Processing メニュー ⇒ Compile Report を選択します。



【図 4-7.2】 Processing メニュー ⇒ Compile Report

## 改版履歴

| Revision | 年月         | 概要 |
|----------|------------|----|
| 1        | 2017 年 1 月 | 初版 |

### 免責およびご利用上の注意

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

- 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。
- 本資料は予告なく変更することがあります。
- 本資料の作成には万全を期していますが、万一ご不明な点や誤り、記載漏れなどお気づきの点がありましたら、本資料を入手されました下記代理店までご一報いただければ幸いです。  
 株式会社アルティマ ホームページ: <http://www.altima.co.jp>      技術情報サイト アルティマ技術データベース: <https://www.altima.jp/members/index.cfm>  
 株式会社エルセナ ホームページ: <http://www.elsena.co.jp>      技術情報サイト ETS : <https://www.elsena.co.jp/elspear/members/index.cfm>
- 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。
- 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。