



**THE IMAGINATION UNIVERSITY PROGRAMME**

# **RVfpga-SoC インストールガイド**

## 目次

1. はじめに	3
2. ラボ1用インストール :	4
3. ラボ2用インストール	6
4. ラボ3用インストール	8
5. ラボ4用インストール :	9
6. 付録A : ドライバをWindowsでインストールして、PlatformIOを使用する	11
7. 付録B : WindowsでのVerilatorおよびGTKWaveのインストール	13



## 1. はじめに

本ガイドには、Ubuntu 18.04オペレーティングシステム（OS）でRVfpga-SoCに必要なツールおよびハードウェアをインストール/設置する方法が、示されています。以下の手順はUbuntu 18.04 OS用ですが、他のLinuxオペレーティングシステムやWindowsオペレーティングシステムも同様のステップに従います（まったく同じではないとしても）。場合によっては、Windows OS用の特定の手順が含まれているボックスを挿入することになります。Ubuntuを使用している場合は、これらのボックスを無視してください。

このプロセスには数時間かかることがあります（または数時間以上、ダウンロード速度によって異なる）、この時間の大部分は、プログラムがダウンロードおよびインストールされる間の待機に費やされます。

表1にRVfpga-SoCに必要なソフトウェアおよびハードウェアがリストされています。

**表1：RVfpga-SoCに必要なソフトウェアおよびハードウェア**

ソフトウェア		
名前	Webサイト	コスト
Vivado 2019.2 WebPACK	<a href="https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2019-2.html">https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2019-2.html</a>	無償
VSCode	<a href="https://code.visualstudio.com/Download">https://code.visualstudio.com/Download</a>	無償
PlatformIO	<a href="https://platformio.org/">https://platformio.org/</a> VSCode内にインストール済み	無償
Verilator（HDLシミュレータ）およびGTKWave	<a href="https://github.com/verilator/verilator">https://github.com/verilator/verilator</a> <a href="http://gtkwave.sourceforge.net/">http://gtkwave.sourceforge.net/</a>	無償
FuseSoC	<a href="https://github.com/olofk/fusesoc">https://github.com/olofk/fusesoc</a>	無償
RISC-V ToolchainおよびOpenOCD	<a href="https://github.com/riscv/riscv-gnu-toolchain">https://github.com/riscv/riscv-gnu-toolchain</a> <a href="https://github.com/riscv/riscv-openocd">https://github.com/riscv/riscv-openocd</a> PlatformIO内にインストール済み	無償
Zephyrプロジェクト	<a href="https://github.com/zephyrproject-rtos/zephyr">https://github.com/zephyrproject-rtos/zephyr</a>	無償
ハードウェア*		
名前	Webサイト	コスト
Nexys A7 FPGAボード*	<a href="https://store.digilentinc.com/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/">https://store.digilentinc.com/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/</a>	\$265 （学生・教師用 価格： \$199）
RISC-Vコアおよびシステムオンチップ（SoC）**		
名前	Webサイト	コスト
Western Digital製 SweRV EH1コア	<a href="https://github.com/chipsalliance/Cores-SweRV">https://github.com/chipsalliance/Cores-SweRV</a>	無償
SweRVolf	<a href="https://github.com/chipsalliance/Cores-SweRVolf">https://github.com/chipsalliance/Cores-SweRVolf</a>	無償

\* ハードウェアはオプションです。

\*\* SweRV EH1コアおよびSweRVolfはRVfpga-SoCパッケージの一部として提供されます。

## 2. ラボ1用インストール :

このセクションでは、RVfpga-SoCコースのラボ1の実行に必要なソフトウェアのインストール方法を示します。

### 1. Vivadoをインストール :

Vivadoは、Verilogコードを表示、変更、合成するためのXilinxツールです。これは後のラボで大規模に使用することになります。インストール手順は

<https://reference.digilentinc.com/vivado/installing-vivado/start>で入手でき、以下に要約されています。

**Windows :** 上記に参照されているWebページ

(<https://reference.digilentinc.com/vivado/installing-vivado/start>) にも、WindowsでのVivadoのインストールの詳細な手順が含まれています。Windows用に特定の命令が必要な場合は、以下にボックスが挿入されています。

**ステップ1 :** <https://reference.digilentinc.com/vivado/installing-vivado/start>に移動します。

**ステップ2 :** Xilinxのダウンロードページ<https://www.xilinx.com/support/download.html>へガイドされます。

**ステップ3 :** 「自己抽出Webインストーラ」をインストールすることを推奨します。これは、このドキュメントを書き込むときに、次記のダウンロードページにあります。  
[Xilinx Unified Installer 2019.2 : Linux自己抽出Webインストーラ](#)。

**WINDOWS :** このドキュメントを書き込むときに、Windows用「自己抽出Webインストーラ」はダウンロードページの次記のリンクにあります。[Xilinx Unified Installer 2019.2 : Windows自己抽出Webインストーラ](#)。

**ステップ4 :** インストーラをダウンロードする前に、お使いのXilinxアカウントにログインするように、求められます。まだアカウントがない場合は、これを作成する必要があります。

**ステップ5 :** バイナリファイルを実行します。ターミナルを開き、これをルートにします (「sudo su」を入力)。バイナリファイル (Xilinx\_Unified\_2019.2\_1106\_2127\_Lin64.bin) をターミナルにドラッグします。このファイルを実行可能にして使用するようプロンプトで指示されたら、OKを選択します。

**トラブルシューティング :** 許可が拒否されたらターミナルに言われたら、ターミナル (バイナリファイルと同じディレクトリにある) に以下を入力します。

- `sudo chmod +x ./Xilinx_Unified_2019.2_1106_2127_Lin64.bin`
- `sudo ./Xilinx_Unified_2019.2_1106_2127_Lin64.bin`

**WINDOWS :** Windowsでは、ステップ3および4でダウンロードしたexeファイルをダブルクリックするだけで、これを実行できます。

**ステップ6 :** Vivadoインストーラを使用して、インストールプロセスを通り抜けます。  
重要な注記 :

- **Vivado** (Vitisではなく) をインストールする製品として選択します。
- Vivado HL **Webpack** (Vivado HL System Editionではなく) を選択します。  
Webpackは無償です。
- そうしない場合は、デフォルトを選択する必要があります。

**ヒント** : Vivadoのインストールディレクトリを変更する場合は、パスを以下のステップで適切に変更する必要があります。

**WINDOWS** : Windowsではステップ7および8は必要ありません。これらの2つのステップを無視して直接ステップ9に進むだけです。

**ステップ7** : Vivadoをインストールした後、環境をセットアップする必要があります。ターミナルを開いて次記を入力します。

```
> source /tools/Xilinx/Vivado2019.2/settings64.sh
```

その行 (source /tools/Xilinx/Vivado2019.2/settings64.sh) を ~/.bashrc ファイルに追加します。これにより、ターミナルを起動するたびに、実行されます。

**ステップ8** : 以下をターミナルに入力して、Vivadoをテストします。

```
> vivado
```

**トラブルシューティング** :

- 実行可能ファイルをシステムによって見つけることができない場合は、使用するパスに以下を追加する必要があります。

```
/tools/Xilinx/DocNav  
/tools/Xilinx/Vivado/2019.2/bin
```

- 「application-specific initialization failed…」 (アプリケーション固有の初期化に失敗しました...) などのエラーが表示されたら、ターミナルで以下を入力します。

```
> sudo ln -s /lib/x86_64-linux-gnu/libtinfo.so.6 /lib/  
x86_64-linux-gnu/libtinfo.so.5
```

## 2. ケーブルドライバのインストール :

**ステップ9** : Nexys A7 FPGAボードのケーブルドライバを手動でインストールする必要があります。ターミナルウィンドウで下記を入力します。

```
cd /tools/Xilinx/Vivado/2019.2/data/xicom/cable_drivers/lin64/  
install_script/install_drivers/
```

```
sudo ./install_drivers
```

**WINDOWS** : PlatformIOに適合しない、Nexys A7ボード用ドライバが、WindowsでのVivadoインストールによって、自動的にインストールされます。Windowsを使用している場合、ドライバを、インストールガイドの付録Aに説明されているように更新する必要があります。

## 3. Digilentボードファイルをインストール :

Digilentボードファイルも手動でインストールする必要があります。

**ステップ10** : Githubリポジトリからvivadoボードの [アーカイブ](#) をダウンロードし、これを抽出します。

**ステップ11** : アーカイブから抽出したフォルダを開いて、new/board\_filesディレクトリに移動します。このディレクトリ内のすべてのフォルダを選択して、それらをコピーします。

**ステップ12** : Vivadoがインストールされていた先のフォルダを開きます (デフォルトでは/tools/Xilinx/Vivado) 。このフォルダの下で <version>/data/boards/board\_filesディレクトリに移動し、ボードファイルをこのディレクトリに貼り付けます。

**ステップ13** : `new/board_files`ディレクトリに移動して次記を入力することにより、このターミナルを使用することもできます。

```
> sudo cp -r * /tools/Xilinx/Vivado/2019.2/data/boards/board_files
```

**WINDOWS** : ステップ10で説明されているように、ダウンロードしたフォルダをコピー/貼り付けします。Windowsでは、Vivadoの`board_files`ファイルは次記にあります。  
`C:\Xilinx\Vivado\2019.2\data\boards\board_files`

### 3. ラボ2用インストール

このセクションでは、RVfpga-SoCコースのラボ2の実行に必要なソフトウェアのインストール方法を示します。このセクションには、Visual Studio Code (VSCode)、PlatformIO、Verilator、GTKWaveなどのソフトウェアのインストール手順が含まれています。

#### 1. VSCodeをインストール :

以下のステップに従って、VSCodeをインストールします。

**ステップ1** : 次記のリンクから`.deb`ファイルをダウンロードします。

<https://code.visualstudio.com/Download>

**ステップ2** : ターミナルを開き、ターミナルに以下を入力して、VSCodeをインストールして実行します。

```
> cd ~/Downloads
> sudo dpkg -i code*.deb
> code
```

**Windows** : VSCodeパッケージは、Windows (.exeファイル) 用に<https://code.visualstudio.com/Download>でも使用できます。これらのオペレーティングシステムでのアプリケーションのインストールおよび実行に使用される一般的なステップに従います。

#### 2. VSCodeに加えてPlatformIOをインストール :

PlatformIOは、MicrosoftのVisual Studio (VS) コードに加えて構築される内蔵システム用の統合開発環境 (IDE) です。これにより、C言語またはアセンブリ言語を使用して、RISC-Vプロセッサ (FPGAに配置されている) を、プログラムできます。PlatformIOはクロスプラットフォームであり、内蔵されたデバッガが含まれています。


以下のステップに従ってPlatformIOをインストールします。

**ステップ3** : 以下をターミナルに入力して、`python3`ユーティリティをインストールします。

```
> sudo apt install -y python3-distutils python3-venv
```

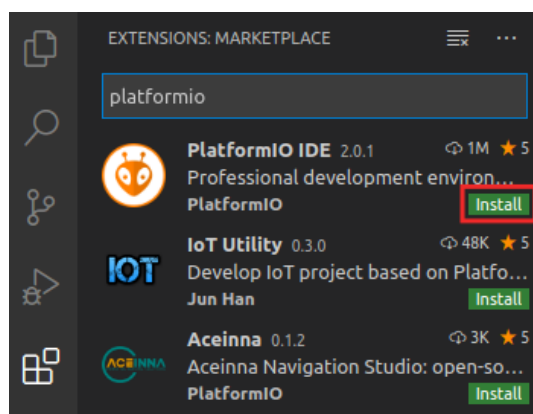
**Windows** : ステップ3はWindowsでは必要ありません。

**ステップ4** : まだ開いていない場合は、Startボタンを選択して検索メニューに「VSCode」を入力してからVSCodeを選択するか、ターミナルで`code`を入力して、VSCodeを起動します。

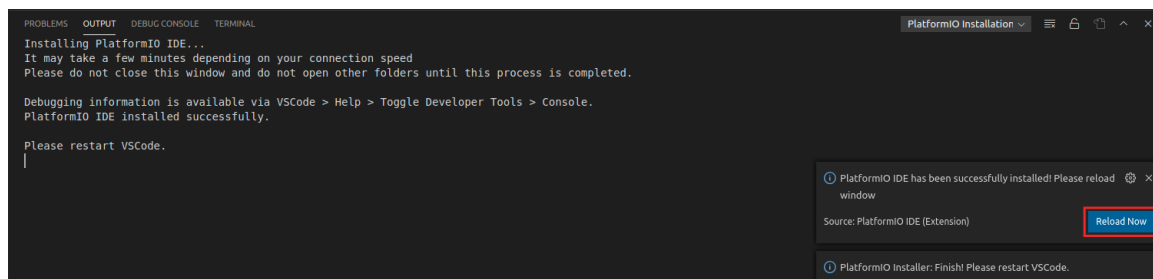
**ステップ5** : VSCodeで、VSCodeの左サイドバーにあるExtensionsアイコン (図1を参照) をクリックします。

**図1 : VSCodeのExtensionsアイコン**

**ステップ6** : 検索ボックスにPlatformIOを入力し、その横のinstallボタン（図2を参照）をクリックして、PlatformIO IDEをインストールします。

**図2 : PlatformIO IDE Extension**

**ステップ7** : 下部のOUTPUTウィンドウにより、インストールプロセスについて通知されます。完了すると、右下のウィンドウ「Reload Now」をクリックします。すると、PlatformIOがVSCode内にインストールされます（図3を参照）。

**図3 : PlatformIOインストール後のReload Now**

### 3. GTKWaveのインストール :

次のステップに従って、GTKWaveをUbuntu 18.04 Linuxシステムにインストールします。使用するUbuntuターミナルを開き、以下のコマンドを入力します。

- `sudo apt-get install git make autoconf g++ flex bison libfl2 libfl-dev`
- `sudo apt-get install -y gtkwave`

**Windows** : Windows用GTKWaveのインストールについては、付録Bを参照してください。

#### 4. Verilatorをインストールする :

次のステップに従ってVerilatorをインストールします（手順は<https://www.veripool.org/projects/verilator/wiki/Installing>で入手できますが、以下にも要約されています）。

- `git clone https://git.veripool.org/git/verilator`
- `cd verilator`
- `git pull`
- `git checkout v4.106`
- `autoconf`
- `./configure`
- `Make`（またはより速く進めるために`make -j$(nproc)`を使用できます）
- `sudo make install`
- `export PATH=$PATH:/usr/local/bin`（使用するシステムでパスを変更します）

使用するパスに`/usr/local/bin`を恒久的に追加するには、最終行を使用する`~/.bashrc`ファイルに追加します。

**Windows** : Windows用Verilatorのインストールについては、付録Bを参照してください。

## 4. ラボ3用インストール

ラボ3以降のインストール手順は、特にUbuntu 18.04オペレーティングシステム用です。Windows 10ユーザーは、[Linux用Windowsサブシステム](#)を使用するラボのシミュレーション部分を実行できます。この手順は、他のあらゆる最新バージョンのUbuntuに対して、同様に機能します。これらの後続するラボすべてをUbuntuオペレーティングシステムで実行することを、強くお勧めします。

このセクションでは、RVfpga-SoCコースのラボ3の実行に必要なソフトウェアのインストール方法を示します。

#### 1. pipをインストール :

「**pip**」は「FuseSoC」および「west」のインストールに必要です。Ubuntuターミナルを開き、以下のコマンドを入力します。

- `sudo apt install python3-pip`

#### 2. pyelftoolsをインストール :

pyelftoolsはwestの構築に必要です。pyelftoolsは以下の2つの方法でインストールできます。

- `pip3 install pyelftools`  
または
- `sudo apt-get install -y python3-pyelftools python-pyelftools`



### 3. FuseSoCをインストール :

現在の安定バージョンのFuseSoCをインストールするには、ターミナルウィンドウを開き、以下のコマンドを実行します。使用するシステムに古いバージョンのFuseSoCが見つかった場合、このバージョンは最新の安定リリースにアップグレードされます。

```
> sudo pip3 install --upgrade fusesoc
```

### 4. OpenOCDをインストール :

OpenOCDはオープンオンチップデバッガであり、これによって内蔵されている対象デバイスを、プログラムおよびデバッグできます。次のステップに従って、RISC-V OpenOCDをコンピュータにインストールします。

**ステップ1 :** 「apt-get」を使用して、次記の必要な依存性をインストールします。

```
> sudo apt-get install libusb-1.*
> sudo apt-get install pkg-config
```

**ステップ2 :** riscv-openocd githubリポジトリをクローンします。

```
> git clone https://github.com/riscv/riscv-openocd.git
> cd riscv-openocd
> ./bootstrap
```

**注意 :** コマンドが見つからないエラーが発生した場合は、以下のコマンドの実行を試行してください。これにより、次記の別の依存性がダウンロードされます。

```
sudo apt-get install libtool
```

**ステップ3 :** ユーザスペースUSBプログラミングライブラリ開発ファイルをダウンロードしてインストールします。

```
> sudo apt-get install libusb-1.0-0-dev
```

**ステップ4 :** OpenOCDを接続できる先のJTAGサーバを設定します。

```
> ./configure --enable-jtag_vpi --enable-ftdi
> make
> sudo make install
```

## 5. ラボ4用インストール :

このセクションでは、RVfpga-SoCコースのラボ4の実行に必要なソフトウェアのインストール方法を示します。

使用するUbuntuターミナルを開き、以下のコマンドを入力します。

### 1. 必要条件および依存性 :

**ステップ1 :** 以下を使用してリポジトリを更新します。

```
> sudo apt update
> sudo apt upgrade
```

**ステップ2 :** 「apt」を使用して、次記の必要な依存性をインストールします。

```
> sudo apt install --no-install-recommends git cmake ninja-
  build gperf \
```

- > ccache dfu-util device-tree-compiler wget \
- > python3-dev python3-pip python3-setuptools python3-tk  
python3-wheel xz-utils file \
- > make gcc gcc-multilib g++-multilib libsdl2-dev

## 2. \*cmake\*のバージョンを3.13.1以上に更新：

[Kitwareサードパーティaptリポジトリ](#)を追加する手順に従って、aptを使用するcmakeの更新バージョンを取得できます。

**ステップ3：**以下のコマンドを入力して、cmakeをダウンロードしてインストール/更新します。

- > wget -O - https://apt.kitware.com/keys/kitware-archive-latest.asc 2>/dev/null |  
sudo apt-key add -
- > sudo apt-add-repository 'deb https://apt.kitware.com/  
ubuntu/ bionic main'
- > sudo apt update
- > sudo apt install cmake

## 3. westをインストール：

**ステップ4：**westをインストールし、使用するPATH環境変数に~/local/binがあることを、確認します。

- > pip3 install --user -U west
- > echo 'export PATH=~/local/bin:"\$PATH"' >> ~/.bashrc
- > source ~/.bashrc

## 4. Zephyr SDKをインストール：

Zephyrソフトウェア開発キット（SDK）には、Zephyrのサポートされているアーキテクチャそれぞれ用のツールチェーンが、含まれています。これには、カスタムQEMUバイナリ、ホストコンパイラなどの追加のホストツールも、含まれています。Zephyr SDKバージョン0.12.4をインストールします

使用するホームディレクトリに以下のコマンドを入力します。

**ステップ5：** [0.12.4バージョンのSDKインストーラ](#)をダウンロードします：

- > Wget https://github.com/zephyrproject-rtos/sdk-ng/releases/  
download/v0.12.4/zephyr-sdk-0.12.4-x86\_64-linux-setup.run

**ステップ6：**インストーラを実行し、SDKを~/zephyr-sdk-0.12.4にインストールします。

- > chmod +x zephyr-sdk-0.12.4-x86\_64-linux-setup.run
- > ./zephyr-sdk-0.12.4-x86\_64-linux-setup.run -- -d ~/zephyr-sdk-  
0.12.4

Zephyr SDKをこれらの場所の外にインストールする場合は、次記を読んでください：  
[Zephyrソフトウェア開発キット（SDK）のインストール](#)。SDKをインストールした後は、このディレクトリを移動できません。

## 5. PuTTYをインストール：

ステップ7：以下のコマンドを入力して、PuTTYをインストールします。

```
➤ sudo apt-get install -y putty
```

## 6. 付録A：ドライバをWindowsでインストールして、PlatformIOを使用する

実行可能なZadigをダウンロードするには、次記のWebサイトを参照します（図4を参照）：

<https://zadig.akeo.ie/>

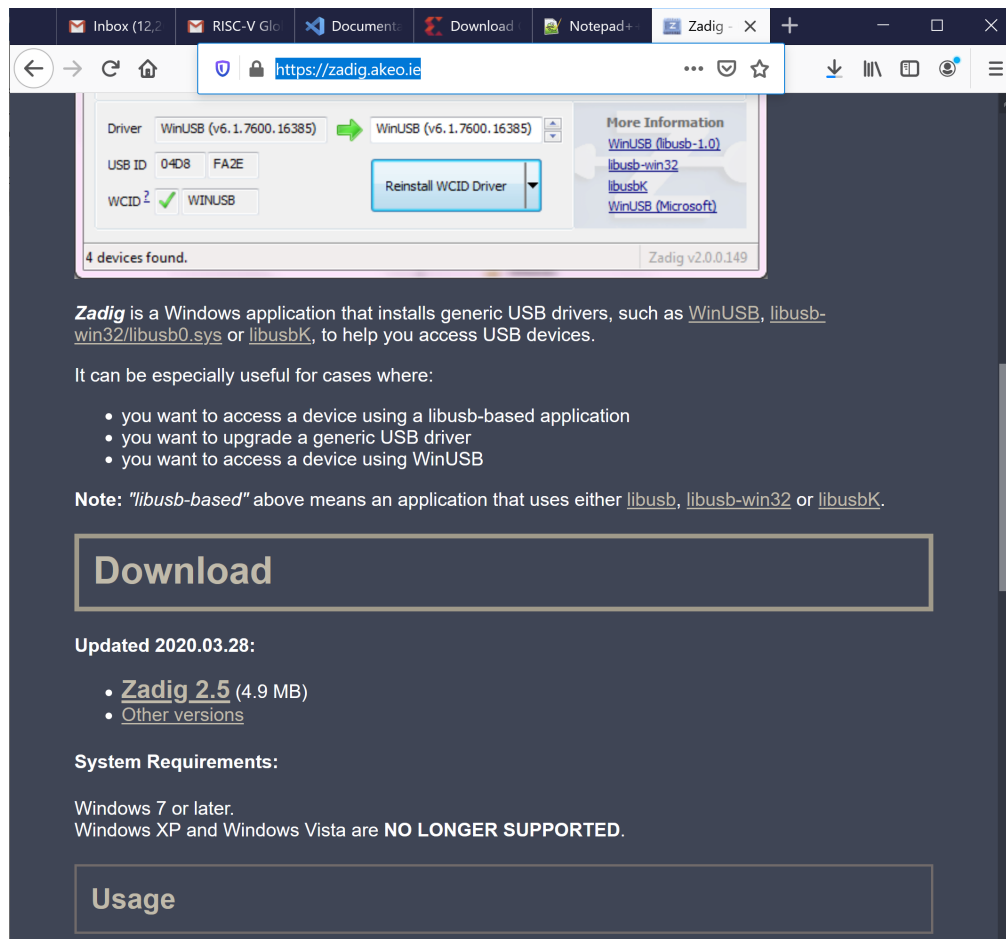


図4：PlatformIOによって使用されるNexys A7ボードドライバをインストールする

Zadig 2.5をクリックして、実行可能ファイルを保存します。これ（zadig-2.5.exe）を実行します（ダウンロードした場所にあります）。zadigをStartメニューに入力してこれを見つけることもできます。Zadigがコンピュータを変更することを許可するかや、これによって更新を確認するかを、多分尋ねられるでしょう。いずれの場合もYesをクリックします。

Nexys A7ボードを、使用するコンピュータに接続し、電源を入れます。Zadigで、Options → List All Devicesをクリックします（図5を参照）。

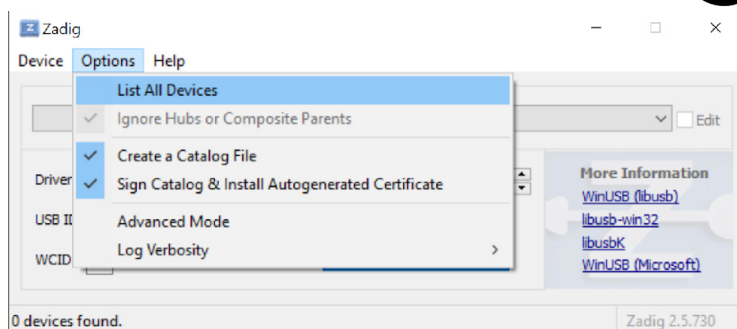


図5 : Zadigのすべてのデバイスを一覧表示

ドロップダウンメニューをクリックすると、Digilent USBデバイス（インターフェース0）およびDigilent USBデバイス（インターフェース1）がリストされて表示されます。新しいドライバをDigilent USBデバイス（インターフェース0）専用インストールします（図6を参照）。

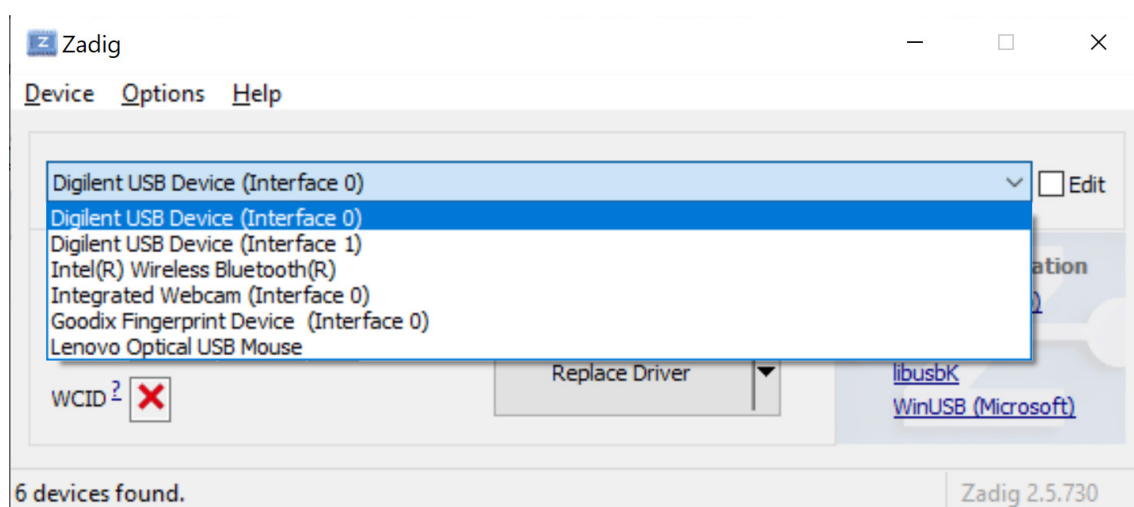


図6 : WinUSBドライバをDigilent USBデバイス（インターフェース0）用にインストールする

図7に示されているように、FTDIドライバをWinUSBドライバと交換します。Digilent USBデバイス（インターフェース0）用Replace Driver（またはInstall Driver）をクリックします。Nexys A7ボード用ドライバをインストールしようとしているか、以前にVivadoをインストールした場合に、Vivadoによって使用されるFTDIドライバをPlatformIOによって使用されるWinUSBドライバと交換しようとしています。

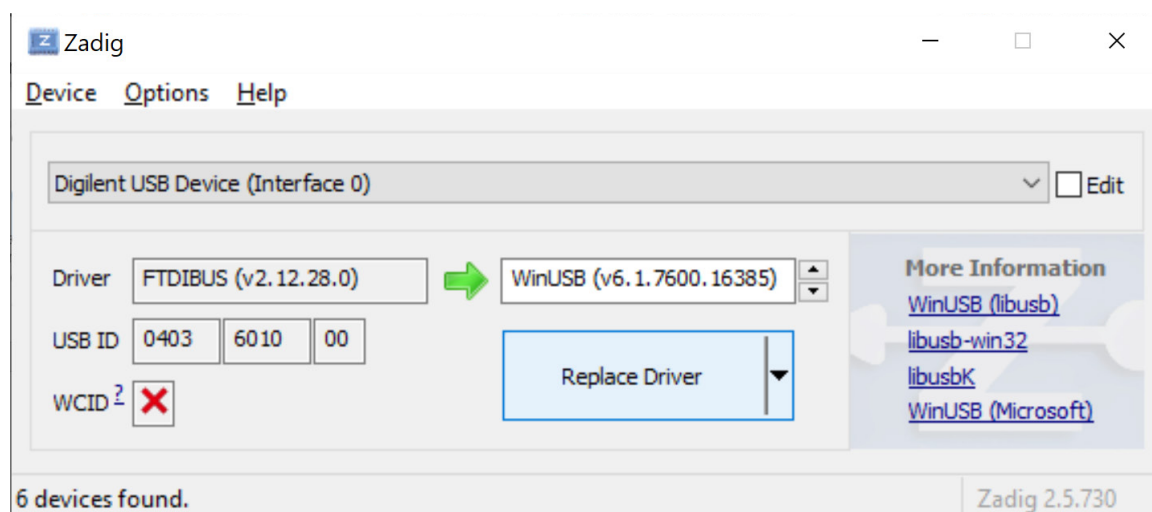


図7 : Nexys A7ボード用ドライバの交換

しばらくした後（通常数分間）、ドライバが適切にインストールされたことが、Zadigによって示されます。Closeをクリックすると、Zadigウィンドウが閉じます。

次にPlatformIOを使用するとき、ドライバを再インストールする必要はありません。ただし、このドライバはWindowsでVivadoに適合しないことに、注意してください。

## 7. 付録B : WindowsでのVerilatorおよびGTKWaveのインストール

このセクションでは、Windows 10でVerilatorおよびGTKWaveをインストールする方法を、説明します。Windowsでは、VerilatorのインストールにCygwinを使用する必要があります。このため、このプログラミング/実行環境のインストール方法を最初に説明します。

### 1. Cygwinのインストール :

そのWebページ (<https://www.cygwin.com>) で説明されているように、CygwinはGNUおよびオープンソースのツールで構成されます。これらにより、WindowsでLinuxディストリビューションと同様の機能が提供されます。次のステップに従って、CygwinをWindows 10でインストールします。

1. インストールのWebページ (<https://cygwin.com/install.html>) に移動して、`setup-x86_64.exe`と呼ばれるインストールファイルをダウンロードします（図8）。

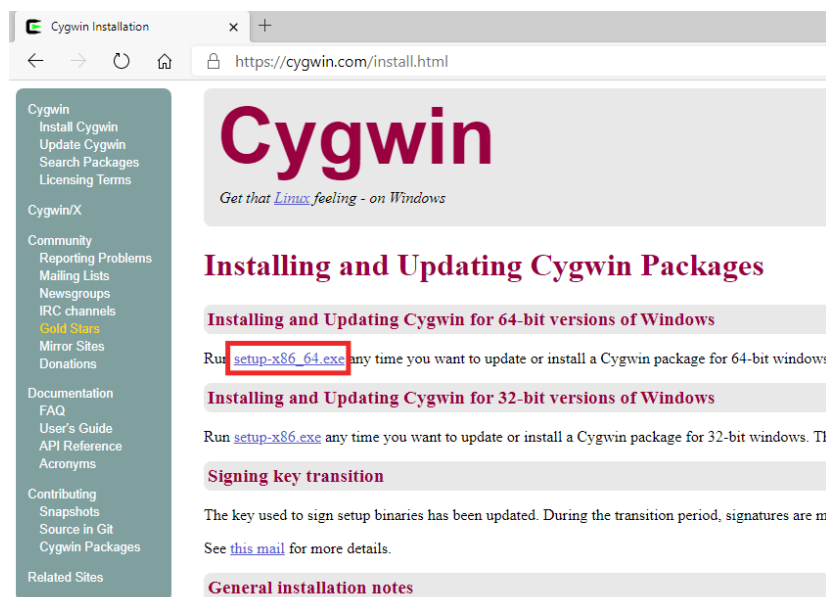


図8 : CygwinのインストールのWebページ

2. 使用するマシンでこれをダブルクリックして、セットアップファイルを実行します（図9）。Nextを数回クリックして、デフォルトのオプションを維持します。インストーラによってダウンロードサイトを選択するように要求され（図10）、任意の1つを選択できます。

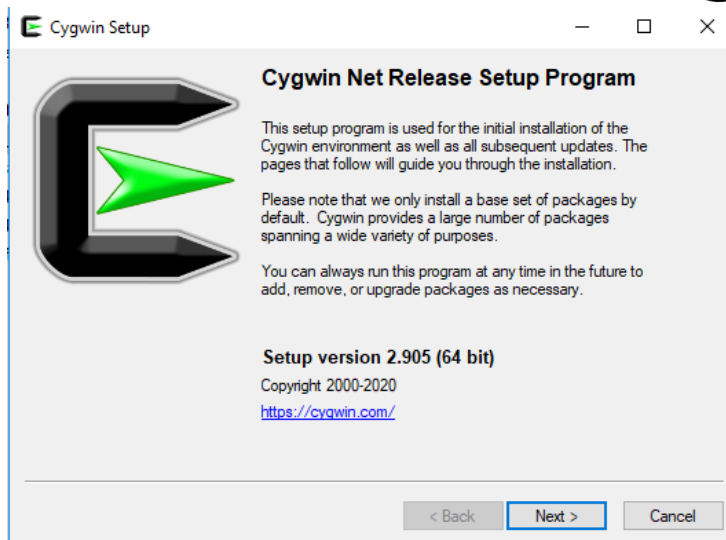


図9 : Cygwinインストールウィンドウ

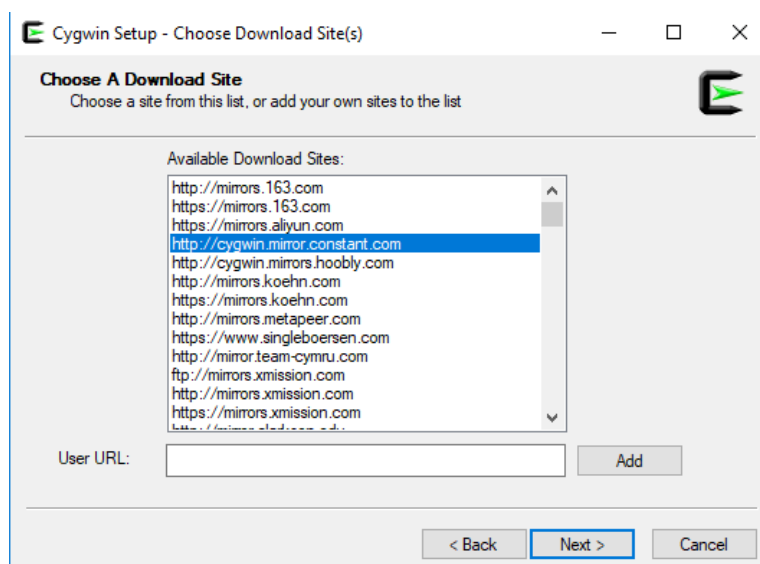


図10 : ダウンロードサイトの選択

3. 数ステップの後、**Select Packages**ウィンドウに到達します（図11）。図11に示されているように、**Fullビュー**を選択します。

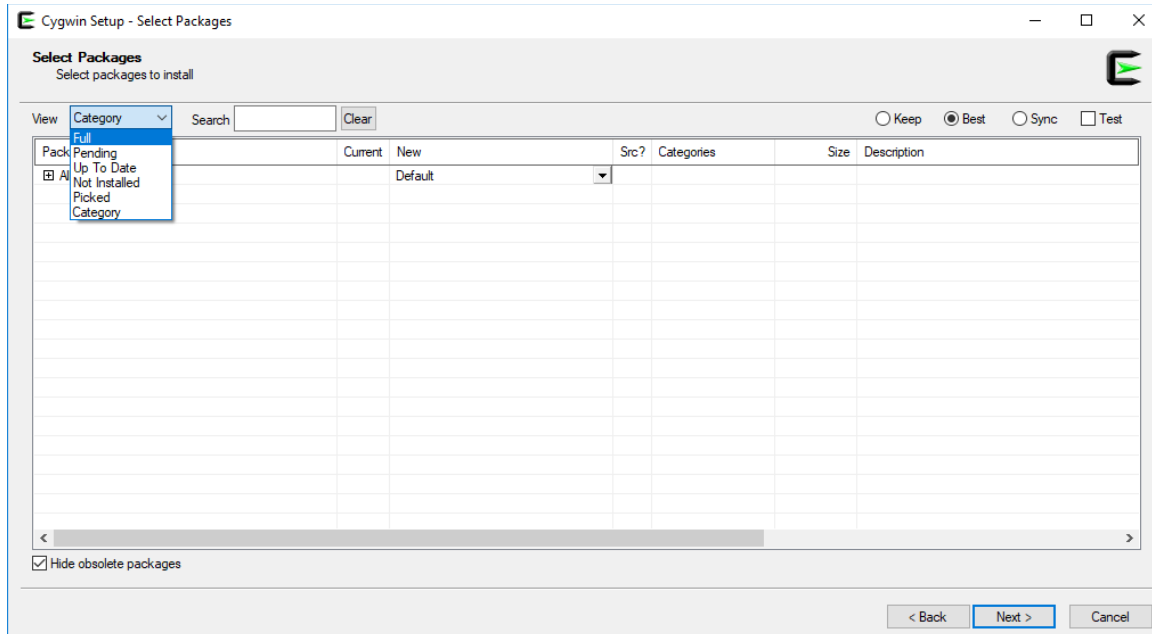


図11 : Select Packagesウィンドウ

4. インストールできるパッケージのリストが表示されます（図12）。**Search**ボックスで、インストールする特定のパッケージを、選択します。

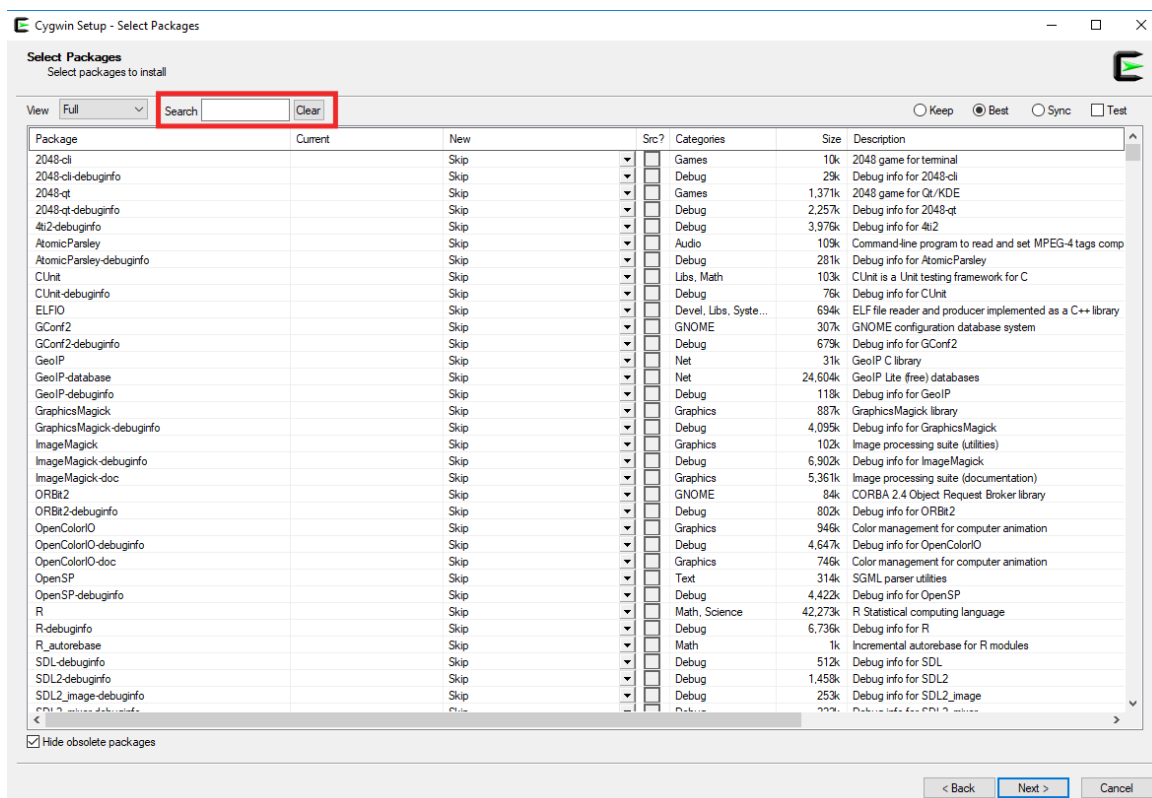


図12 : Select Packagesウィンドウ - 全体表示

Verilatorをコンパイルして新しいシミュレータバイナリを生成できるためには、以下のパッケージをインストールする必要があります。

- git
- make



- autoconf
- gcc-core
- gcc-g++
- flex
- bison
- perl
- libargp-devel

使用するCygwinインストールに、少なくともこれらのパッケージを含めます。以下のステップに従って、これらを1つずつ選択します（リストgitの最初のパッケージの詳細ステップのみを示します。このプロセスは他のパッケージでも同じです）。

- **Search**ボックスでgitパッケージ探します（図13）。

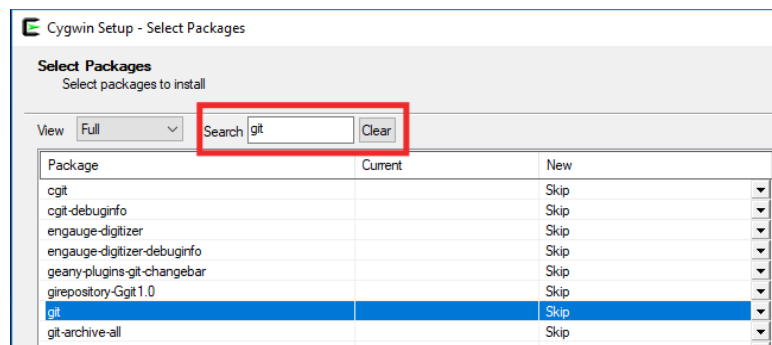


図13：gitパッケージを探す

- 最新バージョンをドロップダウンメニューで選択してからボックスにチェックマークを付けます（図14）。



図14：最新バージョンを選択してからボックスにチェックマークを付ける

- 上記リストの残りのパッケージに、同じことをします。
5. 9つのパッケージを選択したら、後続するウィンドウで**Next**をクリックし、これらのパッケージをCygwinインストールに含めて（インストールプロセス（図15を参照）には数分間かかることがあります）、Finishをクリックしてインストールを終了します（図16を参照）。



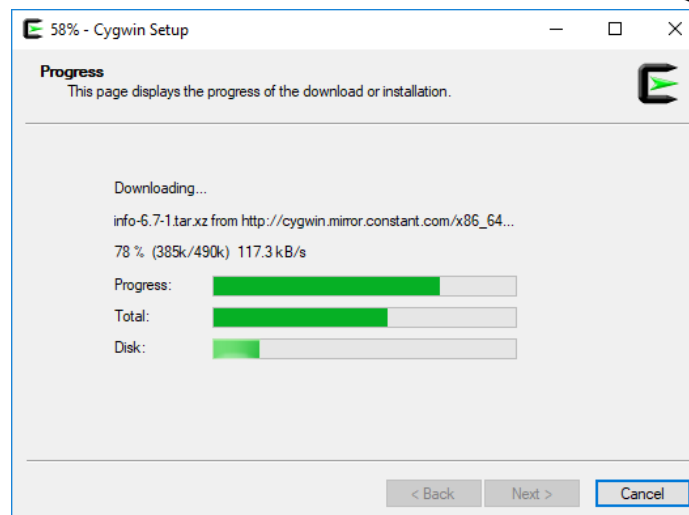


図15 : Cygwinのセットアップ

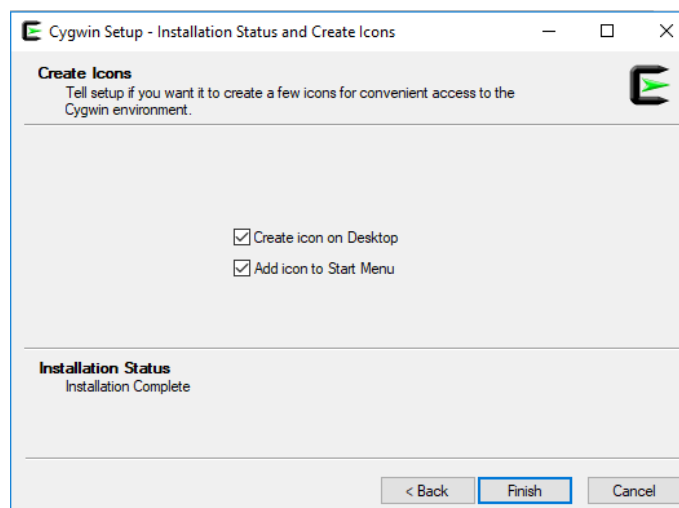


図16 : インストールを終了

6. パッケージをCygwinインストールに追加する必要がある場合は、そのパッケージに対してステップ2～5を繰り返します。

## 2. WindowsでのVerilatorのインストール :

次のステップに従って、Windows 10でVerilatorをインストールします。

1. 使用するWindowsデスクトップで利用できるまたはStartメニューから、Cygwinターミナル（図17）を開きます。



図17 : Cygwinターミナル

2. これらのステップに従って、Verilatorを構築してインストールします。使用するコンピュータの速度によって異なりますが、これにはある程度時間がかかる（何時間ものさえ）ことがあります。

```
➤ git clone https://git.veripool.org/git/verilator
➤ cd verilator
➤ git pull
➤ git checkout v4.020
➤ autoconf
➤ ./configure
➤ make
➤ make install
```

### 3. WindowsでのGTKWaveのインストール :

GTKWaveは、事前にコンパイルしたパッケージとして、  
<https://sourceforge.net/projects/gtkwave/files/>からダウンロードできます。最新のWindowsパッケージ（このドキュメントが書かれた時点では**gtkwave-3.3.100-bin-win64**と呼ばれた）を探してダウンロードし、それを解凍（圧縮から復元）します。フォルダbin内に**gtkwave**と呼ばれる実行可能ファイルがあります。これは、使用するWindowsマシンで実行および使用できます。