



copyright@2019 - 2020

IN



#### • 修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2019/10/9

※ この文書の情報は、文書を改善するため、事前の通知なく変更されることがあります。最新版は 弊社ホームページからご参照ください。「https://www.csun.co.jp」 ※ (株)日昇テクノロジーの書面による許可のない複製は、いかなる形態においても厳重に禁じられ ています。

ホームページ:<u>https://www.csun.co.jp</u>



株式会社日昇テクノロジー

# 目录

第一章 パッケージソフトの説明と開発ボードのテスト	7
1.1 パッケージソフトの説明	7
1.2 開発ボードテスト	7
1.1.2 テストに必要なもの	7
1.2.2 開発ボードのケーブル接続	10
1.2.3 テストする	10
第二章 ZYNQの紹介	15
2.1 PS と PL の相互連結技術	15
2.2 ZYNQ チップの開発流れの紹介	22
2.3 ZYNQ の勉強はどんなスキルが需要	24
第三章 Vivado 開発環境	
3.1 Vivado ソフトの紹介	25
3.2 Vivado ソフトバージョン	25
3.3 Vivado ソフトの Windows でのインストール	25
第四章 PLの"Hello World"LEDテスト	
4.1 LED ハードウェアの紹介	
4.2 Vivado プロジェクトを作成	
4.3 Verilog HDL ファイルを作成し、LED を点灯する	38
4.4 ピン制約を追加	42
4.5 シーケンス制約を追加 <b>.</b>	46
4.6 BIT ファイルを生成する	50
4.7 ダウンロードとデバッグ <mark>.</mark>	52
第五章 HDMI 輸出実験	
5.1 ハードウェアの紹介	55
5.2 Vivado プロジェクトを <mark>作</mark> 成する	55
5.2.1 HDMI コンパイラー IP コアを追加する	55
5.2.2 ピクセルクロックの PLL モジュールを追加する	60
5.2.3 カラーバー発生モジュールを追加する	62
5.2.4 トップモジュールを追加する	63
5.3 XDC 制約ファイルを追加する	64
5.4 ダウンロードとデバッグ	66
5.5 実験のまとめ	67
第六章 ARM を体験	
6.1 パードウェアの紹介	68
6.2 Vivado プロジェクトを作成	68
6.2.1 UART 配置	72
6.2.2 クロック配置	72
6.2.3 DDR3 配置	73
6.3 SDK デバッグ	80
6.4 実験のまとめ	
6.5 よくある問題	93
第七章 PS で PL の LED を点灯する	95
7.1 Vivado プロジェクトを作成する	



7.1.2 DDR3 配置	
7.1.3 AXI GPIO を追加する	
7.2 XDC ファイルで PL ピンを制約する	105
7.3 SDK プログラムをコンパイルする	106
7.4 ダウンロード及びデバッグ	110
7.5 実験のまとめ	112
第八章 PS タイマーインタラプト実験	113
8.1 Vivado プロジェクトを作成する	113
8.2 SDK をプログラミング	114
8.3 ダウンロードとデバッグ	118
8.4 実験纏め	119
第九章 PL キーインタラプト	
9.1 Vivado プロジェクトを作成する	120
9.2 ダンロードとデバッグ	124
9.3 実験まとめ	130
第十章 イーサネット実験(LWIP)	131
10.1 Vivado プロジェクトを作成	131
10. 1. 1 PS 側のイーサネット配置	
10.2 SDK プログラム	133
10.2.1 LWIP テンプレートに基づく APP を作成	133
10.3 ダウンロードとデバッグ	133
10.3.1 イーサネットテスト	133
10.4 実験のまとめ	136
第十一章 ユーザー定義 IP テスト	137
11.1 PWM 紹介	137
11.2 Vivado プロジェクトの作成 <mark></mark>	139
11.2.1 一つの vivado プロジェクトを作成	139
11.2.2 ユーザー定義 IP を作成する。	139
11.3 SDK ソフトのプログラミングとデバッグ	152
11.4 実験結果	161
11.5 よくある問題	161
11.5.1 AXI IPのペースアドレスを調べる	
第十二章 VDMA を使用して HDMI ディスプレイを駆動する	162
12.1 Vivado プ <mark>ロ</mark> ジェクトの設立	162
12. 1.1 UART のコンフィグ	165
12.1.2 12C EMIO のコンフィグ	165
12.1.3 DDR3 のコンフィグ	166
12.1.4 コンフィグ割り込み	
12.1.5 VDMA のコンフィク	
12.1.6 カスタム IP を追加する	
2. ./ HDMI エンコーターを追加する	
IZ.Z SDK ソフトウェアの作成とナハック	
第十二草 フロクラムの復帰	185
13.1 Vivado ブロジェクトの設立	185
13.2 FSBL を生成する	
IJ.J DUUI ノアイルを作成する	
I3.4 3U フートの起動ナムト	
19.9 YOFI アヘア開炉	



13.7 バッチファイルを使用して QSPI をすばやく書き込む
第十四章 仮想マシンと Ubuntu システムをインストールする
14.1 仮想マシンソフトウェアのインストール
14.2 Ubuntu のインストール
14.2.1 システムのインストール
14.2.2 ソフトウェアソースサーバーを変更する
14.2.3 bash をデフォルトの sh に設定する
14.2.4 画面ロック時間を設定する212
14.3 よくある問題
14.3.1 仮想マシンには仮想化サポートが必要である。
第十五章 Ubuntu で Linux バージョンの Vivado ソフトウェアをインストールする
15.1 Linux バージョンの Vivado をインストールする
15.2 許可設定
15.3 ダウンローダードライバーをインストールする
15.4 Vivado をテストする
15.5 よくある問題
15.5.1 Linux ダウンローダーのダウンロード時にプロンプトが表示される
15.5.2 ZYNQ に合うクロスコンパイラ
第十六章 Petalinux ツールのインストール
16.1 Petalinuxの概要
16.2 インストールに必要なライブラリ
16.3 Petalinux をインストールする
第十七章 NFS サービスソフトウェアのインストール
$17 \text{ 1 NFS } + -i - 54 + 22^{-1}$
$17.2 \text{ NFS } \hat{s} - \hat{z} - \hat{z}$
17.3 よくある問題 229
17.3.1 NFS マウントできない
第十八章 Petalinux で Linux システムをカスタマイズする
18.1 Vivado プロジェクト
18.2 Petalinux でプロジェクトを作成する
18.3 Linux カーネルをコンフィグする
18.4 ルートファイルシステムのコンフィグ
18.5 コンパイルする
18.6 BOOT ファイルが生成される
18.7 Linux をテストする
18.8 よくある問題
18.8.1 Bad FIT kernel image format!が表示され、カーネルが起動できない。…244
18.8.2 ファイルとコンフィグが保存できない
第十九章 SDK で Linux プログラムを開発する245
19.1 SDK を使って Linux アプリケーションを作成する
19.2 NFS 共有を実行する
19.3 TCF-Agent を介してデバッグを実行する250
19.4 TCF-Agent 問題253
第二十章 Linux 環境で GPIO 実験254
20.1 SHELL コントロールを使用する
20.2 C 言語を使ってコントロールする
20.2.1 GPIOのコードの確認255



20.2.2 物理 GPIO との関係の確認	
20.3 実験のまとめ	
第二十一章 Petalinux での HDMI ディスプレイ	259
21.1 Petalinux のコンフィグ	
21.2 Linux カーネルをコンフィグする	
21.3 デバイスツリーを変更する	
21.4 テスト petalinux プロジェクトのコンパイル	
21.5 よくある問題	
21.5.1 システムのスリープを防ぐ方法	
第二十二章 Debian デスクトップシステムの使用	270
22.1 Petalinux のコンフィグ	
22.2 linux カーネルをコンフィグする	
22.2.1 USB WIFI モジュールドライバをコンフィグする	
22.2.2 USB カメラドライバをコンフィグする	
22.3 Petalinux プロジェクトのコンパイルとテスト	
22.4 SD カードファイルシステムを作成する	
22.4.1 SD カードのパーティションを変更する。	
22. 4. 2 ルートファイルシステムを SD カード EXT4 パーティ	ションに同期する。 279
第二十三章 QSPI Flash から起動の Linux の作成	
23.1 Petalinux プロジェクトをコピーする	
23.2 Petalinux のコンパイルとコンフィダ	



第一章 パッケージソフトの説明と開発ボードのテスト

パッケージソフトウェアに主に含むものを紹介する。

## 1.1 パッケージソフトの説明

- 1) CP210x\_Windows\_Drivers.zip シリアルドライバー
- 2) Xilinx\_Vivado\_SDK\_2017.4\_1216\_1.tar.gz Vivado 2017.4 インストールパッケージ

これはWindows と Linux が通用するバージョン。Windows を使う場合、解凍ソフトが必要になる

- 3) petalinux-v2017.4-final-installer.run petalinux インストールパッケージ
- 4) qt-opensource-windows-x86-mingw530-5.7.1.exe Windows バージョンQt
- 5) qt-opensource-linux-x64-5.7.1.run Linux バージョンQ t
- 6) imageUSB.exe ミラーリカバリーツール
- 7) 00\_resource 中にLinuxのソースコード、ルートファイルシステムが含まれている
- 8) VMware-workstation-full-12.1.1-3770994.exe バーチャルマシンインストールパッケージ
- 9) ubuntu-16.04.3-desktop-amd64.iso Ubuntuのインストールパッケージ、PC だけにインストール

できる。開発ボードにはインストールできない

## 1.2 開発ボードテスト

開発ボードを手に入れ、正常に働けるかどうかをテストする。これから、簡単に行えるテストを紹介 する。

1.1.2 テストに必要なもの

1) コンピューター



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



 解像度は1920×1080より高い、HDMIをサポートするモニターをもう一台を用意してください。
 電源なしで HDMI から VGA コンバーターを変換することは開発ボードがサポートできないので、給電が 独立なコンバーター需要である。



3)HDMI ケーブルーつ





4) USB ポートのマウスとキーボード



5) ルーター

ネットワークをテストするため、インターネットを繋いたほうがいい。DHCP をサポートする。





- 1.2.2 開発ボードのケーブル接続
- 1) HDMI モニターと接続
- 2) LAN ポートはルーターと接続する
- 3) 電源を接続



## 1.2.3 テストする

1) テスト前、USB からシリアルポートへの変換のドライバソフト(ソフト/CP210x\_Windows\_ Driver\_zip)をインストールください。そうしなければ、シリアル通信テストができない。インストー ル完了後、USB ケープルを使い、コンピューターの USB ポートと開発ボードの UART ポート (J1)をコ ネクトしてください。これから、コンピューターのデバイスマネージャを開き。そこでシリアル設備 CP210x がみつけられる。サンプルでは COM3。





2) ターミナルツールはたくさんある。例えば、putty、tetaterm、Windows内部のターミナルツール と SecureCRT などである。ここでは putty を利用する。

3) Serial と Serial line で COM3 を選んで、スビートは 115200 を書いてください。COM3 のシリア ルナンバーはデバイスマネージャで表示されたものと同じく書いて、"Open"をクリックしてください。

😰 PuTTY Configuration	
Category:	
	Basic options for your PuTTY session
	Specify the destination you want to connect to
Kenhoard	Serial line Speed
	COM3 115200
Features	Connection type:
E-Window	🔘 Raw 🔘 Telnet 🔘 Rlogin 🔘 SSH 🛛 💿 Serial
Behaviour	Load, save or delete a stored session
- Translation	Saved Sessions
Selection	
Colours	Default Settings Load
Proxy	Save
Telnet	Delete
Rlogin	
Senal	Close window on exit:
	Always Thever O Only on clean exit
About	Open Cancel

4) 開発ボードの起動モードは SD モードかどうかを確認する(出荷時、デフォルトで開発ボードの カードスロットにカードがある、スタートモードも SD モードがデフォルト値になっている)、ジャンパ ーでスタートモードを変更できる。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



5) 開発ボードの電源スイッチを入れて、puttyの画面ではu-bootとLinuxシステムのスタートメッ

セージが表示される。

🚱 COM3 - PuTTY
U-Boot 2017.01 (Mar 22 2018 - 20:06:16 +0800)
Model: Zvng ALINX AX7015 Development Board
Board: Xilinx Zynq
I2C: ready DRAM: ECC disabled 1 GiB
MMC: sdhci_transfer_data: Error_detected in status(0x208000)!
SF: Detected w25q256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB
*** Warning - bad CRC, using default environment
In: serial
Out: serial Err: serial
Net: ZYNQ GEM: e000b000, phyaddr 1, interface rgmii-id
U-BOOT for ax_peta
ethernet@e000b000 Waiting for PHY auto negotiation to complete
continences of a for the about registration to complete
· · · · · · · · · · · · · · · · · · ·

6) シリアルターミナルでシステムに登録できる。ユーザー:root、パスワード:root

大勢な人は putty を使用するのが初めて、あるいは、初めてシリアルを使うのである。説明しな ければならないのは、putty でコマンドの入力はホストキーボードで実現する。開発ボードと接続して いるキーボードではない。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



7) 起動した後、開発ボードと接続している HDMI モニターは Debian のデスクトップが表示される。 パネルを選択するの画面が出る場合、デフォルトパネルを選んでいい。





 ここに来て、もうマウスとキーボードを使って、操作出来る。マウスで Web ブラウ ザーをダブルクリックしてください。ブラウザーの起動は時間がかかりますので、し ならくお待ちください。



9) アドレス欄で URL を入力してください。正常に開けば、開発ボードはもう普通に使える。



10) 開発ボードの簡易テストはここで終了する。



## 第二章 ZYNQの紹介

Zynq シリーズの特長は FPGA にある完璧な ARM サブシステム (PS) である。各 Zynq システムのプ ロセッサーは Cortex-A9 プロセッサーが含んでいて、プロセッサーの構造は全部これを中心にしてい る。そして、サブシステムにメモリーコントローラと大量なペリフェラル統合されていて、Cortex-A9 のコアが Zynq-7000 内でプログラマブルロジックユニットから独立できるようになった。つまり、し ばらくの間で、プログラマブルロジックユニット (PL) を使っていなければ、ARM プロセッサーのサ ブシステムも独立で働ける。これは以前の FPGA と本質的な違いがあり、プロセッサーを中心にして います。

Zynq は PS 部分と PL 部分、二つの機能ブロックである。はっきり言うと、ARM の SOC 部分と、FPGA 部分である。その中に、PS は二つの ARM Cortex<sup>™</sup>-A9 プロセッサー、AMBA®の相互連結、内部メモリ、 外部メモリとペリフェラルを統合している。ペリフェラルの方は主に USB バスインターフェス、イー サネットインタフェース、SD/SDIO インタフェース、I2C バスインターフェス、CAN インタフェース、 UART インタフェースと GPIO などを含んでいる。



PS: プロセッシングシステ (Processing System)、FPGA と無関連の ARM の SOC 部分。 PL: プログラマブルロジック、FPGA 部分。

#### 2.1 PS と PL の相互連結技術

ZYNQ は高性能 ARM Cortex-A9 シリーズと高性能 FPGA をシングルチップで厳密に結びつける製品 である。AMR プロセッサーと FPGA との高速通信とデータ交換を実現し、両方の性能優位を発揮する ため、効率が高いチップ内高性能プロセッサーと FPGA の連絡通路が需要である。そのため、効率が 高い PL と PS のだデータ交換通路を設計することは ZYNQ チップの設計で最も重要なことと同時に、



製品の設計が成功できるかどうかに関わっている。この節、弊社が紹介する内容は主に PS と PL の連結で、ユーザーに連結技術を了解させる。

実は、具体的な設計には、連結での工夫はそんなに多くない。IP コアを加わったあと、システムは自動的に AXI インタフェースを使って、弊社の IP コアとプロセッサーを連結する。あとは少し 補充すればいい。

AXI のフルネームは Advanced eXtensible Interface で、Xilinx がシリーズ 6 の FPGA から導入するインタフェースプロトコルである。主にメインデバイスとサブデバイスのデータエントリー方法を説明した。ZYNQ で使っているバージョンは AXI4 であるから、AIX4.0 と ZYNQ の内部デバイスにAXI インタフェースがあることをよく見られる。実際に、AXI は ARM 会社が提出していたAMBA (Advanced Microcontroller Aechitecture)の一部で、高性能、高帯域幅とローディレイのチップインタナルバスである。これも前の AHB と APB バスの代わりになっている。初代パッジョンAXI (AXI3) は 2003 でリリースされた AMBA3.0 に、二代の AXi (AXI4) も 2010 年リリースされた AMBA4.0 に含まれてる。

AXI プロトコルは主にメインデバイスとサブデバイスのデータエントリー方法を説明しました。 両者はハンドシェーキングシグナルを利用して連結する。サブデバイスがデータの受け入れ準備を出 来た時、READY 信号を出す。メインデバイスのデータが用意した時、VALID を維持する信号を出して、 データ有効を示している。VALID と READY 信号両方も有効の時だけ、データは伝送を始まる。この二 つも信号が有効のままでいたら、メインデバイスは次のデータを伝送する。両方のどちらが信号を取 り消したら、伝送は停止になる。AXI のプロトコルは図に表したように、T2 のときは、サブデバイス の READY 信号が有効となり、同じく T3 のときはメインデバイスの VALID 信号が有効となる。データ 伝送も始まる。



ZYNQ の中で、AXI-Lite、AXI4 と AXI-Stream 三つのバスをサポート出来る。表 5-1 から、この三 つの AXI インタフェースの特性が見られる。

インタフェース	特性	使用状況
プロトコル		

ホームページ: <u>https://www.csun.co.jp</u>



AXI-Lite	アドレス/シングルデータを転送	低速ペリフェラルやコントロール
AXI4	アドレス/バーストデータを伝送	アドレスのロット伝送
AXI-Stream	データ伝送とバースト転送	データフローとメディア伝送

AXI4-Lite:

軽量級、構造簡単の特徴があって、小ロットデータや簡単なコントロールの状況に適切である。 ロット伝送はサポートしない。読み込んと書き込みをするとき、一回で文字ひとつしかリードライト できない。主に一部の低速ペリフェラルを訪問とペリフェラルのコントロールに使う。

AXI4 :

インタフェースは AXI-Lite とだいたい同じである。ただ一つの機能、ロット伝送を増えた。この機能は連続でひとつのエリアのアドレスを一気にリードライトができる。つまり、データリードライトの burst 機能。

以上の二つはメモリマッピングでコントロールしてる。すなわち、ARM はユーザーデファイン IP をあるアドレスに編入して訪問する。リードライトの時もチップ内の RAM にしているように、プログ ラミングも便利で、開発も難しくない。それを代わりに、資源を取り過ぎで、他のアドレス読み出し コード、アドレス書き込みコード、データ読み出しコード、データ書き出しコード、リプライ書き込 みコードなどの信号線が需要である。

AXI4-Steam :

これはコンティニュアスストリームインタフェースの一種で、アドレスコードが必要ない(FIFO に似ていて、ひたすらに読み出しあるいは書き込みすればいい)。このタイプの IP を対象にして、ARM は以上のようにメモリマッピングでコントロールできない(FIFO はアドレスのコンセプトがない)、 コンバージョンデバイスが欠かせない。例えば、AXI-DMA モジュールでメモリマッピングからストリ ームインタフェースに変換を実現する。AXI-Stream が適用の場合は多い:ビデオストリームの処理; コミュニケーションプロトコルの変換;デジタルー眼レフカメラシグナルの処理;無線通信あどで使 います。本質ではニューメリカルフローにあって、構築されたデータ通路で、ソース(例えば ARM メ モリ、DMA、無線レシーバーフロントエンドなど)からシンク(例えば HDMI モニター、高速 AD サウン ド出力など)まで連続データストリームをつくった。こんなインタフェースは実時間信号処理に向い ている。

AXI4 と AXI4-Lite インタフェースは五つの通路を含んでいる:

- Read Address Channel
- ➢ Write Address Channel



- ≻ Read Data Channel
- ≻ Write Data Channel
- ➢ Write Response Channl

中に、各通路は独立の AXI ハンドシェーキングプロトコル。下の図はそれぞれ読み出し と書き込みの模型を表している:



ZYNQ チップ内部はハードウェアで AXI-バスプロトコルを実現した。9 つの物理インタフェースがあって、それぞれは AXI-GP0~AXI-GP3、AXI-HP0~AXI-HP3, AXI-ACP インタフェースである。

AXI\_ACP インタフェースは、ARM マルチコアアーキテクチャの一つのインタフェース、DMA のよう なバッファが付けていない AXI ペリフェラルを管理する。PS ソケットは Slave インタフェースであ る。

AXI\_HP インタフェースは、高性能/帯域幅の AXI3.0 標準のインタフェースである。全部は四つあって、PL モジュールをメインデバイスに連結する。主に PL が PS (DDR と On-Chip RAM) のメモリを アクセスするときに使用する。

AXI\_GP インタフェースは、汎用な AXI インタフェースである。全部で四つあって、32 ビットのメ インデバイスのインタフェースと 32 ビットサブデバイスのインタフェースが 2 つずつある。





ここで見られるのは、二つの AXI-GP だけは master Port、即ちメインフレームインタフェース。 残るの 7 つは Slave Port (サブフレームインタフェース)。メインフレームインタフェースはリー ドライトを起こす権限があって、ARM は二つの AXI-GP メインフレームインタフェースっを利用して、 自発的に PL ロジックを訪問できる。PL をあるアドレスにマッピングしてると、PL のレジスタをリー ドライトする時は自分のレジスタをリードライトしているようで。他のサブフレームインタフェース は被動インタフェースで、PL からのリードライトを受ける。

また、この9つのAXIインタフェースの性能も違う。GPインタフェースは低性能インタフェース、理論的な帯域幅は600MB/sである。HPとACPの方は64ビット高性能インタフェースで、理論上で帯域幅は1200MB/s。ここで疑問を持っている人もいるだろう、なんで高性能インタフェースはメインフレームインタフェースにしないのだろう?こうすれば、ARMで高速データ伝送を発起するのも可能になる。答えは、高性能インタフェースはARMCPUがデータ伝送をする必要がない。本当の運ぶ屋はPLにあるDMAコントローラーである。

PC ソケットに位置する ARM は AXI インタフェースをサポートえできるハードウェアがある。PL はロジックを使って、相応の AXI プロトコルが実現する。Xininx は Vivada の開発環境に元々あった IP を提供する。例えば、AXI-DMA、AXI-GPIO、AXI-Dataover、AXI-Steam はそれぞれのインタフェー スを実現した後、使用の時は直接で Vivado の IP リストから添加すれば、相応の機能が実現できる。 下の図は Vivado にある DMA IP:





次はよく使われる AXI インタフェース IP の機能紹介:

AXI-DMA: PSメモリから PL 高速伝送通路 AXI-HP<---->AXI-Streamの変換を実現する。

AXI-FIF0-MM25: PS メモリから PL 汎用伝送通路 AXI-GP<---->AXI-Streamの変換を実現する。

AXI-Datamover : PS メモリから PL 高速伝送エクスプレス通路 AXI-HP<---->AXI-Stream の変換を 実現する。ただし、これは完全に PL にコントロールされて、PS はかんぜん被動的である。

AXI-VDMA: PS メモリから PL 高速伝送エクスプレス通路 AXI-HP<---->AXI-Stream の変換を実現 する。ただし、ビデオ動画、画像などの二次元データにだけに応用できる。

AXI-CDMA:この仕事はデータをメモリのある場所から別の場所へ移動することである。PL で完成して、CPU が動き出す必要がない。

これらの IP をどう使うかについて、後の章で例を挙げ、説明する。時々ユーザー定義の IP を開発して PS と通信する必要がある。この場合はウィーザー生成を利用し、それ相応の IP を<u>る</u>。ユーザ 一定義の IP コアは AXI4-Lite、AXI4、AXI-Steam、PLB と FSL などのインタフェースを手に入れる。 PLB と FSL は ARM がサポートできないため、使わない。

以上のオフィシャル IP とウィーザー生成のユーザー定義 IP があれば、ユーザーは AXI シーケン スに詳しく知る必要がある(問題が出る以外)。Xilinx は既に AXI シーケンスと関係あるディテール をパケージングしたから、ユーザーは自分のロジックを実現できればいい。

厳しくいえば、AXI プロトコルは PTP のマスター・スレーブインタフェースプロトコルである。 いくつかのペリフェラルが互いにデータ交換を必要する時、AXI Interconnect モジュールに加入す る需要がある、いわゆる相互関連マトリクス。これの作用は一つあるいは多数の AXI メインデバイス と一つあるいは多数の AXI サブデバイスを連結する交換メカニズム(チェンジャーにある交換マトリ ックスに似ている)。

この AXI Interconnect コアは最大限でメインデバイス 16 個、サブデバイス 16 個をサーポート できる。これ以上のインタフェースを必要する場合、IP コアを増えればいい。



- AXI interconnect の基本的連結モードは以下の4種類ある:
  - > N-to-1 Interconnect
  - $\succ$  to-N Interconnect
  - > N-to-M Interconnect (Crossbar Mode)
  - > N-to-M Interconnect (Shared Access Mod)



多対多リードライトアドレスチャネル







多対多リードライトデータチャネル

ZYNQ 内部の AXI インタフェースデバイスは相互関連マトリクスの方法で繋がっている。データ 伝送の高効率を確保した同時に、連結の柔軟性も保証できる。Xilinx は Vivado でこの相互関連マト リクスを実現する IP コア axi\_interconnet を提供したので、必要の時はコールするだけでいい。

Search: Qr axi inter 🕓 (6 matches)
<ul> <li>➡ AHB-Lite to AXI Bridge</li> <li>➡ AXI4-Stream Interconnect</li> <li>➡ AXI AHBLite Bridge</li> <li>➡ AXI ABB Bridge</li> </ul>
AXI Interrupt Lontroller
ENTER to select, ESC to cancel, Ctrl+Q for IP details AXI Interconnect IP
ENTER to select, ESC to cancel, Ctrl+Q for IP details AXI Interconnect IP

## 2.2 ZYNQ チップの開発流れの紹介

ZYNQ は CPU と FGPA を統合したため、開発エンジニアは ARM の OS アプリケーションとデバイス ドライバーだけてなく、FPGA 部分のハードウェアロジックをプログラミングする。開発中は LinuxOS とシステムの構造を理解し、FPGA と ARM システムのハードウェプログラムプラットフォームを構築 するのも需要である。というわけで、ZYNQ の開発はソフトエンジニアがハードウェアエンジニアと 協同して、プログラミングと開発することである。即ち、ZYNQ 開発中が言っていた "HW-SW Co-design" である。

ZYNQ システムのハードシステムとソフトシステムをプログラミング、開発するに必要な開発環



### 境とデバッグツール: Xilinx Vivado

Vivado デザインスーツは FPGA 一部のプログラミングと開発、ピンとシーケンスの制限、コンパ イルとシミュレーション、そして RTL からビットストリームのプログラミング流れを実現する。 Vivado は ISE デザインスーツを簡単にアップグレードしたものではなく、新たなデザインスーツで ある。Vivado が ISE デザインスーツにある全ての重要ツールにとって代わった。例えば、Project Navigator、Xilinx Synthesis Technology、Implementation、CORE Generator、Constraint、Simulator、 Chipscope Analyzer、FPGA Editor などのデザインツールなどである。

XiliInx SDK (Software Development Kit)、SDK は XiliInx ソフトウェア開発スーツ (SDK) で ある。VIvado ハードウェアシステムのもとで、システムは自動的に一部の重要なパラメータを配置 する。中にツールとベースパス、コンパイラーオプション、JTAG とフラッシュメモリ設定、デバッ ガー連結とベアメタルボードサポートパケージ (BSP) が含まれている。サポートしている全ての XilinxIP ハードコアにも、SDK はドライバープログラムを提供している。SDK は IP ハードコア (FPGA) とプロセッサーソフトウェアの協同デバッグをサポートしている。高級 C 又は C++言語を使って、AEM と FGPA システムを開発とデバッグすることができる。こうしてハードウェアシステムが順序に働い ているかをテストする。SDK ソフトも Vivado 内部にあるもので、別々でインストールの必要がない。

ZYNQ に開発のハードウェア先で、ソフトウェアが後にする方法である。具体的な流れはこうなる:

1) Vivadoに新規プロジェクトを作成し、組込み型のソースファイルを一つ増加する。

2) Vivado に PS と PL の一部基本的なペリフェラルを添加、配置しする。あるいは、ユーザー 定義のペリフェラルを配置する。

3) Vivado にトトップファイル HDL を生成し、制約ファイルを加わる。そして、ビットストリ ームファイル (\*. bit) をコンパイルする。

4) ハードウェアメッセージを SDK ソフト開発環境に書き出する。SDK 環境にデバッグソフト を数個プログラミングして、これらのソフトでハードウェアとソフトウェアを検証したあと、ビッ トストリームファイルと合わせて ZYNQ システムを単独にデバッグする。

5) SDK に FSBL ファイルを生成する。

6) VNware バーチャルマシンに u-boot.elf、bootloader のカーネルミラーイメージを生成する。

7) SDK に FSBL ファイル、ビットストリームファイル system.bit と u-boot.elf ファイルで BOOT.bin ファイルを生成する。

8) VMware に Ubuntu のカーネルミラーファイル Zimage と Ubuntu のルートファイルシステム を生成する。他にも、FPGA のユーザー定義 IP を対象にドライバーをプログラミングする必要 がある。



9) BOOT、カーネル、デバイスツリー、ルートファイルシステを SD カードに書き込み、開発 ボードの電源を入れたら、LinuxOS は SD カードから作動する。

以上は典型的な ZYNQ 開発ボード流れである。でも、ZYNQ も単純に ARM として使われる。こうす れば、PL 側の資源が必要なくなり、伝統的な ARM 開発とそんなに区別を付けていない。ZYRQ も PL 部分だけを使うが、PL の配置はやはり PS で完成する。つまり、PL だけ必要とするファームを固体化 するには、伝統的な Flash の固体化方法で実現できない。

### 2.3 ZYNQ の勉強はどんなスキルが需要

ZYNQ の勉強は FPGA、MCU、ARM などの従来のツールよりレベルが高い、ZYNQ を使いこなすのもすぐ 達成できることではない。

2.3.1 ソフト開発エンジニア

- ✓ コンピューター基本知識
- ✔C、C++言語
- ✓コンピューターOS
- ✓ tcl スクリプト

✓ 優れな英語を閲覧する基礎

#### 2.3.2 ロジック開発エンジニア

- ✓コンピューター基本知識
- ✓C言語
- ✓ デジタル回路基礎
- ✓ Verilog、VHDL 言語
- √ 優れな英語を閲覧する基礎



#### 第三章 Vivado 開発環境

## 3.1 Vivado ソフトの紹介

Vivado は Xilinx 会社が 2012 年発表された新世代インテグレーションデザイン環境である。Vivado Design Suite User Guide の Getting Started (UG910) にこう書いていた: プログラマの効率を上 がる為、Vivado を出すことにした。このソフトは大幅に Xilinx の 28nm工芸のプログラマブルロジ ックデバイスのデザイン、総合と実現効率を上げる事ができる。予想できるのは、FPGA が 28nm時代 に入り、ISE ツールは時代から外されているように見える。ハードウェアがレベルアップしたのに、 ソフトウェアがそのままだと、デザイン効率は必ず影響を受ける。

3.2 Vivado ソフトバージョン

ZYNQ 開発ボードの全てのサンプルと教程は Vivado2017.4 の開発環境で完成するので、ソフト バージョンで解明できない問題を避けるよう、勉強中はこちらと一致してください。使用する前に Vivado 2017.4をインストールする必要ある。Xilinxのオフィシャルサイトからダウンロードできる。 オフィシャルサイトのダウンロードはアカウント登録が必要である。

Vivado ソフトの Xilinx オフィシャルダウンロードサイト:

https://japan.xilinx.com/support/download.html

VivadoのLinux バージョンとWindows バージョン、それとツーワンタイプのも提供している。 ここで使うのはツーワンタイプで、Windows とLinux 両方の開発を満足できる。Vivado を使用するに は 0S は 64 ビットになければならない。

## 3.3 Vivado ソフトの Windows でのインストール

1) Vivado ソフトの圧缩ファイルダウンロードして、解凍する。xsetup.exe をクリックし、イン ストール開始。

0	api-ms-win-crt-environment-l1- 1-0.dll 10.0.10240.16384		api-ms-win-crt-filesystem-l1-1-0 .dll 10.0.10240.16384	03	api-ms-win-crt-heap-l1-1-0.dll 10.0.10240.16384 ApiSet Stub DLL
Q	api-ms-win-crt-private-l1-1-0.dll 10.0.10240.16384 ApiSet Stub DLL	Q	api-ms-win-crt-process-l1-1-0.d II 10.0.10240.16384		api-ms-win-crt-runtime-l1-1-0.dl l 10.0.10240.16384
0	api-ms-win-crt-utility-l1-1-0.dll 10.0.10240.16384 ApiSet Stub DLL	(C)	concrt140.dll 14.0.23506.0 Microsoft® Concurrency Runti	0	<b>msvcp140.dll</b> 14.0.23506.0 Microsoft® C Runtime Library
	xsetup 文件 2.55 KB	8	xsetup.exe 2017/12/21 20:15 434 KB		



- 2) バージョンアップを提示されたら、それを無視して、"continue"をクリックする。
- 3) "next"をクリックする、Vivadoからのシステムに対する要求が見える。

Vivado 2017.4 Installer	- Welcome
λ.	Welcome
VIVADO.	
HLx Editions	We are glad you've chosen Milinx as your platform development partner. This program can install the Wivado Design Environment, Software Development Kit and Documentation Navigator.
	Supported operating systems for Vivado 2017.4 are:
	- Windows 7.1: 64-bit
	- Windows 10.0 Creators Update: 64-bit
	- Red Hat Enterprise Linux 6.6-6.9: 64-bit
	- Red Hat Enterprise Linux 7.2-7.3: 64-bit
	- CentOS Linux 6.6-6.9: 64-bit
	- CentOS Linux 7.2-7.3: 64-bit
	- SUSE Enterprise Linux 11.4: 64-bit
	- SUSE Enterprise Linux 12.2: 64-bit
	- Ubuntu Linux 18.04.2 LIS: 64-bit - Additional library installation required
	Note: This release requires upgrading your license server tools to the Flex 11.14.1 versions. Flease confirm with your license admin that the correct version of the license server tools are installed and available, before running the tools.
	Note: 32-bit machine support is now only available through HW Server standalone product installers
	Note: This installation program will not install cable drivers on Linux. This item will need to be installed separately, with administrative privileges.
	To reduce installation time, we recommend that you disable any anti-virus software before continuing.
EXILINX ALL PROGRAMMABLE.	
Copyright © 1986-2017 Xilinx,	Inc. All rights reserved. Preferences (Back Next > Cancel

4) "I Agree"をクリックして、各条項を受ける。



5) "Vivado HL Design Edition"を選択する 。





6) デフォルト配置を使用して、"next"をクリックする。



7) ここではインストールパスを変えていない。インストールパスには日本語、スペース などの特殊文字を入れないようにする。コンピューターのユーザーネームも日本語、スペースを含ま ないようにする。Vivado はハードディスクへの大きさ要求はおよそ 33G である。



8)

# 株式会社日昇テクノロジー

低価格、高品質が不可能? 日昇テクノロジーなら可能にする



9) インストールを待つ時間がながく、ウィルス防止ソフトを止めなかったら、インストール中ブロックされ、インストール後使えない可能性がある。





10) ダウンローダーや開発ボードの JTAG 線を切る画面が出て、"確定"をクリックする。



11) インストール完了の画面が出る。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



12) License ファイルをインストールする。 "Copy License"をクリックして、 "xilinx\_ise\_vivado.lic"というファイルを選択する。





13) インストール成功のウィンドウが見える。

License installation was successful: C:\Users\Mei\AppData\Roaming\XilinxLicense\xilinx_ise_vivado.lic
确定

14) もう一度ダウンローダードライバーをインストールする必要があったら、Vivadoのインストールパス "X:Xilinx¥Vivado¥2017.4¥data¥xicom¥cable\_drivers¥nt64¥digilent"に入って、
"install\_digilent.exe"をクリックしてインストールする。その前に Vivado ソフトをしめる。
Vivado がダウンローダーを認識できない場合、ファイアウォールとウィルス防止ソフトを閉めてください。同時にいくつかのバージョン違いの Vivado と ISE を開けることもいけない。

ホームページ:<u>https://www.csun.co.jp</u>

#### 第四章 PLの"Hello World" LED テスト

Vivado をテストするプロジェクトは"led"

ZTNQ にとって、PL (FPGA) 開発はとても重要である。これも ZYNQ は他の ARM よりメリットある ところで、いろんな ARM 側のペリフェラルをカスタマイズできる。カスタマイズする前に、まずは LED サンプルで PL (FPGA) の開発流れと Vivado ソフトの基礎操作を慣れましょう。この開発流れは ARM なしの FGPA チップと完全一致している。

本サンプルにするのはLED ライトをコントロールする実験である。秒ことで開発ボード上のLED ライトをコントロールし、反転させる。ライトの点灯、消灯のコントロールを実現する。LED のコン トロールができれば、ほかのペリフェラルも徐々に把握できる。

### 4.1 LED ハードウェアの紹介

1) 開発ボードの PL 部分は赤い LED ライトを4つ接続した。ライトは完全に PL でコントロール されている。 PL LED

2) 回路図による接続関係でLEDとPLピンの連携関係を確認できる。

17P_T2_AD5P_36 17N_T2_AD5N_35 8P_T2_AD13P_35 3N_T2_AD13P_35 10_L19P_T2_35 19N_T3_VREF_35 20P_T3_AD6P_35 20N_T3_AD6N_35 1_DQS_AD14P_35 1_DQS_AD14P_35 22P_T3_AD7P_35	H20 G19 G20 H15 G15 K14 J14 N15 N16 L14 L15	<ul> <li>IO2_9P</li> <li>IO2_9N</li> <li>IO2_3P</li> <li>IO2_3N</li> <li>IO2_16P</li> <li>IO2_16N</li> <li>IO2_17P</li> <li>IO2_17N</li> <li>KEY1</li> <li>KEY2</li> <li>RTC_DATA</li> </ul>	15 15 15 15 15 15 15 12 12 11
2210_T3_AD710_55 IO_L23P_T3_35 IO_L23N_T3_35 4P_T3_AD15P_35 4N_T3_AD15N_35	M14 M15 K16 J16	LED1 LED2 LED3 LED3 LED4	12 12 12 12

対応のピンのインフォメーション

3) 回路図に PS\_MIO をはじめの IO は PS 側のもので、連携を必要ないし、連携できない。





3) 画面に新規プロジェクトを作成するガイダンスが出て、 "Next"をクリックする。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

New Project	
	Create a New Vivado Project
HLx Editions	This wizard will guide you through the creation of a new project.
	To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.
<b>XILINX</b>	
ALL PROGRAMMABLE.	< Back Nett> First Carcel
0	

ポップアップしたダイアログにプロジェクトのネームと保存する場所を入力する。ここは led というプロジェクトネームを付ける。注意すべきことであるが、 "Project location"に汉字とスペースを入れないように、パスの名称も長くつけてはいけない。

New Project		Recent Property	
Project Name Enter a name for your project and specify a directo	bry where the project data files will	l be stored.	
Project name:			8
Project location: F/ax7016/demo			
Project will be created at: F:/ax7015/demo/led			



5) プロジェクトタイプで"RTL Project"を選択する。

·	New Project
	Project Type Specify the type of project to create.
	<ul> <li>BTL Project         <ul> <li>You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.</li> <li>Do not specify sources at this time</li> </ul> </li> <li>Bost-synthesis Project You will be able to add sources, view device resources, run design analysis, planning and implementation.</li> <li>Do not specify sources at this time</li> <li>JO Planning Project         <ul> <li>Do not specify design sources. You will be able to view part/package resources.</li> <li>Imported Project</li></ul></li></ul>
6) ターゲッ 5月できる。マノ	②
	Use Add Files, Add Directories or Create File buttons below
	Add Files       Add Directories       Create File         Scan and add RTL include files into project       Copy gources into project         ✓ Add sources from subdirectories



7)"Next"をクリックして、ファイルを追加しない。

🔥 New Pro	ect	S YIINY	
Add Cor Specify or	straints (optional) create constraint files for physical and timing constraints.	<b></b>	
+,	-   ±   ∓		
	Use Add Files or Create File buttons below		
Cop	Add Files Create File y constraints files into project		-
?	< Back	ext> Einish Cancel	

8) "Default Part"オプションを開き、Filterの"Family"で"Zynq-7000"、パッケージ ングタイプ"Package"で"clg400"を選択する。AX7020の方はそれぞれプルダウンリストで "xc7z020clg400-2"を選ぶ。"-2"はスビートレベルを表している。数字が大きいければ大きいほ

ど、性能はよくなる。スビート高いチップは下位互換性がある。

New Project						-	Property.		l	_
Default Part Choose a default Xilinx part or	board for yo	ur project. Ti	his can be ch	anged later.						
Select. 👜 Parts 🔳 🖪	pards									
Produ <u>c</u> t category: All		~		Spe	ee <u>d</u> grade	All Re	emaini	~		
Eamily: Zyr	Iq-7000			✓ <u>T</u> er	np grade:	All R	emaini	~		
<u>S</u> earch: Q Part	I/O Pin Count	Available IOBs	LUT Elements	et All Filters FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transc	
xc7z014sclq400-1	400	125	40600	81200	107	0	170	0	0	^
@ xc7z020clg400-3	400	125	53200	106400	140	0	220	0	0	
@ xc7z020clg400-2	400	125	53200	106400	140	0	220	0	0	
xc7z020clg400-1	400	125	53200	106400	140	0	220	0	0	I
xc7z020iclg400-1L	400	125	53200	106400	140	0	220	0	0	~
•				< <u>B</u> ack		<u>N</u> ext >		<u>F</u> inish	Cance	el


### 9) "Finish"をクリックして、"led"という名のプロジェクトは完成できる。





### 4.3 Verilog HDL ファイルを作成し、LED を点灯する

1) Project Manager の Add Sources をクリックする (あるいはショットキーAlt+A を使う)

ow Navigator 😤	PROJECT MANAGER - led	
PROJECT MANAGER	Sources ? _ D 🗆 X	Project Sum
Add Sources	Q ≍ ≑ + ⊠ ● ° ♦	Settings
Language Templates	<ul> <li>Design Sources</li> <li>Constraints</li> </ul>	Project nar
<sup>‡</sup> <sup>‡</sup> <sup>‡</sup> <sup>†</sup> <sup>₽</sup> <sup>1</sup> <sup>P</sup> <sup>2</sup> <sup>1</sup> <sup>2</sup> <sup>4</sup>	Simulation Sources sim_1	Project loc: Product far
IP INTEGRATOR		Project par
Create Block Design	Hierarchy Libraries Compile Order	Top modul
Open Block Design	Properties ? _ D D X	Target lang
Generate Block Design	+ + <b>O</b>	Simulatori
SIMULATION		Synthesis
Run Simulation	Select an object to see properties	Status:
		Messages:

2) 設計ソースを追加また作成の "Add or create design sources"を選択して、 "Next"をクリックする。

	Add Sources
HLx Editions	This guides you through the process of adding and creating sources for your project
	Add or create constraints
	<u>A</u> dd or create design sources
	Add or create simulation sources
	X
ALL PROGRAMMABLE	
9	< <u>B</u> ack <u>E</u> inish

3) ファイル作成 "Create File"を選択する。



Cancel

inish

 Add Sources
 X

 Add or Create Design Sources
 Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

 Image: the end of the en

< <u>B</u>ack

<u>N</u>ext >

4) ファイル名 "File name" を "led" にして、 "OK" をクリックする。

 $\checkmark$  Add so<u>u</u>rces from subdirectories

?

n disk and add it to your pr	
<u>+</u> , - +	Create source file and add it to your project
	File name: led S File location: S Local to Project> <
Scan and add RTL includ Copy govrees into project Add sources from subdir	

5) "Finish"をクリックして、"led.v"の追加は完成する。





日昇テクノロジーなら可能にする

🔥 Add S	ources			-	×
Add o Specify file on d	<b>r Creat</b> e HDL, net lisk and a	<b>e Desig</b> i list, Block idd it to yo	n Sources Design, and IP f ur project.	files, or directories co	ntaining those file types to add to your project. Create a new source
+	-	<b>↑</b>   ↓			
	Index	Name	Library	Location	
ve	1	led.v	xil_defaultlib	<local project="" to=""></local>	
				Add Files	Add Directories Create File
S	can and a	add RTL ir	nclude files into	project	
C	opysour	ces into p	roject		
✓ A	dd so <u>u</u> rce	es from su	Ibdirectories		
?					<back next=""> Finish Cancel</back>

6) ポップアップしたモジュール定義 "Define Module" に、ファイル "led.v"のモジュール名 を "Module name"に指定することができる。ここでは、 "led"にする。一部のソケットも指定でき るが、今は暫く指定しなく、"OK"をクリックする

For each po	rt specified:	ianoro	dunloce	ito Pue	column is c	bockod	
Ports with	blank names will	I not be	written.	s its Dus	columnis c	lecked.	
Module Def	inition						
Module	name: led	Z					$\otimes$
I/O Port	Definitions						
+							
Port N	ame Direction	Bus	MSB	LSB			
	input	-	0	0			

7) ポップアップしたダイアログボックスに "Yes"を選択する。



8)



株式会社日昇テクノロジー

9) "led v"を編集する。ここで 32 ビットのレジスターtimer を定義した。0<sup>~</sup>499999999(1秒) のループカウントに使い、49999999(1秒)に数えたら、レジスターtimer は0になり、四つの LED を反転する。こうして、消した LED は点灯される。逆に、元々点灯している LED が消される。プロ グラミングしたコードは下のように:

`timescale 1ns/1ps

module led(

input sys\_clk, output reg [3:0] led );



reg[31:0] timer\_cnt; always@(posedge sys\_clk) begin if(timer\_cnt >= 32' d49\_999\_999) begin led <= ~led;</pre> timer cnt  $\leq 32' d0;$ end else begin led <= led;</pre> timer\_cnt <= timer\_cnt + 32'd1;</pre> end end endmodule 10) コードを編集したあと、メニュー "File->Save ALI Files"をクリックする。 Edit Flow Tools Window Layout View Help Q- Quicks File Σ # ø New Project. × Open Project. PROJECT MANAGER ? \_ 0 Project Summary × led.v\* × × Open Ex<u>a</u>mple Project. Q. **≚ ≑ +** 2 F:/ax7350/demo/led/led.srcs/sources\_1/new/led.v Save Project As. Write Project Tcl. Desig res (1) Q 🖬 🛧 🦽 🗶 🖬 🛍 🖊 🌆 🖓 led (led.v) Archive Project... timescale ins / ips Constraints 2 🖯 module led( Close Project nulation Sources (1) input sys clk Open Checkpoint output reg [3:0] led 1(1) reg[31:0] timer\_cnt always@(posedge sys\_clk) begin New IP Lo if(timer\_cnt >= 32' d49\_999\_999) begin led <= ~led: Open IP timer\_cnt <= 32' d New File. end else pen File. Hierarchy Libraries Compile Order begin led <= led: timer\_cnt <= timer\_cnt + 32'dl: Open Recent File Source File Propert ? \_ 🗆 🗆 X T File end 18 白 + 0 @ led.v ave File 19 20 - end Save File As ✓ Enabled 21 endmodule 22 Save All Files.. Location: F:/ax7350/demo/led/lec urces.

4.4 ピン制約を追加

Vivada が使っている制約ファイルのフォーマットは xdc ファイルである。xdc ファイルは主にピンの制約、タイマーの制約、それとグループの約束である。ここでは led. v に入出カポートを FPGAの実在なピンに分配することである。

1) "Open Elaborated Design"をクリックする



2)

3)

### 株式会社日昇テクノロジー







<u>File E</u> dit F <u>l</u> ow <u>T</u> ools	<u>W</u> indo	w Layout ⊻iew Help Q- Quid
	Σ	Project Su <u>m</u> mary Σ
Flow Navigator 🗧 😤		Sources 148
PROJECT MANAGER	₽	IP <u>C</u> atalog
🌣 Settings		Prop <u>e</u> rties Ctrl+E
Add Sources		Selection
Language Templates		RTL <u>N</u> etlist
		Devi <u>c</u> e Constraints
T II Gatalog		P <u>h</u> ysical Constraints
IP INTEGRATOR		<u>P</u> ackage Pins
Create Block Design		I/ <u>O</u> Ports
Open Block Design		Clock Regions
Conorata Block Design		Metrics
Generale block Design	٢	De <u>v</u> ice
<ul> <li>SIMULATION</li> </ul>		Package
Run Simulation		Tcl Consol <u>e</u> Ctrl+Shift+T
		Mess <u>ag</u> es

4) I/O Ports がポップアップされ、そこでピンの分配状況が見える

Tcl Console	Messages	Log	Reports	Design Runs	Package Pins	I/O Port	ts	×												
Q <u>∓</u> ≑	-4 -4	H H																		
Name	Dire	ction		Neg Diff Pair	Package Pin	Fi	ed	Bank	I/O Std		Vcco	Vref	Drive Strength		Slew Type		Pull Type		Off-Chip Termination	IN_TERM
🖌 😡 All ports (3	))						$\mathbf{\nabla}$													
<ul> <li>V led (2)</li> </ul>	OUT						1	(Multiple)	LVCMOS33*		3.300		12	~	SLOW	~	NONE	~	FP_VTT_50 V	
· led[1	I] OUT				A16	~ ,		35	LVCMOS33*	*	3.300		12	~	SLOW	~	NONE	~	FP_VTT_50 ~	
Ied[0	OUT				R7	~ `	1	13	LVCMOS33*	•	3.300		12	×	SLOW	~	NONE	~	FP_VTT_50 V	
🗸 🐼 Scalar (	ports (1)																			
> sys	clk IN				Y9	~ .	1	13	LVCMOS33*		3.300						NONE	~	NONE ~	

5) LED とタイマーにピン、レベル標準を分配する。完成したら、保存アイコンをクリックする。

Tcl Console	Mess	ages Log Report	s Design Runs	s Package Pins	I/O	Ports	×									
Q   ¥	\$ H	: + ',								_						
Name		Direction	Neg Diff Pair	Package Pin		Fixed	Bank	I/O Std		Vcco	Vref	Drive Strength		Slew Type		Pull
🗸 🐼 All po	rts (5)															
🗸 😼 lec	1 (4)	OUT				$\checkmark$	35	LVCMOS33*	Ŧ	3.300		12	~	SLOW	~	NON
<u>1</u>	led[3]	OUT		J16	~	$\checkmark$	35	LVCMOS33*	Ŧ	3.300		12	~	SLOW	~	NON
-2	led[2]	OUT		K16	~	$\checkmark$	35	LVCMOS33*	Ŧ	3.300		12	~	SLOW	~	NON
-3	led[1]	OUT		M15	~	$\sim$	35	LVCMOS33*	Ŧ	3.300		12	~	SLOW	~	NON
<u>a</u>	led[0]	OUT		M14	~	$\checkmark$	35	LVCMOS33*	Ŧ	3.300		12	~	SLOW	~	NON
🗸 🐼 Sc	alar ports	(1)														
<b>D</b> -	sys_clk	IN		U18	~	$\checkmark$	34	LVCMOS33*	Ŧ	3.300						NON

6) ウインドウがポップアップされ、制約の保存を要請する。ファイル名は"led"を書いて、フ ァイルタイプはデフォルトの"XDC"。"OK"をクリックする。



株式会社日昇テクノロジー

	1	Save Constraints		x	
		Select a target file to write new unsave Choosing an existing file will update th constraints.	d constraints to. at file with the new	4	
		• <u>C</u> reate a new file			
		<u>F</u> ile type: I XDC	~		
		F <u>i</u> le name: Ied	$\otimes$		
		Fil <u>e</u> location: 🛜 <local< th=""><th>to Project&gt; 🗸 🗸</th><th></th><th></th></local<>	to Project> 🗸 🗸		
	1	○ <u>S</u> elect an existing file			
		<select a="" file="" target=""></select>	~		
		?	ок с	Cancel	
7) 先程生成したつ	ファ	イル"led.xdc"を開いて	、TCLスクリプ	トが見える	。これらの構文を分か
れば、自分で led. xdc を	編	集することでピンを制約で	きる。		
Flow Navigator	ŧ \$	? _ PROJECT MANAGER - led			
PROJECT MANAGER		Sources	? _ 0 Ľ × /	Project Summary	× led.xdc ×
Settings		Q   ≭   ≑   + 図   ●0	٥	F:/AX7021/2017/cour	se_s1/01_led/led.srcs/constrs_1/new/led.xdc
Language Template	s	✓		Q   🔛   🛧   -	▶   🄏   🖬   🛍   🖊   🎟   ♀
👎 IP Catalog		Constraints (1)		1 set_property 2 set_property	IOSTANDARD LVCMOS33 [get_ports {led[1]}] IOSTANDARD LVCMOS33 [get_ports {led[0]}]
✓ IP INTEGRATOR		led.xdc (target)		<pre>set_property set_property set_property</pre>	PACKAGE_PIN A16 [get_ports fled[1]]
Create Block Design		> 🕞 Simulation Sources (1)		6 set_property	PACKAGE_PIN Y9 [get_ports ys_clk]
Open Block Design	<b>7</b>			1 ( I	

Generate B Compile Order Hierarchy Libraries SIMULATIC Run Sim Source File Properties ? \_ 🗆 🖒 × led.xdc  $\leftarrow | \rightarrow |$ ø RTL ANALYSIS

次は最も基礎的な XDC コーディングのグラマーを紹介する。普通な IO ポートはピン番号と電圧を 制約すればいい。

ピンの制約は下のように:

set\_property PACKAGE\_PIN "ピン番号" [get\_ports "ポートネーム"] レベル信号の制約は下のように: *IOSTANDARD "*レベル標準" [get\_ports "ポートネーム" ] set\_property

ここで注意するのは文字のフォントである。ポートネームはアレイだとしたら、 {} で括る。ポートネームは 必ずソースコードの名前と一致して、キーワードと同じくしてはならない。

レベル標準で、"LVCMOS33"のあとの数字は FPGA の BANK 電圧である。LED がいる BANK 電圧は 3.3V から、レベ ル標準は"LVCMOS"である。*Vivado のデフォルトコマンドは全ての IO に正確なレベル標準とピン番号を分配する ことである。* 

#### 4.5 シーケンス制約を追加

FPGA のデザインはピンの分配以外、もう一つ重要な制約ある。それはシーケンス制約である。ここはガイダンス方法でシーケンス制約を行う。

1) "Run Synthesis"をクリックして、統合を開始する。



### 2) ダイアログがポップアップされて、 "Ok"をクリックする



3)

# ● 株式会社日昇テクノロジー



Launch the selected synthesis or implementation runs.	
Launch <u>d</u> irectory: Solution Content of the second	
Options	
● Launch runs on local host: Number of jobs: 4	
<u>G</u> enerate scripts only	
Don't show this dialog again	
OK Cancel	
統合を完成したら、"Cancel"をクリックする Synthesis Completed	
Synthesis successfully completed.	
Next     O     Run Implementation	
Open Synthesized Design	
Dept show this dialog again	
Cancel	
4) "Constraints Wizard"をクリックする	







5) ポップアップされたウィンドウで "Next"をクリックする



ホームページ:<u>https://www.csun.co.jp</u>



6) シーケンス制約ガイダンスを使ってデザイン中のタイマーを分析する。ここは "sys\_clk" 頻度を 50Mhz にして、 "Skip to Finish"をクリックしシーケンス制約ガイダンスは終了。

ues												
F	Recommend	ed Constra	aints									
	Q Da	0 II										
	✓ Objec	t N	lame	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)				
	✓ I∐[ sy	S_CIK S	ys_cik	50.000	20.000	0.000	10.000					
	Constraints f	or Pulse V	Vidth Ch	eck Only								
	0 10-	_/ \ m	1	oon only								
	Objec	t Name	Frequ	Jency (MHz) Peri	od (ns) Rise	At (ns) Fall	t (ns) Jitter	(ns)				
1												
	Tcl Comma	nd Preview	w (1)	Existing Create Clo	ock Constraint	s (0)	$\langle$					
	Tcl Comman	nd Preview	w (1)	Existing Create Clo	ock Constraint	s (0)	$\langle$					
	Tcl Comman	nd Preview	<b>v (1)</b> ).000 -na	Existing Create Ck	ock Constraint orm (0.000 10.0	s (0)	sys_clk)]	Ż				
	Tcl Comman Q create_clock	nd Preview -period 20	<b>v (1)</b> ).000 -na	Existing Create Clo me sys_clk -wavefo	ock Constraint	s (0) 000} [get_ports	sys_clk]]					
	Tcl Comman	nd Preview -period 20	v (1) 9.000 -na	Existing Create Clo me sys_clk -wavefo	ock Constraint	s (0) 000) [get_ports	sys_clk)]					
	Tcl Comman	Id Preview	w (1) 9.000 -na	Existing Create Clo me sys_Clk -wavefo	ock Constraint	s (0) D00) [get_ports	sys_clk)]					
	Tcl Comman	nd Preview	v (1) 0.000 -na	Existing Create Clo me sys_clk -wavefo	ock Constraint	s (0)	sys_Clk]]					
	Tcl Comman	Id Preview	w (1) ).000 -na	Existing Create Cld me sys_clk -wavefo	ock Constraint	s (0) 000) [get_ports	sys_clk}]					
(***	Tcl Comman	d Preview	w (1) 0.000 -na	Existing Create Clo me sys_clk -wavefo	ock Constraint	s (0)	sys_clk]]		ext>	Skipto	Finish >>	Carr
(2	Tcl Comman	nd Preview	w (1) 0.000 -na	Existing Create Clo me sys_clk -wavefo	ock Constraint	s (0)	sys_Clk}]		ext>	Skipto	Finish >>	Car
(7)	Tcl Comman	nd Preview	v (1) 0.000 -na	Existing Create Clo me sys_Clk -wavefo	ock Constraint	s (0) 000) [get_ports	sys_clk]]		ext >	<u>Skip to</u>	Finish >>	Car
ダイ	Tcl Comman Q create_clock	period 20	w (1) 0.000 -na	Existing Create Ck me sys_clk-wavefr	ock Constraint orm {0.000 10.(	s (0) 000) [get_ports Dk"をク	sys_clk]] < <u>B</u> ack	」 ■ する	ext>	<u>Skip</u> to	Finish >>	Can
(T ) ý1	Tcl Comman Q create_clock	nd Preview -period 20	v (1) 0.000 -na	Existing Create Clo me sys_clk -wavefo アップされ	ock Constraint orm {0.000 10.0	s (0) 000) [get_ports Dk"をク	sys_clk] <u> &lt; Back</u> リックマ	」 1 する	ext >	Skipto	Finish >>	Can
ý 1	Tcl Comman Q create_clock	nd Preview -period 20 יישלי דרייי Timung	v (1) ).000 -na ツプ	Existing Create Clo me sys_clk-wavefo アップされ traints Wizard	ock Constraint orm {0.000 10.0	s (0) 000) [get_ports Dk"をク	sys_clk] < <u>Back</u> リックマ	」 1 する	ext >	Skip to	Finish >>	Car
ý 1	Tcl Comman Q create_clock	nd Preview period 20 ביילילילי Tinting	w (1) 0.000 -na 9 Cons	Existing Create Ck me sys_clk-wavefr アップされ traints Wizard	ock Constraint סידה {0.000 10.0	s (0) 000) [get_ports Dk"をク	sys_clk]] < <u>Back</u> リックマ	」 ■ する	ext>	Skip to	Finish >>	Can

8) "Finish" をクリックする





10) この時、led.xdc ファイルがアップデートされる。"Reload"をクリックして、改めて ファイルをアップロードする

Sources × Netlist Device Constraints ? _ C	Package × Device × Ied.v × Ied.xdc ×
	F:/ax7015/demo/led/led.srcs/constrs_1/new/led.xdc
V 🖨 Design Sources (1)	This file has been changed Reload
Constraints (1)	$ \mathbf{Q}_{\mathbf{x}}  \sqsubseteq   \Leftrightarrow   \Rightarrow   \mathbf{X}_{\mathbf{x}}   \blacksquare   \blacksquare   \mathbf{M}_{\mathbf{x}}   \blacksquare   \mathbf{Q}_{\mathbf{x}}  $
	1 set_property IOSIANDARD LVCMOS33 [get_ports {led[3]}]
$\sim \equiv \text{constrs}_1(1)$	2 set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
Ied.xdc (target)	3 set_property_IOSTANDARD_LVCMOS33 [get_ports {led[1]}]
<ul> <li>Simulation Sources (1)</li> </ul>	4 set property IOSTANDARD LVCMOS33 [get ports {led[0]}]
	5 set property IOSIAHDARD LVCMOS33 [get ports sys clk]
$\rightarrow \equiv sim_1(1)$	6 set property PACKAGE PTN B8 [get ports {led[3]}]
	7 set property PACKAGE PIN AS [get ports (led[2])]
	A Sec_property Remonstrate AS [sec_points (led[2])]
	8 set_property fRCKAGE_fill A/ [get_ports [led[1]]]
	9 set_property FAUKAGE_FIN A5 [get_ports {led[U]}]
	10 set_property PACKAGE_PIN V14 [get_ports sys_clk]
Hierarchy Libraries Compile Order	11
Source File Properties 🗙 Clock Regions ? _ 🗆 🖆	

4.6 BIT ファイルを生成する

1) プログラミング中は総合、レイアウト、BIT ファイルを生成することなどに細分化できる。 ここでは、直接"Generate Bitstream"をクリックして、BIT ファイルを生成する。





2) ポップアップされたダイアログに任務の数を選べる。ここと CPU コア数に関係している。一般的には、数字が大きいければ大きいほど、プログラミングは早くなる。"OK"をクリックする。



Launch Runs	x
Launch the selected synthesis or implementation runs.	4
Launch <u>d</u> irectory: 🛜 < Default Launch Directory>	~
Options	
Generate scripts only	
Don't show this dialog again	
OK	el

3) コンパイルを開始して、右上のとこは状況を反映するメッセージ欄が見える。コンパイル 中はウィルス防止ソフトにブロックされる可能性があって、コンパイルができない、又は長時間にコ ンパイルが成功できないことになる。

<u>F</u> ile	<u>E</u> dit	Flow	Tools	<u>W</u> indow	Layout	View	<u>H</u> elp	Q+ Quick Access		Initializing Design Cancel	С
-	* /	•	10	X	a 🖕	Σ	1 11	×	•	😬 Default Layout	~

4) コンパイルはノーミスで完成したらダイアログがポップアップされて、引き続きの操作 を選ぶ。"Open Hardware Maneger"を選択し、"OK"をクリックする。もちろん"Cancel" も選べ る。左のガイド欄で"Open Hardware Manger"をすればいい。

	Bitstream Generation Completed
	Bitstream Generation successfully completed.
$\wedge$ $\checkmark$	Open Implemented Design
$\langle \wedge \rangle$	◯ <u>V</u> iew Reports
$\mathbf{X}$	Open <u>H</u> ardware Manager
×	◯ Generate Mem
	On't show this dialog again
	OK Cancel

4.7 ダウンロードとデバッグ

1) 開発ボードの JTAG インタフェースを接続して、電源を入れる。

2) "HARDWARE MANAGER"インタフェースで、"Auto Connect"をクリックして、自動的に設備と接続する。





3) JTAG が ARM と FGPA のカーネルをスキャンする(図では xc7z015 が、AX7020 開発ボードの 場合は xc7z020\_1 である)。また一つの XADC があって、システムの電圧と温度を検測できる。

		M Dashboard -
Flow Navigator ≚ ♦ 3	HARDWARE MANAGER - loo There are no debug core	calhost/xilinx_tcf/Digilent/2102498 es. Program device Refresh dev
Y SYNTHESIS	Hardware	? _ 🗆 🖒 X
Run Synthesis	Q ₹ ♦ Ø	▶  >   <b>■</b>   <b>○</b>
> Open Synthesized Design	Name	Status
MPLEMENTATION	<ul> <li>localhost (1)</li> <li>ilinx_tcf/Digile</li> </ul>	Connected
Run Implementation	@ arm_dap_0 (	• <b>arm</b> 纳核
<ul> <li>Open Implemented Design</li> <li>Constraints Wizard</li> </ul>	<ul> <li>✓ (i) xc7z015_1 (1)</li> <li>iii XADC (System)</li> </ul>	FPGA pranting
Edit Timing Constraints	1	) )
The Report Timing Summary		

4) xc7z015\_1 (xc7z020\_1) を選択して、右クリックすると"Program Device"のオプションが 出る。



	Hardware	? _ 🗆 🖒 X	led.v × led.xdc
	Q   ¥   ♦   ∅   ▶   3	* <b>•</b>	F:/ax7015/demo/led/le
	Name	Status	
	🗸 📱 localhost (1)	Connected	
	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	Open	1 ¦ timescale i 2 — module led(
	arm_dap_0 (0)	N/A	3 input sys
	✓ @ xc7z015_1 <sup>(**)</sup>		4 output re
	VADC (S	ardware Device Properties	Ctri+E
	Pr	ogram Device	di
	Ve	erify Device	
	Hardware Device Prop C R	efresh Device	
	<pre>xc7z015_1 Ac</pre>	ld Configuration Memory Dev	ice
	Bo	oot from Configuration Memo	ry Device
	General Properties Pr	ogram BBR Key	
	Tcl Console × Mes	ear BBR Key	
r)			h + 7
5) ホッノアッノさ	れたワイントワで、	Program をクリ	ツクタる。
Pri	ogram Device		
Sel sele pro	ect a bitstream programming file ar ect a debug probes file that corresp gramming file.	id download it to your hardwa onds to the debug cores con	are device. You can optionally tained in the bitstream
	Bitstream file: F:/ax7015/dem	o/led/led.runs/impl_1/led.bit	Ø
	Debug probes file:		
	✓ Enable end of startup check		
3			Program Cancel
6) ダウンロード	を待つ。		
Progra	im Device		×
			]
Program	nming the device		
			6504
			05%
			Background Cancel

7) ダウンロード完成したあと、四つの LED は秒ことで変化し始める。ここまで、Vivadoの簡単 な流れ体験が終わる。後の章はこんなことを紹介する:プログラムを Flash に書き込む場合、PS シ ステムの協力が必要になる。PL のみのプロジェクトは Flash に書き込めない。

#### 第五章 HDMI 輸出実験

可能への挑戦

実験用 Vivado プロジェクトは"hdmi\_output\_test"。

前の章で、led の点滅実験を紹介した。これはただ Vivado の基本の開発流れを了解する ため。本章の実験は LED 点滅事件よりすこし複雑で、内容は HDMI でのカラーバーの表示であ る。これも今後表示、ビデオ処理の基礎である。

#### 5.1 ハードウェアの紹介

開発ボードは HDMI コーディングチップを使っていない。その代わりに、FPG の 3.3V 差分 IO を 直接 HDMI コネクタに接続する。

FPGA が 24 ビット RGB コードの出力で TMDS 差分信号を実現する。



### 5.2 Vivado プロジェクトを作成する

1) "hdmi\_output\_test"というプロジェクトを作成する。

### 5.2.1 HDMI コンパイラーIP コアを追加する

2) repo フォルダ(このフォルダは提供したサンプルプロジェクトにある)をプロジェク トメニューにコピーする。フォルターに HDMI コンパイラーの IP がある



名称	修改日期	类型	大小
퉬 hdmi_out_test.cache	2018/9/19 13:46	文件夹	
🌗 hdmi_out_test.hw	2018/9/19 13:46	文件夹	
퉬 hdmi_out_test.ip_user_files	2018/9/19 13:46	文件夹	
퉬 hdmi_out_test.sim	2018/9/19 13:46	文件夹	
📔 repo	2018/9/19 13:47	文件夹	
🝌 hdmi_out_test.xpr	2018/9/19 13:46	Vivado Project Fi	6 KB

### 3) "IP Catalog" をクリックして、標準設置でこれらの IP はすべて Xilinx が提供 されている。いまは第三者の IP、あるいは自分で作った IP を追加する。







	≍ ≑ ¥ + √	2 🔅 🔳 🔍			
	Name			∧1	AXI4
	🗸 📄 Vivado Repository	Properties	Ctrl+F		
	> 📄 Alliance Partne	<u>_</u>			
	> 📄 Automotive & In	IP Settinas			
	> 🗅 AXI Infrastructu	Add Repository			
	> 🖹 AXIS Infrastruct	Refresh All Repositories			
	> 🖴 BaselP	Export to Spreadsheet			
	> 📄 Basic Elements				
	> 📄 Communication &	Networking			
	> 📄 Debug & Verificatio	n			
	> 📄 Digital Signal Proce	essing			
よ先コ Reposit	→ Digital Signal Proce	essing ルターにする。			-
よ先コ Reposit ecent.	> Digital Signal Proce	essing ルターにする。 ndmi_out_test/repo	× 1 3		
よ先⊐ Reposit ecent:	> Digital Signal Proce	essing ルターにする。 ndmi_out_test/repo imi_out_test/repo	× 3 3		
またコ Reposit ecent: irectory:	> Digital Signal Proce	essing ルターにする。 ndmi_out_test/repo mi_out_testvrepo ut_test.cache			

- > i hdmi\_out\_test cache > hdmi\_out\_test hw > hdmi\_out\_test ip\_user\_files > i hdmi\_out\_test sim > i repo > df ip > i repo > Sele
- 6) IP を追加して、成功に追加した IP の数を提示する。

Add Re	pository 1 repository was added to the project		x
Reposi	tory		
Ŧ	f./ax7020/2017/course_s1/hdmi_out_test/repo		
¢	<ul> <li>&gt; IPs (2)</li> <li>&gt; Interfaces (1)</li> </ul>		
			_
		ОК	

Cancel



7) "RGB to DVI Video Encoder (Source)"を見つけ、ダブルクリックする。

Project Summary × IP Catalog ×		
Cores   Interfaces		
≍   ≑   ≇   •€   ∥   ∅   <b>0</b>   Q.		
Name	∧1 AXI4	Status
<ul> <li>User Repository (f./ax7020/2017/course_s1/hdmi_out_test/repo)</li> </ul>		
🗸 🖆 FPGA Features and Design		
> 🚍 Clocking		
∽ 🖆 UserIP		
👎 Dynamic Clock Generator	AXI4	Pre-Productio
🗸 🖆 Video & Image Processing		
👎 RGB to DVI Video Encoder (Source)		Pre-Productio
🗸 🚍 Vivado Repository		
> 🖹 Alliance Partners		
🕥 🖻 Automotive & Industrial		

8) 下のウィンドウがポップアップされ、コンポーネント名"Component Name"を変更せず、 他のパラメータもそのまま。"OK"をクリックする。

Customize IP RGB to DVI Video Encoder (Sou Documentation IP Location C =	rce) (1.3) Switch to Défaults	ZZ
Show disabled ports	Component Name rgb2dvi_0 MMCM/PLL MMCM PLL Reset active high Generate SerialClk internally from pixel clock.	
	TMDS clock range                • >=120 MHz (1080p)              · < 120 MHz (720p)	
		OK Cancel

9) "Generate Output Products"のウィンドウが出る。中にも"Number of jobs"はスレット数で、数値が高ければスビートも早い。





#### 5.2.2 ピクセルクロックの PLL モジュールを追加する

HDMI コンパイラーをドライブされる為、ピクセルクロックと5倍のピクセルクロックが需要になる。5倍の方は10:1シリアル化に使用される。

11) "IP Catlog" ウィンドウでキーワード "clock" をサーチして、 "Clocking Wizard" をダブ ルクリックする。



12) 今回はコンポーネントに名前を付ける。这次给元件起个名字, "Component Name" 欄で"video\_clock"、"clk\_in1"欄で50を入力する。ここの 50Mhz は開発ボード PL 側のク リスタル頻度と一致している。



Nonemanan - Anno 2011 (1990) (1997) (1997)			1
Documentation 📄 IP Location C Switc	th to Defaults		
P Symbol Resource	Component Name video_clock		6
] Show disabled ports	Clocking Options Output Clocks Port Renaming	MMCM Settings Summary	
	MMCM      PLL		
	Clocking Features	Jitter Optimization	
	Frequency Synthesis Minimize Power	Balanced	
	Phase Alignment Spread Spectrum	O Minimize Output Jitter	
	Dynamic Reconfig Dynamic Phase St	nift O Maximize Input Jitter filteri	ng
	Safe Clock Startup		
- reset clk_out1 -	Dynamic Reconfig Interface Options		
	Phase Du	ity Cycle Config	
	AXI4LITE     DKP	Enablement of Phase and Du	ty Cycle dynamic
	Input Clock Information	AXI4-Lite. Enabling this uses	DSP resources o
	Input Clock Port Name Input Freque	ency(MHz) Jitter O	ptions
	Input Clock Port Name Input Erequine Primary Clk_In1 50	ency(MHz) Jitter O	ptions
	Input Clock         Port Name         Input Erequination           Primary         clk_in1         50           Secondary         clk_in2         100.000	Pency(MH2) Jitter O € 10.000 - 800.000 UI 60.000 - 144.000	ptions (
	Input Clock         Port Name         Input Erequine           Primary         clk_in1         50           Secondary         clk_in2         100.000	ency(MHz) Jitter O 10.000 - 800.000 UI 60.000 - 144.000	ptions (

13) アウトプットクロック "clk\_out1" はビデオピクセルクロックに用いられる。ここで 74.25 を入力する。これは 1280x720@60 解析度のピクセルクロックである。各解析度のピクセ ルクロックが違っていて、ビデオ標準 かなり詳しい人だけが、各ビデオ解析度のピクセルクロックを知ってい る。 "clk\_out2" はコンパイラーのシリアルかに使われている。ピクセルの 5 倍は 371.25 を入力する。次は "OK" をクリックして、IP を生成する。







Documentation 📄 IP Location C Sw	ritch to Defaults					
IP Symbol Resource	Component Name	video_clock				(
Show disabled ports	<b>Clocking Options</b>	Output Clock	s Port Renaming	MMCM Settings	Summary	
	The phase is calc	ulated relative to	the active input clock			
	Output Clock	Port Name	Output Freq (MHz)	1	Phase (degrees)	
		alk auti	Requested	Actual	Requested	Act
	CIK_OUT1	CIK_OUT1	74.25	74.219	0.000	0.00
	✓ clk_out2	clk_out2	371.25	371.094	0.000 🚫	0.00
	clk_out3	clk_out3	100.000	NVA	0.000	N/A
	clk_out4	clk_out4	Too.ooo			N/A
- reset clk_out1 - clk_in1 clk_out2 - locked -	clk_out5	clk_out5	100.000	N/A	0.000	N/A
	clk_out6	clk_out6	100.000	N/A	0.000	N/A
	clk_out7	clk_out7	100.000	N/A	0.000	N/A
	USE CLOCK	SEQUENCING	C	locking Feedback		
	Output Cloc	k Sequence	Number	Source		
	clk_out1	1		<ul> <li>Automat</li> </ul>	ic Control On-Chip	

5.2.3 カラーバー発生モジュールを追加する

14) カラーバー発生モジュールは一列の Verilog コードで、ビデオシーケンスと用 于产生视频时序和水平方向の八つのカラーバーを生成するに使われている。FPGA は本開 発の ポイントではないので、コードのことを詳しく説明しない。提供したサンプルに

ホイントではないので、コートのことを詳しく説明しない。提供したサンあるコートをコピーできる。



5. 2. 4

# 株式会社日昇テクノロジー

低価格、高品質が不可能? 日昇テクノロジーなら可能にする

3) gb2dvi_0.xci) (1) solor_bar.v) (video_clock.xci) us (3)	F:/ax7020/ Q	2017/course_s	s1/hdmi_out	test/hdmi_out_	test.srcs/sources_1/new/color_bar.v
3) gb2dvi_0.xci) (1) iolor_bar.v) (video_clock.xci) is (3)	Q      1    // 2  // 3   / 4   / 5   / 6   /	◆ ◆ Description 彩条发生模块	<b>X E</b>		E   Q
gb2dvi_0.xci) (1) olor_bar.v) (video_clock.xci) vs (3)	1   // 2 // 3 // 4 // 5 // 6 //	Descriptiou 彩条发生模块	σι το ************ 1.	*****	
(video_clock.xci) es (3)		Description 彩条发生模块	1. 1. 1.		
(video_clock.xci) 25 (3)	2 // 3 // 4 // 5 // 6 //	Description 彩条发生模块	ı: !		
s (3)	4 // 5 // 6 //	彩条发生模块	5		
<b>is</b> (3)	5 //	n · · · v			
(3)	6 //	77			
		Revision A	story:	Provide the second	
	7 1/	Date	Бу	Revision	Change Description
	9 //	2013/5/7		1.2	remove some wening
	10 //	2013/4/18		I. 1	vs timing
s Libraries ⊲ ▶ =	11 //.	2013/4/16		1.0	Original
	12 🛱 //	*******	*********	**************	************************************
0	13 🖵 mo	dule color_bar	• (	口俗来就是我	
? _ O G X	14	input clk,	2	// 像素町守辙/ 。 // 信点 含古赤	1280x720@60P时像素时钟为74.25
← → Φ	15	output rst,		// <u>多世</u> , 尚有双 // 行同步 [ 宴有]	+
~	17	output vs.		//场同步、高有交	
	18	output de,		(/ 数据有效	
7020/2017/course_s1/t	19	output[7:0]	rgb_r,	1像素数据、红色	日分量
	20	output[7:0]	rgb_g,	1/像素数据、绿色	分量
og	21	output[7:0]	rgb_b	//象素数据、蓝色	的量
ofaultlib	22 1;	222222222 <b>2</b> 3日 11日	「虎条粉女人		***************************************
	24 pa	rameter H AC	IIVE = 16' d	1280: //行有效	, 长度(像素时钟周期个数)
(В 🗸	25 <b>pa</b>	rameter H_FP	= 16' d110;	//行同步	前肩长度
·····	26   pa	rameter H_SY	$\mathrm{MC} = 16'  \mathrm{d}40$	; //行同步	长度
			KAN MINA MAND		
	s Libraries ( ) → Ξ ? _ □ Ľ × ( ) → ( ) × ( ) → ( ) × ( ) → ( ) × ( ) ×	s Libraries (4 ) ≡ ? - □ □ × ? - □ □ × 11 /// 12 0 // 13 0 mo 14 15 16 17 18 19 20 21 22 ); 23 //× 24 pa 25 pa 26 pa	9       //2013/3//18         10       //2013/4/18         11       //2013/4/18         12       //***********************************	s Libraries 4 ▶ = ? _ □ □ ☆ ↑ ↑ □ //2013/4/18 11 //2013/4/18 11 //2013/4/18 12 □ //**********************************	9 //2013/3// 1.2 9 //2013/3/4/18 1.1 10 //2013/4/18 1.1 11 //2013/4/16 1.0 12 白 //**********************************



5.3 XDC 制約ファイルを追加する

下の xdc 制約ファイルをプロジェクトに追加する、ファイルにクロックと HDMI に関するピンも加わっている。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする





set\_property PACKAGE\_PIN U18 [get\_ports {sys\_clk}] set property IOSTANDARD LVCMOS33 [get ports {sys clk}] create\_clock -period 20.000 -waveform {0.000 10.000} [get\_ports sys\_clk] set\_property IOSTANDARD TMDS\_33 [get\_ports TMDS\_clk\_n] set property PACKAGE PIN N18 [get ports TMDS clk p] set property IOSTANDARD TMDS 33 [get ports TMDS clk p] set property IOSTANDARD TMDS 33 [get ports {TMDS data n[0]}] set\_property PACKAGE\_PIN V20 [get\_ports {TMDS\_data\_p[0]}] set\_property IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_p[0]}] set\_property IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_n[1]}] set\_property PACKAGE\_PIN T20 [get\_ports {TMDS\_data\_p[1]}] set\_property IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_p[1]}] set property IOSTANDARD TMDS 33 [get ports {TMDS data n[2]}] set property PACKAGE PIN N20 [get ports {TMDS data p[2]}] set\_property IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_p[2]}] set property PACKAGE PIN V16 [get ports hdmi oen] set\_property IOSTANDARD LVCMOS33 [get\_ports hdmi\_oen]

株式会社日昇テクノロジー

### 5.4 ダウンロードとデバッグ

プロジェクトを保存し bit ファイルにコンバインする、HDMI インタフェースを HDMI モ ニターに接続する。注意するのは、ここで 1280x720@60Hz を使うので、自分のモニターはこ の解析度をサーポートできることを確保してください。

e	Status		
localhost (1)	Connected		
d xilinx_tcf/Digilent/21	Open	1 ; timescale lns / lps 2 — module led(	
@ arm_dap_0 (0)	N/A	3 input sys_clk,	
v () xc7z015_1 (1)	Programm	gram Device	X
W XADC (System			
		a debug prebeb me and berrebpende to are debug bereb berramed in the bible but	
	prog	a a cody process no marcon oppondo to the cody cores contained in the shorteant	-
$\sum$	prog	amming file.	
$\bigcirc$	prog	itstream file:	
$\Diamond$	progr B D	amming file. itstre <u>a</u> m file: F:/ax7015/demo/led/led.runs/impl_1/led.bit	
$\diamond$	prog	amming file. itstream file: F:/ax7015/demo/led/led.runs/impl_1/led.bit	]
	progr	itstre <u>a</u> m file:          F:/ax7015/demo/led/led.runs/impl_1/led.bit         ebug probes file:	]
ware Device Properties	progr B D	amming file.          itstream file:       F:/ax7015/demo/led/led.runs/impl_1/led.bit         ebug probes file:       ?         Enable end of startup check	]
ware Device Properties 7/z015_1	prog	itstre <u>a</u> m file:          F:/ax7015/demo/led/led.runs/impl_1/led.bit         @         @ Enable end of startup check	] ]

ダウンロード後モニターは下の画面を表示する。





### 5.5 実験のまとめ

本実験は初歩にビデオ表示を接触した。ビデオ知識関わっているが、zynq を学ぶキー ポイントではないので、詳しく紹介していない。しかし、zynq はビデオ紹介領域でよく使 われているから、勉強者は良き基礎知識が必要である。実験中は PL だけ使用して HDMI チッ プをドライブして、第三者がカスタマイズする IP の使い方を初歩に学んだ。引き続きはカ スタマイズ IP の作り方を学ぶ。



#### 第六章 ARM を体験

実験用 Vivado プロジェクトは "ps\_hello"。

前の実験は PL 側で行わている。はっきり見えるのは、普通の FGPA 開発流れと何の区別 がないことである。ZYNQ の優勢は FPGA と ARM を合理的に結ぶことなので、開発エンジニア ににより高く要求している。本章から、ARM いわゆる PS の利用を始める。今回は簡単なシリ アルプリントで Vivado SDK と PS の特性を体験する。

#### 6.1 ハードウェアの紹介

原理図から見えるのは、ZYNQ チップは PL と PS に分けられる。PS 側の IO 分配は相対的 に固定で、任意で分配するにはいけない。Vivado ソフトでピンの配りも必要ない。本実験で は PS しか使っていないが、Vivado プロジェクトの作成は需要で、PS ピンの配置に用いられ る。

#### 6.2 Vivado プロジェクトを作成

- 1) "ps\_hello" というプロジェクトを作成する。
- 2) "Create Block Design"をクリックし、Block デザインを作成する。





3) ここで"Design name"は変更しないで、デフォルトの"design\_1"にする。変更の需要 があったら、名前はできるだけ短くする。そうしなければ、Windows でのコンパイルは問題が 出る。







7) "PS-PL Configuration"で全ての項目を表にする。



Documentation       Presets       IP Location       Summary Report         Page Navigator				
Page Navigator       PS-PL Configuration       Summary Report         Zyng Block Design	ocumentation 🔅 Presets 📄	IP Location 🏽 🌀 Import XPS Settings		
Zyng Block Design	ge Navigator _ PS-P	PL Configuration		Summary Report
PS-PL Configuration Peripheral I/O Pins MIO Configuration Clock Configuration DDR Configuration SMC Timing Calculation Interrupts	q Block Design 🗧 🗲	Search: Q-		
Peripheral I/O Pins <ul> <li>MIO Configuration</li> <li>Clock Configuration</li> <li>DDR Configuration</li> <li>MIO Configuration</li> <li>DDR Configuration</li> <li>SMC Timing Calculation</li> <li>PS-PL Cross Trigger interface</li> <li>PS-PL Cross Trigger interface</li> <li>Enables PL cross trigger signals to PS and vice-versa</li> </ul>	PL Configuration	Name	Select	Description
AND Configuration <ul> <li>AND Non Secure Enablement</li> <li>GP Stave AXI Interface</li> <li>GP Stave AXI Interface</li> <li>ACP Stave AXI Interface</li> <li>ACP Stave AXI Interface</li> <li>ACP Stave AXI Interface</li> <li>DMA Controller</li> <li>PS-PL Cross Trigger Interface</li> </ul>	inheral I/O Pins	> General		
MIO Configuration Clock Configuration DDR Configuration SMC Timing Calculation Interrupts		> AXI Non Secure Enablement	0 ~	Enable AXI Non Secure Transaction
Clock Configuration       > HP Slave AXI Interface         DDR Configuration       > ACP Slave AXI Interface         SMC Timing Calculation       > DMA Controller         Interrupts       > PS-PL Cross Trigger Interface	Configuration	> GP Slave AXI Interface		
ACP Slave AXI Interface     DDR Configuration     ACP Slave AXI Interface     DAA Controller     PS-PL Cross Trigger interface     PS-PL Cross Trigger interface     Interrupts	ck Configuration	> HP Slave AXI Interface		
DDR Configuration       > DMA Controller         SMC Timing Calculation       > PS-PL Cross Trigger interface       Enables PL cross trigger signals to PS and vice-versa         Interrupts		> ACP Slave AXI Interface		
SMC Timing Calculation > PS-PL Cross Trigger Interface Enables PL cross trigger signals to PS and vice-versa	R Configuration	> DMA Controller		
interrupts	C Timing Calculation	> PS-PL Cross Trigger interface		Enables PL cross trigger signals to PS and vice-versa
	mpts			

8) "M AXI GPO interface"インタフェースを取り消す。このインタフェースは PL 側の AXI インタフェースペリフェラルを拡張できる。この原因で、PL は PS とデータ交 換をするなら、AXI バスインターフェスプロトコルに沿いで交換を行う。Xilinx は大量 な AXI インタフェースの IP コアを提供した。

ZYNQ7 Processing System	n (5.5)		
Documentation 🔅 Presets	IP Location Gumport XPS Settings		
Page Navigator _ P	S-PL Configuration		Summary Re
Zynq Block Design	Search. Q-		
PS-PL Configuration	A Name	Select	Description
	FCLK_RESET0_N	~	Enables general purpose reset signal 0 for PL logic
Peripheral I/O Pins	FCLK_RESET1_N		Enables general purpose reset signal 1 for PL logic
WIO Configuration	FCLK_RESET2_N		Enables general purpose reset signal 2 for PL logic
	FCLK_RESET3_N		Enables general purpose reset signal 3 for PL logic
Clock Configuration	<ul> <li>AXI Non Secure Enablement</li> </ul>	0 ~	Enable AXI Non Secure Transaction
DDR Configuration	<ul> <li>GP Master AXI Interface</li> </ul>		
	<ul> <li>M AXI GP0 interface</li> </ul>		Enables General purpose AXI master interface 0
SMC Inning Calculation	Static remap	0	Enables static remap for GP0 interface
nterrupts	Thread ID Width	12	Thread ID Width for GP0 interface
	<ul> <li>M AXI GP1 interface</li> </ul>		Enables General purpose AXI master interface 1
	Static remap	0	Enables static remap for GP1 interface
	Thread ID Width	12	Thread ID Width for GP1 interface
	<ul> <li>GP Slave AXI Interface</li> </ul>		
	S AXI GP0 interface		Enables General purpose 32-bit AXI Slave interface 0
	S AXI GP1 interface		Enables General purpose 32-bit AXI Slave interface 1
	<ul> <li>HP Slave AXI Interface</li> </ul>		
	<		



### 6.2.1 UART 配置

9) 原理図から、シリアルは PS の MI048-MI049 上に接続していることが見つける。 "Peripheral I/O Pins"オプションで UART1 (MI048 MI049) を開く。Bank 0 電圧は "LVCMOS 3. 3V"、Bank 1 电压 は "LVCOMS 1.8 V"を選択する。本実験はシリアル機能は一つだけ使 っている。ここでは他の設備は使用しない。



### 6.2.2 クロック配置

10) "Clock Configuration"のオプション欄で、PS クロックのインプット頻度を配置 できる。デフォルト値はボートと同じく 33. 333333 で、変更する必要ない。CPU 頻度のデ フォルト値は 666. 666666Mhz、ここも変更しない。同時に PS は PL 側にクロック四重も提 供でき、頻度は配置できる。ここは必要ないから、デフォルト値にすればいい。




YNQ7 Processing Sys	stem (S	5.5)					
Documentation 🔅 Prese	ets 📄	IP Location 🏻 🍈 Import XPS S	ettings				
Page Navigator _	Clock	Configuration				Summar	y Rej
Zynq Block Design	Basi	c Clocking Advanced Clock	king				
PS-PL Configuration	+	Input Frequency (MHz) 33.33	3333 🚫 CPI	J Clock Ratio 6:2:1	~		
Peripheral I/O Pins	Q	Search: Q					
MIO Configuration	Ŧ	Component	Clock Source	Requested Frequ	Actual Frequency(	Range(MHz)	
	\$	<ul> <li>Processor/Memory Clock</li> </ul>	s				
Clock Configuration	1	CPU	ARM PLL 🗸	666.666666 📀	666.666687	50.0 : 800.0	
DDR Configuration		DDR	DDR PLL 🗸	533.33333 🛞	533.333374	200,000000 : 534.000	
		✓ IO Peripheral Clocks					
SMC Timing Calculation		SMC	IO PLL	100	10.000000	10.000000 : 100.000000	
Interrupts		QSPI	IO PLL	200	10,000000	10.000000 : 200.000000	
		ENET0	IO PLL	1000 Mbps	10.000000		
		ENET1	IO PLL	1000 Mbps	10.000000		
		SDIO	IO PLL	100	10.000000	10.000000 : 125.000000	
		SPI	IO PLL	166.666666	10.00000	0.000000 : 200.000000	
		~ CAN					
		CAN CLK	IO PLL	100	10.00000	0.100000 : 100.000000	
		CAN0 MIOCLK	External	-1	23.8095	-2:-1	
		CAN1 MIOCLK	External	-1	23.8095	-2:-1	

#### 6.2.3 DDR3 配置

11) "DDR Configuration"オプションで PS側の ddr パラメータを配置できる。 AX7010 は DDR3 型番を "MT41J128M16 HA-125" に、AX7020 は DDR3 型番を "MT41J256M16 RE-125" にする。ここの ddr3 型番はボートでの ddr3 型番ではなく、パラメータが 一番近いタイプである。 "Effective DRAM Bus Width" で "32 Bit" を選択する。





AX7020 ddr3 选择



12) "Run Block Automation"をクリックして、vivado ソフトは自動的に一部のポート書き出す仕事を完成する。



14) "OK"をクリックしたあとは、PS 側からエクスポートたピンが見える。DDR と FIXED\_IO も含んでいる。"Ctrl + s"を押してデザインを保存する。



 Diagram
 ×

 Q
 Q
  $\gtrsim$  

 Q
 Q
  $\lesssim$  

 Q
 Q
  $\lesssim$  

 Q
 Q
  $\lesssim$  

 Q
  $\lesssim$   $\Rightarrow$  

 Q
  $\Rightarrow$  

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q

 Q



ホームページ:<u>https://www.csun.co.jp</u>

メール:info@csun.co.jp



18)

17) デザインを開け、PS は普通な IP で使われているのが見える。



19) "Generate"をクリックする。



● ★〒18~048 株式会社日昇テクノロジー

	A Generate Output Products	
	The following output products will be generated.	
	Preview	
	Q ≚ ♦	
	✓ ▲ design_1.bd (OOC per IP)	
	🕥 Synthesis	
	Implementation	
	🗇 Simulation	
	Synthesis Options	
	🗌 Global	
	Out of context per IP	
	Out of context per <u>B</u> lock Design	
	Run Settings	
	Number of jobs: 4 🗸	
	Cancel	
		1
20) メニュー欄で"F	ile -> Export -> Export Hardware"の順番	でハードウェア
インフォメーショ	ンを書き出す。ここに PS の配置も含んでいる。	
	X	
×		





21) ポップアップしたダイアログに "OK"をクリックする。実験はただ PS のシリアルを 使って、PL が参加していないから、ここでは "Include bitstream"を作動させてない。





2) SDK 起動して、ファイルふが見える。 "system. hdf"というファイルがあって、Vivado ハ ードウェアデザインのメッセージが含んでいる。ソフト開発にも使用できて、PS 側ペリ フェラルのレジスタリストも見える。





低価格、高品質が不可能? 日昇テクノロジーなら可能にする



4) "Project name"は "hello" 書き込んで、他のはデフォルト値にする。 "Next" を クリックする。











6) SDK は "hello" という目次ぐを作成した。もう一つの目次ぐは "hello\_bsp" である。
 "hello\_bsp"の目次ぐで使えるもインフォメーション大量である。中の "BSP
 Documentation" に一部の PS ペリフェラルの API 説明が入っている。







7) "system.mss"をダブルクリックする。一部の PS ペリフェラルはサンプルも提供している。 これは対応のペリフェラルを学ぶ大切な資料である。







8) JTAG ケープルを開発ボードに接続して、UART 的 USB ケープルを PC に接続する。

9) PuTTY ソフトをシリアルターミナルデバッグツールにする。PuTTY は無料ダウンロードの ソフトである。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

Real PuTTY Configuration		×	
Category:			
Session	Basic options for your PuTTY se	ssion	
	Specify the destination you want to conne	ct to	
- Kevboard	Serial line	Speed	
Bell	COM3	115200	
Features	Connection type:		
⊡. Window	🔘 Raw 🔘 Telnet 🔘 Rlogin 🔘 SSF	I   Serial	
Behaviour	Load, save or delete a stored session		
Translation	Saved Sessions		
Selection			
Colours	Default Settings	Load	
Data		Save	
Proxy			
Telnet		Delete	
Hlogin			
Serial	Channel and an and		
	<ul> <li>Always</li> <li>Never</li> <li>Only on cl</li> </ul>	ean exit	
About	Open	Cancel	

10) Serial, Serial line を COM3 欄にデータを入れる。Speed 欄に 115200 を書き込む。 COM3 シリアル番号はデバイスマネージャに表示したように写す。"Open"をクリッ クする。

		3
	文件(F) 操作(A) 查看(V) 帮助(H)	
	a → Mei-PC > ← IDE ATA/ATAPI 控制器 > ☆ Jungo > ● ① 处理器	
	● 二 磁型級切解 ● 17 詳□ (COM 11 LPT) 17 Silicon Labs CP210x USB to UART Bridge (COM3) 17 USB Serial Port (COM7) USB Serial Port (COM7)	
	- 小学学/連環語(UCOMI) - 29 伊賀机 - 29 短辺器 	
$\wedge$	2 → ● 一章 刊版和放送到器 > - ● 創版和其他指针设备 > - ● 通用單行总线控制器	
V	<ul> <li></li></ul>	

### 11) 電気を入れる前に、開発ボードの起動モードを JTAG モードに設置する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



12) 開発ボードに電気を入れて、プログラムの作動に準備をする。出荷の時、開発ボードは既に プログラムがつけられている。ここで、作動モードを JTGA モードに選び、改めて電気いれをする。 "hello"を選択し、右ボタンを押せば、オプションがたくさん見える。本実験に使うのは"Run as" で、プログラムを起動するとこと。"Run as"にもオプションたくさんあって、一番目"Launch on Hardware (System Debuger)"を選択する。システムでデバッグをして、直接プログラムを作動する。



ホームページ: https://www.csun.co.jp



13) この時は PuTTY ソフトを観察する。アウトプットが表示されるかも、アウトプットがないかも 知らない。



14) システムのデバッグを保証するため、配置一つ追加必要がある。右ボタンを押し、 "Run As -> Run Configuration..."の順番で操作する。





15) "Reset entire system"を選択して、システムをリセットする。システムにまだ PL デザインがあったら、"Program FPGA"を選択してから、再び"Run"をクリックする。



17) 今回見知りの "Hello World" が画面に表示された。





18) "Run As"以外、"Debug As"もう使用できる。こうして、ブレイクポイントを設置でき、 シングルステップで作動できる。



19) Debug モードに入る。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

e kalla adl. Dakus kalla (na fallaunaid a Vilau CDV		
File Edit Source Refactor Navigate Search Project Run Xilinx Too	ols Window Help	
		Ouick Access
A E System Debugger using Debug bello elf on Local (Local)	🕻 🚧   17	H ALE SI SU
APU	Name	Type
a 🧬 ARM Cortex-A9 MPCore #0 (Breakpoint: main)		
0x001005ec main():/src/helloworld.c, line 55		
= 0x0010078c _start() = 0x0010078c _start()		
<b>=</b>		
ARM Cortex-A9 MPCore #1 (Suspended)	۰ ( m	•
@ xc/z055		<u>^</u>
	4	<del>*</del>
🕞 hallaundel a 😢 🔂 austan mar		
* I UART TYPE BAUD RATE		
*		stdio.h
* uartlite Configurable only in HW design		platform.h
<pre>* ps7_uart 115200 (configured by bootrom/bsp) */</pre>		💾 xil_printf.h
		main() : int
<pre>#include <stdio.h> #include "platform b"</stdio.h></pre>		
#include "xil_printf.h"		E
⊖int main()		
<pre>init platform();</pre>		
ncint("Hello Horld'));		
		-
🖳 Console 🕮 🧑 Tasks 🖳 SDK Terminal 🎇 Problems 🚺 Executables	SDK Log 1 Memo	iny 🕮 📲 🔛 🔝 🖼 👻 🗖 🗖
ICF Debug Virtual Terminal - ARM Cortex-A9 MPCore #1	Monitors 🖶 💥 💥 –	
	Writable Smart Insert 49 · 2	2 1
	Startinger 4312	

20) IDEを開発は他のC言語と同じ、ステップバイステップで作動するやブレイクポイントを設 Total

### 置できる。

Edit Source Refactor Navigate Search Project	un Xilinx Tools Window Hel	p		
• 🖩 🕼 🕷 • 🕥 • 🖄 🖿 💷 📲 🧏	Instruction Stepping Mode		] 🍫 🗢 • 🗢 •	
Debug 🛛 🔶	Resume at Line (C/C++)		s 🛛 💊 Breakpo 👭 Regi	sters 📷 XSCT Co 🗗
System Debugger using Debug_hello.elf on Local (Loc	Resume	F8		後 🔿
a 🔅 APU	L Suspend			Туре
ARM Cortex-A9 MPCore #0 (Breakpoint: main)	Terminate	Ctrl+F2		
0x001005ec main():/src/helloworld.c, line 5	Disconnect			
0x0010078c_start()	Resume Without Signal			
= 0x0010078c_start()	Step Into	F5		
ARM Cortex-A9 MPCore #1 (Suspended)	Step Over	F6	III	
	Step Return	F7		
	Run to Line	Ctrl+R		
	Use Step Filters	0.034		
elloworld.c 🖄 🖬 system.mss	Step Into Selection	Ctrl+F5		- D 📴 Outline 🛛
* UART TYPE BAUD RATE	Run	Ctrl+F11		
*	Debug	F11		💵 stdi
* uartlite Configurable only in HW desig	Run History			💶 plat
<pre>* ps7_uart 115200 (configured by bootrom */</pre>	Run As			🖬 xil_p
	Run Configurations			• mai
<pre>#include <stdio.h> #include "slatform b"</stdio.h></pre>				
#include "xil_printf.h"	Debug History	•		-
	Debug As	•		-
int main()	Debug Configurations			-
{	Toggle Breakpoint	Ctrl+Shift+B		
init_placionm(),	Toggle Line Breakpoint			
<pre>print("Hello World\n\r");</pre>	Toggle Method Breakpoint			-
4	Toggle Watchpoint	AL 145 141 1515		•
Console 🕱 🧔 Tasks 📃 SDK Terminal 🔝 Problems 🖹	Skip All Breakpoints	Ctrl+Alt+B	D SDK Log 🚺 Mer	nory 🕴 👔 📑 🖻
Debug Virtual Terminal - ARM Cortex-A9 MPCore #1	Remove All Breakpoints		Monitors 👍 😠 🕉	
	Breakpoint Types	•	A	



#### 21) 右上で IDE モードが変更できる。

ps_hello.sdk - Debug - hello/src/helloworld.c - Xilinx SDK		
le Edit Source Refactor Navigate Search Project Run Xilinx Tools Window Help		
3 - 📓 📾 🔯 - 🔍 - 🔍 💷 💷 🚳 📓 🐨 - 🕲 - 🕸	8 4 - 3	Quick Access 🔡 🖬 🖬
<ul> <li>Debug IS</li> <li>System Debugger using Debug_hello.elf on Local (Local)</li> <li>S APU</li> <li>APU</li> <li>ARM Cortex-A9 MPCore #0 (Breakpoint: main)</li> </ul>	O ↔ Variables ⊠ ⁰o Breakpo III Registers ■ Name	XSCT Co ◙ Emulatio ≧ Module
	< III	
<pre>Bhelloworld.c ☆ is system.mss U AAT TYPE BAUD RATE U AATTYPE BAUD RATE U artn5559 9600 wartlike Configurable only in Hel design ps7_uart 115200 (configured by bootcom/bsp) */ #include <stdio.h></stdio.h></pre>		P Outline 23 □ stdioch platform.h via printf.h main() : int
<pre>#include "pittorm.h" #include "xil_printf.h" @ int main() { init_platform(); print("Hello World\n\r"); </pre>		
Q Console ஜ ⊘ Tasks 및 SDK Terminal 🦹 Problems 🕡 Executables 💽 F Debug Virtual Terminal - ARM Cortex-A9 MPCore ≠1	SDK kap () Memory () Monitors () () () () () () () () () () () () ()	2 - V -
	40.00	

#### 6.4 実験のまとめ

本実験は簡単な Hello World で SDK の使い方を説明した。SDK は強い機能があって、一つ一つ で紹介できないが、使用の繰り返しでだんだん把握できると思う。

#### 6.5 よくある問題

- 6.5.1 vivadoに通じて SDK を起動後、ウィンドウがポップアップされていない
  - 1) Vivado ソフトをインストールする時はかならず SDK もインストールする。
- SDK を起動する前に sdk 目次ぐがあったら、SDK が起動できないことになる。この目次ぐ を削除してから試す。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

📙 .metadata	2018/3/5 15:03	文件夹	
Ji .Xil	2018/3/5 19:32	文件夹	
🎉 ps_hello.cache	2018/3/5 19:29	文件夹	
🌗 ps_hello.hw	2018/3/5 19:29	文件夹	
퉬 ps_hello.ip_user_files	2018/3/5 19:29	文件夹	
퉬 ps_hello.runs	2018/3/5 19:29	文件夹	
🍌 ps_hello.sdk	2018/3/5 19:32	文件夹	
ps_hello.sim	2018/3/5 19:29	文件夹	
퉬 ps_hello.srcs	2018/3/5 19:29	文件夹	
퉬 RemoteSystemsTempFiles	2018/3/5 15:03	文件夹	
ip_upgrade.log	2018/3/5 14:27	wrifile	5 KB
À ps_hello.xpr	2018/3/5 16:20	Vivado Project Fi	7 KB
SDK.log	2018/3/5 15:03	wrifile	1 KB
📄 vivado.jou	2018/3/5 19:32	JOU 文件	1 KB
📄 vivado.log	2018/3/5 19:44	wrifile	2 KB
📄 vivado_8944.backup.jou	2018/3/5 16:22	JOU 文件	7 KB
📄 vivado_8944.backup.log	2018/3/5 16:52	wrifile	15 KB



**第七章 PS で PL の LED を点灯する** 実験用 Vivado プロジェクトは "ps\_axi\_led"。

zynq の使用に一番大きい疑問はどうすれば PS と PL を結んで使用できることである。一 般的に、ほかの SOC チップに GPIO がある。本実験は AXI GPIO の IP コアを一つ使って、PS 側を AXI バスで PL 側の LED ライトをコントロールする時は。実験は簡単が、PL と PS と結 びつく方法を示す。

7.1 Vivado プロジェクトを作成する

1) "ps\_axi\_led" Vivado という名のプロジェクトを作成して、PS が AXI パスで LED ライト をコントロールすることを表示する。

2) Block デザインを作成する。



3) ZYNQ プロセッサーを追加する。





Zynq Block Design	← 🔍 🛬 🔶	0																		
PS-PL Configuration	Search: Q-																			
Peripheral I/O Pins		. 4	Bank 1	LVCMO	S 1.8V	~														
	Peripherals	15	16 17	18 19	20 21	22 23	24 25	28 27	28 29	30 31	32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47	48 49	
MIO Configuration	USB 1																US	B1		^
Clock Configuration	SD 0			SD0						SD0						SD0				
DDB Configuration	> SD 1						SD1		$\square$				SD1		$\square$				SD1	
DOX Configuration	> SPI 0		SPI0		ma	15			SPI0		mo				SPIO		mos			Ĩ.
SMC Timing Calculation	> SPI 1					SPI	1					SPI						SPI	1	
Interrupts	> UART 0	RTO		UARTO		UARTO		UARTO		UARTO		UARTO		UARTO		UARTO		UARTO		
	> 🗹 UART 1		UART1		UART1	$\square$	UART1		UART1		UART1		UART1		UART1	<u> </u>	UART1		UART1	ſIJ
	> I2C 0	200		12C0		12C0		12C0		12C0		12C0		12C0	ET.	12C0		12C0		1
	> 🗌 I2C 1		12C1		1201		1201		1201		1201		1201		1201		1201		1201	n"
-	> CAN 0																			-

### 7.1.2 DDR3 配置

 AX7010 は DDR3 の型番を "MT41J128M16 HA-125"に、AX7020 は DDR3 の型番を "MT41J256M16 RE-125"に配置する。ここの ddr3 タイプはボードでの ddr3 タイプではな く、 パラメータが一番近いタイプである。







低価格、高品質が不可能? 日昇テクノロジーなら可能にする

🔥 Re-customize IP			
ZYNQ7 Processing System	(5.5)		4
1 Documentation 🔅 Presets 📄	IP Location  🍈 Import XPS Settings		
Page Navigator _ DDR	Configuration		Summary Report
Zynq Block Design 🖌 🖕	✓ Enable DDR		
PS-PL Configuration	Search: Q-		
Peripheral I/O Pins	Name	Select	Description
۵	<ul> <li>DDR Controller Configuration</li> </ul>		<u> </u>
MIO Configuration	Memory Type	DDR 3 🗸	Type of memory interface. Refer to UG585 Zyng Technical Ref
Clock Configuration	Memory Part	MT41J256M16 RE-1 🗸	Memory component part number. For unlisted parts choose "C
	Effective DRAM Bus Width	32 Bit 🗸	Data width of DDR interface, not including ECC data width. Re
DDR Configuration	ECC	Disabled	Enables error correction code support. ECC is supported only
SMC Timing Calculation	Burst Length	8 🗸	Minimum number of data beats the controller should use whe
Interrunts	DDR	533.333333 🛞	Memory clock frequency. The allowed freq range is (200.0000
interrupto	Internal Vref		Enables internal voltage reference source. Disable to use exte
	Juntion Temperature (C)	Normal (0-85) 🗸 🗸	Intended operating temperature range. Controls the DDR refre
	<ul> <li>Memory Part Configuration</li> </ul>		
	DRAM IC Bus Width	16 Bits	Width of Individual DRAM components.
	DRAM Device Capacity	4096 MBits	Storage capacity of individual DRAM components.
	Speed Bin	DDR3_1066F	Speed bin of the individual DRAM components.
	Bank Address Count (Bits)	3	Number of bank address pins.
	<	40	×1
			OK Cancel



#### AX7020 ddr3 选择

2) AXI GPIO の IP コアを一つ追加する。



ZYNQ7 Processing System



"All Outputs"を選択する。 LED をコントロールしたから、アウトプットするだけていい。 2) "GPIO Width"欄で4を入れて、4つのLEDをコントロールする。 'OK"をクリックする。

1) Documentation 🛛 📄 IP Location		
Show disabled ports	Component Name axi_gpio_0	
	All Inputs	
	✓ All Outputs	
	GPIO Width 4 (1 - 32)	
	Default Output Value 0x0000000 (i) [0x00000000.0xFFFFFFF	
		]
+ S_AXI	Enable Dual Channel	
• s_axi_aresetn	GPIO 2	
	All Inputs	
	All Outputs	
		1
	Default Tri State Value 0xFFFFFF 0 [0x0000000.0xFFFFFFF	י ר
		· ·
	Enable Interrupt	
		OK Cancel
		OK Cancel
3) "Run Connection A Diagram × Address Eddor	Automation"をクリックして、一部の自動連線ができ	ok Cancel
3) "Run Connection A Diagram × Address Edder Q Q S N Q Q * Designer Assistance available: R	Automation"をクリックして、一部の自動連線ができ (	ok Cancel
3) "Run Connection A Diagram × Address Editor Q Q X X Q Q * Designer Assistance available: R	Automation"をクリックして、一部の自動連線ができ (	ok Cancel
3) "Run Connection A Diagram × Address Editor Q Q X: S: Q Q * Designer Assistance available." R	Automation"をクリックして、一部の自動連線ができ ( *   +   P、 / ビ C ( な ) む   む   ひ   ひ   ひ   ひ   ひ   ひ   ひ   ひ	ok Cancel
3) "Run Connection A Diagram × Address Edder × Q Q S: N: O Q * Designer Assistance available. R	Automation"をクリックして、一部の自動連線ができ (ま) + P ア ア C C G C Run Block Automation Run Connection Automation axi_gpio_0	ok Cancel ≛る。
3) "Run Connection A Diagram × Addross Eddor Q Q 2: 02 00 * Designer Assistance available. R	Automation"をクリックして、一部の自動連線ができ (*) + P ア ア C C G C Aun Block Automation Run Connection Automation axi_gpio_0	ok Cancel ≛る。
3) "Run Connection A Diagram × Address Editor × Q Q 3: No Q Q * Designer Assistance available: R	Automation"をクリックして、一部の自動連線ができ ( + I P ) ア I C I C I C I C I C I C I C I C I C I	ok Cancel ≛る。
3) "Run Connection A Diagram × Addross Eddor Q Q X Q Q * Designer Assistance available. R	Automation"をクリックして、一部の自動連線ができ (ま) + P ア ア C G G Run Block Automation Run Connection Automation axi_gpio_0 (+ S_AXi s_axi_aresetn AXI GPIO	ok Cancel ≛る。
3) "Run Connection A Diagram × Address Editor Q Q X NO 2000 * Designer Assistance available: R	Automation"をクリックして、一部の自動連線ができ (*) + P、 ア C C C C Run Block Automation Run Connection Automation	ok Cancel ≛る。
3) "Run Connection A Diagram × Address Editor Q Q 2: 0.000 * Designer Assistance available. R	Automation"をクリックして、一部の自動連線ができ (ま) + P ア ビ C ビ P Run Block Automation	ok Cancel ≝る。
3) "Run Connection A Diagram × Address Editor Q Q X & Q A * Designer Assistance available." R	Automation"をクリックして、一部の自動連線ができ (*) + 「 ) 」 ご ご ご ご Run Block Automation Run Connection Automation	ok Cancel ≛る。
3) "Run Connection A Diagram × Addrees Editor × Q Q 3: No Q Q * Designer Assistance available: R	Automation をクリックして、一部の自動連線ができ (* * * * * * * * * * * * * * * * * * *	ok Cancel ≛る。
3) "Run Connection A Diagram × Addross Eddor Q Q X Diagram / * Designer Assistance available."	Automation' をクリックして、一部の自動連線ができ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	oK Cancel ≛る。 sing_system7_0 DDR FIXED_IO
3) "Run Connection A Diagram × Addrees Editor × Q Q 3: No 2 Q * Designer Assistance available: R	Automation をクリックして、一部の自動連線ができ (* * * * * * * * * * * * * * * * * * *	oK Cancel ≛る。 sing_system7_0 DDR FIXED_IO M_AXI_GPO M_AXI_GPO M_AXI_GPO



4) 自動的に接続するポートを選択する。ここはどこ全部選んで、"OK"をクリックする。

X Run Connection Automation Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its λ configuration options on the right. Q. 🛨 🌲 All Automation (2 out of 2 selected) ✓ ✓ ₽ axi\_gpio\_0 🖌 🕀 GPIO V 🕼 S\_AXI Select an interface pin on the left panel to view its options ? Cancel 5) "Run Block Automation" をクリックする。 Diagram × Address Editor × Q Q X X ↔ Q X + · · / / / C 9 ÷ 🕈 Designer Assistance availab axi\_gpio\_0 S\_AXI axi\_a rst\_ps7\_0\_50M ps7\_0\_axi\_periph ps7\_0\_axi\_perig est\_sync\_clk mb\_reset \_\_\_\_\_\_ bus\_struct\_reset[0:0] \_\_\_\_\_ peripheral\_reset[0:0] \_\_\_\_\_ ext\_reset\_in
 aux\_reset\_in
 mb\_debug\_sys\_rst
 dcm\_locked nterconnect\_aresetn[0:0] Pro tem Reset ocessing\_system7\_0 DDR + FIXED\_IO + M\_AXI\_GP0 + FCLK\_CLK0 FCLK\_RESET0\_N xi\_gp0\_aclk ZYNQ ZYNQ7 Processing System "OK"をクリックする 6)



Q, , , , , , , , , , , , , , , , , , ,	Provide and the second s
✓ ✓ All Automation (1 out of 1 selected)	Description
	This option sets the board preset on the Processing System. All current properties will overwritten by the board preset. This action cannot be undone. Zynq7 block automation applies current board preset and generates external connections for FIXED_IO, Trigger and DDR interfaces.
	NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this b if you wish to retain previous configuration.
	Instance: /processing_system7_0
	Options
	Make Interface External: FIXED_IO, DDR
	<u>C</u> ross Trigger In: Disable v
	Cross <u>T</u> rigger Out: Disable ↓
3	С.
)ptimize Routing"をクリッ	クして、レイアウトを最適化できる。
Optimize Routing"をクリッ	クして、レイアウトを最適化できる。
Dptimize Routing" をクリッ	クして、レイアウトを最適化できる。
Optimize Routing"をクリッ agram × Address Editor ×	クして、レイアウトを最適化できる。



8) GPI0 ポートの名前を変更する。







11)

12)

## 株式会社日昇テクノロジー



小一トートノンを聞る。





2) ファイルネームは led にする。



Specify or create constraint files for physic	cal and timing constraint to add to your project.	<u> </u>
Specify constraint set	(active)	
+ -  +   +		
	A Create Constraints File	
	Create a new constraints file and add it to your	
	project	
	Eile type: 🗈 XDC 🗸	
	File name: led 🛛 🔍 📑	
	Fil <u>e</u> location: 🛜 <local project="" to=""></local>	
Copy constraints files into project	? ОК Сапсе	

3) led. xdc に内容を追加する。ポートネームはからなずトップファイルポートと一致する。



2) ハードウェアを書き出す。







6) 慣れていない AXI GPIO をどうコントロールする?SDK 内部のサンプルを試しておこう。
7) "system.mss"をダブルクリック、 "axi\_gpio\_0"を見つかる。 "Documentation" をクリ


platform\_config.h

platform.c

Hardware Specification: F:\ax7350\demo\ps\_axi\_led\ps\_axi\_led.sdk\design\_1\_wrapper\_hw\_platfo

ックして、それに関するファイルを見る。 ユ <del>現体出す</del> i i # 「Import Examples"をクリックす る。

Target Processor: ps7\_cortexa9\_0

🔪 lscript.ld	Operating System
Xilinx.spec	Board Support Package OS.
🛛 🎒 axi_led_bsp	Name: standalone
b i BSP Documentation	Version: 6.3
b B ps7_cortexa9_0	Description: Standalone is a simple, low-level software layer. It provides access to basic pro
à Makefile	features of a hosted environment, such as standard input and output, profiling,
System.mss	Documentation: <u>standalone v6 3</u>
design_1_wrapper_nw_platform_0	
	Peripheral Drivers
	Drivers present in the Board Support Package.
	axi_gpio_0 gpio <u>Documentation</u> <u>Import Examples</u>
	ps7_afi_0 generic <u>Documentation</u>
	ps7_afi_1 generic <u>Documentation</u>
	ps7_afi_2 generic <u>Documentation</u>
	ps7_afi_3 generic <u>Documentation</u>
	ps7_coresight_comp_0 coresightps_dcc <u>Documentation</u>
	ps7_ddr_0 ddrps <u>Documentation</u>
	ps7_ddrc_0 generic <u>Documentation</u>
	ps7_dev_cfg_0_devcfgDocumentation_Import_Examples
	ps7_dma_ns_dmaps <u>Documentation_Import_Examples</u>
8) ポップアップされたダイアログはいく	つかのサンプルがあり、ネームからどんな内容が
見当がつける。ここは一番目の"xgpio_exa	mple"を選ぶ。
son Examples for gpio	
Import Examples	
Import Examples	
Import Examples Select the examples to be imported into work	spsace. Double click on the file to view the source.
<b>Import Examples</b> Select the examples to be imported into work	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Import Example         Import Example	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Import Example         Import Example	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Import Example         Import Example	spsace. Double click on the file to view the source.
Import Examples         Select the examples to be imported into work         ▷       ♥ 🗁 xgpio_example         ▷       □       >> xgpio_intr_tapp_example         ▷       □       >> xgpio_low level_example         ▷       □       >> xgpio_tapp_example	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the examples to be imported into work Select the example Select the example of the example Select the example of the example Select the example of the example	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example of the example Select the example Select the example Select the example Select the example Select the example Select the example of the example Select the example of the example Select the example of th	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example Select the example of the example Select the example of the example Select the example of the example	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Import Example         Import Example	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example Select th	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example Select the example Sele	spsace. Double click on the file to view the source.
Import Examples         Select the examples to be imported into work         ▷       ☑ > xgpio_example         ▷       ☑ > xgpio_intr_tapp_example         ▷       ☑ > xgpio_low level_example         ▷       ☑ > xgpio_tapp_example         ▷       ☑ > xgpio_tapp_example	spsace. Double click on the file to view the source.
Import Examples         Select the examples to be imported into work         ▷       ☑ ➢ xgpio_example         ▷       ☑ ➢ xgpio_intr_tapp_example         ▷       ☑ ➢ xgpio_low level_example         ▷       ☑ ➢ xgpio_tapp_example	spsace. Double click on the file to view the source.
Import Examples         Select the examples to be imported into work         ▷       ☑ > xgpio_example         ▷       ☑ > xgpio_intr_tapp_example         ▷       ☑ > xgpio_low level_example         ▷       ☑ > xgpio_tapp_example	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example Select the example Sele	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example Select the example Sele	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example Select th	spsace. Double click on the file to view the source.
Import Examples Select the examples to be imported into work Select the example Select th	spsace. Double click on the file to view the source.
Import Examples         Select the examples to be imported into work         Import Example	spsace. Double click on the file to view the source.
Import Examples         Select the examples to be imported into work         Import Example	spsace. Double click on the file to view the source.
Import Examples         Select the examples to be imported into work <ul> <li></li></ul>	psace. Double click on the file to view the source.

9) サンプルは簡単で、少ないコードで、AXI GPIOの操作を完成した。





中に GPIO に関する API 関数を使用している。ファイルで詳しいことを了解できる。この関数を選んで F3 を押せば具体的な定義も見える。これだけのインフォメーションがあって、まだ AXI GPIO の使用を理解できないなら、C 言語基礎を補充すべきである。

#### 7.4 ダウンロード及びデバッグ

1) SDK は数多くのサンプルを提供できるが、一部のサンプルは自分で変更するものである、この簡単な LED サンプルは変更せず、作動してみよう。予想の効果とかなり距離があって、エラーの提示も出た。





株式会社日昇テクノロジー

2) 前の内容はもう "Run As" がシステムのリセットに効果抜群ということを説明した。 PL のデ ザインは "Program FPGA" が必要である。PL が何度も更新されたら、改めてハードウェアを書き出す のが忘れないよう。下の画像のように配置して、再起動する。開発ボードの LED1 が素早くブリンクし ている。

	Name: System Debugger using Debug_axi_led_bsp_xgpio_example_1.elf on Local
type filter text	📀 Target Setup 🔲 Application 🕬 Arguments 🚾 Environment 🚋 Symbol Files 🦆 Source 🔭
formance Analysis get Communication Framework	Debug Type: Standalone Application Debug 🔻
nx C/C++ application (GDB) nx C/C++ application (System De nx C/C++ application (System De	Connection: Local   New
System Debugger using Debug	Hardware Platform: design_1_wrapper_hw_platform_0
	Bitstream File: design_1_wrapper.bit Search Browse Gene
	Initialization File: ps7_init.tcl Search Browse
	FPGA Device: Auto Detect Select
	PS Device: Auto Detect Select
	Summary of operations to be performed
	Program FPGA
	Image: Second and Second an
	Run ps7_post_config 4. Runs ps7_post_config. Enables level shifters from PL to PS. (Recommen to use this option only after system reset or board power ON).
	5. All processors in the system will be suspended, and Applications will be
	1) ps7_cortexa9_0
	(F:\ax7350\demo\ps_axi_led\ps_axi_led.sdk\axi_led_bsp_xgpio_example_1\ ug\axi_led_bsp_xgpio_example_1.elf)
	III
Filter matched 6 of 10 items	Revert Appl
?	Run Clo



#### 7.5 実験のまとめ

実験で PS は AXI パスに通じて PL をコントロールすることを了解した。しかし、ZYNQ の優位を 表現されていない見たい。その原因は、LED ライトをコントロールすることなら、ARM も FGPA も 楽に完成できる。でも LED をシリアルポートにしたらどうなるか?100 重シリアルポート通信や 8 重イーサネットなどのアプリを制御するようなことは、どこの SDC も完成できないと思う。こ れは ZYNQ しかできないこと。これも ZYNQ と普通の SDC との違いである。

ホームページ:<u>https://www.csun.co.jp</u>



#### 第八章 PS タイマーインタラプト実験

実験用 Vivado プロジェクトは "ps\_timer"。

たくさんの SDC は内部にタイマーがある。ZYNQ の PS にもある。ZYNQ にはどんなペリフェラル があるか、これらのペリフェラルはどんな特性あるか?それはエンジニアが了解すべくことなの で、よく xilinx のマニュアル UG585 を読むのがお勧めです。

#### 8.1 Vivado プロジェクトを作成する

- 数個 Vivado プロジェクトを作成する。中にも同じ仕事は大量にあって、最優な解 決法はプロジェクトをコピーして、変更する。 "ps\_hello"を開く。
- 2) メニューで"File -> Save Project As... をクリックする。🛌



3) ポップアップされたダイアログに新規プロジェクト名"ps\_timer"を入れる。サ ブディレクトリの作成を選択する。PSのタイマーはピンでアウトプットをするとき必要 がないので、ピンを配置しない。



\lambda Save Project As	production and the	
Save this project to	a new name and location.	4
r.		
<u>P</u> roject name:	ps_timer	
Project location:	F:/ax7350/demo	<b>8</b>
✓ Create project	t subdirectory	
Project will be cre	ated at: F:/ax7350/demo/ps_timer	
	ок	Cancel

#### 8.2 SDK を プ ロ グ ラ ミ ン グ

- 1) SDK を起動する。前のサンプルと違っていて、ここはハードウェアプラットフォームの
  - ファイルが一つ多くなった。



2) 他人の SDK プロジェクトを使うときも似たようなことがあって、全てを削除する。

		×		
Projec	Expl	orer 🛛 🕒 🚖 🔽 🗵 🗖	le helloworld.c	system.ms
de 💷 de	sign_1	_wrapper_hw_platform_0	ps7_d	ma_s dmaps
de de	sign_1	wrapper_hw_platform_1	ps7_globaltin	ner_0 generic
he he			ps7_g	pv_0 generic
		New	•	t_0 generic
		Go Into		g_0 generic
		Сору	Ctrl+C	c_0 generic
	Ē	Paste	Ctrl+V	c_0 generic
	×	Delete	Delete	0_0 generic
		Move		u_0 generic
		Rename	F2	h_0 generic
	2	Import		n_1 generic
		Export		c_0 generic
	11			IC U SCUQIC

3) ファイルも削除する。



Son Delete Resources
Are you sure you want to remove these 4 projects from the workspace?
☑ Delete project contents on disk (cannot be undone)
Project locations:
F:\ax7350\demo\ps_timer\ps_timer.sdk\design_1_wrapper_hw_platform_0 F:\ax7350\demo\ps_timer\ps_timer.sdk\design_1_wrapper_hw_platform_1 F:\ax7350\demo\ps_timer\ps_timer.sdk\hello F:\ax7350\demo\ps_timer\ps_timer.sdk\hello_bsp
Preview > OK Cancel

4) もう一度 Vivado に SDK を起動した、新しいハードウェアプラットフォームメッセージが 見えてくる。

ps_timer.sdk - C/C++ - design_1_wrapper_hw_platform_0/system.hdf - Xilinx SDK
File Edit Navigate Search Project Run Xilinx Tools Window Help
🖆 🕶 🔚 🐚   🕲 🖛 🔦 🖛 🔯 🖬 🛃 🖬 🚱 🍬 🕶 🔕 🖛 🚱
🖻 Project Explorer 🛛 💦 🦳 🦳 🔚 🖻
<ul> <li>design_1_wrapper_hw_platform_0</li> <li>ps7_init_gpl.c</li> <li>ps7_init_gpl.h</li> <li>ps7_init.c</li> <li>ps7_init.html</li> <li>ps7_init.tcl</li> <li>ps7_init.tcl</li> </ul>
5) 新規プロジェクトを作成する。"ps_timer_test"という名で、テンプレートは Hello World にする。
File Edit Navigate Search Project Run Xilinx Tools Window Help
📑 👻 🔚 🐚   🗞 × 🌭 × 📾   🔌   🖸   🚔 🔯 🖬 🚱 × 💽 × 🔗 × I
🕒 Project Explorer 🛛 📄 🔄 🔽 🔍 🗖
Image: book and the second
⊳ 👺 ps_timer_test
▷ 🧶 ps_timer_test_bsp

6) コーディングのときが来て、どこから始めるかは知らないでしょう。タイマーの使



## い方も、インタラプトする方法も見当をつけない。いつもの通り、例をみよう。

ps7_ram_0 generic	Documentation
ps7_ram_1 <mark>generic</mark>	Documentation
ps7_scuc_0 generic	Documentation
ps7_scugic_0 scugic	Documentation Import Examples
ps7_scutimer_0 scutimer	Documentation Import Examples
ps7_scuwdt_0 scuwdt	Documentation Import Examples
ps7_slcr_0 generic	Documentation
ps7_uart_0 uartps	Documentation Import Examples
ps7_xadc_0 xadcps	Documentation Import Examples
Libraries	

7) タイマーインタラプトの例一つあった。この例はインタラプトの例ってことはどう 確認するか? "intr"から推測した。ということで、基礎がとても重要で、把握できないと、 サンプルも探せない。



次はコードを読むと変更することである。すぐにコードを全部理解できないのも当然である。 これから使用中で何度も練習するしかない。

8) 本実験はタイマーが一秒ことにインタラプトして、メッセージをプリントアウト する。30 秒後おわり。まずカウンターを変更するから。最大値は CPU 頻度の半分にし て、つまりカウンターのクロック頻度値になる。こうして、1秒ことにインタラプト ことになる。





#include "xparameters.h" #include "xscutimer.h"
#include "xscugic.h"
#include "xil\_exception.h" #include "xil\_printf.h' \* The following constants map to the XPAR parameters created in the \* xparameters.h file. They are only defined here such that a user can easily \* change all the needed parameters in one place. #ifndef TESTAPP GEN #define TIMER\_DEVICE\_ID #define INTC\_DEVICE\_ID #define TIMER\_IRPT\_INTR XPAR\_XSCUTIMER\_0\_DEVICE\_ID XPAR\_SCUGIC\_SINGLE\_DEVICE\_ID XPAR\_SCUTIMER\_INTR #endif #define TIMER\_LOAD\_VALUE (XPAR\_PS7\_CORTEXA9\_0\_CPU\_CLK\_FREQ\_HZ/2 - 1) /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Macros (Inline Functions) Definitions \* 9) カウンターの回数は3から30に変更する。 LastTimerExpired = TimerExpired; / \* If it has expired a number of times, then stop the timer \* counter and stop this example. \*/ if (TimerExpired == 30) { XScuTimer\_Stop(TimerInstancePtr); break; } }  $(\Box)$ \* Disable and disconnect the interrupt system. TimerDisableIntrSystem(IntcInstancePtr, TimerIntrId); プリントのインフォメーションを追加する。 static void TimerIntrHandler(void \*CallBackRef) XScuTimer \*TimerInstancePtr = (XScuTimer \*) CallBackRef; check if the timer counter has expired, checking is not necessary since that's the reason this function is executed, this just shows \* how the callback reference can be used as a pointer to the instance \* of the timer counter that expired, increment a shared variable so \* the main thread of execution can see the timer expired. \*/ if (XScuTimer\_IsExpired(TimerInstancePtr)) { XScuTimer ClearInterruntStatus(TimerInstancePtr); printf(" %d Second\n\r",TimerExpired); limerFynired±±• if (TimerExpired == 30) { xscurimer\_DisableAucoReload(TimerInstancePtr); 3 } }

10)



#### 8.3 ダウンロードとデバッグ

- 1) PuTTY シリアルポートターミナルを開く。
- 2) デバッグプログラムのダンロード前の章でもう説明した。





SCU Timer Interrupt Example Test	
0 Second	
1 Second	
2 Second	
3 Second	
4 Second	
5 Second	
6 Second	
7 Second	
8 Second	
9 Second	
10 Second	
11 Second	
12 Second	
13 Second	
14 Second	
15 Second	
16 Second	
17 Second	
18 Second	
19 Second	
20 Second	
21 Second	
22 Second	
23 Second	
24 Second	
25 Second	
26 Second	
27 Second	
28 Second	
29 Second	
Successfully ran SCU Timer Interrupt Example Test	

#### 8.4 実験纏め

実験中 SDK のサンプルうを少し変更するしてタイマーとインタラプトの応用をかんせいした。操作簡単に見えるが、豊かな知識が含んている。タイマー、インタラプトの原理詳しく知る必要がる。これらの基本知識は ZYNQ を把握する必要条件である。





#### 第九章 PL キーインタラプト

実験用 Vivado プロジェクトは"ps\_axi\_key"。

前の章にしたタイマーインタラプト実験は PS 内部のインタラプトで、本実験のインタラプトは PS から出てくる。PS は PL からのインタラプト信号は 16 も受けられて、立ち上がりエッジとハ イレベルを通じてトグルする。本実験はボタンインタラプトを使用して LED をコントロールする。

Source	Interrupt Name	IRQ ID#	Status Bits (mpcore Registers)	Required Type	PS-PL Signal Name	ı/o
DI	PL [2:0]	63:61	spi_status_0[31:29]	Rising edge/ High level	IRQF2P[2:0]	Input
SourceInterrupt NamePLPL [2:0]PL [7:3]TimerTimerTTC 1DMACDMAC[7:4]USB 1Ethernet 1Ethernet 1Ethernet 1IOPI2C 1SPI 1UART 1CAN 1	PL [7:3]	68:64	spi_status_1[4:0]	Rising edge/ High level	IRQF2P[7:3]	Input
Timer	TTC 1	71:69	spi_status_1[7:5]	High level		~
DMAC	DMAC[7:4]	75:72	spi_status_1[11:8]	High level	IRQP2F[27:24]	Output
	USB 1	76	spi_status_1[12]	High level	IRQP2 <mark>F[</mark> 7]	Output
	Ethernet 1	77	spi_status_1[13]	High level	IRQP2F[6]	Output
	Ethernet 1 Wake-up	78	spi_status_1[14]	Rising edge	IRQP2F[5]	Output
TOP	SDIO 1	79	spi_status_1[15]	High level	IRQP2F[4]	Output
101	I2C 1	80	spi_status_1[16]	High level	IRQP2F[3]	Output
	SPI 1	81	spi_status_1[17]	High level	IRQP2F[2]	Output
	UART 1	82	spi_status_1[18]	High level	IRQP2F[1]	Output
	CAN 1	83	spi_status_1[19]	High level	IRQP2F[0]	Output
PL	PL [15:8]	91:84	spi_status_1[27:20]	Rising edge/ High level	IRQF2P[15:8]	Input
SCU	Parity	92	spi_status_1[28]	Rising edge	~	~
Reserved	~	95:93	spi_st <mark>at</mark> us_1[31:29]	~	~	~

#### 9.1 Vivado プロジェクトを作成する

1) 本実験使用している Vivado プロジェクトは "ps\_axi\_led" だけを使う。この プロジェクトにボタンインプット用の AXI GPIO を追加すればいい。メニューの "File -> Save Project As..."をクリックする。

	<u>F</u> ile	<u>E</u> dit	F <u>l</u> ow	<u>T</u> ools	<u>W</u> indow	La <u>v</u> out	<u>V</u> iew	<u>H</u> elp	Q- Quick Acce
$\bigcirc$		<u>N</u> ew Pro <u>O</u> pen Pr Open <u>Re</u> Open Ex	ject oject ecent Pro <u>a</u> mple P	oject roject	Þ	ECT MAI	∑ NAGER -   ♦   -	ps_axi_led	• 0
		Write Pro Archive F Close Pr Open Ch	pject As Project Tcl. Project roject nec <u>k</u> poin	t	Þ	> @ > @ Const	n Source design_' traints (1) ation Sou	es (1) 1_wrapper ) urces (1)	(design_1_wrap;

2) 新規プロジェクトは "ps\_axi\_key" にする。







6) ポートネームは keys に変更する。







- 10) デザインを保存して、bit ファイルを生成する。
- 9.2 ダンロードとデバッグ

1) SDK を起動する、他のプロジェクトからコピーしたもので、SDK に必要ないファイルが入っている。それを全部削除してから、再起動する。







# ₩₩₩₩₩ 株式会社日昇テクノロジー

New Project		- 0 X	
Anniantian Design			
Application Project		G	
Create a managed make	application project.		
Desired and the second			
Project name: key_test			
Use default location			
Location: F:\ax7015\dem	no\ps_axi_key\ps_axi_key.sdk\key_test	Browse	
Choose file sys	tem: default 👻		
OS Platform: standalor	ne	•	
New Project Application Project Create a managed make application project. Project mane: [ex_test U do default location Locator flag typernel and heypen, and keyvelkkey_test Decrease flag typernel default Create a managed make application project. Create a managed make application project. Create a managed make application project. Project mane: [ex_test U do default location Locator flag typernel default Create a managed make application project. Project mane: [ex_test Create a managed make application project. Create application project. Create a managed make application project. Create a managed make application project. Create applicatio			
Project name: key_test     Project name: key_test nam			
Deserves	-7		
Processor: p	s/_cortexa9_0		
T 10.0			
larget Software			
Language:			
Compiler:	32-bit 👻		
Hypervisor Guest:	N/A 👻		
Board Support Package	e:      O Create New key_test_bsp		
	⊘ Use existing		
			•
<	Back Next > Finish	Cancel	
4) 前の言っ	た通り、SDK プログラムに慣れ	っていない	状況で、できるだけ SDK 内部のサンプ
ルを使って変	更する。		
ן Project Explorer ב שלים design_1_wrap	aper_hw_platform_0	system.hdf	system.mss 🛛
key_test 1 key_test		Operating System Board Support Pac	ckage OS.
i BSP Docum	nentation	Name: s	tandalone
Makefile	32.0	Version: 6 Description: S	5.5 Standalone is a simple, low-level software layer. It provides a
🔝 system.mss		b Documentation:	pasic features of a hosted environment, such as standard inp <u>standalone v6 5</u>
·		Peripheral Drivers	•
		Drivers present in t	the Board Support Package.
		axi_gp axigp	pio_0 gpio <u>Documentation Import Examples</u>
		ps7_e	afj. O service
		ps7_i ps7_i	ari_x generic afi_2 generic
		ps7_a ps7_coresight_com	afi_3 generic np_0 coresightps_dcc <u>Documentation</u>
		ps7_d	ldr_0 ddrps <u>Documentation</u>
		ps7_dd ps7_dev_c	rfg_0 devcfg <u>Documentation Import Examples</u>
		ps7_dm ps7_dn	a_ns dmaps <u>Documentation Import Examples</u> ma_s dmaps <u>Documentation Import Examples</u>
		ps7_globaltim	er_0 generic
		637 U	···-



5) "xgpio\_intr\_tapp\_example" を選択する。



7) 下の図に従って GPIO とインタラプト番号のマクロ定義をへんこうする。



8)







9) シリアルターミナルを開き、プログラムを起動する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



10) 長時間ボタンを押さないと、シリアルポールは "No button pressed."を表示する。 "PLKEY1" を押すと、 "Successfully ran Gpio Interrupt Tapp Example" が表示される。





#### 9.3 実験まとめ

PL 側は PS にインタラプト信号を伝送でき、PL と PS のデータ交換 PL の効率を高めた。大数、 低遅延の APP にインタラプト処理が必要である。

ホームページ:<u>https://www.csun.co.jp</u>



#### 第十章 イーサネット実験(LWIP)

実験用 Vivado プロジェクトは "net\_test"。

開発ボードに1重ギガイーサネットがあって、RGMIIインタフェースで接続する。本実 験は SDK 内部の LWIP テンプレートでギガイーサネット TCP 通信する方法を説明する。LWIP は軽量級プロトコルが、一度も使用したことがないなら使うときは少し難しい。こちらの意見 は LWIP に関する知識を了解すべき。

10.1 Vivado プロジェクトを作成

1) 新規 vivado プロジェクト "net\_test"を作成して、ZYNQ を追加する。前の教程 にシリアルポールを配置する。詳しいパラメータはサンプル内部のプロジェクトを 参考できる。

#### 10.1.1 PS 側のイーサネット配置

1) "EnetO" (MI016-MI027) と "MDIO" (MI052-MI053) を起動させる。

Re-customize IP	stem (5.5)												
Documentation 🔅 Pres	ets 📄 IP Location (	🚳 Import XP	S Settings										
Page Navigator —	Peripheral I/O F	Pins									Sumr	nary Rep	port
Zynq Block Design	+ Q ¥	<b>e</b>   <b>0</b>											
PS-PL Configuration	<u>S</u> earch: Q												
Peripheral I/O Pins			Bank 1	LVCMOS 1	.8V 🗸								
	Peripherals	13 14	15 16 17	18 19 20	21 22	23 24 25	26 27	28 29 3	0 31 3	2 33	34 35	36 37	38
AIO Configuration	Quad SPI Flat												
Clock Configuration	SRAMMOR F					SRAN	I/NOR Flash,	addr[0-24]					
ODR Configuration	NAND Flash							1					
SMC Timing Calculation	Ethernet 0				Enet0								
	Fthornot 1												
nterrupts							_	-	-	Ene	et1	-	
×	USB 1				-					USI	30		
	> SD 1			SD0			_		SD0				
	> SPI 0	51				SD1		0.510				SD1	
	> SPI 1		SPIO		mos	·D14		SPIO		mos	CRIT		
		1				971 (					aril		>

2) Enet0 のレベル標準を STL 1.8V にし、Speed を fast に変更する。これらのパラメータ は非常に重要なので、変更しないと、ネットが繋がらない。



ZVNOZ Brossesing Su	atom (E.E.)						
Documentation      Decumentation	stem (5.5) sets 📄 IP Location 🊳 Impo	rt XPS Settings					
Dana Navinatar —							Gumman
	wio configuration						Summary
Zynd Block Design	Bank 0 I/O Voltage LVCMO	S 3 ♥	Bank 1 I/O Volta	ige LVCMOS 1	~		
PS-PL Configuration	← Q 품 ≑ +	: 0					
Peripheral I/O Pins	Search: Q-						
MIO Configuration	Peripheral	ю	Signal	Ю Туре	Spe	eed	Pullup
Clock Configuration	> 🖉 MDIO	MIO 52 53	*				
	Enet 0	MIO 16	tx_clk	HSTL 1.8V	<ul> <li>✓ fas</li> </ul>	t 🗸	en 🗸
DDR Configuration	Enet 0	MIO 17	txd[0]	HSTL 1.8V	👻 fas	t v	en 🗸
SMC Timing Calculation	Enet 0	MIO 18	txd[1]	HSTL 1.8V	<ul> <li>✓ fas</li> </ul>	t v	en 🗸
Interrupts	Enet 0	MIO 19	txd[2]	HSTL 1.8V	<ul> <li>✓ fast</li> </ul>	t ~	en 🗸
	Enet 0	MIO 20	txd[3]	HSTL 1.8V	<ul> <li>✓ fas</li> </ul>	t v	en 🗸
	Enet 0	MIO 21	tx_cti	HSTL 1.8V	<ul> <li>✓ fas</li> </ul>	t v	en 🗸
	Enet 0	MIO 22	rx_clk	HSTL 1.8V	✓ fas	t v	en 🗸
	Enet 0	MIO 23	rxd[0]	HSTL 1.8V	✓ fas	t v	en 🗸
	Enet 0	MIO 24	rxd[1]	HSTL 1.8V	✓ fas	t v	en 🗸
	Enet 0	MIO 25	rxd[2]	HSTL 1.8V	✓ fasi	t v	en 🗸
	Enet 0	MIO 26	rxd[3]	HSTL 1.8V	✓ fast	t v	en 🗸
	Enet 0	MIQ 27	rx_ctl	HSTL 1.8V	✓ fast	t v	en 🗸
ファイルを作成 ンプ	esign Sources (1)	1 bd) (0) ode Properties	Ctrl+E				
ファイルを作成 くっし くっこ くっこ こ	esign Sources (1) sources (1) source No source No source No Source No Source No Source No Source No Source No Source No Sources (1)	1. bd) (0) ode Properties	Ctrl+E Alt+O				
ファイルを作成 く く く く く く く く く く く く	esign Sources (1) sources (1) source No source No source No source No source No source No source No source No co Co Co Co Co Co Co Co Co Co C	1. bd) (0) ode Properties )L Wrapper	Ctrl+E Alt+O				
ファイルを作成 、、の 、、この 、、この 、、この 、、この 、、この 、、この 、、この	esign Sources (1) esign Sources (1) onst co C Open File imult Create HE sir View Insta	1. bd) (0) ode Properties )L Wrapper intiation Template	Ctrl+E Alt+O		-0		
ファイルを作成 、 D 、 C 、 C 、 S 、 S 、 S	esign Sources (1) esign Sources (1) onst Source No co Co Open File imula Create HE sit View Insta Generate	1. bd) (0) ode Properties DL Wrapper Intiation Template Output Products	Ctrl+E Alt+O		-0		
ファイルを作成 、、・ ク 、・ こ 、 、 、 、 、 、 、 、 、 、 の 、 、 の 、 、 の 、 、 の 、 、 の 、 、 の 、 、 の の 、 、 の の 、 、 の の 、 、 の の 、 、 の の 、 つ 、 の 、 の	esign Sources (1) esign Sources (1) onst Source No co Co Open File imul: Create HE sit View Insta Generate Reset Out	2. bdv.(0) ode Properties DL Wrapper DL Wrapper intiation Template Output Products iput Products	Ctrl+E Alt+O		-0		
ファイルを作成 、 の 、 。 の 、 。 。 、 。	esign Sources (1) Source No onst co co Source No Open File imult Sir View Insta Generate Reset Out Replace F	2 bdV (0) ode Properties DL Wrapper antiation Template Output Products iput Products	Ctrl+E Alt+O		-0		
ファイルを作成 、、・・・ 、、・・・ 、、・・・・ 、、・・・・・・・・・・・・・・・・	esign Sources (1) sources (1) source Na source Na S	DL Wrapper DL Wrapper antiation Template Output Products tput Products file Into Project	Ctrl+E Alt+O		-9		
ファイルを作成 、 の 、 。 の 、 。 。 、 。	esign Sources (1) Source No onsti co co sir Source No Open File imult Create HE Sir View Insta Generate Reset Out Replace F Copy File Copy All F	1 bdV (0) ode Properties DL Wrapper antiation Template Output Products tput Products 'ile Into Project iles Into Project	Ctrl+E Alt+O		-0		



2) bit ファイルを生成する。そして、ハードウェアインフォメーションをエクス ポートする。SDK を起動する。

#### 10.2 SDK プログラム

10.2.1 LWIP テンプレートに基づく APP を作成



10.3 ダウンロードとデバッグ

テスト中環境は dhcp をサポートするルーターが必要である。開発ボードはルーターと接続す ると自動にIPアドレスを獲得できる。実験中メインフレームと開発ボードは同じネットワークに いて、お互いに通信できる。

#### 10.3.1 イーサネットテスト

1) シリアルポールと接続して、シリアルでバッグターミナルを開く。PS 側のイーサネットケーブ ルをルーターに接続して、SDK を起動する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

Run or Debug a program using	g System Debugger.							
Ĩ 🗎 🗙   🖻 ‡≻ 🔹 🗍	Name: System Debugg	er using Debug_lwip.elf on Local						
type filter text	🧿 Target Setup 📘	Application 🗱 Arguments 🖾 Er	nvironment 👼 Symbo	ol Files 💦				
formance Analysis				*				
nx C/C++ application (GDB)	Hardware Platform:	design_1_wrapper_hw_platform_0						
nx C/C++ application (System	Bitstream File:	design_1_wrapper.bit	Search	Browse				
System Debugger using Deb	Initialization File:	ps7_init.tcl	Search	Browse				
	FPGA Device:	Auto Detect	Select					
	PS Device:	Auto Detect	Select					
	-	Summary of operations to	be performed	=				
	Reset entire syste	m Following operations will b	be performed before	launching the de				
	V Program FPGA	2. Program FPGA fabric (P	ears the FPGA fabric ( L).	Ρι.).				
	Run ps7_init	3. Runs ps7_init to initialize	e PS.					
	Run ps7_post_cor	fig 4. Runs ps/_post_config. E	nables level shifters fi	rom PL to PS. (Re				
	*			•				
4 III	N							
ilter matched 6 of 10 items			Revert	Apply				

2) シリアルはインフォメーションをプリントアウトして、自動的に "192. 168. 1. 68"というアドレスを獲得した。接続速度は 1000Mbps で、tcp ポー トは 7 である。

B COM3 - PuTTY	
lwIP_TCP_echo_server	
TCP packets sent to port 6001 will be echoed back	
WARNING: Not a Marvell or TI Ethernet PHY. Please verify the initialization sequ	
ence	
Phy 1 13 KS29031	
Start PHY autonegotiation	
Waiting for PHY to complete autonegotiation.	
autonegotiation complete	
auto-negotiated link speed: 1000	
Board 1P: 192.168.1.68	
Netmask : 255.255.255.0	
Gateway : 192.168.1.1	
TCP echo server started @ port 7	
	۳.



3) telnet を使って接続する。



4) 文字一つを書き込む時、開発ボードは同じ文字を表示する。





#### 10.4 実験のまとめ

実験をつうじて、SD プログラムの開発を深く理解できた。本実験はただ簡単にLWIP APP の 作成を説明しただけである。LWIP は UDP、TCP 等のプロトコルを完成できる。続きの教程にこち らはイーサねに基づく具体的な APP を提供する。例えば、イーサネットを通じて、ADC が集めた データを伝送することとか、カメラデータを上位機械に伝送することとか。

ホームページ:<u>https://www.csun.co.jp</u>



#### 第十一章 ユーザー定義 IP テスト

実験用 Vivado プロジェクトは "custom\_pwm\_ip"。

オフィシャル Xilinx はたくさんの IP コアを提供した。Vivado の IP Catalog にこれら の IP コアを見ることができる。ユーザーは自分のシステムを構築する時、オフィシャル Xilinx のフリーIP コアだけを使うのができない。ユーザー自身の IP コアを作成するときこ とが必要で、いい点もある。例えば、システムデザインをカスタマイズする;デザインを重 複に使用できる。IP コアに license を追加して、有料で提供する;システムデザインをシン プルにして、デザイン時間も縮める。ZYNQ システムで IP コアをデザインするには、よく使 われているのが AXI バスで PL と PL 部分の IP コアを接続することである。本実験は Vivado に AXI バスタイプの IP コアを構築する方法を紹介する。この IP コアは PWM を生成して、開 発ボード上の LED をコントロールし、呼吸ライトの効果をつくる。

#### 11.1 PWM 紹介

PWM はよく LED やブザーなどのコントロールに使われている。パルスのデューティ比を調節して LED の明るさを調節する。開発ボードに使われた pwm モジュールは下のように:







```
timescale <u>lns</u> / <u>lps</u>
module ax_pwm
 #(
    parameter N = 32 //pwm bit width
 )
 (
 input
                      clk.
 input
                     rst.
 input[N - 1:0]period,
 input[N - 1:0] duty, output
                     pwm out
 );
 reg[N - 1:0] period_r;
 reg[N - 1:0] duty_r;
 reg[N - 1:0] period_cnt;
 reg pwm_r;
 assign pwm_out = pwm_r;
 always@(posedge clk or posedge rst)
begin
 if(rst==1)
       begin
           period_r <= { N {1'b0} };
duty_r <= { N {1'b0} };</pre>
 end
       else begin
           period_r <= period;</pre>
          duty r <= duty;</pre>
        end
     end
     always@(posedge clk or posedge
    rst) begin
if(rst=1)
        period_cnt <= { N { [ ] b0 } };</pre>
       else
           period_cnt <= period_cnt + period_r;</pre>
     end
 always@(posedge clk or posedge rst)
     begin
        if (rst=1)
       begin
           pwm_r <= 1' b0;
  end
        else
       begin
       if(period_cnt >= duty_r)
       pwm_r <= 1'b1;
else
               pwm_r <= 1'b0;
        end
  end
```

```
endmodu l e
```



この PWM モジュールは"period"、"duty"という二つのパラメータを利用して、頻度と デューティ比をコントロールする。レジーをデザインしてこれらのパラメータをコント ロールする。ここは AXI バスを使用して、レジスターのリードライトをする。

- 11.2 Vivado プロジェクトの作成
  - 11.2.1 一つの vivado プロジェクトを作成

"custom\_pwm\_ip" という名のプロジェクトを作成する。zynqPS システムを追加して、パラメータを配置する。具体な方法は前の章に参考できる。



2) "Next を"を選択する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



4) ネームは "ax\_pwm"にして、デスクリプションは "alinx pwm"にする。そして、適切な位置を 選んで IP を入れる。



# ▼■▲へ●▲■ 株式会社日昇テクノロジー

低価格、高品質が不可能? 日昇テクノロジーなら可能にする

🔥 Create and Packag	ge New IP				10	
Peripheral Deta Specify name, versio	ils on and description for the new peripheral					
Name: Version:	ax_pwm 1.0					8
Display name:	ax_pwm_v1.0					$\otimes$
Description:	alinx pwm					$\otimes$
IP location:	F:/ax7350/demo/custom_pwm_ip/repo				8	
Overwrite e	ásting					
?		< <u>B</u> ack	Next :	Eirlis	h	Cancel

5) 下のパラメータはインタフェースタイプやレジスターの数とかを指定できる。ここは変更 せずに、 AXI Lite Slave インタフェースとレジスター4つを使う。

Enable Interrupt Support	+ -		Name	S00_AXI	
	Interfaces		Interface Type	Lite	~
	I S00_AXI		Interface Mode	Slave	~
			Data Width (Bits)	32	~
		<	Memory Size (Bytes)	64	~
ax_pwm_v1.0		>	Number of Registers	4	[4512]

6) "Finish" をクリックして、IP の作成を完成する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



8) この時の IP は簡単なレジスターをリードライトする機能しかないので、IP を変更する必要ある。IP を選択して、右ボタンで"Edit in IP Packager"を探し出す。





9) ダイアログがポップアップされて、プロジェクトネームとパスを書き込める。ここはど こデフォルト値にして、 "OK"をクリックする。

A Edit in IP Packager	×
Choose a project name and location for editing.	4
Project name: ax_pwm_v1_0_project Project_location: F:/ax7350/demo/custom_pwm_ip/custom_pwm_ip.tmp	8
Edit/IP project will be created at: F://custom_pwm_ip.tmp/ax_pwm_v1_0_project	Cancel
10) Vivado は新しいプロジェクトを開いた。	



低価格、高品質が不可能? 日昇テクノロジーなら可能にする




12) コード追加する時、コードを IP 目次ぐにコピーする。

Add or	r Create Design Sources	
Specify H	HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a n	ew source
file on dis	lisk and add it to your project.	
+.		
	Index Name Library Location	
ve	1 ax_pwm.v xil_defaultiib F:/ax7350/demo/custom_pwm_ip	
	Add Files Create File	
Sca	can and add RTL include files into project	
Co	copy sources into IP Directory	
🖌 Ado	dd so <u>u</u> rces from subdirectories	
	- Desta	iniah Canaal
?	< <u>B</u> ack <u>N</u> ext > E	inish Cancel
?	< <u>B</u> ack <u>N</u> ext > E	inish Cancel
?	< Back Next > E	inish Cancel
(?) "ax_p	< <u>₽ack</u> №ext> E _pwm_v1_0.v"を変更し、 pwm アウトプットポートを追加する。	inish Cancel
(?) "ax_p	<back next=""> E .pwm_v1_0. v"を変更し、 pwm アウトプットポートを追加する。</back>	inish Cancel
(? "ax_p	< Back Next > E pwm_v1_0. v"を変更し、pwm アウトプットポートを追加する。	jinish Cancel
(?) "ax_p	<pre></pre>	inish Cancel
(?) "ax_p Sources Q ≚ ≑	Eack Next> E     pwm_v1_0. v"を変更し、 pwm アウトプットポートを追加する。     ? _ D び ×     f/ax7350/demo/custom_pwm_ip/repo/ax_pwm_v1_0/v*     f/ax7350/demo/custom_pwm_ip/repo/ax_pwm_v1_0/v	inish Cancel
(?) "ax_p Sources Q ₹ ≑ > Design Sc > @4 ax (	<pre></pre>	inish Cancel
(?) "ax_p Sources Q ₹ ≑ ∨ ⊡ Design Sc > @A ax_p ⊛ ax_pw	<pre></pre>	inish Cancel
(?) "ax_p Sources Q ≚ ≑ > ⊕ A ax_1 ⊛ ax_pw > □ IP-XAC	<pre></pre>	inish Cancel
?       "ax_p       Sources       Q     ₹       > Design Sc       > @ ax_pw       > □ IP-XAC       > □ Constraint	<pre></pre>	inish Cancel
?         "ax_p         Sources         Q       ₹         > Design Sc         > @A       ax_p         > Destraint         > Constraint         > Simulator	(型ack Next> E     ()	inish Cancel
?         "ax_p         Sources         Q       ₹         > Design Sc         > @A ax_p         > @ P-XAC         > @ Constraint         > @ Simulator	(型ack Next> E     (	inish Cancel
?         "ax_p         Sources         Q       ₹         > □ Design Sc         > □ A ax_p         > □ IP-XAC         > □ Constraint         > □ Simulator	(型ack Next> E     (Next> E     (Next> )     (Next) )     (Ne	inish Cancel
?         Sources         Q       ₹         >       Design Sc         >       ax_pw         >       IP-XAC         >       Constraint         >       Simulator	Librariesa Compile Order     Librar	inish Cancel
?       Sources       Q     ₹       >     Design Sc       >     ax_pw       >     IP-XAC       >     Constraint       >     Simulation	▲ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	inish Cancel
?         Sources         Q       ₹         Q       ₹         > □ Design Sc         > □ P-XAC         > □ P-XAC         > □ Constraint         > □ Simulator         Hierarchy       Li         Source File Prop	(型ack Next> E     (	inish Cancel
?         Sources         Q       X         Q       X         >       Design Sc         >       @ ax_pw         >       IP-XAC         >       Constraint         >       Simulator         Hierarchy       Li         Source File Prop       @ ax_pwm_v1_c	() 「「「「「」」」」」」     () 「「」」」     () 「「」」」     () 「     () 「」     () 「     () 「」     () 「	inish Cancel
?         Sources         Q       ₹         Q       ₹         >       Design Sc         >       @ ax_pw         >       IP-XAC         >       Constraint         >       Simulation         Hierarchy       Li         Source File Prop       @ ax_pwm_v1_c         ?       Enabled	Sources (3)     (2)	inish Cancel
?         Sources         Q       ₹         Q       ₹         >       Design Sc         >       ax_pw         >       IP-XAC         >       Constraint         >       Simulator         Hierarchy       L1         Source File Prop         @ ax_pwm_v1_c         Y Enabled         Location:	■ Back Next> E pwm_v1_0.v" を変更し、pwm アウトプットポートを追加する。 Project Summary × Package IP - ax_pwm_v1_0.v* * * * * * * * * * * * * * * * * * * *	inish Cancel
?         Sources         Q       ₹         Q       ₹         >       Design Sc         >       ax_pw         >       First ax_pw         >       Constraint         >       Simulation         Hierarchy       Li         Source File Prop       ax_pwm_v1_c         Penabled       Location:         Type:       Type:	■ Back Next> E pwm_v1_0. v" を変更し、pwm アウトプットポートを追加する。 Project Summary × Package IP - ax_pwm × ax_pwm_v1_0.v* Fix7350/demo/custom_pwm_ip/repo/ax_pwm_v1_0.v* Fix7350/demo/custom_pwm_ip/repo/ax_pwm_v1_0.v* I i i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps I i e screl i ns / 1 ps <td>x</td>	x
?         Sources         Q       X         Q       X         >       Design Sc         >       Design Sc         >       IP-XAC         >       IP-XAC         >       Simulator         Herarchy       Li         Source File Prop       @ ax_pum_v1_c         V       Enabled         Location:       Type:         Librarr       Librarr	Sources (3)     Compile Order     Compi	inish Cancel

14) "ax\_pwm\_v1\_0.v"を変更し、インスタンス化 "ax\_pwm\_V1\_0\_S00\_AXI" に pwm ポート のインスタンス化を追加する。





16) 修改 "ax\_pwm\_v1\_0\_s00\_AXI.v" ファイルを変更して, pwm 機能コアコードをインス タンス化にする。レジスター slv\_reg0 と slv\_reg1 を pwm モジュールパラメータのコント ロールに用いる。





#### 17) "component. xml"ファイルをダブルクリックする



18) "File Groups"オプションに"Merge changers from File Groups Wizard" をクリックする。



Impatibility       Impatibility	ging Steps	File Groups		
mpatibility   e Groups   stomization Parameters   rts and Interfaces   dressing and Memory   stomization GUI     Q $\bigstar$ Name   Name   Library   Name   Standard $\checkmark$ Advanced   >   Verilog Synthesis (2)   >   >   Verilog Simulation (2)   >   >   Software Driver (6)   >   Block Diagram (1)	entification	Merge changes from File Groups Wizard		
Groups     Name     Library Name     T       Image: Standard	npatibility	Q   ±   €   11 +   C		
Image: Standard       Image: Standard	Groups	Name	Library Name	Туре
omization Parameters <ul> <li>Advanced</li> <li>Verilog Synthesis (2)</li> <li>Verilog Simulation (2)</li> <li>Software Driver (6)</li> <li>Ul Layout (1)</li> <li>Block Diagram (1)</li> <li>Software (1)</li></ul>		Standard		
Ind Interfaces     >     >      >      >      >	stomization Parameters rts and Interfaces dressing and Memory	✓		
Image: Series and Memory     Image: Series and Memory       Image: Series and Memory     Image: Series and S	and Interfaces	> 🔄 Verilog Synthesis (2)		
essing and Memory > Software Driver (6) pmization GUI > UI Layout (1) > Block Diagram (1)		Verilog Simulation (2)		
> Image: UI Layout (1)       > Image: Block Diagram (1)	essing and Memory	> 🕞 Software Driver (6)		
> Block Diagram (1)	mization GUI	> 🔁 UI Layout (1)		
		> 📄 Block Diagram (1)		
w and Package	w and Package			

19) "Customization Parameters"オプションに"Merge changes form Customization Parameters Wizard"をクリックする。

Packaging Steps	Customization Parameters		
✓ Identification	Merge changes from Customization	on Parameters Wizard	
Compatibility			
✓ File Groups	Name	Description	Display Name
	<ul> <li>Customization Parameters</li> </ul>		
Customization Paraméters	C_S00_AXI_DATA_WIDTH	Width of S_AXI data bus	C S00 AXI DATA WID
Ports and Interfaces	C_S00_AXI_ADDR_WIDTH	Width of S_AXI address bus	C S00 AXI ADDR WID
	C_S00_AXI_BASEADDR		C S00 AXI BASEADD
<ul> <li>Addressing and Memory</li> </ul>	C_S00_AXI_HIGHADDR		C S00 AXI HIGHADDI
Customization GUI			
Review and Package			

20) "Re-Package IP"をクリックして、IPの変更を完成する。



Packaging Steps	Review and Package
<ul> <li>Identification</li> </ul>	IP has been modified.
<ul> <li>Compatibility</li> </ul>	Summary
✓ File Groups	Display name: ax_pwm_v1.0
<ul> <li>Customization Parameters</li> </ul>	Root directory: f/ax7350/demo/custom_pwm_ip/repo/ax_pwm_1.0
<ul> <li>Ports and Interfaces</li> </ul>	
<ul> <li>Addressing and Memory</li> </ul>	
<ul> <li>Customization GUI</li> </ul>	
Review and Package	After Packaging
	An archive will not be generated. Use the settings link below to change your preference Project will be removed after completion Edit packaging settings
	Re-Package IP
1) "pwm"をサーチし、"ax_pw Diagram のしの チ Designer	m_v1. 0"を追加する。 × Address Editor × IP Catalog × ∷   №   ♀   ♀   辛   +   癶   〆   ♡   ℃   ♡   ミ Assistance available. Run Block Automation
	wm 💿 (1 match)
₽ ax_pwm_v1.	0 processing_system
3	- M_AXI_GP0_ACLK ZYNQ
×	
ENTER to select,	ESC to cancel, Ctrl+Q for IP details

"Run Block Automation" Wクリックする。 2)









6) xdc ファイルを追加して、P ピンを分配する。 pwm\_0 アウトプットポートを PLLED1 に 配って、呼吸ライトを作る。

set\_property IOSTANDARD LVCMOS33 [get\_ports
pwm\_0] set\_property PACKAGE\_PIN M14 [get\_ports

ホームページ:<u>https://www.csun.co.jp</u>







低価格、高品質が不可能? 日昇テクノロジーなら可能にする



3) 新規 APP を作る。テンプレートは "Hello World" を選ぶ。





pplication Project Create a managed m	ake application project.
Project name: pwm_	test
Use default location	on
Location: F:\ax7015\	demo\custom_pwm_ip\custom_pwm_ip.sdk\r
Choose file	system: default 💌
OS Platform: stand	lalone 👻
Target Hardware	
Hardware Platform:	design_1_wrapper_hw_platform_0
Processor:	ps7_cortexa9_0
Target Software	
Language:	
Compiler:	32-bit
Hypervisor Guest:	N/A 🔹
Board Support Pac	kage:  Create New pwm_test_bsp
	Use existing
	Back Next > Finish Cancel

前の例はすべて xilinx の IP を使っている。xilinx だいたい API をーセットで提供している、ユーザー定義 IP の場合、自分で開発するには必要がある。APP の目次ぐにあるいは資源を見て、ax\_pwm.h というファイルが見つかる。このファイルにユーザー定義UO レジスターのリードライトマクロ定義を含んている。



低価格、高品質が不可能?

日昇テクノロジーなら可能にする



6) レジスターリードライトマクロとユーザー定義 IP のアドレスがあって、コーディングをはじめる。ユーザー定義 IP をテストにするには、レジスター
 AX\_PWM\_SOO\_AXI\_SLV\_REGO\_OFFSET をライトすろことで、, PWM のアウトプット頻度をコントロールする。そして、レジスターAX\_PWM\_SOO\_AXI\_SLV\_REG1\_OFFSET をライトすることで、 PWM アウトプットのデューディリ比をコントロールする。



unsigned int duty; int main() init\_platform(); print("Hello World¥n¥r"); //pwm out period = frequency(pwm\_out) \* (2 \*\* N) / frequency(clk); AX\_PWM\_mWriteReg(XPAR\_AX\_PWM\_0\_S00\_AXI\_BASEADDR, AX\_PWM\_SOO\_AXI\_SLV\_REGO\_OFFSET, 17179) ://200hz while (1) { for (duty = 0x8fffffff; duty < 0xffffffff; duty = duty + 100000)</pre> { AX\_PWM\_mWriteReg(XPAR\_AX\_PWM\_0\_S00\_AXI\_BASEADDR, AX\_PWM\_SOO\_AXI\_SLV\_REG1\_OFFSET, duty); usleep(100); } } cleanup\_platform();
return 0; } コードを作動することで、 PLLED1 が 呼吸ライトの効果を表現している。 7)



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

un or Debug a program using	g System Debugger.										
* ∰ ≍   ■ *	Name: System Debugg	er using Debug_pwm_test.el	f on Local	~							
Performance Analysis	📀 Target Setup 📃	Application 🗱 Arguments	📧 Environment 🚋 Symbo	ol Files »3							
Target Communicatio	Hardware Platform:	design_1_wrapper_hw_platf	orm_0 🔻								
Image: System Debugger	Bitstream File:	design_1_wrapper.bit	Search	Browse							
	Initialization File:	ps7_init.tcl	Search	Browse							
	FPGA Device:	Auto Detect	Select								
	PS Device:	Auto Detect	Select								
		Summary of operations to be performed									
	<ul> <li>Reset entire syste</li> <li>Program FPGA</li> <li>Run ps7_init</li> <li>Run ps7_post_cor</li> </ul>	m Following operations 1. Resets entire syste 2. Program FPGA fab 3. Runs ps7_init to ini 4. Runs ps7_post_cor	will be performed before I m. Clears the FPGA fabric (I rric (PL), itialize PS, nfig. Enables level shifters fr	launching the de PL). rom PL to PS. (Re							
	•			۲							
ter matched 6 of 10 items			Revert	Apply							

8) debugを通じて、レジスターの状況を見てみよう。



低価格、高品質が不可能?

日昇テクノロジーなら可能にする



- = 🗆 🔡 Outline 🖾 - h x ld he 1 s #include "sleep.h" FR N Nº O W stdio.h unsigned int duty; stdio.h
   platform.h
   xil.pintf.h
   ax\_pwm.h
   xil.jo.h
   xparameters.h
   sleep.h
   duty: unsigned int
   main(): int int main() i
  init\_platform(); print("Hello World ); //pmm out peciad - frequency(port ort) \* (2 \*\* N) / frequency(slk); AX\_PWM\_mWriteReg(XPAR\_AX\_PWM\_0\_500\_AXI\_BASEADDR, AX\_PWM\_S00\_AXI\_SLV\_REG0\_OFFSET, 17179);//200hz 1) {
   (duty = 0; duty = 0.5fffffff; duty = duty + 500) {
   AX\_NAM\_MIPTERS(NAM\_AX\_PAN\_0\_500\_AXI\_SACEADOR, AX\_PAN\_500\_AXI\_SLV\_REGI\_OFFSET, duty);
   usilept[1000); while (1) { } } cleanup\_platform();
  return 0; © Console ☆ ⊘ Tasks © SDK Terminal ☆ Problems ② Executables 📑 🔍 \* 🖱 ♥ 🗋 SDK Log ③ Memory ☆ 🔌 👔 🐨 🖏 🖗 🖏 ♥ ♥ ♥ 🗆 TCF Debug Virtual Terminal - ARM Cortex-A9 MPCore #1
  - 10) メニューから Window→Show View→Memoryをクリックする。



低価格、高品質が不可能?

日昇テクノロジーなら可能にする

	New Window	A - 1 1 10 1	> - ⇔ -		Quick Access
S Debug S3	Editor	(x)= Variables S?	Breakno IIII Poo	isters 📰 V	SCT Co 🕅 Emulatio 🛸 Modulos
E System Debugger using Debug own test.elf on Local (Local)	Appearance	· · · · · ·	e breakpoint mit neg		
▲ ③ APU	Show View	Breakpoints	Alt+Shift+Q, B	-	
▲ ARM Cortex-A9 MPCore #0 (Breakpoint: main)	Perspective	Console	Alt+Shift+Q, C		
0x00100624 main():/src/helloworld.c, line 60	Navigation	Debug     Disassembly			
<pre>0x00100874_start()</pre>	Preferences	Executables		-	
<b>=</b>		6 Expressions		-	
ARM Cortex-A9 MPCore #1 (Suspended)		Memory			
₩ xc7z015		Modules			
		E Outline	Alt+Shift+Q, O		
		Problems	Alt+Shift+Q, X		
		Signals			
		Tasks			
i system.hdf i in xparameters.h i helloworld.c 🔀		Son Trace Control			E Outline ⊠
#include sleep.n		(x)= Variables	Alt+Shift+Q, V		
unsigned int duty;		Other	Alt+Shift+Q, Q		I platform.h
eint main()			Quick Access   Sereskpo   Itt Registers   X SCT Co   Envolutio   Alt-Shift+Q, B   Alt-Shift+Q, C   Jype   Alt-Shift+Q, V   Alt-Shift+Q, V		
<pre>init_platform();</pre>					ax_pwm.h
<pre>print("Hello World\n\r");</pre>					<ul> <li>xil_io.h</li> <li>xnarameters h</li> </ul>
//mm out popied - frequency/num out) * (2 ** N) / fr	anuoneu (alk)				u sleep.h
AX_PWM_mWriteReg(XPAR_AX_PWM_0_S00_AXI_BASEADDR, AX_PV	WM_S00_AXI_SLV_REG0_	OFFSET, 17179);//200h	z	100	duty : unsigned int
while (1) {				=	main() : int
<pre>AX_PWM_mWriteReg(XPAR_AX_PWM_0_S00_AXI_BASEAD usleep(10000);</pre>	DR, AX_PWM_S00_AXI_S	LV_REG1_OFFSET, duty)	:	E.	
}					
return 0;				-	
🕽 Console 🛱 🧏 Tasks 📮 SDK Terminal 🦹 Problems 🔘 Executable	25		SDK Log 🚺 Me	mory 🛛	2 🖸 🛃 🔠 🖼 👘 🔹 💙 '
F Debug Virtual Terminal - ARM Cortex-A9 MPCore #1			Monitors 🖶 🗶 🧃	<u> </u>	<u> </u>
	$\land$				
	$\mathbf{}$				
	1				
	()				
	()				
	<pre>3inx Window Help  3inx Window Help  3inx Window Editor Appearance Appear</pre>				
	()				
	()				







#### 11.4 実験結果

本実験より SDK のデバッグ方法を勉強して、ARM と FPGA のデータ交換方法を把握する。

#### 11.5 よくある問題

11.5.1 AXI IP のベースアドレスを調べる





第十二章 VDMA を使用して HDMI ディスプレイを駆動する

実験Vivadoプロジェクトはvdma\_hdmi\_outである。

PSには統合されたディスプレイ制御システムがなく、PLを使用して実現する必要がある。実現の方法はい くつかあるが、どれでもDMAシステムが必要である。DMAシステムは、ddr3からディスプレイへの表示を完了 し、CPUオーバーヘッドを削減できる。VDMAは、ビデオ出力専用のxilinxが開発した特別なDMAであり、 xilinxFPGAビデオ処理を学習する重要な部分である。

前のHDMI表示データはPLによって内部に生成されているが、この実験では、PSによって表示データが生成 され、VDMAを介してPLがHDMIインターフェイスに送信される。

12.1 Vivado プロジェクトの設立

VDMAディスプレイは重要な部分であるため、この実験ではVivadoビルドプロセスの詳細を説明する。

1) vdma\_hdmi\_outというプロジェクトを作成する。

New Project Project Name	tat Nas	interface time-times (Ma	$\wedge$	
inter a name for yo	our project and specify a directory	where the project data files will b	e stored.	
<u>P</u> roject name:	vdma_hdmi_out			0
Project location:	F:/a 5/demo			Ø
Create project	ct subdirectory			
?)	•	< <u>B</u> ack	Next > Einish	Cancel

#### 2) 新しい Block デザインを作成する



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

Q- Quick Access Eile Edit Flow Tools Window Layout View Help 局 18 ÷ Σ % Ø X → □ × ► Flow Navigator ₹ <del>0</del> ? \_ PROJECT MANAGER - vdma\_hdmi\_out PROJECT MANAGER Sources ? \_ 🗆 🖒 X Settings **≑** + ÷ 0 Q ? ø Add Sources Design Sources Language Templates > Constraints P IP Catalog Simulation Sources sim\_1 **IP INTEGRATOR** Create Block Design Open Block Design Generate Block Design ompile Order Libraries Hierarchy ✓ SIMULATION Run Simulation 3) デザイン名はそのまま Settings Edit Add Sources Design Sources Language Templates Project name: Constraints Project location: IP Catalog Simulation Sources Product family 📄 sim\_1 - X Create Block Design IP INTEGRATOR Please specify name of block design. Create Block Des Open Block Desig Generate B Desian Design name: design\_1  $\otimes$ <Local to Project> Directory: Hierarchy Libraries SIMULATION Run Simulation Specify source set Design Sources ~ Properties RTL ANALYSIS ? Cancel > Open Elaborated Design Report Strategy SYNTHESIS 4) ZYNQ プロセッサを追加する



( Income the second				
Search:	Q zynq	(1 match)		
🕈 ZYN	Q7 Processing System	m		
			$\searrow$	
			$\checkmark$	
			This des	ing is empty Press the 🔸 h

5) ZYNQ パラメーターをコンフィグし、VDMA 高速読み取り DDR3 の HPO インターフェイスを有効にする。

YNQ7 Processing Syst	em (5.5)	ΥN		1
Documentation 🌣 Preset	s 🔚 IP Location 🚳 Import XPS Setting:			
Page Navigator –	PS-PL Configuration			Summary Report
Zyng Block Design	• Q ≠ €			
PS-PL Configuration	Search: Q-			
Peripheral I/O Pins	sing System (5.5)  Presets IP Location in import XPP Settings  PS-PL Configuration  Summary Report  PS-PL Configuration  Summary Report  Setted  PS-PL Configuration  Setted  Setted			
IIO Configuration				
into conliguration	Thread ID Width	12	Thread ID Width for GP1 interface	
Clock Configuration	<ul> <li>GP Stave AXI Interface</li> </ul>			
ODR Configuration	8 AXI GP0 Interface		Enables General purpose 32-bit AXI Slave interface 0	
	S AXI GP1 interface		Enables General purpose 32-bit AXI Slave interface 1	
MC Timing Calculation	<ul> <li>HP Slave AXI Interface</li> </ul>		-	
terrupts	<ul> <li>S AXI HP0 interface</li> </ul>		Enables AXI high performance slave interface 0	
	S AXI HP0 DATA WIDTH	64 🗸	Allows HP0 to be used in 32/64 bit data width mode	
	<ul> <li>S AXI HP1 interface</li> </ul>		Enables AXI high performance slave interface 1	
	S AXI HP1 DATA WIDTH	64 ~	Allows HP1 to be used in 32/64 bit data width mode	
	<ul> <li>S AXI HP2 interface</li> </ul>		Enables AXI high performance slave interface 2	
	S AXI HP2 DATA WIDTH	64 🗸	Allows HP2 to be used in 32/64 bit data width mode	
	<ul> <li>S AXI HP3 interface</li> </ul>		Enables AXI high performance slave interface 3	
	S AXI HP3 DATA WIDTH	64 ~	Allows HP3 to be used in 32/64 bit data width mode	



#### 12.1.1 UART のコンフィグ

Bnakレベル標準をコンフィグし、BankOはLVCMOS 3.3V、Bank1はLVCMOS 1.8V、シリアルポートを有効にする

ZYNQ7 Processing System	1 (5.5)																		
Documentation 🔅 Presets	😑 IP Location 🛛 🚳 Import XPS	Settings																	
Page Navigator	Peripheral I/O Pins																	Summ	nary Repr
Zyng Block Design	<b>+</b> Q ₹ \$	0																	
PS-PL Configuration	Search: Q,-	_																	
Peripheral VO Pins			Bank 1	LVCMC	IS 1.8V	v													
	Peripherals	15	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31	32 33	34 35	38 3	7 38 35	9 40 41	42 43	44 45	46 47	40 40
MIO Configuration	USB 1		_			_										_	U	501	
Clock Configuration	> _ SD 0			SDO						800						500			
	> _ SD 1		_			-	501					1	501					1	501
DDR Configuration	> _ SPI 0		SPID		-	_			SPID						500		-	_	
SMC Timing Calculation	> _ SPI 1				110					L	_							-	
Interrupts	> UART 0	ATO		UARTO		UARTO		UARTO		UARTO		UARTO		LIARTS		LIARTO		LIARTO	
	> 🗹 UART 1		UARTI		UARTI		UARTI		LIART1		UARTI		UART		UARTI		UARTI		UARTI
	> 12C 0	200		1200	2.5471	1200		1200		1200		1000		1200	2.4111	1200	-	1200	
	> [] I2C 1		1201		1201		1263		1201		1201	-	1201		1201	200	1203		1201
	> CAN 0								201						Carlo I		13101		and the f

#### 12.1.2 12C EMIO のコンフィグ

### 1) 12COを有効にし、EMIOを選択して、12CをPL側に接続できるようにする。

Re-customize IP	_																								_				
ZYNQ7 Processing Sys	stem (5.5)																								2				
Documentation 🔅 Pres	ets 📄 IP Location	6 Import	KPS S	Settin	igs																								
Page Navigator —	Peripheral I/O	Pins																				Su	umma	ary Repo	ort				
Zynq Block Design	+ 9 =	<b>\$</b> 0																											
PS-PL Configuration	Search: 0-																												
Peripheral I/O Pins																													
MIO Configuration	Peripherals	10 31	32	33	34 3	5 3	30 37	38	39	40	41	42	43	44	45	48	47	48	49	50	51	52	53	EMIO	I,				
X																			US	B1		-	-	-	-				
Clock Configuration	2 SD 1	SD0		-						-	-	SC	0	-	-									EMIO	ų				
DDR Configuration	tion > SPI 0					-	SD1	-	_						_			S	D1	-				EMIO	ų				
SMC Timing Calculation	> SPI0			mos						SP	10			_	mos									EMIO	ų				
				_	SF	PI1											SPI1							EMIO	ų				
Interrupts	> UARTO	JARTO			UART	0		UAR	то			UAR	то			UAP	RTO	L.,		UA	RTO			EMIO	ų				
	> 🔽 UART 1		UAR	RT1		-	UART1			UAR	RT 1		-	UAR	T1			UA	RT1	_	_	UA	RT1	EMIO					
	> 🗹 I2C 0	12C0			1200			120	æ			120	0			12	00			12	co			EMIO					
	7 - 12C I		120	21			1201			120	21			120	:1			12	C1			12	C1	EMIO	Ī				
	> CAN 0	CANO			CAN	0		CA	10			CA	NO			CA	ND			CA	NO			EMIO	1				
	> _ CAN 1		CAL	N1		٦	CAN1			CA	N1			CA	N1			C	AN1			CA	N1	EMIO	1				
	TTC0	TTCO				T						TT	00					1						EMIO	1				
	TTC1	<												-								-		_	>				
																			Г	-	K			Cancel	_				
																				0	ars.			Cancel					



#### 2) クロックを設定し、FCLK\_CLKOを100Mhzに、FCLK\_CLK1を142MHzに設定し、このクロックの使用はVDMAデ ータを読み取るためである。

Documentation 🎄 Pres	ets 📄 IP Location 🚳 Impor	t XPS Settings				
Page Navigator —	Clock Configuration					Summary Repor
lyng Block Design	Basic Clocking Advance	ed Clocking				
S-PL Configuration	Input Frequency (MHz) 33.33	33333 🚫 CP	U Clock Ratio 6:2:1	~		
Peripheral I/O Pins	← Q 素 ≑ M	:				
MIO Configuration	Search: Q-					
Clock Configuration	Component	Clock Source	Requested Frequ	Actual Frequency(	Range(MHz)	Class
_	CAN1 MIOCLK	External	-1	23.8095	-2:-1	CAN
DDR Configuration	FCLK_CLK0	IO PLL 🗸	100	100.000000	0.100000 : 250.000000	PL Fabric Cloc
SMC Timing Calculation	FCLK_CLK1	IO PLL 🗸	142	142,857132	0.100000 : 250.000000	PL Fabric Cloc
interrupts	FCLK_CLK2	IO PLL	50	10.000000	0.100000 : 250.000000	PL Fabric Cloc
	FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.000000	PL Fabric Cloc
	TPIU	External	200	200.000000	10.000000 : 300.000000	System Debug
	WDT	CPU_1X	133.333333	111 111115	0.100000 : 200.000000	Timers
	TTC0 CLKIN0	CPU_1X	133.333333	111.111115	0.100000 : 200.000000	TTC0
	TTC0 CLKIN1	CPU_1X	133.333333	111.111115	0.100000 : 200.000000	TTC0
	TTC0 CLKIN2	CPU_1X	133.333333	111.111115	0.100000 : 200.000000	TTC0
	TTC1 CLKIN0	CPLL 1X	133 333333	111 111115	0 100000 - 200 000000	TTC1

12.1.3 DDR3 のコンフィグ

DDR Configurationタブで、PS端子ddrのパラメーターをコンフィグできる。AX7010コンフィグDDR3モデルは MT41J128M16 HA-125、AX7020コンフィグDDR3モデルはMT41J256M16 RE-125である。ここでは、ddr3モデルは ボード上のddr3モデルではない、パラメーターに最も近いモデルである。Effective DRAM Bus Width、32 Bit を選択する。



YNQ7 Processing Sy	stem (5.5)		4
Documentation 💠 Pres	ets 🛯 IP Location 🔞 Import XPS Setti	ngs	
Page Navigator —	DDR Configuration		Summary Report
Zyng Block Design	✓ Enable DDR		
PS-PL Configuration	<ul> <li>← Q 王 ≑</li> </ul>		
Peripheral VO Pins	Search: Q-		
MIO Configuration	Name Select  V DDR Controller Configuration		Description
Clock Configuration	Memory Type	DDR 3 v	Type of memory interface. Refer to UG585 Zyng Technical
DDR Configuration	Memory Part	MT41J256M16 RE-125 🗸	Memory component part number. For unlisted parts choose
SMC Timing Calculation	Effective DRAM Bus Width	32 Bit 🗸	Data width of DDR interface, not including ECC data width
	ECC	Disabled	Enables error correction code support. ECC is supported
Interrupts	Burst Length	8 ~	Minimum number of data beats the controller should use
	DDR	533 333333 🔕	Memory clock frequency. The allowed freq tange is (200.0
	Internal Vref		Enables internal voltage reference source. Disable to use
	Juntion Temperature (C)	Normal (0-85) 🗸	Intended operating temperature range. Controls the DDR
	<ul> <li>Memory Part Configuration</li> </ul>		
	DRAM IC Bus Width	16 Bits	Width of individual ORAM components.
	DRAM Device Capacity	4096 MBits	Storage capacity of individual DRAM components.
	<		

### 12.1.4 コンフィグ割り込み

### 割り込みの設定、IRQ\_F2Pを有効にし、PL側で割り込みを受信

Documentation OPreset	s 🐚 IP Location 🚱 Import XPS Se	ettings		
Page Navigator —	Interrupts		Summar	y Repor
Zyng Block Design	• Q = +			
	Sasth D.			
PS-PL Configuration	Internint Port	ID	Description	
Peripheral VO Pins	Eabric Internuots	10	Enable PL Interrunts to PS and vice versa	
	<ul> <li>PL-PS Interrupt Ports</li> </ul>			
alo consguration	V IRQ_F2P[15:0]	[91:84], [6	Enables 16-bit shared interrupt port from the PL. MSB is assigned th	
Clock Configuration	Core0_nFIQ	28	Enables fast private interrupt signal for CPU0 from the PL	
DDB Configuration	Core0_nIRQ	31	Enables private interrupt signal for CPU0 from the PL	
DDR Configuration	Core1_nFIQ	28	Enables fast private interrupt signal for CPU1 from the PL	
SMC Timing Calculation	Core1_nIRQ	31	Enables private interrupt signal for CPU1 from the PL	
	> PS-PL Interrupt Ports			
nterrupts				



12.1.5 VDMA のコンフィグ

1) VDMA IP を追加する



### 2)以下に示すように、VDMA基本パラメーターをコンフィグする。

XXI Video Direct Memory Access (6.3)					1
Documentation 🔚 IP Location					
Show disabled ports	Component Name axi_vdma_0				
	Basic: Advanced:				
	Address Width (32-64) 32	bits			
	Frame Buffers 1 ~				
	Enable Write Channel		Z Enable Read Channel		
	Memory Map Data Width	64 ~	Memory Map Data Width	64 🛩	
	Write Burst Size	8 ~	Read Burst Size	16 🗸	
+ s_wo_LITE	Stream Data Width (Auto)	32 ~	Stream Data Width	24 ~	
HADS_FRAME_PTH_IN_0     M_A01_MADS -     z_aci_Bo_acis     M_A01_MADS -     M_A01_MADS	+ Line Buffer Depth	512 ~	Line Buffer Depth	4096 ~	
< au, nowin	Note:				_
	Configuring the IP with streaming width w	hich is not a pow	er of 2		
	results in significant increase in resource	s			

### 3) VDMA高度なパラメーターのコンフィグ



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

Re-customize IP 1 AXI Video Direct Memory Access (6.3) O Documentation IP Location Show disabled ports Component Name axi\_vdma\_0 Basic Advanced 2 Enable Asynchronous Mode (Auto) 🔳 🔲 Enable Single AXI4 Data Interface Enable Vertical Flip Write Channel Options **Read Channel Options** Fsync Options s2mm tuser v Fsync Options None v ¥. GenLock Mode Slave ~ GenLock Mode Master + S\_AO\_LITE + MADS\_FRAME\_PTR\_IN\_P M\_AU\_MADS + M\_AUS\_MADS + 3 MADS\_FRAME\_PTR\_OUT + 1 Allow Unaligned Transfers Z Allow Unaligned Transfers a jacijita jacik m jacijimn2s jacik avis\_mm2s\_avik mm2s\_introd Cancel OK 4) ビデオタイミングコントローラーを追加する ayram A Muureaa Luitor ~ a a 💥 🔯 🕂 a 😤 P. 🕨 🖸 C 의 🗈 Designer Assistance available. Run Block Automat Run Connection Automation xi vdma 0 M\_AXI\_MM2S + Search: Q- timing (1 match) M\_AXIS\_MM2S + ♀ Video Timing Controller s\_frame\_ptr\_out[5:0] mm2s\_introut rect Memo ENTER to select, ESC to cancel, Ctrl+Q for IP details proce + S\_AXI\_HP0\_FIF0\_CTRL

5) ビデオタイミングコントローラーパラメーターのコンフィグ



日昇テクノロジーなら可能にする



7) AXIストリーミングビデオ出力コントローラーパラメーターのコンフィグ



23 Re-customize IP AXI4-Stream to Video Out (4.0) A Ocumentation IP Location Show disabled ports Component Name v\_axi4s\_vid\_out\_0 Pixels Per Clock 1 v Auto Video Format RGB v Auto AXI4S Video Input Component Width 8 Native Video Output Component Width 8 ÷ + video\_in + vtiming\_in FIFO Depth 2048 vid\_io\_out + acik aciken aciken vtg ce -Clock Mode looked overflow - tid - vid\_io\_out\_clk - vid\_io\_out\_ce Common 

Independent underflow status[31:0] vid\_io\_out\_reset Timing Mode 🔿 Slave 💿 Master [0 - 2047] Hysteresis Level 12 OK Cancel

#### 12.1.6 カスタム IP を追加する

1)ビデオには多くの解像度があるため、さまざまなクロック周波数が異なるため、ダイナミッククロック コントローラーを使用する必要がある。このIPはオープンソースソフトウェアから提供されている。サンプ ル内のrepoディレクトリを見つけて自分のディレクトリにコピーする。

Herry.	15° KA 14 /93	~=	~~ ~
🔰 repo	2018/3/8 11:41	文件夹	
🎉 vdma_hdmi_out.cache	2018/3/8 11:24	文件夹	
🍌 vdma_hdmi_out.hw	2018/3/8 11:24	文件夹	
🍌 vdma_hdmi_out.ip_user_files	2018/3/8 11:24	文件夹	
퉬 vdma_hdmi_out.sim	2018/3/8 11:24	文件夹	
🍌 vdma_hdmi_out.srcs	2018/3/8 11:27	文件夹	
À vdma_hdmi_out.xpr	2018/3/8 11:27	Vivado Project Fi	7 KB

#### 2) IPウェアハウスを追加する



Y PROJECT MANAGER	Sources Design × Signals ? _ □ Ľ	Diagram × Address Editor × IP Catalog ×
Settings	이 폰 병 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이	Cores   Interfaces
Add Sources	🛦 design_1	
Language Templates	Paxi_vdma_0 (AXI Video Direct Memory Access:6.3)	
P Catalog	> Processing_system7_0 (ZYNQ7 Processing System:5.5)	Name
	>      v_axi4s_vid_out_0 (AXI4-Stream to Video Out.4.0)     v_axi4s_vid_out.4.0     v_axi4s_vid_out.4.0	Allia Properties Ctrl+E
Y IP INTEGRATOR	> V_tc_0 (video Timing Controller:6.1)	>  Auto IP Settings
Create Block Design		AXI Add Repository
Open Block Design		> AXIS Refresh All Repositories
Generate Block Design		Bas     Export to Spreadsheet
		Communication & Networking
Run Simulation		> S Debug & Verification
	Repository Properties ? _ D D X	> 🗁 Digital Signal Processing
Y RTL ANALYSIS	🗅 User Repository 🔶 🔶 🔅	> 😑 Embedded Processing
Diagram x Address E	ditor × IP Catalog ×	
Cores   Interfaces		
	F 2 0	
Name		^ 1 AXI4
User Repository (f./a	x7015/demo/vdma_hdmi_out/repo)	
> 🚍 FPGA Features ar	nd Design	
✓		
👎 Dynamic Clock	Generator	AXI4
🗸 📄 Video & Image Pr	ocessing	
P RGB to DVI Vid	eo Encoder (Source)	
👻 📄 Vivado Repository		
> 📄 Alliance Partners		
> 🖹 Automotive & Indu	Istrial	
> 📄 AXI Infrastructure		
> 📄 AXIS Infrastructure	9	

#### 4) ダイナミッククロックコントローラーを追加する







株式会社日昇テクノロジー

Re-customize IP	a sublishing the Connection Subscription	2
GB to DVI Video Encoder (S	ource) (1.3)	4
Documentation 📄 IP Location		
Show disabled ports	Component Name rgb2dvi_0	
	MMCM/PLL	
	MMCM     PLL	
	Reset active high     Generate SerialClk internally from pixel clock.	
	TMDS clock range                • >=120 MHz (1080p)              · < 120 MHz (720p)	TMDS clock range
		Cancel
ivadoへの接続がクロッ	フ信号に自動的に接続しない場合がある	
axi_dyndk_0	v_b_0 + cl ck ck ck ck ck ck ck ck ck ck	processing_system7_0 FO_CTRL FI: .K ZYNQ MA FC FC FC ZYNQ7 Processing System
al, ack al, ansoth Nock Generator (Pre-Production) Sector (Pre-Production) (+ S, AX m, acl, m acl, rosc	axi_vdma_0          intk         M_AXI_MW2S + if          intk         M_AXI_MW2S + if          intk         M_AXIS_MW2S + if	rgb2dvi_0 + RGB • Rot_n TMOS +

2)他の重要な信号を接続する

fid vid\_io\_out\_ck vid\_io\_out\_ce vid\_io\_out\_reset

AXI4-Stream to Video Out



# 株式会社日昇テクノロジー



3)割り込み信号に接続するには、最初に信号接続用のConcat IPを追加する必要がある。

Diagram × Address Editor × IP Catalog ×	
$\mathbf{Q} \mid \mathbf{Q} \mid \mathbf{X} \mid \mathbf{\Sigma} \mid \odot \mid \mathbf{Q} \mid \mathbf{X} \mid \Leftrightarrow \mid \mathbf{H} \mid \Box_{\mathbf{X}}$	⊁   ⊠   C   ⊴   ₽
* Designer Assistance available. Run Block Automation Ru	un Connection Automation
Search: Q- concat (1 match) Concat	
ENTER to select, ESC to cancel, Ctrl+Q for IP details	processing_system7_0 .HP0_FIF0_CTRL .HP0 P0_ACLK 20_ACLK 0:0] FCL ZYNQ7 Processing System avii dunally 0



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



#### 5) すべてのモジュールを選択して自動的に接続する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



<sup>7)</sup> ポートを選択して、エクスポート。





8) IIC\_0ポートをエクスポート



10)別のポートの名前を変更する





12) HDMI出力のxdcファイルを追加する。



<u>File Edit</u> Flow <u>T</u> ools <u>W</u> indow	Layout View Help Q- Quick Access	
	☑ ▶ # ✿ Σ ≤ ∅ ¥	
Flow Navigator — 🗄 🔍 _	BLOCK DESIGN - design_1	
<ul> <li>PROJECT MANAGER</li> <li>Settings</li> <li>Add Sources</li> <li>Language Temples</li> <li>IP Catalog</li> </ul>	Sources × Design Signals ? Q ¥ + Add Sources > Design Sources (1) > @ A design_1_wrapp > @ design_1 : > @ design_1	Diagram × Address Editor × IP Catalog × design_1_wrapp      Add Sources      This guides you through the process of adding and creating sources for your project      Add or greate constraints
<ul> <li>IP INTEGRATOR</li> <li>Create Block Design</li> <li>Open Block Design</li> <li>Generate Block Design</li> </ul>	<ul> <li>Constraints</li> <li>constraint</li> <li>Constraint</li> <li>Simulation Sources</li> <li>Sim_1(2)</li> </ul>	<ul> <li>Add or create design sources</li> <li>Add or create <u>s</u>imulation sources</li> </ul>
<ul> <li>SIMULATION</li> <li>Run Simulation</li> </ul>		
Y RTL ANALYSIS	Hierarchy IP Sources	
> Open Elaborated Design	Source File Properties	
<ul> <li>SYNTHESIS</li> <li>Run Synthesis</li> </ul>	edesign_1_wrapper.v         ∑ Enabled	X
> Open Synthesized Design	Location: F/ax7015	s Bark
	Type: Verilog	S inter 105 ek n.
<ul> <li>Nun implementation</li> </ul>		

### 13) xdcファイル内容は以下通りである。

set\_property IOSTANDARD TMDS\_33 [get\_ports TMDS\_clk\_n] set\_property PACKAGE\_PIN\_N18 [get\_ports TMDS\_clk\_p] set\_property IOSTANDARD TMDS\_33 [get\_ports TMDS\_clk\_p] set\_property IOSTANDARD\_TMDS\_33 [get\_ports {TMDS\_data\_n[0]}] set\_property PACKAGE\_PIN\_V20 [get\_ports {TMDS\_data\_p[0]}] set\_property IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_p[0]}] set\_property IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_n[1]}] set\_property PACKAGE\_PIN T20 [get\_ports {TMDS\_data\_p[1]}] set\_property.IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_p[1]}] set property IOSTANDARD TMDS 33 [get ports {TMDS data n[2]}] set\_property PACKAGE\_PIN N20 [get\_ports {TMDS\_data\_p[2]}] set\_property IOSTANDARD TMDS\_33 [get\_ports {TMDS\_data\_p[2]}] #set\_property PACKAGE\_PIN Y19 [get\_ports {hdmi\_hpd\_tri\_i[0]}] #set\_property IOSTANDARD LVCMOS33 [get\_ports {hdmi\_hpd\_tri\_i[0]}] set\_property PACKAGE\_PIN V16 [get\_ports hdmi\_oen] set\_property IOSTANDARD LVCMOS33 [get\_ports hdmi\_oen] set\_property PACKAGE\_PIN R18 [get\_ports hdmi\_ddc\_scl\_io] set\_property IOSTANDARD LVCMOS33 [get\_ports hdmi\_ddc\_scl\_io] set\_property PACKAGE\_PIN R16 [get\_ports hdmi\_ddc\_sda\_io] set\_property IOSTANDARD LVCMOS33 [get\_ports hdmi\_ddc\_sda\_io]

### 14) bitファイルのコンパイルと生成


#### 12.2 SDK ソフトウェアの作成とデバッグ

### 1) ハードウェアのエクスポート

1	Export Hardware
	Export hardware platform for software development tools.
	✓ Include bitstream
	Export to: So <local project="" to=""></local>
	Ск Cancel
2) SDKを実行する	
[	Launch SDK
	Launch software development tool.
	Exported location: 🛜 <local project="" to=""> 🗸</local>
	Workspace: 🔂 <local project="" to=""> 🗸</local>
	OK   Cancel

3) vdma\_hdmiというアプリを作成する



低価格、高品質が不可能?

日昇テクノロジーなら可能にする

pplication Proj	ect			Tr
Create a manag	ed make	application proje	ct.	
Project name:	/dma_hdn	ni		
✓ Use default	location			
Location: F:\ax	7015\dem	io\vdma_hdmi_ou	t\vdma_hdmi_out.sdk\v	Brows
Choo	se file syst	tem: default 👻		
r				
OS Platform:	standalor	ne		
Target Hardw	are			
Hardware Plat	form: de	esign_1_wrapper_	hw_platform_0	▼ Ne
Processor:	ps	s7_cortexa9_0		
Target Softwa	re		$\langle \rangle$	
Language:		. C © C++		
Compiler:		32-bit	-	
Hypervisor Gu	est:	N/A	-	
Board Suppor	t Package	e: 🙆 Create New	vdma_hdmi_bsp	
		O Use existing		
	K			
		la contra la contra de la contr		

4)多くのプログラムファイルがあるため、具体的に紹介しなく、直接にサンプルのソースコードをコピーする。srcディレクトリ内のファイルを削除し、代わりにサンプルのsrcディレクトリファイルを使用する。

nulue (r.) , a. , demo ,	vuma_numi_out v vuma_nu	ami_out.suk v vame	a_nomi • src
共享 ▼ 新建文件夹			
名称	修改日期	类型	大小
🎉 display_ctrl	2018/3/8 16:01	文件夹	
🎍 dynclk	2018/3/8 16:01	文件夹	
h display_demo.h	2018/3/8 16:54	C++ Header file	3 KB
script.ld	2018/3/8 15:31	LD 文件	6 KB
main.c	2018/3/8 16:53	C Source file	8 KB
h pic_800_600.h	2017/1/14 22:27	C++ Header file	7,208 KB
Xilinx.spec	2018/3/8 15:31	SPEC 文件	1 KB



5) SDKで更新する



6) HDMI出力ポートをディスプレイに接続し、コンパイルして実行する。

reate, manage, and run configura	ations			
Run or Debug a program using Sy	stem Debugger.			
" 🗎 🗙 🖻 Þ 🔹	Name: System Debugg	er using Debug_vdma_hdmi.elf	on Local	
type filter text	📀 Target Setup 🔲	Application 🗱 Arguments	Environment "4	
Performance Analysis				^
Silinx C/C++ application (G	Hardware Platform:	design_1_wrapper_hw_platfor	m_0 👻	
Strip Xilinx C/C++ application (Sy Xilinx C/C++ application/(Sy Xilinx C/C++ application/(Sy Xilinx C/C++ application/(Sy Xilinx C/C++ application/(Sy Xilinx C/C++ application/(Sy Xilinx C/C++ application (Sy Xilinx C/C++ applicati	Bitstream File:	design_1_wrapper.bit	Search	Brow:
E System Debugger using	Initialization File:	ps7_init.tcl	Search	Brow
	FPGA Device:	Auto Detect	Select	
	PS Device:	Auto Detect	Select	
		Summary of operations	to be performed	1
	Reset entire syste	m Following operations w	ill be performed before l	aunching
×	Program FPGA	1. Resets entire system. 2. Program FPGA fabric	. Clears the FPGA fabric (F : (PL).	ν <b>L</b> ).
	Run ps7_init	3. Runs ps7_init to initia	lize PS.	
	Run ps7_post_cor	fig 4. Runs ps7_post_config	g. Enables level shifters fr	om PL to
	•			•
•				
ilter matched 6 of 10 items			Revert	Apply
?)			Run	Close

7) ディスプレイに画像が表示される







#### 第十三章 プログラムの復帰

実験Vivadoプロジェクトはvdma\_to\_qspi\_sdである。

JTAGデバッグを通して、SDカードにプログラムを書き込む、またはQSPI FLASHに書き込む方法について、 多くの実験を行った。まず、PLにはPSコンフィグが必要であるため、以前のFPDA書き込みのようにFlashに 直接書き込むことはできない。この実験では、プログラムを復帰する方法について説明する。

13.1 Vivado プロジェクトの設立

この実験では、VDMAテストエンジニアリングを選択して復帰したが、VDMAテストプロジェクトを設立する際、QSPIおよびSDカードを有効にできなかった。プログラムを復帰するには、QSPIまたはSDカードを有効に する必要がある。

1) プログラムの復帰テストのため、VDMAテストプロジェクトのコピーを保存し、名前を vdma\_to\_qspi\_sdに変更する。

Project name:	vdma_to_qspi_sd	C
Project location:	F:/ax7015/demo	8
✓ Create project	t subdirectory	
	astad at Every 7015/demokidmenta applied	
Project will be cr	eated at P./atro ib/demo/vdma_to_dspi_sd	

QSPIを追加し、MI01-6を使用する。



ZYNQ7 Processing Sy	stem (5.5)													¢
Documentation OPres	sets 📄 IP Location 🚳	Import X	PS Settings											
Page Navigator —	Peripheral I/O Pin	5										Sum	imary	Report
Zynq Block Design	+ Q ₹ \$	0										200		
PS-PL Configuration	Search: Q-													
Peripheral VO Pins			Bank 0	LVCMOS 3	3V ~	•		Bank	1 L	VCMO	S 1.8V	v		
MIO Configuration	Peripherals	0 1	2 3 4 5 6	78	9 10	11 12 1	13 14 1	5 16 1	17 18	19	20 21	22	23	24 25
Clock Configuration	Single SS		Single SS 4bit IO											
DDR Configuration	O Dual Quad	-	si_b											
100000000000000000000000000000000000000	O Dual Quad	0	lual Quad SPI (8bit)		Oual Qu	ad SP1 (8bit	5							
SMC Timing Calculation	Feedback			6										
interrupts	> SRAMNOR F		SR	MINDR Flash								•		SRAM
	> 🔄 NAND Flash	a		NAND FIL	ats .					1.1				
	> Ethernet 0	_									E	netO		
	> Ethernet 1													
	USB 0	<u> </u>							-					
	USB 1								-			-		-
	> 🗹 SD 0									800				
	3 SD 1	¢ ===				-				×				>

3) SD0コントローラーを追加、MI040-45及びTFカードインターフェイスを使用する





#### 4) SDカードの検出ピンMI047を追加する



5) デザインを保存、bitファイルをコンパイルして生成し、ハードウェアを再度エクスポートする。

### 13.2 FSBL を生成する

FSBLは、MIOの割り当て、DDRコントローラーの初期化、SD、QSPIコントローラーの初期化を完了し、FPGA をコンフィグして、ユーザープログラムを読み込むセカンダリブートローダーである。

1) SDKソフトウェアを起動する。SDKソフトウェアは他のプロジェクトからコピーしてきたため、以前のSDKプロジェクトであり、パスの変更により、もう1つのhw\_platform\_1がある。



低価格、高品質が不可能?

日昇テクノロジーなら可能にする

vdma_to_qspi_sd.sdk - C/C++ - Xilinx SDK					
le Edit Navigate Search Project Run Xilir	x Window Help				
3•30000•%•800000	おえる法 見図 回着	🛃 🖬 🚱 🎄 🔹 (	0 • 🛷 • 🐤	6.0.	
					Quick Access 🕴 🖽 🗔 🕇
🖢 Project Explorer 🕮 😑 😫 🖤 🔍 🗖 🔲	a system.hdf 🖾			- 0	2:02 ×2 = E
<pre>design_1_wrapper_hw_platform_0     design_1_wrapper_hw_platform_1</pre>	design_1_wrapper_hw_p	latform_1 Hardwa	are Platform S	pecificati	An outline is not available.
vdma_hdmi	Design Information			E	
	Target FPGA Device: 7z015 Part: xc7z015 Created With: Vivado Created On: Thu Ma	clg485-2 2017.4 r 8 20:28:20 2018			
	Address Map for processor ps	7_cortexa9_[0-1]			
	Cell	Base Addr	High Addr	Slave I/f	
	ps7_intc_dist_0	0xf8f01000	0xf8f01fff		
	axi dynclk 0	0x43c10000	0x43c1fff	\$00 au	
	ps7_scutimer_0	0xf8f00600	0xf8f0061f		
	ps7 slcr 0	0xf8000000	0xf8000fff		
	axi onio 0	0x41200000	0x4120/00	S AXT	
	or7 roundt 0	0×68600620	0~1810061	2,000	
	ps7 12carber 0	0xf8f02000	Oxf8ND2fff		
	particounerio	N NIGIOZOGO	GARDIGETT		
	Overview				
Target Connections 🛛 🦨 🖉 😁 🗖	👷 Pr., 🖉 T., 🖸 C., 🖾 🗖	] Pr 🛄 S 😬 🕻		Debug 23	· · · · · · · · · · · · · · · · · · ·
▷ 🧽 Hardware Server ▷ 🍐 Linux TCF Agent ▷ 🎥 QEMU TcfGdbClient	CDT Build Console [vdma_hdmi_t	sep]	*		
vdma_hdmi_bsp			1		
2)fsblという新しいアプリ 択する。	を作成し、ハードウェ	ェアプラットス	フォームにま	最新のプ <sup>:</sup>	ラットフォームを



低価格、高品質が不可能?

日昇テクノロジーなら可能にする

Use default locati	on
Location: F:\ax7015\	demo\vdma_to_qspi_sd\vdma_to_qspi_sd.sdl Browse
Choose file	system: default *
OS Platform: stand	alone
Target Hardware	
Hardware Platform:	design_1_wrapper_hw_platform_1
Processor:	ps7_cortexa9_0
Target Software	
Language:	
Compiler:	32-bit
Hypervisor Guest:	N/A -
Board Support Pack	cage:      Create New fsbl_bsp
	Use existing
-	





4) デバッグマクロ定義FSBL\_DEBUG\_INFOを追加すると、FSBLの出力ステータス情報を開始できる。これ はデバッグには役立つが、起動時間が長くなる。



低価格、高品質が不可能?

日昇テクノロジーなら可能にする

e Edit Navigate Search Pro	ject Run Xilinx Window Help	in the second		
e Edit Navigate Search Pro Project Explorer 22  Project Explorer 22  Gesign 1_wrapper_hw_pla Gesign 1_wrapper_hw_pla Fisbl Binaries Binar	iect Run Xilinx Window Help	£ □ : # D ■ @ + • • • •		Quick Access
<ul> <li>is tsbl_handott.S</li> <li>is fsbl_hooks.c</li> <li>is fsbl_hooks.h</li> <li>is fsbl.h</li> <li>is image_mover.c</li> <li>is image_mover.h</li> <li>is main.c</li> <li>is md5.c</li> <li>is md5.h</li> <li>is nand.c</li> <li>is nand.h</li> <li>is nand.h</li> <li>is nand.h</li> <li>is nand.h</li> <li>is nand.h</li> </ul>	<pre>#rtern "C" { #endif #define DEBUG_GENERAL 0xi #define DEBUG_INFO 0x00000 #define FSBL_DEBUG_INFO #if defined (FSBL_DEBUG_INFO #if defined (FSBL_DEBUG_INFO #define fsbl_dbg_current_t; #else #define fsbl_dbg_current_t; #else</pre>	00000001 /* general debug mes 0002 /* More debug information EO) ypes ((DEBUG_INFO)   (DEBUG_GENER ypes (DEBUG_GENERAL) ynes R m	ssages */	¥ fsbl_printf()
Target Connect 22 P D & L Hardware Server Chinux TCF Agent C QEMU TcfGdbClient	Problems Tasks Console CDT Build Console [fsb] 'Invoking: ARM v7 Print Size' arm-none-eabi-size fsbl.elf [t text data bss det	tee "fsbl.elf.size" → mex filename		g ‡o Debug ⊠ ⊟ E Ng #r  i+ ≺

5) SDKはデフォルトで自動的にコンパイルされ、fsbl.elfファイルが生成される。

### 13.3 BOOT ファイルを作成する

1) fsblプロジェクトを選択し、Create Boot Imageを選択する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



2) 生成されたBIFファイルパスは、ポップアップウィンドウに表示される。BIFファイルは、BOOTファ イルを生成するためのコンフィグファイルであり、生成されたBOOT.binファイルパスである。BOOT.binファ イルは、必要なスタートアップファイルであり、SDカードに入れることができ、 QSPI Flashにも書き込み 可能である。





eate Boot Image		
reates Zynq Boot Image in .bin format from	given FSBL elf and partition files in specified output folder.	20
chitecture: Zynq 🔹		
Create new BIF file	BIF file	
asic Security		
Dutput BIF file path: F:\ax7015\demo\vdma_t	to_qspi_sd\vdma_to_qspi_sd.sdk\fsbl\bootimage\fsbl.bif	Browse
DF data:		Browse
Split Output format: BIN •	-	
Dutput path: F:\ax7015\demo\vdma t	to aspi sd\vdma to aspi sd.sdk\fsbl\bootimage\BOOT.bin	Browse
oot image partitions		
oot image partitions File path	Encrypted Authenticat	Add
bot image partitions File path bootloader) F:\ax7015\demo\vdma_to_qspi_s F:\ax7015\demo\vdma_to_qspi_sd\vdma_to_q	Encrypted Authenticat sd\vdma_to_qspi none none qspi_sd.sdk\desig none none	Add
bot image partitions File path bootloader) F:\ax7015\demo\vdma_to_qspi_sd\vdma_to_q	Encrypted Authenticat sd\vdma_to_qspi none none qspi_sd.sdk\desig none none	Add
oot image partitions File path (bootloader) F:\ax7015\demo\vdma_to_qspi_s F:\ax7015\demo\vdma_to_qspi_sd\vdma_to_q	Encrypted Authenticat sd\vdma_to_qspi none none qspi_sd.sdk\desig none none	Add Delete Edit
oot image partitions File path bootloader) F:\ax7015\demo\vdma_to_qspi_s F:\ax7015\demo\vdma_to_qspi_sd\vdma_to_q	Encrypted Authenticat sd\vdma_to_qspi none none qspi_sd.sdk\desig none none	Add Delete Edit Up
oot image partitions File path bootloader) F:\ax7015\demo\vdma_to_qspi_s F:\ax7015\demo\vdma_to_qspi_sd\vdma_to_q	Encrypted Authenticat sd\vdma_to_qspi none none qspi_sd.sdk\desig none none	Add Delete Edit Dowr
oot image partitions File path bootloader) F:\ax7015\demo\vdma_to_qspi_s F:\ax7015\demo\vdma_to_qspi_sd\vdma_to_q	Encrypted Authenticat sd\vdma_to_qspi none none qspi_sd.sdk\desig none none	Add Delete Edit Dowr

3) Boot image partitionsリストに合成するファイルがあ。最初のファイルはbootloaderファイルで、 上記で生成されたfsbl.elfファイルである。2番目のファイルはFPGAコンフィギュレーションファイルであ る。[Add]をクリックして、VDMAテストプログラムvdma\_hdmi .elfを追加する。



Who Cre	ate Boot Image				
a mb Creat	e Boot Image				
Crea	tes Zynq Boot Image in .bin format from	given FSBL elf and partition files in specifie	ed output folder.	_٤	Ŏ.
and Add partition					
Add new boot	image partition				
Add new boo	image partition				
File path:	5\demo`vdma_to_qspi_sd\vdma_to_qspi	_sd.sdk\vdma_hdmi\Debug\vdma_hdmi.elf	Browse	Browse	
Partition type:	datafile 🔹			Browse	
Authentication:	none Encryption:	none v			
Checksum	none		bin	Browse	
Deselant	in the second se				
Other			Browse		
Alignment:		Offset:			
Reserve:		Load:			
Startup:		1	ypt	ed A Ad	a
			1		
				Edi	
?		ОК	Cancel	Up	
11140				, Dov	YIN I
/* _F					
?		Preview BIF Changes	te Image	Cancel	
; J/i    A.					
[Creat	a [mage]をクリック」	、て生成する、			
[Creat	e Image]をクリックし	て生成する。			
[Creat	e Image]をクリックし <sup>mage</sup>	て生成する。			X
[Creat Create Boot	e Image]をクリックし <sup>Image</sup> 1 <b>age</b>	て生成する。			
[Creat Create Boot Treate Boot In Creates Zynq	e Image]をクリックし Image Boot Image in .bin format from give	.て生成する。 en FSBC elf and partition files in speci	fied output fold	er.	
[Creat Create Boot In Creates Zynq	e Image]をクリックし Image Boot Image in .bin format from give	・て生成する。 en FSBC elf and partition files in speci	fied output fold	er.	
[Creat Create Boot reate Boot In Creates Zynq rchitecture:	e Image]をクリックし Image Boot Image in .bin format from give	って生成する。 en FSBC elf and partition files in speci	fied output fold	er.	
[Creat Create Boot reate Boot In Creates Zynq rchitecture:	e Image]をクリックし Image Boot Image in .bin format from give	.て生成する。 en FSBC elf and partition files in speci file	fied output fold	er.	
[Create Boot reate Boot In Creates Zynq rchitecture:	e Image]をクリックし Image Boot Image in .bin format from give Iynq マ NF file © Import from existing BIP ty	・て生成する。 en FSBC elf and partition files in speci file	fied output fold	er.	
[Creat Create Boot In Create Boot In Creates Zynq rchitecture:	e Image]をクリックし Image Boot Image in .bin format from give Lyng ・ SIF file © Import from existing BIP by path: Ft\ax7015\demo\vdma_to_c	.て生成する。 en FSBC elf and partition files in speci (file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif	er.	rowse
[Creat Create Boot In Create Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data:	e Image]をクリックし Image Boot Image in .bin format from give Zynq マ SIF file © Import from existing BIF ity path: F:\ax2015)demolydma.to.c	.て生成する。 en FSBL elf and partition files in speci file gspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif	er. B	
[Create Boot reate Boot In Creates Zynq rchitecture:	e Image]をクリックし Image Boot Image in .bin format from give Lyng 、 BIF file © Import from existing BIF ity path: F:\ax7015\demo\vdma_to_c	.て生成する。 en FSBL elf and partition files in speci (file	fied output fold otimage\fsbl.bif	er. B	rowse
[Create Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output pathe	e Image]をクリックし Image Boot Image in .bin format from give Lynq ・ SIF file Import from duisting BIP ity ty the Ft/ax7015/demo/vdma_to_c Output format: BIN ・ Ft/ax7015/demo/vdma_to_c	.て生成する。 en FSBC elf and partition files in speci file gspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif otimage\BOOT.	er. f B bin B	
[Create Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ SIF file © Import from existing BIP ity path: F:\ax2015)demo\vdma_to_c Output format: BIN ・ F:\ax2015\demo\vdma_to_c	.て生成する。 en FSBL elf and partition files in speci file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif otimage\BOOT.	er. B bin B	
[Create Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path	e Image]をクリックし Image Boot Image in .bin format from give Zyng ・ BIF file © Import from existing BIFI ity : path: F:\ax7015\demo\vdma_to_c Output format: BIN ・ F:\ax7015\demo\vdma_to_c	.て生成する。 en FSBL elf and partition files in speci (file gspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo gspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif otimage\BOOT.	er. B bin B	rowse
[Creat Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ 31F file O Import from existing BIP ity path: F:\ax7015\demo\vdma_to_c Output format: BIN ・ F:\ax7015\demo\vdma_to_c	.て生成する。 en FSBC elf and partition files in speci file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif otimage\BOOT.	er. f B bin B	
[Create Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ 3IF file ① Import from existing BIP ity > path: F:\ax2015)demo\vdma_to_c Output format: BIN ・ F:\ax2015\demo\vdma_to_c	.て生成する。 en FSBL elf and partition files in speci file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif otimage\BOOT.	er. f B bin B	rowse
[Create Boot reate Boot In Creates Zynq rchitecture: Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ 3IF file © Import from existing BIF ity path: F:\ax7015\demo\vdma_to_c Output format: BIN ・ F:\ax7015\demo\vdma_to_c	.て生成する。 en FSBL elf and partition files in speci file aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif otimage\BOOT.	er. B bin B	rowse
[Create Boot reate Boot In Creates Zynq rchitecture: [: 0 Create new I Basic Secur Output BIF file UDF data: ] Split Output path Boot image pu	e Image]をクリックし Image Tage Boot Image in .bin format from give Zyng ・ BIF file Import from existing BIF ity Path: F:\ax7015\demo\vdma_to_c Output format: BIN・ F:\ax7015\demo\vdma_to_c	・て生成する。 en FSBC elf and partition files in speci file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo	fied output fold otimage\fsbl.bif otimage\BOOT.	er. B bin B Authenticat.	rowse rowse
[Create Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path Soot image pu	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ BIF file O Import from existing BIF ity Path: F:\ax7015\demo\vdma_to_c F:\ax7015\demo\vdma_to_c intitions spi_sd\vdma_to_qspi_sd.sdk\fsbl\Do to_qspi_sd.sdk\design_1_wrapper_h	.て生成する。 en FSBL elf and partition files in speci file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo ebug\fsbl.elf nw_platform_1\design_1_wrapper.bit	fied output fold otimage\fsbl.bif otimage\BOOT.	er. F B bin B Authenticat. none none	rowse rowse rowse
[Create Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path Soot image puno/vdma_to_c	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ SIF file ① Import from dvisting BIF ity a path: F:\ax2015)demo\vdma_to_c Output format: BIN ・ F:\ax2015\demo\vdma_to_c artitions spi_sd\vdma_to_qspi_sd.sdk\fsbl\De to_qspi_sd.sdk\vdma_hdm\Debug\	.て生成する。 en FSBC elf and partition files in speci file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo ebug\fsbl.elf mv_platform_1\design_1_wrapper.bit vdma_hdmi.elf	fied output fold otimage\fsbl.bif otimage\BOOT.	er. f B bin B Authenticat. none none none	rowse rowse rowse
[Creat Create Boot reate Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path Boot image pu no\vdma_to_c aspi_sd\vdma, aspi_sd\vdma,	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ 3IF file Import from existing BIF ity s path: F:\ax7015\demo\vdma_to_s Output format: BIN ・ F:\ax7015\demo\vdma_to_c urtitions spi_sd\vdma_to_qspi_sd.sdk\fsbl\Deto to_qspi_sd.sdk\vdma_hdm\Debug\vdma_to_to	.て生成する。 en FSBC elf and partition files in speci file aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo ebug\fsbl.elf nw_platform_1\design_1_wrapper.bit vdma_hdmi.elf	fied output fold otimage\fsbl.bif otimage\BOOT. Encrypted none none none	er. B bin B Authenticat. none none none	rowse rowse rowse rowse total
[Creat Boot In Create Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path Boot image pu no\vdma_to_c qspi_sd\vdma, qspi_sd\vdma,	e Image]をクリックし Image Boot Image in .bin format from give Zyng ・ BIF file Import from existing BIF ity s path: F\\ax7015\demo\vdma_to_c Output formats BIN ・ F\\ax7015\demo\vdma_to_c urtitions	・て生成する。 en FSBC elf and partition files in speci (file hspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo ebug\fsbl.elf mv_platform_1\design_1_wrapper.bit vdma_hdmi.elf	fied output fold otimage\fsbl.bif otimage\BOOT.	er. B bin B Authenticat. none none none	rowse rowse rowse rowse rowse tourse rowse rowse
[Creat Boot In Create Boot In Creates Zynq Inchitecture: [] Create new I Basic Secur Output BIF file UDF data: Split Output path Boot image pi mo\vdma_to_c aspi_sd\vdma, aspi_sd\vdma,	e Image]をクリックし Image Boot Image in .bin format from give Zynq ・ BIF file O Import from existing BIF ity a path: F:\ax7015\demo\vdma_to_c Output format: BIN ・ F\ax7015\demo\vdma_to_c intitions spi_sd\vdma_to_qspi_sd.sdk\fsbl\Debug\vdma_to_c	・て生成する。 en FSBC elf and partition files in speci file qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo qspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo ebug\fsbl.elf nv_platform_1\design_1_wrapper.bit vdma_hdmi.elf	fied output fold otimage\fsbl.bif otimage\BOOT.	er. F B bin B Authenticat. none none	rowse rowse rowse rowse rowse Labelete Edit Up
[Creat Boot reate Boot In Creates Boot In Creates Zynq rchitecture: D Create new I Basic Secur Output BIF file UDF data: Split Output path Boot image pu no\vdma_to_c aspi_sd\vdma, aspi_sd\vdma,	e Image]をクリックし Image Tage Boot Image in .bin format from give Zynq ・ 3IF file ① Import from existing BIF ity a path: F:\ax2015)demolvdma_to_co Output format: BIN ・ F:\ax2015)demolvdma_to_co Intitions spi_sd\vdma_to_qspi_sd.sdk\fsbl\Do to_qspi_sd.sdk\vdma_hdmi\Debug\vdma_to_co	.て生成する。 en FSBC elf and partition files in speci file aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo ebug\fsbl.elf mv_platform_1\design_1_wrapper.bit vdma_hdmi.elf	fied output fold otimage\fsbl.bif otimage\BOOT. Encrypted none none none	er. f B bin B Authenticat. none none none	rowse rowse rowse rowse ddd Delete Edit Up Down
[Creat Boot Create Boot In Create Boot In Creates Zynq rchitecture: Create new I Basic Secur Output BIF file UDF data: Split Output path Boot image pu no\vdma_to_c spi_sd\vdma spi_sd\vdma	e Image]をクリックし Image Tage Boot Image in .bin format from give Zynq ・ 3IF file Import from existing BIF ity s path: F:\ax7015\demo\vdma_to_s Output format: BIN・ F:\ax7015\demo\vdma_to_s spi_sd\vdma_to_qspi_sd.sdk\fsb\Det to_qspi_sd.sdk\vdma_hdm\Debug\	・て生成する。 en FSBL elf and partition files in specia (file aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo aspi_sd\vdma_to_qspi_sd.sdk\fsbl\boo ebug\fsbl.elf mw_platform_1\design_1_wrapper.bit vdma_hdmi.elf	fied output fold otimage\fsbl.bif otimage\BOOT. Encrypted none none none	er. B bin B Authenticat. none none none	Add Delete Edit Down



5) BOOT.binファイルは、生成されたディレクトリにある。



- 13.4 SD カードの起動テスト
- 1) SDカードをフォーマットし、FAT32にフォーマットする。他のフォーマットは開始できない。

ſ	格式化 FAT (J:)	
	容里(P):	
	4.65 GB ▼ 文件系统 (F)	
	FAT32 (默认) 🗸	
	分配单元大小(A) 4096 字节	
	<u></u> 建原设面的款以值 (4)	
	RAT	
	格式化选项(0)	
	<ul> <li>快速格式化(Q)</li> <li>创建一个 MS-DOS 启动盘(M)</li> </ul>	
	王始の二〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇	

2) BOOT.binファイルをルートディレクトリに入れる。





- 3) SDカードを開発ボードのSDカードスロットに挿入する。
- 4) 開始モードをSDカードブートに調整



- 5) HDMIディスプレイを接続し、ボードの電源を入れると、ディスプレイに子猫の写真が表示される。
- 13.5 QSPI テスト開始
- 1) SDK $\forall = = -\mathcal{O}$ [Xilinx]-> [Program Flash]



File Edit Navigate Search Project Run	Xilir	Window Help
<ul> <li>➡ □</li> <li>□</li> <li>➡ □</li> <li>➡ □</li></ul>	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Generate linker script Board Support Package Settings Repositories
<ul> <li>         design_1_wrapper_hw_platform_0      </li> <li>         design_1_wrapper_hw_platform_1      </li> <li>         fsbl     </li> <li>         fsbl_bsp     </li> </ul>	## 20 20 20 20 20 20 20 20 20 20	Program FPGA Dump/Restore Data File Program Flash
⊳ 🚰 vdma_hdmi ⊳ 🎒 vdma_hdmi_bsp	> •	Launch She Program parallel flash memory XSCT Console
		#define _FSBL_DEBUG_H

2) Hardware Platformは最新のを選択、Image FIIeファイルは書き込んだBOOT.binを選択、FSBL file はコア電子カスタムの特別バージョンfsbl.elfを選択し、このfsblしか書き込めない

Program Flash N	lemory	
Program Flash N	Memory via In-system Programmer.	
Hardware Platfor	n: design_1_wrapper_hw_platform_1	
Connection:	Local	New
Device:	Auto Detect	Select
image File:	pi_sd\vdma_to_qspi_sd.sdk\fsbl\bootimage\BOOT.bi	Browse
Offset:		
Flash Type	gspi_single	
FSBL File:	F:\ax7015\zynq_fsbl_flash.elf	Browse
Convert ELF to	bootloadable SREC format and program	
Verify after flag	h	

3) [Program]をクリックして、プログラミングが完了するのを待つ。

4) ブートモードをQSPIに設定して、もう一度起動すると、ディスプレイにディスプレイ出力が出てくる。

13.6 Vivado のもとで QSPI を書き込む



1) HARDWARE MANGERでデバイスを選択し、[Add Configuration Memory Device]をクリックする。

There are no debui	g cores. Program dev	ice Reifesh device	
Hardware		2 _ 0 & X	
Q X 0	0   •   »   🔳	0	
Name		Status	
🗸 📱 localhost (1)		Connected	
✓ Image: Value > V	Digilent/2102498547	Open	
📵 arm_da	<b>p_0</b> (0)	N/A	
Si XAD(	Program Device Verify Device Refresh Device	e	
Hardware Device P	Boot from Cont	figuration Nemory Device	
<pre>(i) xc7z015_1 Name: Part:</pre>	Program BBR Clear BBR Key Program eFUS	Key E Registers	

2) Winbondを選択し、タイプをaspi、幅をx4-single入力すると、w25q128が表示され、開発ボードは w25q256を使用、書き込みには影響がない。



Add Configuration Memory Device Choose a configuration memory part. This can be changed later. Device: () xc7z015\_1 Filter Manufacturer Winbond Туре qspi Density (Mb) All Width x4-single Reset All Filters Select Configuration Memory Part Search: Q-Manufact. Alias Name Part 🛊 w25q128-qspi-x4-single w25q128 Winbond 0 < = (?) Cancel OF 右クリックしてプログラミングファイルを選択する。 3) Iocalhost (1) Connected v I or xilinx\_tcf/Digilent/2102498547... Open arm\_dap\_0 (0) N/A xc7z015\_1 (2) Not programm MADC (System Monitor) w25q128-cspi-x4 Configuration Memory Device Properties... Ctrl+E Remove Configuration Memory Device Delete Program Configuration Memory Device... Readback Configuration Memory Device. Configuration Memory Device Export to Spreadsheet... w25q128-qspi-x4-single uncentra anni vi sinala Mama:

4) 書き込みたいファイルとコア電子がカスタムのfsblファイルを選択すると、書き込みができる。書 き込む時、JTAGブートモードじゃないと、警告が出る。QSPIを書き込むときにJTAGブートモードを設定する ことをお勧めする。



低価格、	高品質が不可能?
日昇テクノロジ	ーなら可能にする

Program Configuration Memory Device	X
Select a configuration file and set programming options.	•
Memory Device: @w25q128-qspi-x4-single -	-
Configuration file: /ax7015/demo/vdma_to_qspi_sd/vdma_to_qspi_sd.sdk/fsbl/bootimage/BOOT.bin	-
Zynq FSBL: F:/ax7015/zynq_fsbl_flash.elf	-
Program Operations	
Address Range: Configuration File Only	
✓ <u>E</u> rase	
Blank Check	
✓ Program	
✓ Verify	
OK         Cancel         Apply	

13.7 バッチファイルを使用して QSPI をすばやく書き込む

1) 新しいprogram\_qspi,txtテキストファイルを作成し、拡張名はbatに変更、コンテンツは次のように入力する。Set XIL\_CSE\_ZYNQ\_DISPLAY\_UBOOT\_MESSAGES = 1はプログラミングプロセス中にuboot印刷情報 を設定する。C:¥Xilinx¥SDK¥2017.4¥bin¥program\_flashはツールパス、インストールパスに従って変更する。 -fは書き込むファイル、-fsblは書き込まれるfsblweファイル(コア電子特定のファイル)、-blank\_check -verifyはチェックオプションである。

```
set XIL_OSE_ZYNQ_DISPLAY_UBOOT_MESSAGES=1
call C:\Xilinx\SDK\2017.4\bin\program_flash -f BOOT.bin -fsbl
zynq_fsbl_flash.elf -offset 0 -flash_type qspi_single -
blank_check -verify
pause
```

2) 書き込むBOOT.bin、fsbl、batファイルをまとめる



名称	修改日期	类型	大小
BOOT.BIN	2018/3/27 20:25	BIN 文件	14,879 KB
🚳 program_qspi.bat	2018/3/22 11:04	Windows 批处理	1 KB
zynq_fsbl_flash.elf	2018/1/5 22:04	ELF 文件	181 KB

3) JTAGケーブルを差し込んで電源を入れ、batファイルをダブルクリックしてflashを書き込める。







#### 第十四章 仮想マシンと Ubuntu システムをインストールする

この後のマニュアルには、組み込みLinux開発が含まれ、通常、u-bootまたはLinux-kernelをコンパイル するにはLinuxオペレーティングシステムホストが必要である。Windowsオペレーティングシステムに仮想 マシンをインストールしてから、仮想マシンにLinuxオペレーティングシステムをインストールするのは一 番簡単な方法である。

#### 14.1 仮想マシンソフトウェアのインストール

私立ち提供する仮想マシンのインストールソフトウェアバージョンはVIMware-worksation-full12.1.1で、 ユーザーは提供された情報でそれを見つける。VIMware-worksation-full12.1.1-3770994 exeをダブルクリ ックすると、インストールが開始される。 比較的簡単であるため、インストール手順は紹介せず、ユーザ ーは[Next]ボタンをクリックするだけでインストールできる。 完了後のインターフェースに、VMware12の シリアル番号を入力するためにライセンスを選択する必要がある。

インストール完了後、デスクトップにVMware Workstation Proのアイコンが表示される。



14.2 Ubuntu のインストール

14.2.1 システムのインス トール

仮想マシンをインストールしたら、仮想マシンにLinuxオペレーティングシステムをインストールする。 Ubuntuデスクトップオペレーティングシステムのインストールが簡単のため、ubuntuデスクトップオペレー ティングシステムを選択した。

このマニュアルでは、Ubuntu 16.04.3 LTS 64ビットオペレーティングシステムを使用する。

他のバージョンを使用している場合、予期しないエラーが発生する可能性があるから、バージョンの一貫 性を維持してください。システムをアップグレードしないでください。

Ubuntuのインストール手順は次のとおりである。

1) デスクトップのVMware Workstation Proのアイコンをダブルクリックし、VMwareの作業インター フェイスで[新しい仮想マシンの作成]アイコンをクリックする。





N	您希望使用什么类型的配置?
	<ul> <li>         ·</li></ul>
Þ	○ 自定义(高级)(⊆) 仓健带有 SCSI 控制器类型、虚拟磁盘类型 以及与旧版 VMware 产品兼容性等高级选项 的虚拟机。

下一步(N) >

取消

3) 「Installer CD Image (iso)」アイテムを選択し、ダブルクリックしてubunt CDイメージファイ ルubuntu-16.04.3-desktop-amd64.isoを見つける。

<上一歩(8)

4) 仮想マシンウィザードで、仮想マシンのフルネーム、ユーザー、およびパスワードを入力する。 ここでのフルネーム、ユーザー名、パスワードはユーザーが設定できる。

帮助





新建虚拟机向导	-	Concernant, Concernat, Concernant, Concernat, Concer	-	×
<b>简易安装信息</b> 这用于安装	桋 Ubuntu 64 位。			
个性化 Linux				
全名(F):	alinx			
用户名(U):	alinx			
密码(P):	•••••			
确认 <b>(C)</b> :	•••••			
舞助	<	上一步(B) 下一	- (N) →	取消

5) 仮想マシン名は変更できるが、十分な容量のあるディスクにインストールしてください。





6) 最大ディスクサイズを300Gに設定する。システムをインストールするため、ここのスペースを大きくする必要がある。ユーザーは、ハードディスクの空き容量に応じて適切な容量を選択できるが、300G以上にすることをお勧めする。



7)

# 株式会社日昇テクノロジー

低価格、高品質が不可能? 日昇テクノロジーなら可能にする

				and the second s	
<b>指定磁盘容里</b> 磁盘大小为多	少?				
虚拟机的硬盘作为— 小,随着您向虚拟机	-个或多个文件  中添加应用程	存储在主机的物理 序、文件和数据而	磁盘中。这些5 逐渐变大。	<b>C件最初很</b>	
最大磁盘大小(GB)(	5):	120.0 🚔			
针对 Ubuntu 64 位	的建议大小:2	0 GB			
◎ 将虚拟磁盘存储>	为单个文件 <mark>(0)</mark>				
将虚拟磁盘拆分质 拆分磁盘后,可 性能。	成多个文件(M) 以更轻松地在计	计算机之间移动虚	以机,但可能会	隆低大容里磁盘的	n
			0		
帮助		<上一步(B)	下一步(N) >	取消	
	・選択する ┪	<上一步(B)	下一步(N) >	取消	
帮助 スタムハードウェアを 建成拟机向导	E選択する	<上一步(B)	下—步(N) >	取消	
帮助 スタムハードウェアを 建虚拟机向导 <b>已准备好创建虚拟</b> 单击"完成"创	を選択する 2 <b>れ</b> 圭虚拟机・并チ	<上一步(B) T始安装 Ubuntu	下一步(N) > 64 位 和 VMv	▶ 取消 ware Tools。	
帮助 スタムハードウェアを 建虚拟机向导 已准备好创建虚拟 単击"完成"创始 将使用下列设置创建	を選択する 2 <b>机</b> 重虚拟机・并升 虚拟机:	<上一步(B) 干始安装 Ubuntu	下一步(N) > 64 位 和 VMv	▶ 取消 ware Tools。	
朝助 スタムハードウェアを 建虚拟机向导 已准备好创建虚拟 単击"完成"创 将使用下列设置创建 名称: 位置:	E選択する E 動 重虚拟机・并引 虚拟机: Ubuntu16_ ENVirtual N	<上一步(B) 干始安装 Ubuntu _64 Jachines\Ubuntu	下一步(N) > 64 位 和 VMv	▶ 取消 ware Tools。	
朝助 スタムハードウェアを 建虚拟机向导 已准备好创建虚拟 单击"完成"创 将使用下列设置创建 名称: 位置: 版本:	E選択する AU 重虚拟机・并引 虚拟机: Ubuntu16_ F.\Virtual M Workstatio	<上一步(B) 干始安装 Ubuntu _64 fachines\Ubuntu n 12.0	下一步(N) > 64 位 和 VMv	▶ 取消 ware Tools。	
<ul> <li>帮助</li> <li>スタムハードウェアを</li> <li>建虚拟机向导</li> <li>已准备好创建虚拟</li> <li>単击"完成"创建</li> <li>将使用下列设置创建</li> <li>名称:</li> <li>位置:</li> <li>版本:</li> <li>操作系统:</li> </ul>	E選択する AU 重虚拟机・并引 虚拟机: Ubuntu16_ F、Virtual M Workstatio Ubuntu 64	<上一步(B) 干始安装 Ubuntu _64 fachines\Ubuntu n 12.0 位	下一步(N) > 64 位 和 VMv	▶ 取消 ware Tools。	
<ul> <li>         帮助         <ul> <li>スタムハードウェアを</li> <li>建虚拟机向导</li> </ul> </li> <li>             已准备好创建虚拟             <li>单击"完成"创握</li> </li></ul> <li>移使用下列设置创建</li> <ul> <li>名称:</li> <li>位置:</li> <li>版本:</li> <li>操作系統:</li> <li>硬盘:</li> <li>内存:</li> </ul>	E選択する AU 重虚拟机・并引 虚拟机: Ubuntu16_ Fs\Virtual M Workstatio Ubuntu 64 120 GB, 拆 1024 MB	<上一步(B) 干始安装 Ubuntu _64 Machines\Ubuntu n 12.0 位 分	下一步(N)> 64 位 和 VMv	▶ 取消 ware Tools。	
朝助 スタムハードウェアを 建 虚拟机向导 已准备好创建虚拟 単击"完成"创成 将使用下列设置创建 名称: 位置: 版本: 操作系统: 研究: 研究: 一項 一項 留書: 一方 一句	E選択する RM 重虚拟机,并升 虚拟机: Ubuntu16_ F、Virtual M Workstatio Ubuntu 64 120 GB,拆 1024 MB NAT	<上一步(B) 干始安装 Ubuntu 64 tachines\Ubuntu n 12.0 位 分	下一步(N)> 64 位 和 VMv 1_16	▶ 取消 ware Tools。	
<ul> <li>         帮助         <ul> <li>スタムハードウェアを</li> <li>建虚拟机向导</li> </ul> </li> <li> <b>已准备好创建虚拟</b></li></ul>	E選択する AM 重虚拟机・并引 虚拟机: Ubuntu16_ F\Virtual M Workstatio Ubuntu 64 120 GB, 拆 1024 MB NAT CD/DVD, U	<上一歩(B) 干始安装 Ubuntu 64 Machines\Ubuntu 加12.0 位 ジカ USB 控制器, 打印	下一步(N) > 64 位 和 VMv 1_16 机, 声卡	▶ 取消	
<ul> <li>         帮助         <ul> <li>スタムハードウェアを</li> <li>建虚拟机向导</li> <li><b>已准备好创建虚拟</b> 单击"完成"创握</li> <li>移使用下列设置创建</li> <li>各称:</li> <li>位置:</li> <li>版本:</li> <li>操作系統:</li> <li> </li> <li></li></ul></li></ul>	E選択する AU 重虚拟机: 重虚拟机: Ubuntu16_ Fs\Virtual M Workstatio Ubuntu 64 120 GB, 拆 1024 MB NAT CD/DVD, U	<上一步(B) 干始安装 Ubuntu 64 Machines\Ubuntu n 12.0 位 分	下一步(N) > 64 位 和 VMv u_16 机, 声卡	ware Tools。	
帮助 スタムハードウェアを 建虚拟机向导 已准备好创建虚拟 単击"完成"创想 将使用下列设置创建 名称: 位置: 版本: 操作系統: 頑盤: 内存: 网络适配器: 其他设备: 」 自定义硬件(C 図 创建后开启此虚抑	E選択する ま虚拟机: 建虚拟机: Ubuntu16_ Fs\Virtual M Workstatio Ubuntu 64 120 GB, 拆 1024 MB NAT CD/DVD, U	<上一歩(B) 干始安装 Ubuntu _64 Machines\Ubuntu n 12.0 位 分 SB 控制器, 打印	下一步(N) > 64 位 和 VMv u_16 机, 声卡	ware Tools。	
郡助     スタムハードウェアを     建虚拟机向导     已准备好创建虚抑     单击"完成"创     単击"完成"创     和     で     取     名称:     位置:     版本:     操作系統:     飛盘:     内存:     网络适配器:     其他设备:        自定义硬件(C     又) 创建后开启此虚抑	E選択する ま 建 虚 拟 机 し し し し し し し し し し し し し	<上一步(B) 干始安装 Ubuntu _64 Machines\Ubuntu 加12.0 位 分 USB 控制器, 打印	下一步(N) > 64 位 和 VMv L_16 机, 声卡	vare Tools。	
朝助   スタムハードウェアを   建虚拟机向导 <b>已准备好创建虚抑</b> 単击"完成"创想   将使用下列设置创建   名称:   位置:   版本:   操作系统:   ·	E選択する AL 重虚拟机: Ubuntu16_ Fs\Virtual M Workstatio Ubuntu 64 120 GB, 拆 1024 MB NAT CD/DVD, U	<上一步(B) 干始安装 Ubuntu _64 Machines\Ubuntu n 12.0 位 分 USB 控制器, 打印	下一步(N) > 64 位 和 VMv 1_16 机, 声卡	vare Tools。	



8) 変更されたメモリサイズとプロセッサコア、ネットワークアダプタオプション、ネットワーク接 続に基づいてブリッジモードを選択できる。

设备	接要	设备状态
■内存 □处理器 ●新 CD/DVD (SA	2 GB 2 T正在使用文件 F:\ax7015\ubuntu-16	□ 已连接(C) 図 启动时连接(O)
國際議員	NAT	网络连接
○ USB 控制器 ● 声卡 ● 打印机	存在 自动检测 存在	<ul> <li></li></ul>
	自动检测	<ul> <li>NAT 模式(N):用于共享主机的 IP 地址</li> <li>仅主机模式(H):与主机共享的专用网络</li> <li>自定义(U):特定虚拟网络</li> </ul>
		VMnet0 (自动桥接)
	1010/D	● LAN 区投(L): LAN 区投(S) 高級(V)
	警添加(A) 移除(R)	

9) 「完了」をクリックして、インストールが開始する。

新建處拟机向导 已准备好创建虚 单击"完成"@	<b>以机</b> 刚建虚拟机,并开始安装 Ubuntu 64 位 和 VMware Tools。	X
谷使用下列设置回知 名称: 位置: 版本: 操作系统:	Ubuntu16_64 F: Virtual Machines\Ubuntu_16 Workstation 12.0 Ubuntu 64 位	
硬盘: 内存: 网络适配器: 其他设备:	120 GB, 拆分 2048 MB 桥接模式(自动) 2 个 CPU 核心, CD/DVD, USB 控制器, 打印机, 声卡	
自定义硬件 ▼ 创建后开启此虚	(C) 拟机(P)	
	< 上一步(B) 完成 耳	则消

10) 時間かかるから、しばらく待ってください。





11) インストール完了後、システムに入る。

位主页	×E	Ubunt	u16_64	4 ×				Λ	_							
ubuntu												0 ti	En	<b>4</b> )))	12:30	ам 🔱
T.A.																
÷.																
•																
*																
		alinx														
		Passv	vord		-				۱.							
	, c	Guest	Sess	ion					,							
*																
ut	bùn	tų®	16.0	04 LT	s											
▶ 单击虚 可发送	討以屏幕 注按键	简	易安装	正在安装	t Ubun	tu 64 (	ž o								帮助	



14.2.2 ソフトウェアソースサーバーを変更する

1) ソフトウェアを便利にインストールするには、ソフトウェアソースを設定し、システム設定をクリックする。



2) Software&UpdatesにOtherを選択する。

All Settings					Q,		
Appearance	inghtness & Lock	Linguage Uncort	Online Accounts	Security A T Privacy	Part Entry		
ardware Bluetooth	Color	Displays	Keyboard	Mouse & Touchpad	Network	Power	
Printers	Sound			O Software &	Updates Other Softwar in the Intern-	re Updates Au et	uthentication Additional Drivers Developer Options
Backups	Details	Software &	O Time & Date	Community m Proprietary dr Software rest	aintained free rivers for devic ricted by copy	e and open-source ces (restricted) right or legal issu	ee software (universe) ues (multiverse)
		opostes		Download from:	Main server Server for U Other	nited States	
			1	To install from a	CD-ROM or D	VD, insert the me	relium into the drive.



3) [select best sever]をクリックして、最速のサーバーが出て、次に、choose serverを選択する。 これらの操作は、仮想マシンがインターネットに接続できるという事実に基づいている。

Ubuntu Software Other S	oftware Updates Authentication Ad ②	ditional Drivers Developer Options
<ul> <li>Canonical-supported</li> <li>Community-maintain</li> <li>Proprietary drivers f</li> <li>Software restricted</li> <li>Source code</li> <li>Download from: Other</li> <li>Installable from CD-ROF</li> </ul>	<ul> <li>Canada</li> <li>Chile</li> <li>China ftp.sjtu.edu.cn mirror.neu.edu.cn</li> <li>mirrors.aliyun.com</li> <li>mirrors.cqu.edu.cn mirrors.skyshe.cn mirrors.sohu.com</li> <li>mirrors.tuna.tsinghua.edu.cn</li> </ul>	Select Best Server
To install from a CD-RO	Protocol: http	Cancel Choose Server Reve

4) パスワードを入力して、ソフトウェアソースの変更を完了する

Canonica Commun Proprieta	e from the inte al supported free hity-maintained f ary drivers for de	<ul> <li>Chile</li> <li>China</li> <li>ftp.sjtu.edu.cn</li> <li>mirror.neu.edu.cn</li> </ul>	Select Best S
	Authenticate	mirrore aliano com	
	To change	software repository settings	you need to
Dowr	authentica	ite.	you need to
Dowr Install	An applicatio privileges. Au	n is attempting to perform an action th thentication is required to perform thi	nat requires is action.
Dowr Install To in	An applicatio privileges. Au Password:	n is attempting to perform an action th thentication is required to perform thi	nat requires is action.



#### 14.2.3 bash をデフォルトの sh に設定する

1) Ctrl + Alt + Tでターミナルを開く。

		ix@ubuntu:-				
	To run a co See "man su	mmand as administra ido_root" for detail	ator (user "root"). ls.	use "sudo «comar	nd>"+	
	elinxgubunt	101-5 🗌				
Configuri	ıg dashコマン	ッドを入力し、	、NOを選択し	、Enterを押	して確認す	-3
Configurii sudo dpł	ng dashコマン :g-reconfigu	ッドを入力し、 re dash	、NOを選択し	、Enterを押	して確認す	-3
Configurii sudodpl	ng dashコマン g-reconfigui Inx@ubuntu:	ンドを入力し、 re dash	、NOを選択し	、Enterを押	して確認す	- 3
Configurin sudodph Configurin sudodph ackage c	ng dashコマン :g-reconfigui inx@ubuntu: onfiguratio	ンドを入力し、 re dash n	、NOを選択し	、Enterを押	して確認す	-3
Configurin sudodpl sudodpl ackage co	ng dashコマン g-reconfigur inx@ubuntu: onfiguratio	ンドを入力し、 re dash ~ n	、NOを選択し	、Enterを押	して確認す	-3
Configurii sudo dpl constant sudo dpl ackage co	ng dashコマン g-reconfigui inx@ubuntu: onfiguratio	ンドを入力し、 re dash ~	、NOを選択し	、Enterを押	して確認す	- 3
Configurin sudo dpl e o al ackage co	ng dashコマン g-reconfigur inx@ubuntu: onfiguratio	ンドを入力し、 re dash ~ n	、NOを選択し	、Enterを押	して確認す	-3
Configuri sudo dpl configuri sudo dpl ackage co	ng dash⊐マ: g-reconfigur inx@ubuntu: onfiguration	ンドを入力し、 re dash <sup>~</sup>	、NOを選択し	、Enterを押	して確認す	- 3
Configurin sudo dpl configurin sudo dpl ackage co	ng dash⊐マン g-reconfigur inx@ubuntu: onfiguration	ンドを入力し、 re dash	、NOを選択し Configuring	、Enterを押	して確認す	- 3
Configurin sudo dpl o al ackage co The sys	ng dash⊐マン g-reconfigur inx@ubuntu: onfiguratio	ンドを入力し、 re dash ~ n ts the defa	、NOを選択し Configuring	、Enterを押 dash	して確認す	nell scripts.
Configurin sudo dpl configurin sudo dpl ackage control The system Using of	ng dash⊐マ: g-reconfigur inx@ubuntu: onfiguration tem she)t lash as the	ンドを入力し、 re dash ~ n ts the defa system she	、NOを選択し Configuring oult command	、Enterを押 dash	して確認す ter for sh	nell scripts.
Configurin sudo dpl configurin sudo dpl ackage control the system Using of perform	ng dash⊐マン g-reconfigur inx@ubuntu: onfiguration stem shell lash as the ances It do	ンドを入力し、 re dash ~ n ts the defa system she oes not alt	、NOを選択し Configuring ault command ault command ault imp ault imp	、Enterを押 dash	して確認す ter for sh system's c	nell scripts.
Configurin sudo dpl sudo dpl ackage co The sys Using of perform Use da	ng dash⊐マ: g-reconfigur inx@ubuntu: onfiguratio stem shell lash as the nance. It do in as the do	ンドを入力し、 re dash ~ n ts the defa system she oes not alt efault syst	、NOを選択し Configuring Pult command ell will imp er the shel er shell ()	、Enterを押 dash	して確認す ter for sh system's c ed to inte	nell scripts.
Configurin sudo dpl sudo dpl ackage co The sys Using of perfort Use dat	ng dash⊐マ: g-reconfigur inx@ubuntu: onfiguration item she)tu dash as the nance. It du ih as the du	ンドを入力し、 re dash ~ n ts the defa system she oes not alt efault syst	、NOを選択し Configuring oult command ell will imp er the shel er shell (/	、Enterを押 dash interpret prove the s l presente 'bin/sh)?	して確認す ter for sh system's c ed to inte	nell scripts.
Configurin sudo dpl configurin sudo dpl ackage control the system Using of perform Use dat	ng dash⊐マン g-reconfigur inx@ubuntu: onfiguration stem shell lash as the hance. It do ih as the do	ンドを入力し、 re dash ~ n ts the defa system she oes not alt efault syst <yes></yes>	、NOを選択し Configuring Pult command ell will imp eer the shel eem shell ()	、Enterを押 dash interpret orove the s i presente bin/sh)?	して確認す ter for sh system's c ed to inte	nell scripts.

#### 14.2.4 画面ロック時間を設定する

Ubuntuシステムに大きなファイルをコピーできるように、画面ロックをキャンセルする。



acocomgs brightness a	Lock	
irn screen off when inactiv	e for: Never 🔹	
ock		
Lock screen after:	Screen turns off 🔻	
	vhen waking from suspend	
🖉 Require my password v		
Require my password v		
🛛 Require my password v		
Require my password v		_

#### 14.3.1 仮想マシンには仮想化サポートが必要である。

1) Ubuntuをインストールし、次のエラーメッセージボックスが表示された場合、ユーザーはコンピューターを再起動し、BIOSに入ってセットアップする必要がある。

コンピューターを再起動した後、BIOSに入り、Intel仮想化を見つけて、[開く]をクリックする。 マザーボードによって名前が異なる場合がある。

Intel 温度监控功能	开启
开启处理器核心	ATT
限制 CPUID 最大位	关闭
执行禁止位	开启
Intel 論與化人士	デル目
硬件预取	开启
相邻高速缓存线预取	开启
启动性能模式	JE Turbo



#### 第十五章 Ubuntu で Linux バージョンの Vivado ソフトウェアをインストールする

WindowsでのVivadoソフトウェアはほとんどの問題を解決できるが、場合によってはLinuxバージョンの vivado、特にSDKを使用する必要があり、多くのアプリケーションをクロスコンパイルできる。

#### 15.1 Linux バージョンの Vivado をインストールする

1) インストールファイルを仮想マシンのubuntuにコピーし、ファイルを抽出する。

-	
han san	
Open With Archive Manager	
 Open With >	
Си <u>т</u> <u>С</u> ору	
Move To Copy To Ma <u>k</u> e Link Rename	
Mo <u>v</u> e to Trash	
Extract Here	
Email	

2) ターミナルを使用して、解凍されたファイルに入る

alinx@alinx: ~/Downloads/Xilinx_Vivado_SDK_2017.4_1216_1
alinx@alinx:~/Downloads/Xilinx_Vivado_SDK_2017.4_1216_1\$

3) コマンドを実行する。



sudo chmod +x xsetup

alinx@alinx: ~/Downloads/Xilinx\_Vivado\_SDK\_2017.4\_1216\_1
alinx@alinx:~/Downloads/Xilinx\_Vivado\_SDK\_2017.4\_1216\_1\$ sudo chmod +x xsetup
sudo: unable to resolve host alinx
[sudo] password for alinx:
alinx@alinx:~/Downloads/Xilinx\_Vivado\_SDK\_2017.4\_1216\_1\$

4) コマンドを実行し、インストールを開始する。注意、ここはsudoインストールである。

#### sudo ./xsetup

 5) 以下のようなウィンドウが表示されたら、Ignoreをクリックする。
 Checking For Latest Version

 Connecting to www.xilinx.com
 Connect To Internet
 Connect to the internet to check if there is a newer version of Xilinx Design Tools available. Please verify connectivity and/or update provise tings. Note: Clicking on Ignore button will let you continue with this installation.
 Change Proxy Settings Ignore

 6) インストールプロセスでは、ウイルス対策ソフトウェアをオフにする必要がある



低価格、高品質が不可能? 日昇テクノロジーなら可能にする








Select Destination Directory

### 10) インストールパスはデフォルトパスを使用する。

stallation Options	Select shortcut and file association options	
/ont Alliny	Viliov Design Tools	
estallation location(c)		
istallation location(s)	Create desktop shortcuts	
opt/Xilinx/Vivado/2017.4		
opt/Xilinx/SDK/2017.4		
isk Space Required		
Download Size: NA		
Disk Space Required: 24.61 GB		
bisk space Available. 05.1 Gb		
yright © 1985-2018 Xiřnx, Inc. All rights reserved.	- Back Nex	t > Cancel
yright © 1985-2018 Xiinx, Inc. All rights reserved. Copy licenseをクリックしてli	<mark>⊂Back Nex</mark> ファイルをインストールする。	t > Cancel
yright e 1986-2018 Xiinx, Inc. All rights reserved. Copy licenseをクリックしてli CO Vivado License Manager 2017.2	<mark>&lt; Back Nex</mark> ファイルをインストールする。	t > Cancel
yright © 1995-2018 Xiinx, Inc. All rights reserved. Copy licenseをクリックしてに Copy Vivado License Manager 2017/2 Help	<mark>≺Back Nex</mark> ファイルをインストールする。	t > Cancel
might © 1986-2018 Xiinx, Inc. All rights reserved. Copy licenseをクリックしてIi B Vivado License Manager 2017.2 Help	<mark>&lt; Back Nex</mark> ファイルをインストールする。	t > Cancel
wight © 1985-2018 Xiinx, Inc. All rights reserved. Copy licenseをクリックしてli CO Vivado License Manager 2017.2 Help	Rack Nex アイルをインストールする。 Manager	t > Cancel
right © 1985-2018 Xinx, Inc. All rights reserved. Copy licenseをクリックしてに Copy License Manager 2017.2 Help COMMAND COMMAND License et License Set Provy	<ul> <li>Back Nex</li> <li>ファイルをインストールする。</li> <li>Manager</li> <li>Select License File</li> </ul>	t > Cancel
right © 1995-2018 Xinx, Inc. All rights reserved. Copy licenseをクリックしてに OVivado License Manager 2017.2 Help VIVADOCO License S Set Prowy Obtain Unense Licentificate B Certificate B	<ul> <li>Back Nex</li> <li>アイルをインストールする。</li> <li>Manager</li> <li>Manager</li> <li>Select License File</li> <li>Vicense's button to Look In: Downloads</li> </ul>	
right © 1985-2018 Xinx, Inc. All rights reserved. Copy licenseをクリックしてに Copy Licenseをクリックしてに UNACODE Conservation Set Proxy Datain Unionse Cool Unio	Manager Manager Ed Licenses y License' button te nic applications audi directory:	
rright © 1995-2018 Xinx, Inc. All rights reserved. Copy licenseをクリックしてIi Copy LicenseをクリックしてIi Deen Copy LicenseをクリックしてIi License Set Proxy Chain Usings Set Proxy Chain Usings Click the 'C Set Proxy Chain Usings Click the 'C Set Proxy Const License Search Paths View License Status Borrow/Restore License Seat	<ul> <li>Back Nex</li> <li>7アイルをインストールする。</li> <li>Manager</li> <li>Manager</li> <li>Genese button te nix applications aut directory.</li> <li>Select License File</li> <li>Ucenses button te nix applications aut directory.</li> </ul>	
right © 1985-2018 Xiinx, Inc. All rights reserved. Copy licenseをクリックしてIi Copy licenseをクリックしてIi Copy License Manager 2017.2 Help Contain Unicense Set Proxy Contain Unicense Set Proxy Contain Unicense Contain U	Anager  Manager  Manager  Ucenses  y Ucenses' button tc nx applications auto directory.  se  se  y Ucense' button tc nx applications auto directory.  se  y Ucense' button tc nx applications auto directory.  se  se  y Ucense' button tc invited of the sec  y Uc	
right © 1985-2018 Xinx, Inc. All rights reserved. Copy licenseをクリックしてに Copy licenseをクリックしてに Copy License をクリックしてに Copy License をクリックしてに Copy License をクリックしてに Copy License をのいった。 Copy License Search Patrix View License Status BorrowRestore License Search Return License to Minx lew System Information View Host Information	Anager  Manager  Manager	
right © 1996-2018 Xiinx, Inc. All rights reserved. Copy LicenseをクリックしてCI Copy LicenseをクリックしてCI Deen Copy LicenseをクリックしてCI License Set Proxy Copy License Search Pather Set Proxy Copy License Search Pather New License to Xinx Borrow/Restore License Search Pather License to Xinx Borrow/Restore License Search Pather License to Xinx Power License to Xinx	<ul> <li>Back Nex</li> <li>Prdルをdンストールする。</li> <li>Manager</li> <li>Manager</li> <li>Select License File</li> <li>Look Jr: Downloads</li> <li>Look Jr: Downloads</li> <li>Look Jr: Downloads</li> <li>Look Jr: Downloads</li> <li>Suitory Marcho SDK 2017.2_0616_1</li> <li>Wilnx_ise_vivado.lic</li> </ul>	
et License Set Prosy Copy license & danager 2017.2 Help Et License Set Prosy Cobtain Information Et License Set Prosy Cobtain Information Et License Set Prosy Cobtain Information Et License Set Prosy Contract Information Et License Set Prosy Set Prosy Contract Information Et License Set Prosy Set Prosy Set Prosy Contract Information Et License Set Prosy Set Prosy Contract Information Et License Set Prosy Set Prosy Contract Information Et License Set Prosy Set Pro	ABack Nex      Anager      Manager      Munager      Select License File      Ucenses      y Licenses      y License	
wight @ 1985-2018 Xiinx, Inc. All rights reserved. Copy licenseをクリックしてに Copy licenseをクリックしてに Copy licenseをクリックしてに Copy licenseをクリックしてに Copy licenseをクリックしてに Copy licenseをのいった。 Copy licenseをのいった。 Copy licenseをのいった。 Copy licenseをのいった。 Copy license Copy licen	Anager  Manager  Manager  Manager  Manager  Manager  Manager  Manager  Manager  Manager  File Name: pilmx_ise_vivado.lic	
eright @ 1986-2018 Xinx, Inc. All rights reserved. Copy License & Dy yo D L Clin Copy License & Dy yo D L Clin Leen Leen Copy License & Dy yo D L Clin Leen Manage Locase Search Patric Set Prosy Chain Usones Load License & Search Patric Manage Locase Search Patric M	Anager  Manager  Manager  Manager  Manager  Manager  He Mana	



#### 15.2 許可設定

コマンドを実行して実行許可を追加する。

sudo	chmod	777	-R	/opt/Xilinx/
sudo	chmod	777	-R	~/.Xilinx/

#### 15.3 ダウンローダードライバーをインストールする

次のコマンドを実行して、ダウンローダードライバーをインストールする

cd /opt/Xilinx/Vivado/2017.4/data/xicom/cable\_drivers/lin64/install\_script/install\_drivers/ sudo ./install\_drivers



15.4 Vivado をテストする

1) 以下のコマンドを実行して、Vivadoを起動する

source /opt/Xilinx/Vivado/2017.4/settings64.sh vivado &





2) ダウンローダーを仮想マシンに接続する



电源(P)							2	
可移动设备(D) ▶		CD/DVD	(SATA)				-	
暂停(U) Ctrl+Shitt+P	~	<ul> <li>✓ 网络适配器</li> <li>&gt; 打印机</li> <li>声卡</li> <li>&gt;</li> </ul>						
发送 Ctrl+Alt+Del(E) 抓取輸入内容(T) Ctrl+G								
中EE(N)	T	Future D	)evices (	Digilent U	JSB Device			连接(断开与 主机 的连接)(C)
捕获屏幕 (C) Ctrl+Alt+PrtScn	Г			- VI	840 201	1.4		更改图标(I)
管理(M)			Eile	Flow	Iools	Window	~	在状态栏中显示(S)
重新安装 VMware Tools(T)						~	1	
设置(S) Ctrl+D				VI۱	ΆĽ	O.		
					HLx Edi	tions		

3) 開発ボードとダウンローダーを接続し、Open Hardware Managerでデストすると、通常の状態でチップが見つかり、Vivadoおよびダウンローダードライバーが正常にインストールされていることがわかる。

<u>File Edit T</u> ools <u>W</u> indow Layou	ut View Help Q- Quick Access
	🖄 🖉 😹 Dashboard 🗸
HARDWARE MANAGER - localhost/xilinx	_tcf/Digilent/210249854629
1 There are no debug cores. Program	device Refresh device
Hardware ?	_ 0 & X
$Q \mid \underbrace{\star} \mid \diamondsuit \mid \varnothing \mid \Vdash \mid \gg \mid \blacksquare \mid$	0
Name	Status
localhost (1)	Connected
✓ ■ vilinx_tcf/Digilent/2102498	Open
@ arm_dap_0 (0)	N/A
✓ @ xc7z035_1 (1)	Not program
XADC (System Monitor)	
<	
Hardware Target Properti	
	»
General Reportion	
deneral properties	

15.5 よくある問題

#### 15.5.1 Linux ダウンローダーのダウンロード時にプロンプトが表示される

1) ハードウェアのテスト時に、ダウンローダーが見つかるが、エラーが表示される。





2) 一部のマザーボードでは、USB互換性を設定し、仮想マシンのUbuntuをオフにし、USB互換性をUSB3.0 に設定して、もう一度試してください。ダウンローダー使用できない場合は、Windowsバージョンでしかダ ウンロードできない。

设备	摘要	JEE 18
画内存	2 GB	USB 兼容性(C): USB 3.0 ▼
■ 处理器 ■ 硬盘(SCSI)	2 120 GB	☑ 自动连接新的 USB 设备(U)
CD/DVD (SATA)	自动检测	□ 显示所有 USB 編入设备(5) ▼ 与走切却共宣转夺恐各(8)
回网络适配器	桥接模式(自动)	
<ul> <li>USB 控制器</li> <li>加声卡</li> </ul>	存在	Linux 内核 3.2 或更新版本。
副打印机	存在	
	自动检测	



15.5.2 ZYNQ に合うクロスコンパイラ

Vivadoをインストール時にSDKをインストールしたため、SDKにはクロスコンパイラarm-linux gnueabihf-gccが含まれている。

source /opt/Xilinx/SDK/2017.4/settings64.sh arm-linux-gnueabihf-gcc -v

@ @ alinx@ubuntu:~ alinx@ubuntu:~\$ source /opt/Xilinx/SDK/2017.4/settings64.sh alinx@ubuntu:~\$ arm-linux-gnueabihf-gcc -v Using built-in spees. COLLECT\_LTO\_WRAPPER=/opt/Xilinx/SDK/2017.4/gnu/aarch32/lin/gcc-arm-linux-gnueabi /bin/../libexec/gcc/arm-linux-gnueabihf/6.2.1/lto-wrapper Target: arm-linux-gnueabihf Configured with: /proj/esdt\_sdk/gnu\_abe/ABE/builds/lin/arm/arm-linux/snapshots// gcc-linaro-snapshot-6.2-2016.11/configure SHELL=/bin/Sh\_=-with-bugurl=https://bu gs.linaro.org --with-mpc=/proj/esdt\_sdk/gnu\_abe/ABE/builds/abe\_check/builds/dest dir/x86\_64-unknown-linux-gnu --with-mpfr=/proj/esdt\_sdk/gnu\_abe/ABE/builds/abe\_check/builds/dest dir/x86\_64-unknown-linux-gnu --with-mpfr=/proj/esdt\_sdk/gnu\_abe/ABE/builds/abe\_check/builds/destdir/x86\_64-unknown-linux-gnu --with-gnu=-proj/esdt\_sdk/gnu\_abe/A BE/builds/abe\_check/builds/destdir/x86\_64-unknown-linux-gnu --with-gnu=-or-with-gnu-as --with -gnu-1d --disable-libstdcxx-pch --disable-libmudflap --with-cloog=no --with-ppl= no --with-isl=no --disable-nls --enable-copy --disable-libstdcxx-pch --enable-clo cale=gnu --enable.libstdcxx-debug --enable-long -long --enable-gnu-indirect-funct ion --disable-multilib --with-tune=cortex-a9 --with-arch=armv7-a --with-fpu=vfpv 3-d16 --with-float=hard --with-mode=arm --disable-multilib --enable-multiarch CF LAGS=-02 CXXFLAGS=-02 --disable-silent-rules --enable-libguadmath --enable-libg2 c --enable-symvers=gnu --enable-libstdcxx-pch --enable-libguadmath --enable-libg2 c --enable-checking=release, --enable-cheaders=c\_global --enable-botstr ap --enable-checking=release, --enable-cheaders=c\_global --enable-libmudfl ap --enable-checking=release, --enable-shared --enable-lib --disable-libmudfl ap --enable-checking=release, --enable-cheaders=c\_global --enable-threads=posix --dis able-multilib --enable-clocale=generic --enable-shared --enable-lib --disable-botstr ap --with-linker-hash-style=gnu --with-ppl=no --with-cloog=no --without-isl --en



1

第十六章 Petal inux ツールのインストール

#### 16.1 Petalinux の概要

Petalinuxは特別なLinuxカーネルではなく、開発環境設定用のツールセットであり、uboot、カーネル、 及びルートファイルシステムの設定作業負荷を軽減できる。Vivadoのエクスポートハードウェア情報から関 連ソフトウェアを自動的に設定できる。

特に注意が必要なのは、Petal inuxがシステムバージョンと設定に厳しいこと。その他のバージョンでシ ステムを操作する場合、問題があったらご自身で解決してください。

#### 16.2 インストールに必要なライブラリ

1) 以下のコマンドを実行して、ライブラリをインストールする

sudo apt-get install tofrodos gawk xvfb git libncurses5-dev tftpd zlib1g-dev zlib1g-dev:i386

libssl-dev flex bison chrpath socat autoconf libtool texinfo gcc-multilib \ libsdl1.2-dev libglib2.0-dev screen pax

🔋 🗐 🗊 alinx@alinx-fpga: ~ alinx@alinx-fpga:~\$ sudo apt-get install tofrodos gawk xvfb git libncurses5-dev tftpd zlib1g-dev zlib1g-dev:i386 \ libssl-dev flex bison chrpath socat autoconf libtool texinfo q cc-multilib \ libsdl1.2-dev libglib2.0-dev screen pax sudo: unable to resolve host alinx-fpga [sudo] password for alinx: 1Sorry, try again. [sudo] password for alinx: Sorry, try again. [sudo] password for alinx: Reading package lists... Done Building dependency tree Reading state information... Done autoconf is already the newest version (2.69-9). bison is already the newest version (2:3.0.4.dfsg-1). flex is already the newest version (2.6.0-11). gawk is already the newest version (1:4.1.3+dfsg-0.1). gcc-multilib is already the newest version (4:5.3.1-1ubuntu1). libncurses5-devits already the newest version (6.0+20160213-1ubuntu1). libsdl1.2-dev is already the newest version (1.2.15+dfsg1-3). libtool is already the newest version (2.4.6-0.1). pax is already the newest version (1:20151013-1).

2) Tftp serverを設定する。TFTPから起動する必要がない場合、この手順は選択せきる。

sudo -s apt-get install tftpd-hpa chmod a+w /var/lib/tftpboot/

reboot



#### 16.3 Petalinux をインストールする

1) インストールのためにコマンドを実行、ユーザー名は「your\_user\_name」、たとえば、下の画像のalinx。

sudo	-S	
mkdir	-p /opt/pkg/petalin	nux
chown	<your_user_name></your_user_name>	/opt/pkg/
chgrp	<your_user_name></your_user_name>	/opt/pkg/
chgrp	<your_user_name></your_user_name>	/opt/pkg/petalinux/
chown	<your_user_name></your_user_name>	/opt/pkg/petalinux/

```
🧐 🗐 🔹 alinx@alinx-fpga: ~/Downloads
```

```
alinx@alinx-fpga:~/Downloads$ sudo -s
sudo: unable to resolve host alinx-fpga
[sudo] password for alinx:
root@alinx-fpga:~/Downloads# mkdir -p /opt/pkg/petalinux
root@alinx-fpga:~/Downloads# chown alinx /opt/pkg/
root@alinx-fpga:~/Downloads# chown alinx /opt/pkg/petalinux/
root@alinx-fpga:~/Downloads# chorp alinx /opt/pkg/petalinux/
root@alinx-fpga:~/Downloads# exit
exit
alinx@alinx-fpga:~/Downloads$
```

2) 実行許可をインストールファイルに追加する。もちろんpetalinux-v2017.4-final-installer.runこのファイルは最初にシステムにコピーする必要がある

sudo chmod +x petalinux-v2017.4-final-installer.run

3) インストールを開始する

./petalinux-v2017.4-final-installer.run /opt/pkg/petalinux/

😣 🗇 回 🛛 alinx@alinx-fpga: ~/Downloads

alinx@alinx-fpga:~/Downloads\$ ./petalinux-v2017.4-final-installer.run /opt/pkg/p etalinux/ INFO: Checking installer checksum...

4) Enterキーを押してプロトコルコンテンツを確認できる



nux/

INFO: Checking installer checksum... INFO: Extracting PetaLinux installer...

LICENSE AGREEMENTS

PetaLinux SDK contains software from a number of sources. Please review the following licenses and indicate your acceptance of each to continue.

You do not have to accept the licenses, however if you do not then you may not use PetaLinux SDK.

Jse PgUp/PgDn to navigate the license viewer, and press 'q' to close

Press Enter to display the license agreements

5) qを押してプロトコルコンテンツを終了する

XILINX, INC. END USER LICENSE AGREEMENT FOR PETALINUX TOOLS

CAREFULLY READ THIS END USER LICENSE AGREEMENT FOR PETALINUX TOOLS ("AGREEMENT") . BY CLICKING THE "ACCEPT" OR "AGREE" BUTTON, ENTERING <93>YES<94> OR <93>YTO ACCEPT THIS AGREEMENT, OR OTHERWISE ACCESSING, DOWNLOADING, INSTALLING OR USING THE SOFTWARE, YOU AGREE ON BEHALF OF LICENSEE TO BE BOUND BY THIS AGREEMENT.

IF LICENSEE DOES NOT AGREE TO ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT, DO NOT CLICK THE "ACCEPT" OR "AGREE" BUTTON, ENTER <93>YES<94> OR <93>Y<94>, OR ACCESS, DOWNLOAD, INSTALL OR USE THE SOFTWARE.

1. Definitions

"Bitstream" means a machine-executable, binary form of a core used to program a Xilinx Device.

"Licensee" means the individual, corporation or other legal entity who has downl oaded and installed the Software.

"User" means a specific human being who is identified by Licensee as a person wh o is authorized to use the applicable Software on behalf of Licensee. In cases /tmp/tmp.Wt4jhy0kou/./etc/license/Petalinux\_EULA.txt

6) yを押して同意する

inux/ INFO: Checking installer checksum...

INFO: Extracting PetaLinux installer...

LICENSE AGREEMENTS

PetaLinux SDK contains software from a number of sources. Please review the following licenses and indicate your acceptance of each to continue.

You do not have to accept the licenses, however if you do not then you may not use PetaLinux SDK.

Use PgUp/PgDn to navigate the license viewer, and press 'q' to close

Press Enter to display the license agreements Do you accept Xilinx End User License Agreement? [y/N] >

7) Licenseはインストールプロセス中にポップアップ表示され、qを押して終了、yを押して同意する



WebTalk Terms and Conditions

By indicating I accept this WebTalk notice, I also confirm that I have read Sect ion 13 of the terms and conditions above concerning WebTalk and have been afford ed the opportunity to read the WebTalk FAQ posted at http://www.xilinx.com/webta lk. I understand that I am able to disable WebTalk later if certain criteria de scribed in Section 13(c) apply. If they don't apply, I can disable WebTalk by u ninstalling the Software or using the Software on a machine not connected to the internet. If I fail to satisfy the applicable criteria or if I fail to take th e applicable steps to prevent such transmission of information, I agree to allow Xilinx to collect the information described in Section 13(a) for the purposes d escribed in Section 13(b).

/tmp/tmp.Wt4jhy0kpU/./etc/license/WebTalk\_notice.txt (END)

INFO: Checking installer checksum... INFO: Extracting PetaLinux installer...

LICENSE AGREEMENTS

PetaLinux SDK contains software from a number of sources. Please review the following licenses and indicate your acceptance of each to continue.

You do not have to accept the licenses, however if you do not then you may not use PetaLinux SDK.

Use PgUp/PgDn to navigate the license viewer, and press 'q' to close

Press Enter to display the license agreementsq Do you accept Xilinx End User License Agreement? [y/N] > y Do you accept Webtalk Terms and Conditions? [y/N] > y Do you accept Third Party End User License Agreement? [y/N] > y INFO: Checking installation environment requirements... INFO: Checking free disk space INFO: Checking installed tools INFO: Checking installed development libraries INFO: Checking network and other services INFO: Installing PetaLinux...



#### 第十七章 NFS サービスソフトウェアのインストール

NFS (Network FileSystem、インタネットファイルシステム)はSUN会社が開発した技術であり、1984年 に導入され、異なるマシンや異なるオペレーティングシステムでファイルを相互に共有する技術である。 NFSはもともと異なるシステム間で使用するように設計されていたため、その通信プロトコル設計はホスト オペレーティングシステムとは関係ない。

NFSサーバーとクライアントは、リモートファイルを使用する場合、mountコマンドを使用して、ローカル ファイルシステムの下のリモートNFSサーバーにファイルシステムをマウントできる。 NFSサーバーによっ て共有されるファイルまたはディレクトリは、/ etc / exportsファイルに記録されている。

組み込みLinux開発では、NFSがよく使用されるが、通常、ターゲットシステムはNFSクライアントとして 使用され、LinuxホストはNFSサーバーとして使用されている。 ターゲットシステムで、NFSを介してサーバ ーのNFS共有ディレクトリをローカルデバイスにマウントし、サーバー上のファイルを直接に実行できる。 システムドライバーモジュールとアプリケーションのデバッグには、NFSが非常に必要である。また、Linux は、リモートNFSルートからシステムを直接起動できるし、NFSルートファイルシステムもサポートできる。 また、組み込みLinuxルートファイルシステムの調整および統合にも必要がある。

#### 17.1 NFS サーバーをインストールする

1) 次のコマンドでNFS サーバーをインストールする

sudo apt-get install nfs-kernel-server



2) NFSの作業ディレクトリとして新しい作業ディレクトリを作成する。将来、このディレクトリにクロ スコンパイルされたプログラムを配置できる。開発ボードはこのディレクトリ内のファイルを簡単に共有で きる。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

⊘ Recent			
🔂 Home	÷		
🖿 Desktop	Desktop	Documents	Downloads
Documents			
Downloads	Videos	work	Examples
J Music			
O Pictures 次のコマンドを使用して	/ etc / exportsファイル	レを編集し、NFSサ <del>ー</del> ビス	パ <mark>ス</mark> をコンフィグする
○ Pictures 次のコマンドを使用して udo gedit /etc/exports	/ etc / exportsファイノ	レを編集し、NFSサービス	パスをコンフィグする
○ Pictures 次のコマンドを使用して udo gedit /etc/exports	/ etc / exportsファイノ	レを編集し、NFSサービス	パスをコンフィグする
<ul> <li>Pictures</li> <li>次のコマンドを使用して</li> <li>udo gedit /etc/exports</li> <li>e alinx@ubuntu:~</li> <li>alinx@ubuntu:~\$ sudo</li> <li>[sudo] password for al</li> </ul>	/ etc / exportsファイノ gedit /etc/exports linx:	レを編集し、NFSサービス	パスをコンフィグする
○ Pictures 次のコマンドを使用して udo gedit /etc/exports ◎ ● ● alinx@ubuntu: ~ alinx@ubuntu:~\$ sudo [sudo] password for al (gedit:60516): IBUS-W/ root!	/ etc / exportsファイノ gedit /etc/exports Linx: ARNING **: The owner	レを編集し、NFSサービス of /home/alinx/.com	パスをコンフィグする

😣 🖱 🗊 🛛 🖉	Open 👻 🖪		exports /etc		
# /etc/expo # #	rts: the ac to NFS	cess control li Clients. See	st for filesystems wh exports(5).	ich may be exported	
<pre># Example fo # /srv/home: # # Example fo</pre>	or NFSv2 and s hos	t NFSV3. tname1(rw,sync,	no_subtree_check) hos	tname2(ro,sync,no_su	ubtree_check)
# /srv/nfs4 # /srv/nfs4 #	gss /homes gss	/krb5i(rw,sync, /krb5i(rw,sync,	fsid=0,crossmnt,no_su no_subtree_check)	btree_check)	
/home/alinx	/work *(rw,	sync,no_root_sq	uash,no_subtree_check		

5) 次のコマンドを実行して、rpcbindサービスを再起動する。 NfsはRPCプログラムで、使用する前にポ ートをマップしてrpcbindで設定する必要がある。

sudo /etc/init.d/rpcbind restart

6) 次のコマンドを実行してnfsサービスを再起動する。

sudo /etc/init.d/nfs-kernel-server restart

17.2 NFS をテストする



1) 次のコマンドでNFSをマウントし、NFS作業パスを/mntディレクトリにマウントする。

mount -t nfs 127.0.0.1:/home/alinx/work /mnt

2) / mntと入力して、新しテストディレクトリtestを作成する。/home/alinx/workディレクトリにtest フォルダーが表示される。

cd /mn	nt	
mkdir t	test	

17.3 よくある問題

17.3.1 NFS マウントできない

最初に、仮想マシンと開発ボードがネットワークセグメントであるかどうかを確認する。

ifconfigコマンドを使用して、仮想マシンのIPアドレスを表示する。次の図の例は、192.168.1.55、 192.168.1.55ネットワークセグメントである。開発環境にはDHCCPサーバーがあるため、仮想マシンのIPア ドレスが自動的に割り当てられる。 このテキストでは、ネットワーク環境が異なるため、ネットワークの コンフィグ方法については説明しない。ネットワークをコンフィグできない場合は、ネットワーク管理者に 相談してください。

800	alinx@ubuntu:~
alinx@ul ens33	<pre>puntu:~\$ ifconfig link encap:Ethernet_HWaddr_00:0c:29:33:14:96 inet_addr:192.168.1.55 Bcast:192.168.1.255 Mask:255.255.255.0 ineto_addr: reso::12a1:91e3:bf6:1cc/64 Scope:Link UP_BROADCAST_RUNNING_MULTICAST_MTU:1500_Metric:1 RX_packets:33916_errors:0_dropped:0_overruns:0_frame:0 TX_packets:11041_errors:0_dropped:0_overruns:0_carrier:0 collisions:0_txqueuelen:1000 RX_bytes:9607327 (9.6_MB)_TX_bytes:1136591 (1.1_MB)</pre>
lo	Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1/128 Scope:Host UP LOOPBACK RUNNING MTU:65536 Metric:1 RX packets:11374 errors:0 dropped:0 overruns:0 frame:0 TX packets:11374 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:1362283 (1.3 MB) TX bytes:1362283 (1.3 MB)
alinx@ul alinx@ul alinx@ul alinx@ul	puntu:~\$ puntu:~\$ puntu:~\$

シリアル端末でifconfigコマンドを使用して、開発ボードのIPアドレスが表示される。次の図の例は、 192.168.1、ネットワークセグメントは192.168.1.46である。IPがない場合、または開発ボードIPに異なる ネットワークセグメントがある場合は、ネットワーク管理者に連絡してください。



root@zynq:	* ifconfig
eth0	Link encap:Ethernet HWaddr 00:0a:35:00:1e:53
	inet addr:192.168.1.46 Bcast:192.168.1.255 Mask:255.255.255.0
_	inetb addr: fe80::20a:35ff:fe00:1e53/64 Scope:Link
	UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
	RX packets:37 errors:0 dropped:0 overruns:0 frame:0
	TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
	collisions:0 txqueuelen:1000
	RX bytes:4597 (4.4 KiB) TX bytes:3782 (3.6 KiB)
	Interrupt:29 Base address:0xb000
eth1	Link encap:Ethernet HWaddr 00:0a:35:00:03:22
	UP BROADCAST MTU:1500 Metric:1
	RX packets:0 errors:0 dropped:0 overruns:0 frame:0
	TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
	collisions:0 txqueuelen:1000
	KX bytes:0 (0.0 B) IX bytes:0 (0.0 B)
10	Link encap:Local Loopback
	inet addr:127.0.0.1 Mask:255.0.0.0
	Ineto addr: :: 1/128 Scope: Host
	UP LOUPDACK KUNNING MIU:05550 Metric:1
	KA packets:8 errors:0 dropped:0 overruns:0 frame:0
	alligional transmolari
	RV bytes: 1104 (1 0 KiB) TV bytes: 1104 (1 0 KiB)
	IX bytes. 1104 (1.0 Kib) IX bytes. 1104 (1.0 Kib)
root@zvna:	~#

シリアルターミナルで仮想マシンにpingし、サンプルでpingする。 192.168.1.55 これは、仮想マシンのIPが192.168.1.155であり、これをpingしてNFSを通常にマウントできる。

root@zynq: ~ # ping 192.168.1.55
PING 192.168.1.55 (192.168.1.55) 56(84) bytes of data.
64 bytes from 192.168.1.55: icmp\_seq=1 ttl=64 time=0.862 ms
64 bytes from 192.168.1.55: icmp\_seq=2 ttl=64 time=0.489 ms
64 bytes from 192.168.1.55: icmp\_seq=3 ttl=64 time=0.779 ms
64 bytes from 192.168.1.55: icmp\_seq=4 ttl=64 time=0.504 ms
64 bytes from 192.168.1.55: icmp\_seq=5 ttl=64 time=0.574 ms
^C
--- 192.168.1.55 ping statistics --5 packets transmitted, 5 received, 0% packet loss, time 4131ms
rtt min/avg/max/mdev = 0.489/0.641/0.862/0.153 ms
root@zynq: ~ #



第十八章 Petalinux でLinux システムをカスタマイズする

実験Vivadoプロジェクトはlinux\_baseである。

前のマニュアルでは、Petalinux環境を構築したが、このマニュアルでは主にPetalinuxの使用方法を示す。 この実験は、Linuxホストがインターネットに接続できた場合のみ完了できる。

18.1 Vivado プロジェクト

Petalinuxを使用すると、組み込みLinuxシステムをカスタマイズするのに非常に便利である。Vivadoソフトウェアがハードウェア情報をエクスポートするだけで、その後、Petalinxはこの情報に従ってuboot、カーネル、ファイルシステムなどをコンフィグする。vivadoプロジェクトの確立手順は、前の実験で説明したため、ここでは説明しない。

- 1) Bitファイルのコンパイルと生成
- 2) ハードウェア情報をエクスポートする

Export hardware platform for s development tools.	oftware
✓ Include bitstream	
Export to: 🔊 <local p<="" td="" to=""><th>roject&gt; 🗸</th></local>	roject> 🗸
<b>Э</b> ок	Cancel

3) vivadoのプロジェクトディレクトリに\*.sdkディレクトリがあり、下に\*.hdfファイルがある。この ファイルには、petalinuxが使用するファイルが含まれている。



▶ 新加祥 ▶ demo ▶ linux_base ▶	linux_base.sdk >
————————————————————————————————————	
名称	修改日期
퉬 .metadata	2018/3/13 19:59
🎉 design_1_wrapper_hw_platform_0	2018/3/13 19:59
퉬 RemoteSystemsTempFiles	2018/3/13 19:59
design_1_wrapper.hdf	2018/3/13 19:55
SDK.log	2018/3/13 19:59
ne Downloads peta_prj linux_base.sdk ターミナルを開き、ディレクトリに入る。	
なぜここにpeta_prjティレクトリかないのか、自分でティ Pルでディレクトリがあるのに、自分が場合、ご自身で作成	レクトリを作成する必要がめる。 してください。
<pre> e linx@ubuntu: ~/Downloads/peta_prj alinx@ubuntu:~/Downloads/peta_prj </pre>	

3) petalinux環境変数を設定し、次のコマンドを実行する。

source /opt/pkg/petalinux/settings.sh



@@@@@ alinx@ubuntu:~/Downloads/peta\_prj
alinx@ubuntu:~/Downloads/peta\_prj\$ source /opt/pkg/petalinux/settings.sh
PetaLinux environment set to '/opt/pkg/petalinux'
WARNING: /bin/sh is not bash!
bash is PetaLinux recommended shell. Please set your default shell to bash.
INF0: Checking free disk space
INF0: Checking installed tools
INF0: Checking installed development libraries
INF0: Checking network and other services
WARNING: No tftp server found - please refer to "PetaLinux SDK Installation Guid
e" for its impact and solution
alinx@ubuntu:~/Downloads/peta\_prj\$

4) 次のコマンドを実行して、vivado環境変数を設定する。

source /opt/Xilinx/Vivado/2017.4/settings64.sh

alinx@ubuntu: ~/Downloads/peta\_prj
alinx@ubuntu: ~/Downloads/peta\_prj\$ source /opt/pkg/petalinux/settings.sh
PetaLinux environment set to '/opt/pkg/petalinux'
WARNING: /bin/sh is not bash!
bash is PetaLinux recommended shell. Please set your default shell to bash.
INF0: Checking free disk space
INF0: Checking installed tools
INF0: Checking installed development libraries
INF0: Checking network and other services
WARNING: No tftp server found - please refer to "PetaLinux SDK Installation Guid
e" for its impact and solution
alinx@ubuntu:~/Downloads/peta\_prj\$

5) 次のコマンドで、ax\_petaという名前のpetalinuxプロジェクトを作成すると、petalinuxはax\_petaという名前のプロジェクトが自動的に作成される。 - は二つで、ご注意してください。

petalinux-create --type project --template zynq --name ax\_peta





6) 次のコマンドを使用して, petal inuxディレクトリーに入る。

cd ax\_peta



7) 次のコマンドで、Petalinuxプロジェクトのハードウェア情報をコンフィグする。../linux\_base.sdk ディレクトリは、vivadoからエクスポートしたハードウェア情報である。

petalinux-config --get-hw-description ../linux base.sdk

alinx@ubuntu:~/Downloads/peta\_prj/ax\_peta\$ petalinux-config --get-hw-description ../linux\_base.sdk INFO: Getting hardware description... INFO: Rename design 1 wrapper.hdf to system.hdf [INFO] generating Kconfig for project

8) ポップアップウィンドウでpetalinuxプロジェクトをコンフィグできる。コンフィグ後に再度コンフィグしたい場合は、コマンドpetalinux-configを実行してコンフィグできる。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



9) ubootおよびLinuxカーネルのソースは、オプションLinux Components Selectionで設定できる。デフ オルトでgithubにダウンロードされる。ダウンロードするにはインターネットとLinuxホストと接続する必 要がある。 このテストはデフォルト設定のまま。



10) 周辺機器と起動モードは、オプションのSubsystem AUTO Hardware Settingsで設定できる。





11) Advanced bootable images storage Settingsオプションで、起動モードをコンフィグする。デバッ グのため、ここでは、デフォルトでSDカードから起動のまま、もしQSPI flashから起動する組み込みLinux を作成する場合は、ここでコンフィグできる。



12) 完了した後も設定を維持する。この実験は基本的にデフォルト設定である。





14) しばらくお待ちください。



🕽 🗇 🗊 🛛 alinx@ubuntu: ~/Downloads/peta\_prj/ax\_peta INFO: Getting hardware description... [INFO] generating Kconfig for project [INFO] menuconfig project /home/alinx/Downloads/peta\_prj/ax\_peta/build/misc/config/Kconfig.syshw:30:warnin g: defaults for choice values not supported /home/alinx/Downloads/peta\_prj/ax\_peta/build/misc/config/Kconfig:568:warning: co nfig symbol defined without type \*\*\* End of the configuration. \*\*\* Execute 'make' to start the build or try 'make help'. [INFO] sourcing bitbake [INFO] generating plnxtool conf [INFO] generating meta-plnx-generated layer ~/Downloads/peta\_prj/ax\_peta/build/misc/plnx-generated ~/Downloads/peta\_prj/ax\_p eta ~/Downloads/peta\_prj/ax\_peta [INFO] generating machine configuration [INFO] generating bbappends for project . This may take time ~/Downloads/peta\_prj/ax\_peta/build/misc/plnx-generated ~/Downloads/peta\_prj/ax\_p eta

#### 18.3 Linux カーネルをコンフィグする

1) 次のコマンドでカーネルをコンフィグする。コマンドを実行したら、少し待ちください。

petalinux-config -c kernel

😣 🗇 💿 alinx@ubuntu: ~/Downloads/peta_prj/ax_peta	
eta ~/Downloads/peta_prj/ax_peta [INFO] generating u-boot configuration files	
[INFO] generating kernel configuration files [INFO] generating kconfig for Rootfs Generate rootfs kconfig [INFO] oldconfig rootfs [INFO] generating petalinux-user-image.bb alinx@ubuntu:~/Downloads/peta_prj/ax_peta\$ petalinux [INFO] generating Kconfig for project	k-config -c kernel
[INFO] sourcing bitbake [INFO] generating plnxtool conf [INFO] generating meta-plnx-generated layer ~/Downloads/peta_prj/ax_peta/build/misc/plnx-generat eta	ted ~/Downloads/peta_prj/ax_p
~/Downloads/peta_prj/ax_peta [INFO] generating machine configuration [INFO] configuring: kernel [INFO] generating kernel configuration files [INFO] bitbake virtual/kernel -c menuconfig Parsing recipes: 14%  ######	ETA: 0:01:29   ETA: 0:01:31

2) しばらく待ってから、カーネルをコンフィグするためのコンフィグインターフェースがポップアップ する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



18.4 ルートファイルシステムのコンフィグ

次のコマンドを実行して、ルートファイルシステムがコンフィグできる。この実験ではデフォルトのコン フィグが維持される。

petalinux-config -c rootfs



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



2) コンパイル完了





スツョメンドを使用して DUUI ノナイルが工具される。 へいてんと - 記方に注思してく たさい。

petalinux-package --boot --fsbl ./images/linux/zyng\_fsbl.elf --fpga --u-boot --force

alinx@ubuntu:~/Downloads/peta\_prj/ax\_peta\$ petalinux-package --boot --fsbl ./ima ges/linux/zynq\_fsbl.elf --fpga --u-boot --force

INFO: File in BOOT BIN: "/home/alinx/Downloads/peta\_prj/ax\_peta/images/linux/zyn
q\_fsbl.elf"
INFO: File in BOOT BIN: "/home/alinx/Downloads/peta\_prj/ax\_peta/images/linux/des
ign\_1\_wrapper.bit"
INFO: File in BOOT BIN: "/home/alinx/Downloads/peta\_prj/ax\_peta/images/linux/u-b
oot.elf"
INFO: Generating zynq binary package BOOT.BIN...
INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
webtalk failed:Invalid tool in the statistics file:petalinux-yocto!
webtalk failed:Failed to get PetaLinux usage statistics!
alinx@ubuntu:~/Downloads/peta\_prj/ax\_peta\$

18.7 Linux をテストする

1) プロジェクト images -> linux ディレクトリの BOOT.BIN と image.ub を SD カードにコピーする。コ ピーする前に SD カードをフォーマットして、開発ボートに挿入し、開発ボードを SD カードブートに設定す る。



低価格、高品質が不可能?

Downloads peta\_prj ax\_peta images linux Q = BOOT.BIN design\_1\_wrapper. image.ub rootfs.cpio bit rootfs.ext3 rootfs.cpio.gz rootfs.cpio.gz.urootfs.ext4 boot rootfs.ext4.gz rootfs.jffs2 rootfs.manifest rootfs.tar.gz 101. J-boot.elf system.dtb System.map.linux u-boot.bin zynq\_fsbl.et vmlinux zimage シリアルターミナルを開き、開発ボードを起動する。 \_ **D** X



3) root でログインし、デフォルトでパスワードは root、インターネットケーブルを接続した後(ルー ターは自動 IP 取得をサポートする)、ifconfig コマンドを使用してネットワークステータスを確認できる。



COM3 - PuTTY	
root@ax_peta:~# macb e000b000.ethernet eth0: link up (1000, IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready	/Full)
root@ax peta:~# ifconfig	
eth0 Link encap:Ethernet HWaddr 00:0A:35:00:1E:53 inet addr:192.168.1.46 Bcast:192.168.1.255 Mas) inet6 addr: fe80::20a:35ff:fe00:1e53%lo/64 Scope: UP BROADCAST RUNNING MULTICAST MTU:1500 Metric: RX packets:20 errors:0 dropped:0 overruns:0 frame TX packets:12 errors:0 dropped:0 overruns:0 carr: collisions:0 txqueuelen:1000 RX bytes:4350 (4.2 KiB) TX bytes:1902 (1.8 KiB) Interrupt:29 Base address:0xb000	k:255.255.255.0 :Link :1 e:0 ier:0
<pre>lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1%1/128 Scope:Host UP LOOPBACK RUNNING MTU:65536 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame TX packets:0 errors:0 dropped:0 overruns:0 carrie collisions:0 txqueuelen:1 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B) root@ax_peta:~#</pre>	10 er:0

18.8 よくある問題



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

#### 18.8.1 Bad FIT kernel image format!が表示され、カーネルが起動できない。

U-Boot 2017.01 (Sep 05 2018 - 15:31:52 +0800) Board: Xilinx Zyng I2C: ready DRAM: ECC disabled 1 GiB MMC: sdhci\_transfer\_data: Error detected in status(0x208000)! sdhci@e0100000: 0 (SD), sdhci@e0101000: 1 (eMMC) SF: Detected w25q256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB \*\*\* Warning - bad CRC, using default environment serial Out: serial Err: serial ZYNQ GEM: e000b000, phyaddr ffffffff, interface rgmii-id Net: Warning: ethernet@e000b000 (eth1) using random MAC address - 5a:51:56:3c:ab:16 eth1: ethernet@e000b000 U-BOOT for ax\_peta ethernet@e000b000 Waiting for PHY auto negotiation to complete ......... TIMEOUT ! Hit any key to stop autoboot: 0 Device: sdhci@e0100000 Manufacturer ID: 41 OEM: 3432 Name: SD16G Tran Speed: 50000000 Rd Block Len: 512 SD version 3.0 High Capacity: Yes Capacity: 15 GiB Bus Width: 4-bit Erase Group Size: 512 Bytes reading image.ub Invalid FAT entry 4096 bytes read in 13 ms (307.6 KiB/s) ## Loading kernel from FLI Image at 10000000 ... Bad FIT kernel image format EPPOP.

解決方法:

SD カードの fat32 を再度フォーマットし、ブートファイルを再配置する

18.8.2 ファイルとコンフィグが保存できない

petal inux のデフォルトファイルシステムは RAM タイプであるため、保存できず、次のマニュアルで SD カ ードタイプに設定し、データは SD カードに保存できる。



#### 第十九章 SDK でLinux プログラムを開発する

前のマニュアルでは、petalinux を使用して組み込み Linux システムを作成したが、この実験は Linux ア プリケーションを作成し、開発ボード上で実行される。 この実験では、上記の実験で Linux オペレーティ ング環境を使用する必要がある。

#### 19.1 SDK を使って Linux アプリケーションを作成する

1) SDK ワークスペース用に/ home / alinx / work にディレクトリ linux\_app を作成する



3) ワークスペースは/home/alinx/work/linux\_app を選択する



#### Eclipse Launcher

#### Select a directory as workspace

Xilinx SDK uses the workspace directory to store its preferences and development artifacts.

Workspace: /home/alinx/work/linux_app	rowse
Use this as the default and do not ask again	cel OK
4) Create Application Projectを選択する	
■ Welcome 🛱	🏠 🗢 o 🕅 🖛 🕫
ALL PROGRAMMABLE. Welcome to Xilinx Software Development Kit	
💽 Create Application Project 📩 Import Project	
Typemals 💮 What's New 🌐 Web I	Resources
Create a new Xilinx® SDK project	1

5) プロジェクト名は hello を入力、OS Platform は Linux を選択する。





**New Project Application Project** Create a managed make application project. Project name: hello Use default location Location: /home/alinx/work/linux app/hello Browse... Choose file system: default 🔅 OS Platform: linux Target Hardware Processor Type: ps7 cortexa9 💿 Little-endian 🔘 Big-endian Endianness: Target Software Language: Compiler: Hypervisor Guest Linux System Root: a Browse Linux Toolchain: Browse ? < Back Next > Cancel Finish

6) Build Project



ello	2 #j
New	'⊖ir
Go Into	{
Open in <u>N</u> ew Win	dow
<u>С</u> ору	Ctrl+C }
Paste	Ctrl+V
<u>D</u> elete	Delete
Source	÷
Move	
Rename	F
Import	
Export	
<u>B</u> uild Project	
Clean Project	

#### 19.2 NFS 共有を実行する

1) 開発ボートにネットワークケーブルを挿入し(ルーターは自動的に IP を取得する必要がある)、電源を入れ、LinuxホストNFSをマウントする。ホスト IPは192.168.1.77、NFS ディレクトリは/home/alinx/work、/mnt は開発ボートのディレクトリである。ここでは、ホストと開発ボートが同じネットワークセグメント上にある必要がある。





Putty Sending select for 192.168.1.46... Lease of 192.168.1.46 obtained, lease time 86400 /etc/udhcpc.d/50default: Adding DNS 192.168.1.1 done. Starting Dropbear SSH server: Generating key, this may take a while... Public key portion is: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC3fwPFZmbH0qhjkpV+J/zQsBk5nYuFDON9kbBkN7d1 B4MEEC/2aV7AmR0S7SFQ1JZedVCh2YBREBy1mBV21d+Up125nGQnwwSOLj7T6kiFLU5fxRqDSHfxoOu5 nQq4ck7yS5kgyDbHE/vUD1yyE22J1AwcARDI91VRYCsv/aoumLWoEL3y85VLIwsrkNPUz8L3sXKICauv IAshRYbkxc2zzsc4HkMGJZ/NXL2MC527taECp5y2DmXHdEUTIDSVpXLH1cFf9PRFzjhaEoJpXgAu3thJ AGTza2ZPDmghv5MTuAr5KVpgu2yTizcmoMDeZWTmdpnh+OtqjU29FrEvhhxl root@ax peta Fingerprint: md5 f0:36:14:f7:ad:e9:48:56:87:71:80:b0:d7:f7:dd:6a dropbear. hwclock: can't open '/dev/misc/rtc': No such file or directory Starting syslogd/klogd: done Starting tcf-agent: OK PetaLinux 2017.4 ax\_peta /dev/ttyPS0 ax peta login: root Password: coot@ax\_peta:~# mount -t nfs -o nolock 192.168.1.77:/home/alinx/work /mnt ooreax\_pera:\*\*

2) ディレクトリ/mnt/linux\_app/hello/Debug に入り、hello.elf を実行する。Hello World がプリント アウトしたことが分かる。

cd /mnt/linux\_app/hello/Debug ./ hello.elf

B COM3 - PuTTY
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC3fwPFZmbH0qhjkpV+J/zQsBk5nYuFDON9kbBkN7dl
B4MEEC/2aV7AmR0575F0172edVCh2YBREBy1mBV21d+Up125nGQnwwSOLj7T6kiFLU5fxRqDSHfxoOu5
nQq4ck7yS5kgyDbHE/vUD1yyEZ201AwcARDI91VRYCsv/aoumLWoEL3y85VLIwsrkNPUz8L3sXKICauv
IAshRYbkxc2zzsc4HkMGJZ/NXLZMC527taECp5y2DmXHdEUTIDSVpXLH1cFf9PRFzjhaEoJpXgAu3thJ
AGTza22PDmghv5MTuAr5KVpgu2yTizcmoMDeZWTmdpnh+OtqjU29FrEvhhxl root@ax_peta
Fingerprint: md5 f0:36:14:f7:ad:e9:48:56:87:71:80:b0:d7:f7:dd:6a
dropbear
hwclock: can't open //dev/misc/rtc': No such file or directory
Starting syslogd klogd: done
Starting tcf-agent: OK
PetaLinux 2017.4 ax_peta /dev/ttyPS0
ax_peta login: root
Password:
rootgax_peta:~# mount -t nfs -o nolock 192.168.1.77:/home/alinx/work /mnt
rootgax_peta:~# cd /mnt
rootgax_peta:/mnt# cg linux_app/nello/Debug/
rootgax_peta:/mnt/linux_app/nello/Debug# random: crng init done
rootday peta:/mpt/lipuy.app/ballo/Dabugt /ballo alf
Holio Norid
root@ax peta:/mnt/linux app/hello/Debug#



#### 19.3 TCF-Agent を介してデバッグを実行する

1) Linux Agent をダブルクリックして、接続パラメーターをコンフィグする。

ြဲ Project Explorer 🛙	- 0	e helloworld.c 😫	•		Beog
8		⊕ * Copyright (c) 2012 Xilinx, Inc. All rights reserved.		8	e P
🖉 😂 hello		<pre>? #include <stdio.h></stdio.h></pre>		-	⊽
🕨 💐 Binaries		⊖int main()			Al si
<ul> <li>Bebug</li> <li>Brc</li> </ul>		<pre>printf("Hello World\n"); return 0;</pre>			e m
		,		in the second	
Target Connection □X					
ä Target Connection 🛱		} Pro ⊠ ⊇ Tas ⊇ Con ⊇ Pro ⊇ SD ⊇ ⊒ SDKLog ⊠ I3:25:08 INFO		Regis	tering
Target Connection ≅		} Pro ☆ Tas ♥ Con ♥ Pro ♥ SD ♥ ■ SDKLog ☆ 13:25:08 INFO 13:25:09 INFO 13:25:09 INFO 13:25:09 INFO		Regis	tering hing X server

2) Host の IP アドレスを入力する。ここで入力するのは開発ボートの IP アドレスである。



😕 回 🛛 Target Connection Details	
Edit Target Connection	
Edit Target Connection	
Target Name Linux Agent	
🧭 Set as default target	
Specify the connection type and properties	
Type Linux TCF Agent	*
Host 192.168.1.46	
Port 1534	
Advanced >>	
?	Test Connection Cancel OK

3) Test Connection をクリックして、テスト接続が成功した場合、Linux TCF Agent サービスが実行されており、デバッグを実行できる。

dit Target Co	nection	essful			_	
Set as	Successfully '192.1684.46	established conr	nection to 'Linux T	CF Agent' on the l	nost	
Type L		X			ок	3)
				_	_	-

4) プロジェクトを選択して、右クリックして実行する。



roject Explorer 🛛 🗖 🗖	े helloworld.c छ
🚍 🔄 🦁 🔻	• * Copyright (c) 2012 Xilinx, Inc. All rights reserved
hello	<pre>#include <stdio.h></stdio.h></pre>
New	* O
Go Into	
Open in New Window	tf("Hello World(n");
	rn θ;
Сору	Ctrl+C
Paste	Ctri+V
Delece	Delete
Source	
Rename	F2
Import	
Export	
Build Project	
Clean Project	
Refresh	F5
Close Project	
Close Unrelated Projects	
Build Configurations	
Run As	1 Launch on Hardware (System Debugger)
Debug As	2 Start Performance Analysis
F Compare With	<ul> <li><u>3</u> Launch on Hardware (System Debugger on QEMU)</li> </ul>
L Restore from Local History	4 Launch on Hardware (GDB)
C/C++ Build Settings	5 Local C/C++ Application




TCF-Agent はLinux アプリケーションを簡単にデバッグできるが、NFS サポートも必要ない。しかし、複数行プログラムのデバッグにはあまり適していない。アプリケーションがクラッシュした場合、デバッグ環境の復元には役立たず、開発ボードを再起動する必要があるため、あまり使用されていない。



#### 第二十章 Linux 環境で GPIO 実験

前のマニュアルでは、SDk を使用して zynq バージョンの helloworld 実験をコーディングする方法につい て説明した。この実験では、zynq のペリフェラルをコントロールする方法を紹介する。実験では、GPIO を 例として使用する。ZYNQ の GPIO は、二つのタイプがあって、一つは PS に付属する GPIO、1 つは PL で実装 された GPIO である。Vivado プロジェクトをビルドする時に、Xilinx の GPIO IP が追加される。Xilinx が提 供する IP コアのほとんどはすでに Linux でドライバがあって、しかも、使える。例えば、AXI GPIO ドライ バはカーネルで再構成せずに使用できる。

この http://www.wiki.xilinx.com/Linux+Drivers サイドに、Linux 下のすべての Xilinx ドライバを見つける。例えば、GPIO ドライバは以下のように、一部ドライバは詳しい使い方もある。

GPIO	Zynq and	GPIO Driver	Yes	drivers/gpio/gpio-zynq.c
	Zynq Ultrascale+ MPSoC			
GPIO	axi_gpio	AXI GPIO Driver	Yes	drivers/gpio/gpio-xilinx.c
HDMI Clocks	SI5324 Clock Multiplier/Jitter Attenuator	CCF SI5324 Driver	No	hdmi-modules/clk/*

GPIO ドライバの詳細ページhttp://www.wiki.xilinx.com/Linux%20GPIO%20Driver にGPIO ドライバの使用 範囲、デバイスツリーの例、及びプログラムの作成方法が紹介されている。

#### 20.1 SHELL コントロールを使用する

Linux は強い SHELL 機能を提供しており、Linux を学習に必要な技能である。ZYNQ を勉強するには、しっかりと SHELL をマスターしないといけない。このマニュアルでは、Linux と SHELL の使用方法を紹介しない。

Is /sys/class/gpio コマンドで GPIO 番号を確認できる。

root@zynq	l:~# ls /:	sys/class/	/gpio		
export	gpio900	gpio903	g <mark>p</mark> io906	gpio957	gpiochip898
gpio898	gpio901	gpio904	gpio919	gpiochip1016	unexport
gpio899	gpio902	gpio905	gpio956	gpiochip1020	

gpio\_test. sh ファイル内容は以下通り、gpio\_test 関数は、パラメーターに従って GPIO をエクスポート し、次に for ループを3回行う。最初に0を書き込み、次に1を書き込み、gpio\_test を5回呼び出してか ら、5つの LED を点灯する。その中で、898 は PS 側で、その他は PL 側である。(次のセクションでは、GPIO 番号を確認する方法について説明する)

```
#!/bin/sh
gpio_test() {
    gpio=11
    echo Sgpio > /sys/class/gpio/export
    echo out > /sys/class/gpio/gpio$(gpio)/direction
    for i in $(seq 1 3)
    do
    echo 0 >/sys/class/gpio/gpio$(gpio)/value
```





	sleep 1
	echo 1 >/sys/class/gpio/gpio <mark>\${gpio}</mark> /value
	sleep 1
	done
	echo \$gpic > /sys/class/gpio/unexport
}	
gpio	test 898
gpio	test 1016
gpio	test 1017
gpio	test 1018
gpio	test 1019
	_

NFS をマウントすることでこの SHELL を実行できる

SHELL が実行できない場合、先に実行許可を追加しておく。コマンドは次のとおり。



20.2 C 言語を使ってコントロールする

ほとんどの場合、C 言語を使用して周辺機器をコントロールする必要がある。Xilinx の Wiki ページ http://www.wiki.xilinx.com/GPI0%20User%20Space%20App には、次の GPI0 テストコードが見つかり、コー ト内容は以下通り。





on open source linux the GPIO can be toggled about every 1sec. 11 // The following commands from the console setup the GPIO to be // exported, set the direction of it to an output and write a 1 // to the GPIO. // // // bash> echo 898 > /sys/class/gpio/export // bash> echo out > /sys/class/gpio/gpio898/direction // bash> echo 1 > /sys/class/gpio/gpio898/value // if sysfs is not mounted on your system, the you need to mount it // bash> mount -t sysfs sysfs /sys // the following bash script to toggle the gpio is also handy for // testing 11 // while [ 1 ]; do // echo 1 > /sys/class/gpio/gpio898/value
// echo 0 > /sys/class/gpio/gpio898/value
// done // to compile this, use the following command
// gcc gpio.c -o gpio // The kernel needs the following configuration to make this work. // CONFIG\_GPIO\_SYSFS=y // CONFIG\_SYSFS=y // CONFIG\_EXPERIMENTAL=y // CONFIG\_GPIO\_XILINX=y int main() int valuefd, exportfd, directionfd; printf("GPIO test running...\n"); // The GPIO has to be exported to be able to see it // in sysfs exportfd = open("/sys/class/gpio/export", O\_WRONLY); (exportfd < 0Ł printf("Cannot open GPIO to export it\n"); exit(1); write (exportfd, "898", 4); close (exportfd) ; printf("GPIO exported successfully\n"); // Update the direction of the GPIO to be an output directionfd = open("/sys/class/gpio/gpio898/direc on", O RDWR); if (directionfd < 0) -{ printf("Cannot open GPIO direction  $c \in \mathbb{R}^{2}$ exit(1); 3 write(directionfd, "out"
close(directionfd); printf("GPIO directi output successfully\n"); // Update the direction of the GRIO to be an output ss/gp.o/gpio898/direction", O\_RDWR); #1 s directionfd = open("/ < 0) if (directionfo ł pen GPIO direction it\n"); printf("
exit(1); anno ъ write(directionfd, "
close(directionfd); "0 ", 4); crion set as output successfully\n"); printf("GPIO d // Get the GPIO value ready to be toggled valuefd = open("/sys/class/gpio/gpio898/value", O\_RDWR); if (valuefd < 0)</pre> 4 printf("Cannot open GPIO value\n"); exit(1); 3 printf("GPIO value opened, now toggling...\n"); // toggle the GPIO as fast a possible forever, a control c is needed // to stop it while (1) write (valuefd, "1", 2); write(valueid, 1, 2);
 sleep(1);
write(valuefd, "0", 2); sleep(1); } }



今回はコンパイルに SDK を使用しない。ソースコードの名前を gpio.c にし、次のコマンドを実行してコ ードをコンパイルする。

source /opt/Xilinx/Vivado/2017.4/settings64.sh arm-linux-gnueabihf-gcc gpio.c - o gpio

コンパイルが完了すると、gpio ファイルが生成される。Windows やLinux とは異なり、拡張名は厳しくない。gpio ファイルは elf ファイルである。

gpio を実行すると、ps 端子 LED が点滅していることがわかる。これは、この 898 が PS 側の最初の LED であることを示している。



20.2.1 GPI0 のコードの確認

次のコマンドを実行すると、gpiochip898 gpiochip1016 gpiochip1020 が表示される。これは、3 つの GPIO コントローラーがあることを示し、数字はコントローラーGPIO の基数である。

ls /sys/class/gpio



### 20.2.2 物理 GPI0 との関係の確認

次のコマンドで、GPI01016 と物理 GPI0 の関係を確認する。デバイスツリーの gpio のノードは/ amba\_pl / gpio @ 41210000 であることがわかる。デバイスツリーのノードでどの物理 GPI0 を確認できる。

cat /sys/class/gpio/gpiochip1016/label

```
root@zynq:/sys/class/gpio/gpiochip1016# cat /sys/class/gpio/gpiochip1016/label
/amba_pl/gpio@41210000
root@zynq:/sys/class/gpio/gpiochip1016#
```



#### 20.3 実験のまとめ

この実験の焦点は、Xilinxから提供された情報を通して ZYNQ を学習すること。技術資料は迅速に更新され、チップメーカーから提供された最新の情報を入手して最新最高の技術を取得できる。 フォローアップ マニュアルでは、PCIe ドライバと PL 側イーサネットドライバが Xilinx から提供されている、これらの資料 は wiki で入手できる。

非 xilinx IP または独自の IP を使用する場合、独自のドライバを開発する必要がある。これは、Linux ドライバを実行していない開発者にとっての課題であるため、Xilinx の IP を使用してシステムを構築したほうがお勧めである。利点は、Linux ドライバを開発する必要がないこと、欠点は、十分な柔軟性がないこと。 もし、IP に問題がある、または、ドライバに問題がある場合、問題をすばやく見つける ことができない。

ホームページ: <u>http://www.csun.co.jp</u> メール: info@csun.co.jp





#### 第二十一章 Petalinux での HDMI ディスプレイ

前のマニュアルで、Petalinux で組み込み linux システムを開発するのを体験したが、使う機能は Petalinux のただ一部である。この実験では、独自のカーネルを使用して Linux を実行する方法を紹介する。 これにより、HDHI ディスプレイなど、多くのドライバをカーネルに追加できる。

開発ボードは HDMI インターフェイスチップを使用せず、PFGA を使用してエンコードを完了し、Core Electronics は Xilinx が提供するカーネルに HDMI コード化 IP ドライバを追加した。 その他のバージョン を使用するソフトウェア開発者に、このマニュアルでは変更された linux-xlnx-xilinx-v2017.4 カーネルの みを提供し、他のバージョンの変更されたバージョン、および変更手順が提供されていないことを注意して ください。

#### 21.1 Petalinux のコンフィグ

この実験は、前のPetalinux プロジェクトの実験で変更する。前の実験コンテンツを習得する必要がある。

1) カーネルソースファイルをLinux ホストにコピーして、次に解凍する。解凍後のカーネルディレクト リはこの実験 Petalinux に使うカーネルである。



2) ターミナルを開き、前の実験中の Petal inux プロジェクトディレクトリに入る。





source /opt/pkg/petalinux/settings.sh

4) 下のコマンドを実行して、vivado環境変数を設定する。

source /opt/Xilinx/Vivado/2017.4/settings64.sh



5) 下のコマンドで Petal inux をもう一回コンフィグする。

petalinux-config



6) Linux Components Selection ---> linux-kernel (linux-xlnx) --->を選択する。



home/alinx/Downloads/peta\_prj/ax\_peta/project-spec/configs/config - misc/conf Linux Components Selection Linux Components Selection Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [\*] built-in [] [\*] First Stage Bootloader [\*] Auto update ps7\_init II-hoot (II-hoot-plax) linux-kernel (linux-xlnx) <Select> < Exit > < Help > < Save Load > 7) ext-local-src を選択してスペースキーを押す



8) External linux-kernel local source settings --->を選択する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

Linux Components Selection Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus --->). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc> to exit, <?> for Help, </> for Search. Legend: [\*] built-in [ ] [\*] First Stage Bootloader Auto update ps7 init [\*] u-boot (u-boot-plnx) linux-kernel (ext-local-src) ---External linux-kernel local source settings <Select> < Exit > < Help > < Save < Load > EXternal linux-kernel local source path を選択する。 External linux-kernel local source settings Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search, Legend: [\*] built-in [ ] EXternal linux-kernel local source path

() EXternal linux-Kernel local source path

9) Linux カーネルソースコードのパスを/home/alinx/work/linux-xlnx-xilinx-v2017.4入力する。実際のパスはカーネルの場所によって異なる。ここに例を示す。



home/alinx/Downloads/peta_prj/ax_peta/project-spec/configs/config - misc/c → Linux Components Selection → External linux-kernel local source settings
EXternal linux-kernel local source path Please enter a string value. Use the <tab> key to move from the input field to the buttons below it. /home/alinx/work/linux-xlnx-xilinx-v2017.4</tab>
< Ok > < Help >
0) 保存して閉じる。 misc/config System Configuration Arrow keys navigate the menu. <enter> selects submenus&gt; (or empr submenus). Highlighted letters are hotkeys. Pressing <y> includes, <n> excludes, <m> modularizes features. Press <esc><esc></esc></esc></m></n></y></enter>
exit, for Help,  for Search. Legend: [*] built-in [] Linux Components Selection> Auto Config Settings> -*- Subsystem AUTO Hardware Settings> DTG Settings> u-boot Configuration> Image Packaging Configuration> irmware Version Configuration>
<pre>     Select&gt; &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt; </pre>

21.2 Linux カーネルをコンフィグする

1) 下のコマンドを実行してカーネルをコンフィグする。

petalinux-config -c kernel





2) ポップアップウインドウが表示され、Device Drivers→Graphics support に入り、Digilent VGA/HDMI DRM Encoder Driver を選択してyをs押す。

→ Device Drivers → Graphics support
Graphics support Arrow keys navigate the menu. <enters selects="" submenus=""> (or empty submenus). Highlighted letters are hotkeys. Pressing <y> includes, <n> excludes, <m> modularizes features. Press <esc><esc> to exit, <? > for Help,  for Search. Legend: [*] built-in []</esc></esc></m></n></y></enters>
Display Interface Bridges> < > DRM Support for STMicroelectronics SoC stiH4xx Series <*> Xilinx DRM -*- Xilinx DRM Display Port Driver -*- Allinx DRM Display Port Subsystem Driver < Xilinx DRM MIPI DSI Driver
<pre>     Clinx DRM SDI Subsystem Driver     Digilanc VGA/HDMI DRM Encoder Driver     ALINX LCD DRM Encoder Driver     ALINX 9134 Encoder Driver     L(+) </pre>
<pre><select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>

3) Device Drivers p Common Clock Framework オプションから Digilent axi\_dynclk Driver を選択して y を押す。





### 21.3 デバイスツリーを変更する

< > /// clkgen driver

< >> Digilent axi\_dynclk Driver

デバイスツリーは、デバイス情報を記述するフォーマットされたテキストの一種で、このテキスト構造は XML や JSON に似ている。

< > Clock driver for CS2000 Fractional-N Clock Synthesizer & Cloc

< Save >

< Load >

1) petalinux プロジェクトファイルで system-user. dtsi という名前のファイルを開く。

< > Clock driver for TT CDCE706 clock synthesizer < > Clock driver for TI CDCE925 devices

Clock driver for Freescale QorIQ platforms

<Select> < Exit > < Help >



nts	syste	em-user.dts			

### 2) デバイスツリーの変更内容は以下通りである。





&gem0 { phy-handle = <&ethernet\_phy>; ethemet\_phy: ethernet-phy@1 { reg = <1>; device\_type = "ethemet-phy"; }; }; &amba\_pl { hdmi\_encoder\_0:hdmi\_encoder { compatible = "digilent,drm-encoder"; digilent,edid-i2c = <&i2c0>; }; xilinx\_drm { compatible = "xInx,drm";  $xlnx,vtc = < &v_tc_0>;$ xInx,connector-type = "HDMIA"; xlnx,encoder-slave = <&hdmi\_encoder\_0>; clocks = <&axi\_dynclk\_0>; dgInt,edid-i2c = <&i2c0>; planes { xInx, pixel-format = "rgb8888"; plane0 { dmas = <&axi\_vdma\_0 0>; dma-names = "dma"; }; }; }; }; &axi\_dynclk\_0 { compatible = "digilent,axi-dynclk"; #clock-cells = <0> clocks = <&clkc 15>; }; &v\_tc\_0 { compatible = "xinx,v-tc-5.01.a";

### 21.4 テスト petal inux プロジェクトのコンパイル

1) 下のコマンドで、uboot、カーネル、ルートファイルシステム、デバイスツリーなどをコンパイルする。

petalinux-build





2) 下のコマンドを実行して BOOT ファイルを作成する。スペースとハイフンに注意してください。

petalinux-package --boot --fsbl ./images/linux/zynq\_fsbl.elf --fpga --u-boot --force

3) BOOT. bin と iamge. ub を SD カードにコピーして、開発ボードを SD モードで起動するように設定し、 HDMI ディスプレイを接続して、開発ボ→ドを起動する。

		1919	pio	1010
IN design	1 wrapper.	mage.ub rooti	rootfs.cpio	io.gz rootfs.cpi bool
		1 Ca		1.0
		1010	targe	101 1918
kt4 rootf	fs.ext4.gz ro	otfs.jffs2 rootfs.i	manifest rootfs.ta	ar.gz system.
				•
oin u-b	boot.elf	vmlinux zim	nage zynq_fsb	Lelf
	IN design ct4 root	IN design 1 Avrapper. Sit ct4 rootfs.ext4.gz ro pin u-boot.elf	IN design if wrapper. Image.ub roots bit ct4 rootfs.ext4.gz rootfs.jffs2 rootfs. jin u-boot.elf vmlinux zin	IN design & wrapper. Image.ub rootfs.cpio rootfs.cp tit term te

4) ディスプレイに以下の内容が表示される。





21.5 よくある問題

21.5.1 システムのスリープを防ぐ方法

スリープ前にコマンドを実行する。

echo -e "\033[9;0]\033[?33l\033[?25h\033[?1c" > echo -e "\033[9;0]\033[?33l\033[?25h\033[?1c" > echo -e "\033[9;0]\033[?33l\033[?25h\033[?1c" > echo -e "\033[9;0]\033[?33l\033[?25h\033[?1c" >

/dev/tty0 /dev/tty1

- /dev/tty
- /dev/console



#### 第二十二章 Debian デスクトップシステムの使用

前のマニュアルで Petal inux で組み込む Linux システム及び HDMI ディスプレイを作成する方法を紹介した。この実験では、Debian に基づいてルートファイルを作成する。ルートファイルシステムの作成が複雑のため、この章では説明しない。生成された Debian ルートファイルシステムを直接使用する。

22.1 Petalinux のコンフィグ

Debian ルートファイルシステムが大きいのため、QSPI flash には保存できず、SD カードまたは emmc にし か保存できない、だから、Petalinux をコンフィグする必要がある。

この実験は、前の Petal inux プロジェクトの実験をまだ変更するため。以前のテストコンテンツをマスターする必要がある。

1) ターミナルを開き、前の実験の Petal inux プロジェクトディレクトリに入る



3) 下のコマンドを実行して vivado 環境変数を設定する

source /opt/Xilinx/Vivado/2017.4/settings64.sh



4) 下のコマンドで Petal inux をもう一回コンフィグする。



petalinux-config

5) Image Packaging Configuration --> Root filesystem type オプションから SD card を選択し、ルートファイルシステムを SD カードに置く。



- 22.2 linux カーネルをコンフィグする
- 1) 下のコマンドを実行してカーネルをコンフィグする

petalinux-config -c kernel



🔵 回 🛛 alinx@ubuntu: ~/Downloads/peta\_prj/ax\_peta \*\*\* Execute 'make' to start the build or try 'make help'. [INFO] sourcing bitbake [INFO] generating plnxtool conf [INFO] generating meta-plnx-generated layer -/Downloads/peta\_prj/ax\_peta/build/misc/plnx-generated ~/Downloads/peta\_prj/ax\_p eta -/Downloads/peta prj/ax peta [INFO] generating machine configuration [INF0] generating bbappends for project . This may take time ! ~/Downloads/peta\_prj/ax\_peta/build/misc/plnx-generated ~/Downloads/peta\_prj/ax\_p eta -/Downloads/peta prj/ax peta [INFO] generating u-boot configuration files [INFO] generating kernel configuration files [INFO] generating kconfig for Rootfs Generate rootfs kconfig [INFO] oldconfig rootfs [INFO] generating petalinux-user-image.bb [INFO] successfully configured project webtalk failed:PetaLinux statistics:extra lines detected:notsent\_nofile! webtalk failed:Failed to get PetaLinux usage statistics! alinx@ubuntu:~/Downloads/peta\_prj/ax\_peta\$ petalinux-config - ker<mark>n</mark>el

22.2.1 USB WIFI モジュールドライバをコンフィグする

1) Networking Support -> Wireless ->オプションの中から cfg80211 - wireless configuration API を選択して、次に Generic IEEE 802.11 Networking Stack (mac80211)を選択する。



2) Device Drivers -> Network device support -> Wireless LAN -> Realtek rtlwifi family of devices オプションから Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter を選択する。





ek RTL8192SE/RTL8191SE PCIE Wireless Network Adapter ( ek RTL8192DE/RTL8188DE PCIE Wireless Network Adapter ( ek RTL8723AE PCIE Wireless Network Adapter (NEW) ek RTL8723BE PCIE Wireless Network Adapter (NEW) ek RTL8188EE Wireless Network Adapter (NEW) ek RTL8192EE Wireless Network Adapter (NEW) ek RTL8192EE Wireless Network Adapter (NEW) ek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) ek RTL8192CU/RTL8188CU USB Wireless Network Adapter	<pre>ealtek RTL8192SE/RTL8191SE PCIe Wireless Network Adapter ( ealtek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter (NEW) ealtek RTL8723BE PCIe Wireless Network Adapter (NEW) ealtek RTL8188EE Wireless Network Adapter (NEW) ealtek RTL8192EE Wireless Network Adapter (NEW) ealtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) ealtek RTL8192CU/RTL8188CU USB Wireless Network Adapter t&gt; &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</pre>	<pre>&lt; &gt; Realtek RTL8192SE/RTL8191SE PCIe Wireless Network Adapter ( &lt; &gt; Realtek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter ( &lt; &gt; Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8723BE PCIe Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter ((+) </pre>	<pre>&gt; Realtek RTL8192SE/RTL8191SE PCIe Wireless Network Adapter ( &gt; Realtek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter (NEW) &gt; Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter +) </pre>	Realtek RTL8192SE/RTL8191SE PCIe Wireless Network Adapter ( Realtek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter ( Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) Realtek RTL8723BE PCIe Wireless Network Adapter (NEW)	
ek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter ( ek RTL8723AE PCIe Wireless Network Adapter (NEW) ek RTL8723BE PCIe Wireless Network Adapter (NEW) ek RTL8188EE Wireless Network Adapter (NEW) ek RTL8192EE Wireless Network Adapter (NEW) ek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) ek RTL8192CU/RTL8188CU USB Wireless Network Adapter	<pre>caltek RTL8192DE/RTL8188BDE PCIe Wireless Network Adapter ( caltek RTL8723AE PCIe Wireless Network Adapter (NEW) caltek RTL8723BE PCIe Wireless Network Adapter (NEW) caltek RTL8188EE Wireless Network Adapter (NEW) caltek RTL8192EE Wireless Network Adapter (NEW) caltek RTL821AE/RTL8812AE Wireless Network Adapter (NEW) caltek RTL8192CU/RTL8188CU USB Wireless Network Adapter ct&gt; &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt; </pre>	<pre>&lt; &gt; Realtek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter ( &lt; &gt; Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8723BE PCIe Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter 4(+) </pre>	<pre>&gt; Realtek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter ( &gt; Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) &gt; Realtek RTL8723BE PCIe Wireless Network Adapter (NEW) &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU VII VII VII VII VII VII VII VII VII VI</pre>	Realtek RTL8192DE/RTL8188DE PCIe Wireless Network Adapter ( Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) Realtek RTL8723BE PCIe Wireless Network Adapter (NEW)	
ek RTL8723AE PCIe Wireless Network Adapter (NEW) ek RTL8723BE PCIe Wireless Network Adapter (NEW) ek RTL8188EE Wireless Network Adapter (NEW) ek RTL8192EE Wireless Network Adapter (NEW) ek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) ek RTL8192CU/RTL8188CU USB Wireless Network Adapter	to the kills of the second sec	<pre>&lt; &gt; Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8723BE PCIe Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &lt;*&gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter 4(+) </pre>	<pre>&gt; Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) &gt; Realtek RTL8723BE PCIe Wireless Network Adapter (NEW) &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter +) </pre>	Realtek RTL8723AE PCIe Wireless Network Adapter (NEW) Realtek RTL8723BE PCIe Wireless Network Adapter (NEW)	
ek RTL87236E PCIE WIFELESS NETWORK Adapter (NEW) ek RTL8188EE Wireless Network Adapter (NEW) ek RTL8192EE Wireless Network Adapter (NEW) ek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) ek RTL8192CU/RTL8188CU USB Wireless Network Adapter	to to the solution of the second seco	<pre>&lt; &gt; Realtek RIL8723BE PCIe Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &lt;*&gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter 4(+) </pre>	<pre>&gt; Mealtek RTL8723BE PCIe Wireless Network Adapter (NEW) &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter +) </pre> <pre> <select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt; </select></pre>	Mealtek KIL8/23BE PCIe WIReless Network Adapter (NEW)	
ek RTL8188EE WIFEless Network Adapter (NEW) ek RTL8192EE Wireless Network Adapter (NEW) ek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) ek RTL8192CU/RTL8188CU USB Wireless Network Adapter	t> < Exit > < Help > < Save > < Load >	<pre>&lt; &gt; Realtek RTL8188EE Wireless Network Adapter (NEW) &lt; &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &lt;&gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &lt;*&gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter 4(+) &lt;<select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	<pre>&gt; Realtek RTL8188EE WIreless Network Adapter (NEW) &gt; Realtek RTL8192EE Wireless Network Adapter (NEW) &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter +) </pre> <pre> <select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt; </select></pre>	Realtak DTI 0400055 Utaalaan Naturak Adaptas (NEU)	
ek RTL8192EE Wtretess Network Adapter (NEW) ek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) ek RTL8192CU/RTL8188CU USB Wireless Network Adapter	altek RTL8192EE WITEless Network Adapter (NEW) altek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) altek RTL8192CU/RTL8188CU USB Wireless Network Adapter	<pre>&lt;    Realtek RTL8192EE wtretess Network Adapter (NEW) &lt;&gt;    Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &lt;*&gt;    Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter 4(+) &lt;<select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	<pre>&gt; Realtek RTL8192EE Wtreless Network Adapter (NEW) &gt; Realtek RTL8821AE/RTL8812AE Wireless Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter +) <select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	Realter RIL8188EE WIREless Network Adapter (NEW)	
ek RTL8192CU/RTL8188CU USB Wireless Network Adapter	altek RTL8192CU/RTL8188CU USB Wireless Network Adapter	<pre>&lt;*&gt; Realtek RTL802IAE/RTL80IZAE Wtretess Network Adapter ((Lew)) &lt;*&gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter ((+) <select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	<pre>&gt; Realtek RTL802IAE/RTL80I2AE wtretess Network Adapter (NEW) &gt; Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter +) <select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	Realter RIL8192EE WIRELESS Network Adapter (NEW)	
ek RIEDISZEU/RIEDISSEU USB WEIELESS NELWOIK AUGPLEI	t> < Exit > < Help > < Save > < Load >	<pre>4(+) <select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	<pre><select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	Realtek RTL8021AE/RTL8012AE WITELESS Network Adapter (NEW)	
	ts < Exit > < Help > < Save > < Load >	<pre><select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	<pre><select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	REDECER RIEDISEO/RIEDIOCO OSO HETELESS RECHOIR RUBPET	
	t> < Exit > < Help > < Save > < Load >	<pre><select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>	<pre><select> &lt; Exit &gt; &lt; Help &gt; &lt; Save &gt; &lt; Load &gt;</select></pre>		
< Exit > < Help > < Save > < Load >				Select> < Exit > < Help > < Save > < Load >	
		/ / / / / / / / / / / / / / / / /			

Arrow keys submenus includes, exit,	navigate ). Hig <n> exclud for Help,</n>	the menu hlighted ts, <n> r s/&gt; for S</n>	<enter> letters a modularize Search. L</enter>	selects s re hotkeys s features egend: [*]	ubmenus . Press . Press built-i	> (or sing <y> s <esc><es< th=""><th>empty sc&gt; to</th></es<></esc></y>	empty sc> to
Mi [*] <m> &lt; &gt; &lt; &gt; &lt; &gt; &lt; &gt;</m>	edia USB Ad *** Webcan USB Vidco USB Vidco USB Philip PiA2 Vide SB ZR364X USB Syntek USB Sensor	apters devices class (U) t events d webcams s Cameras o For Lin X Camera DC1125 ( ay 2255 )	*** device su s (NEW) - s (NEW) nux (NEW) support ( Camera sup video capt	pport (NEW > NEW) port (NEW) ure device	) (NEW)		

- 2) 保存して閉じる。
- 22.3 Petal inux プロジェクトのコンパイルとテスト



1) 下のコマンドを使用して、uboot、カーネル、ルートファイルシステム、デバイスツリーなどをコン フィグする。

petalinux-build

alinx@ubuntu: ~/Downloads/peta\_prj/ax\_peta kipped, 0 masked, 0 errors. NOTE: Resolving any missing task queue dependencies NOTE: Executing RunQueue Tasks NOTE: Tasks Summary: Attempted 2 tasks of which 0 didn't need to be rerun and al L succeeded. [INFO] successfully configured kernel webtalk failed:PetaLinux statistics:extra lines detected:notsent\_nofile! webtalk failed:Failed to get PetaLinux usage statistics! alinx@ubuntu:~/Downloads/peta\_prj/ax\_peta\$ petalinux-build [INFO] building project [INFO] sourcing bitbake INFO: bitbake petalinux-user-image Time: 0:00:01 Parsing of 2466 .bb files complete (2433 cached, 33 parsed). 3259 targets, 226 s kipped, 0 masked, 0 errors. NOTE: Resolving any missing task queue dependencies | ETA: 0:00:08 Initialising tasks: 18% |########

2) 下のコマンドを実行して BOOT. BIN ファイルを生成する。スペースとハイフンを注意してください。

petalinux-package --boot --fsbl ./images/linux/zynq\_fsbl.elf --fpga --u-boot --force

### 22.4 SD カードファイルシステムを作成する

SD カードファイルシステムを作成するには SD カードの内容が失う可能性があるため、事前にバックアップしてください。

- 22.4.1 SD カードのパーティションを変更する。
- 1) 開発ボードの SD カードをカードリーダーに挿入し、次にコンピューターの USB ポートに挿入する。
- 2) 仮想マシン Linux に接続する



0	电源(P)	L.				
0	可移动设备(D)		CD/DVD (SATA)			
	暂停(U) Ctrl+Shift+P	1	网络适配器			
-	发送 Ctrl+Alt+Del(E)		FIEDER	٠		
	抓取输入内容(I) Ctrl+G		声卡	•	ta	project-spec meta-user recip
16	(快頭(N) ►		Super Top Mass Storage Device	•	1	连接(新开与 主机 的连接)(C)
~	捕获屏幕 (C) Ctrl+Alt+PrtScn	Г			-	更改图标(1)
	(M)	L	system-user.dtsi		~	在状态栏中显示(S)
	重新安装 VMware Tools(T)	1				
51	设置(S) Ctrl+D	bs				

3) ubuntu の検索パスに disk と入力すると、Disks のアイコンが表示される。



4) DIsks アイロンをクリックすると、DIsks のダイアログボックスが表示される。この実験で、SD カードは、FAt と EXT という 2 つのパーティションに分けているが、ここではもう一回分け直す。





5) 各パーティションの下のパーティション削除アイコンを選択し、すべてのパーティションを削除する。







6) +アイコンをクリックして最初のパーティションを追加する。この実験は 5000MB を入力し、フォーマットは FAT で、ZYNQ のブートファイル BOOT. bin とカーネルファイル、及びデバイスツリーを保存するため、名前は FAT にする。

	<b>≜</b> () ≡
129 GB Hard Disk VMware, VMware Virtual S CD/DVD Drive VMware Virtual SATA CDRW Drive 8.0 GB Drive Mass Storage Device	Model Mass Storage Beyler (100) Size Create Partition Partitioning Serial Number Volumes Contents Co
$\Diamond$	For example, "My Files" or "Backup Data" Delune /c Contents U

7) ルートファイルシステムを保存するため、第二のパーティションを作成し、フォーマットを EXT4 に し、名前を EXT にする。







9) システムは自動的にパーティションをマウントし、ウィンドウをポップアップする





22.4.2 ルートファイルシステムをSDカードEXT4パーティションに同期する。

1) ルートファイルシステムの圧縮ファイルをLinux ホストにコピーする(実験を/home / alinx / work ディレクトリーにコピーする)

ne work	<u>}</u>		
linux_app	linux-xlnx-xilinx- v2017.4	test	debian_rootfs.tar. gz

2) 解凍コマンド sudo tar -zxvpf debian\_rootfs.tar.gz を入力して、ファイルシステムを抽出する。 解凍には数分かかる場合がある。 解凍方法はマニュアルと一致する必要があることに注意してください。

### alinx@ubuntu:~/work\$ sudo tar zxvpf debian\_rootfs.tar.gz

3) これらのファイルシステムのすべてのファイルを SD カードの EXT パーティションのルートディレクトリにコピーする必要がある。コマンドウィンドウでコマンド cd debian\_rootfs を入力して、ルートファイルシステムのディレクトリに入る。



🤒 🗇 💿 alinx@ubuntu: ~/work/debian_rootfs	
debian_rootfs/usr/share/zoneinfo/Australia/Adelaide	
debian rootfs/usr/share/zoneinfo/Australia/Eucla	
debian_rootfs/usr/share/zoneinfo/Australia/Darwin	
debian_rootfs/usr/share/zoneinfo/Australia/Brisbane	
debian_rootfs/usr/share/zoneinfo/Australia/North	
debian_rootfs/usr/share/zoneinfo/Australia/NSW	
debian_rootfs/usr/share/zoneinfo/Australia/Yancowinna	
debian_rootfs/usr/share/zoneinfo/Australia/Canberra	
debian_rootfs/usr/share/zoneinfo/Australia/West	
debian_rootfs/usr/share/zoneinfo/Australia/Queensland	
debian_rootfs/usr/share/zoneinfo/Australia/Perth	
debian_rootfs/usr/share/zoneinfo/Australia/Sydney	
debian_rootfs/usr/share/zoneinfo/Australia/Tasmania	
debian_rootfs/usr/share/zoneinfo/Jamaica	
debian_rootfs/usr/share/zoneinfo/GB-Eire	
debian_rootfs/usr/share/zoneinfo/PST8PDT	
debian_rootfs/usr/share/zoneinfo/EST	
debian_rootfs/usr/share/zoneinfo/CST6CDT	
debian_rootfs/usr/share/zoneinfo/Hongkong	
debian_rootfs/usr/share/zoneinfo/localtime	
debian_rootfs/usr/share/zoneinfo/Poland	
debian_rootfs/mnt/	
alinx@ubuntu:~/work\$ cd debian_rootfs	
alinx@ubuntu:~/work/debian rootfs\$	

4) コマンドウィンドウで、sudo rsync -av .// media / alinx / EXT (/ media / alinx / EXT を入力 し、これは SD カードの EXT4 パーティションのパスである。異なる場合があり、実際の状況に応じて変更し てください。現在のディレクトリを SD カードの EXT パーティションのルートディレクトリに同期し始め、 同期には数十分かかる場合がある。

var/log/dpkg.log var/log/faillog
var/log/faillog
var/log/fontconfig.log
var/log/lastlog
var/log/wtmp
var/log/apt/
var/log/apt/history.log
var/log/apt/term.log
var/log/fsck/
var/log/tsck/checkts
var/log/tsck/checkroot
var/log/ntpstats/
var/mail/
var/opt/
var/spool/
var/spool/mail >/mail
var/spool/cron/
var/spool/cron/crontabs/
var/spool/rsyslog/
var/tmp/
sent 1,117,338,469 bytes received 809,451 bytes 5,414,759.90 bytes/sec
total size is 1,114,115,109 speedu <u>p is 1.00</u>
alinx@ubuntu:~/work/debian_rootfs\$ sudo rsync -av ./ /media/alinx/EXT

5) コマンドプロンプトがコマンドラインに再表示されると、同期が終了する意味である。

6) BOOT. bin と iamge. ub を SD の FAT32 パーティション (最初のパーティション) にコピーし、開発ボードの sd モードを開始に設定し、HDMI ディスプレイを差し込み、開発ボードを開始する。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする

	FRIM decir	n 1 wrapper	image ub	rootfr coio	rootfs coin at	rootfr cnie
10001	LEIN GESIG	bit	image.ub	10003.000	Toours.cpro.gz	bool
1 18 191	6		* Ch 101 101		harm	
rootfs	.ext4 ro	otfs.ext4.gz	rootfs.jffs2	rootfs.manifest	rootfs.tar.gz	system.
**** ****	0	4	0	* Ch.	4	
u-boo	t.bin	u-boot.elf	vmlinux	zimage	zynq_fsbl.elf	

7) SD カードを作成したら、できた SD カードを開発ボードの SD カードスロットに挿入する。 USB シリ アルケーブルを接続、HDMI ディスプレイを接続して、ボードの電源を入れた後、Debian オペレーティング システムのインターフェースが HDMI ディスプレイに表示される。さらに、シリアルポートツールでは、オ ペレーティングシステムを起動するプロセスも確認できる。u-boot を実行後、Linux の実行を開始する。ア カウント:root、パスワード:root。



8) システムに入ったら、ifconfigコマンドを使用してネットワーク接続を確認できる。



#### root@zynq:~# ifconfig

- eth0 Link encap:Ethernet HWaddr 00:0a:35:00:1e:53 inet addr:192.168.1.46 Bcast:192.168.1.255 Mask:255.255.255.0 inet6 addr: fe80::20a:35ff:fe00:1e53/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:119 errors:0 dropped:0 overruns:0 frame:0 TX packets:86 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:12231 (11.9 KiB) TX bytes:7420 (7.2 KiB) Interrupt:29 Base address:0xb000
- eth1 Link encap:Ethernet HWaddr 00:0a:35:00:03:22 inet addr:192.168.1.62 Bcast:192.168.1.255 Mask:255.255.255.0 inet6 addr: fe80::20a:35ff:fe00:322/64 Scope:Link UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:26 errors:0 dropped:0 overruns:0 frame:0 TX packets:8 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:2771 (2.7 KiB) TX bytes:1184 (1.1 KiB)
- Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1/128 Scope:Host UP LOOPBACK RUNNING MTU:65536 Metric:1 RX packets:236 errors:0 dropped:0 overruns:0 frame:0 TX packets:236 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1 RX bytes:18960 (18.5 KiB) TX bytes:18960 (18.5 KiB)

ホームページ: <u>http://www.csun.co.jp</u> メール: info@csun.co.jp



#### 第二十三章 QSPI Flash から起動の Linux の作成

前のマニュアルで説明した Linux はすべて SD で開始するが、この実験では、Petalinux を使用して QSPI Flash から起動の Linux を作成する方法について説明する。

ここでは、ボード上の QSPI Flash サイズは 32M バイトだが、ZYNQ チップは 16MB しか使用できないため、 ZYNQ 自体によって決定してください。Linux システムは 16MB を超えると使えなくなる。

### 23.1 Petalinux プロジェクトをコピーする

前のマニュアルでは、Petalinux を使用して SD カードのブートのさまざまな実験を行った。SD ブートプロジェクトを保持したいが、新しいプロジェクトを作成したくないため、古いプロジェクトをコピーできる。 1) プロジェクトディレクトリで、Ctrl + Hを同時に押して、シャドウファイルが表示される。



2) project-spec, petalinux、config.project を新しいディレクトリにコピーして、新しい Petalinux プロジェクトにする。





3) PL コンフィギュレーションの BOOT を合成するために、images / linux ディレクトリの bit ファイル も新しいプロジェクトディレクトリにコピーする。



### 23.2 Petal inux のコンパイルとコンフィグ

### 1) 下のコマンドで環境変数を設定する

source /opt/pkg/petalinux/settings.sh



source /opt/Xilinx/Vivado/2017.4/settings64.sh

alinx@ubuntu:~/work/peta/ax7015\_qspi\$ source /opt/pkg/petalinux/settings.sh

alinx@ubuntu:~/work/peta/ax7015\_qspi\$ source /opt/Xilinx/Vivado/2017.4/settings6 4.sh

2) petalinux-config コマンドを使用して petalinux をコンフィグする



3) Subsystem AUTO Hardware Settings ---> Advanced bootable images storage Settings ---> boot image settings ----> image storage media オプションから primary flash を選択する。

Use the arton botkey of the	keys to navigat item you wish t	storage Settings → boot image settin prage media te this window or press the to select followed by the <space< th=""></space<>
BARSOPress	() prim () prim () manu	al information about this mary flash mary sd mal
L	<select></select>	< Help >

4) Subsystem AUTO Hardware Settings ---> Advanced bootable images storage Settings --->kernel image settings ---> image storage media オプションから primary flash を選択する。





5) Subsystem AUTO Hardware Settings →Flash Settings はQSPI Flash のパーティションを変更でき る。デフォルトはほとんど使用可能だが、ファイルサイズがデフォルトのパーティションサイズを超える場 合は、自分で調整する必要がある。



6) Image Packaging Configuration ---> Root filesystem type に INITRAMFS を選択して、RAM タイプ のルートファイルシステムを使用することにより、簡単にパッケージ化して QSPIflash に書き込むことがで きる。



低価格、高品質が不可能? 日昇テクノロジーなら可能にする



7) Firmware Version Configuration --->にHost name 等の情報を修正できる。





\*\*\* Execute 'make' to start the build or try 'make help'. [INFO] sourcing bitbake [INFO] generating plnxtool conf [INFO] generating meta-plnx-generated layer ~/work/peta/ax7015\_qspi/build/misc/plnx-generated ~/work/peta/ax7015\_qspi ~/work/peta/ax7015\_qspi [INFO] generating machine configuration [INFO] generating bbappends for project . This may take time ! ~/work/peta/ax7015\_qspi/build/misc/plnx-generated ~/work/peta/ax7015\_qspi ~/work/peta/ax7015\_qspi [INFO] generating u-boot configuration files [INFO] generating kernel configuration files [INFO] generating kconfig for Rootfs Generate rootfs kconfig [INFO] oldconfig rootfs [INFO] generating petalinux-user-image.bb [INFO] successfully configured project webtalk failed:PetaLinux statistics:extra lines detected:notsent\_nofile! webtalk failed:Failed to get PetaLinux usage statistics! alinx@ubuntu:~/work/peta/ax7015\_qspi\$ petalinux-build

10) 下のコマンドを使用して BOOT を合成する。以前のマニュアルとの違いは、-kernel オプションが追加され、カーネルを BOOT. bin ファイルにパッケージ化することである。

petalinux-package --boot --fsbl ./images/linux/zynq\_fsb1.elf --fpga --u-boot --kernel -force

alinx@ubuntu:~/Downloads/peta\_prj/ax\_peta\$ petalinux-package --boot --fsbl ./ima ges/linux/zyng\_fsbl.elf --fpga --�\_boot\_--kernel --force INFO: Getting system flash information... rlwrap: warning: your \$TERM is 'xterm-256color' but rlwrap couldn't find it in t he terminfo database. Expect some problems.: Inappropriate ioctl for device INFO: File in BOOT BIN: "/home/alinx/Downloads/peta\_prj/ax\_peta/images/linux/zyn q\_fsbl.elf INFO: File in BOOT BIN: /home/alinx/Downloads/peta\_prj/ax\_peta/images/linux/des ign\_1\_wrapper.bit INFO: File in BOOT BIN: "Thome/alinx/Downloads/peta\_prj/ax\_peta/images/linux/u-b oot.elf INFO: File in BOOT BINE //home/alinx/Downloads/peta\_prj/ax\_peta/images/linux/ima ge.ub" INFO: Generating zyng binary package BOOT.BIN... INFO: Binary is ready. WARNING: Unable to access the TFTPBOOT folder /tftpboot!!! WARNING: Skip file copy to TFTPBOOT folder!!! webtalk failed:Invalid tool in the statistics file:petalinux-yocto! webtalk failed Failed to get PetaLinux usage statistics!

11) 第十三章を参照して、BOOT. bin を QSPI flash に書き込むことができる。

12) 起動モードを QSPI に調整する

以上。