

# Zynqberry × ZYNQ 活用セミナー

## 第2章 Zynqberryの使い方

2018年6月13日

特殊電子回路株式会社

# 目次

- ハードウェア
- MIO
- CPLD
- GPIO
- リファレンスデザイン
- サンプルプロジェクト
- Trenz製Debian Linuxの所在
- Trenzスクリプトの使い方
- 最初の書き込み
- Linuxの起動
- Linux上からのROM書き換え
- SDSoCについて
- 作成したFPGAとソフトのROM化

# Zynqberryのハードウェアの特徴①

- XILINX ZYNQ XC7Z010-1CLG225C
  - 割と小規模。より小規模な7S版も最近はある。
- DDR3 SDRAM 512MB
  - ZYNQの最大メモリが1GBなのでMAXの半分
- LAN9514
  - ZYNQのGbEを使わない。USB HUBと兼用
- HDMI出力
  - 標準では720p。1080pもいけるはず。
- PWMによるオーディオ出力

# Zynqberryのハードウェアの特徴②

- DSIコネクタ
- CSIコネクタ
  - CSIを受信するための回路が秀逸
  - CSIを利用するサンプルデザインが提供されている
- マルチプレクサでI2Cを拡張
- アナログ入力(V\_P、V\_N)はオーディオコネクタに接続
- ボードコントロール用にCPLDを搭載
- RTCは非搭載
- ユーザが使えるLEDはない

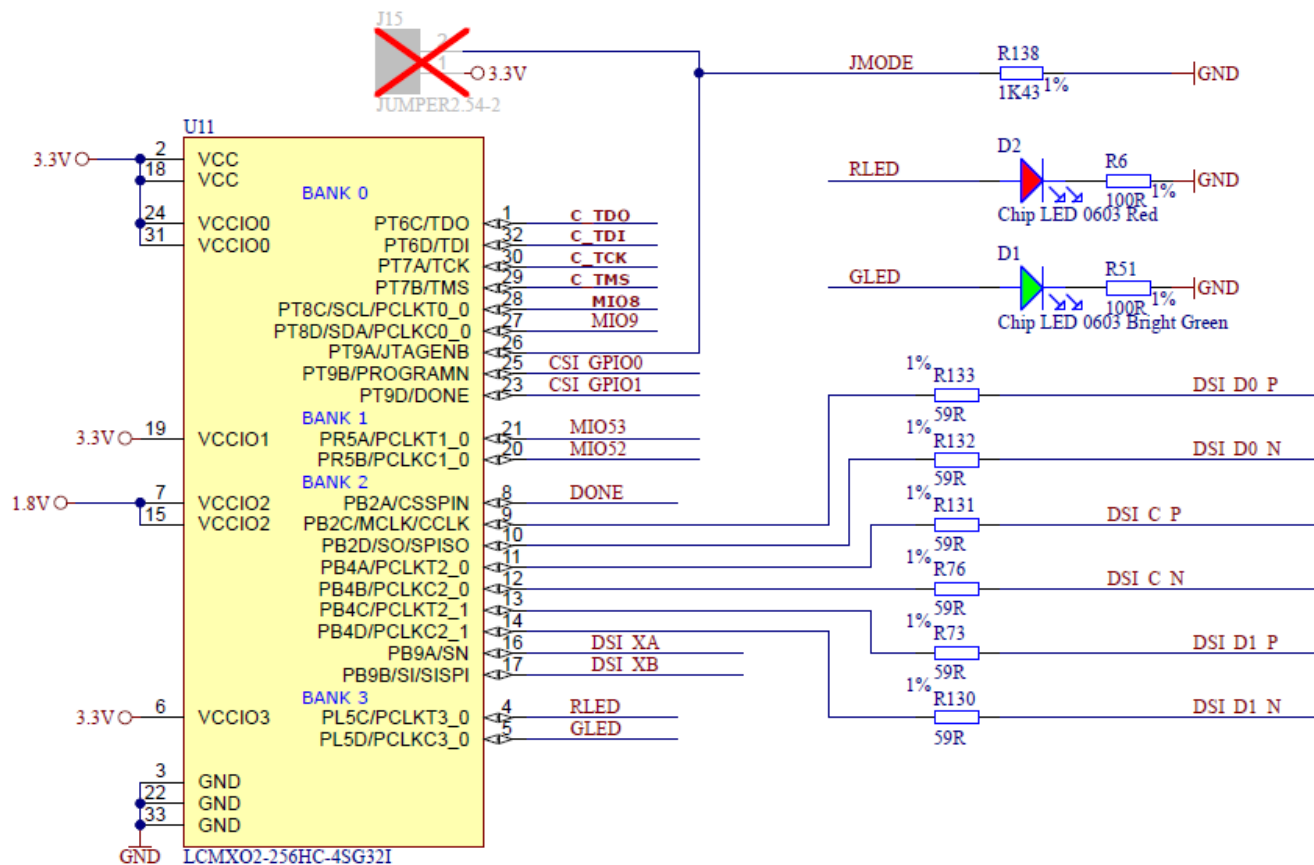
# MIOの割り当て

MIO	機能	
0	I2C MUX	INT信号の受信?
1-6	QSPI FLASH ROM	S25FL127SABMFV10
7	USB RESET	Lでリセット
8-9	UART1	CPLD経由でUSBへ出力
10-15	SD1	
28-39	USB OTG	USB3320C
48-49	I2C1 MUX	TCA9544APWR
52-53	CPLD	
EMIO	I2C0	サンプルデザインでは未接続
EMIO	GPIO	24bit幅
EMIO	TTC	サンプルデザインでは未接続

※電源電圧は3.3V

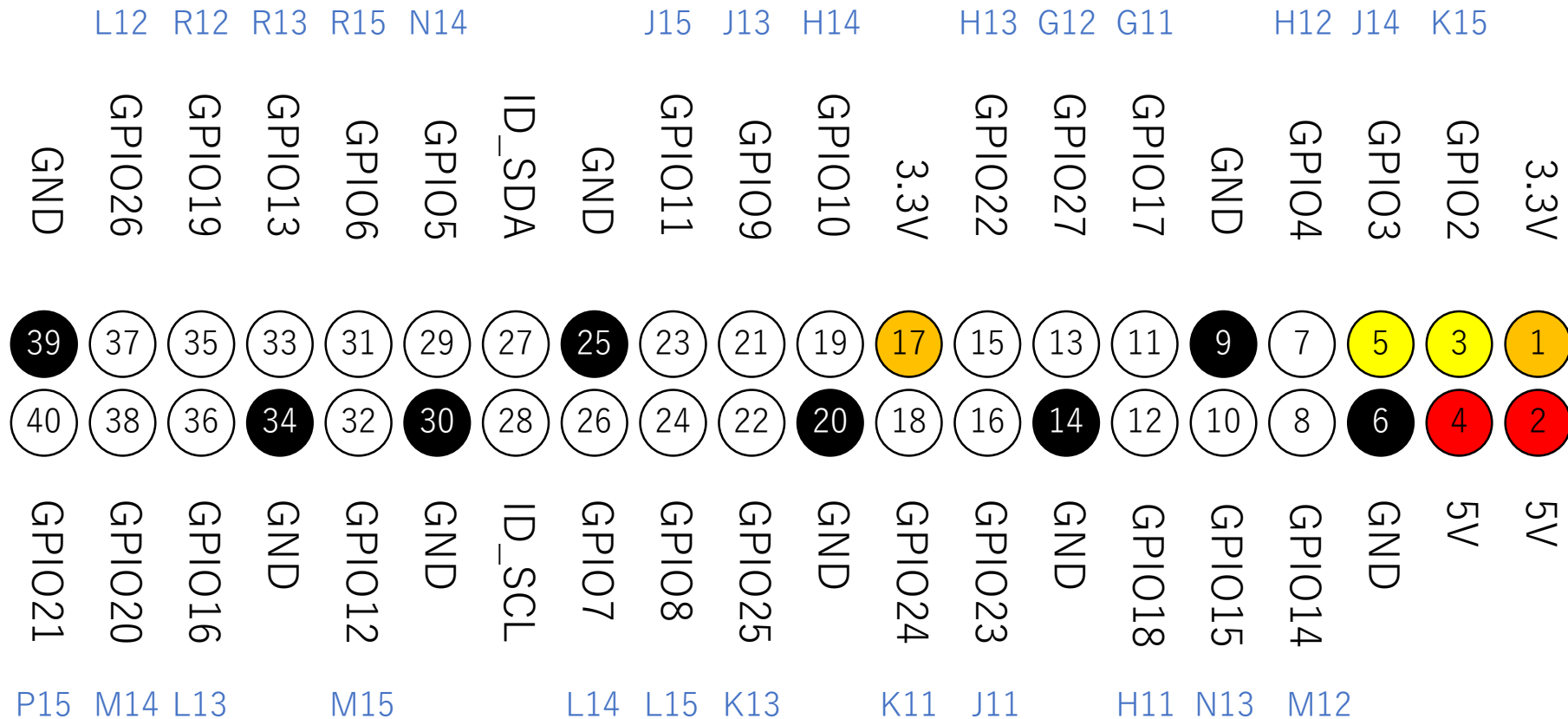
# CPLDについて

- 詳細は不明
  - LEDの点灯
  - DSI(ディスプレイ)のコントロール
  - CSIのGPIO操作
  - MIO8,9をおそらくC\_TDO、C\_TDI経由でUSB-UARTに送っている



# GPIOについて

# GPIOのピン割り当て



※GPIO 14,15はEMIOからは操作できない  
 ※GPIO2,3はプルアップされている



# GPIOの端子割り当て一覧

index	FPGAピン	GPIO番号	ピンヘッダ
0	K15	2	3
1	J14	3	5
2	H12	4	7
3	N14	5	29
4	R15	6	31
5	L14	7	26
6	L15	8	24
7	J13	9	21
8	H14	10	19
9	J15	11	23
10	M15	12	32
11	R13	13	33
12	L13	16	36
13	G11	17	11
14	H11	18	12

index	FPGAピン	GPIO番号	ピンヘッダ
15	R12	19	35
16	M14	20	38
17	P15	21	40
18	H13	22	15
19	J11	23	16
20	K11	24	18
21	K13	25	22
22	L12	26	37
23	G12	27	13
24	F13		DSI_D0_N
25	F14		DSI_D0_P
26	F12		DSI_D1_N
27	E13		DSI_D1_P
28	E11		DSI_C_N
29	E12		DSI_C_P

index	FPGAピン	GPIO番号	ピンヘッダ
30	M11		CSI_D0_N
31	M10		CSI_D0_P
32	P14		CSI_D1_N
33	P13		CSI_D2_P
34	N12		CSI_C_N
35	N11		CSI_C_P
36	N8		PWM_R
37	N7		PWM_L

FPGAのGPIO[23:0]を操作すると  
ピンヘッダに出ている  
GPIO端子の2~27に反映される

GPIO 14,15はEMIOからは操作できない

# GPIOの操作方法

- GPIO汎用ドライバ(/sys/class/gpio)を使う
  - Linuxのシステムコールを使い、/sys/class/gpioにある仮想ファイルを操作します。
  - シェルでの操作をC言語から行うこともできます。
- /dev/memを使う方法
  - GPIOの物理アドレスに直接データを書き込む

# GPIO ドライバの説明

- /sys/class/gpio にPSからGPIOを操作するドライバあり
  - /sys/class/gpio/gpio906~gpio1023が1つ1つのbitに対応
- 906~959はMIO経由のGPIO(54本)
- 960~1023はEMIO経由のGPIO(64本)
  - RasPiピン番号が 2~13の場合："ピン番号+958"番の仮想ファイル
  - RasPiピン番号が16~26の場合："ピン番号+956"番の仮想ファイル
- gpio960 = FPGAのEMIO GPIO[0] = RasPiのGPIO2
  - EMIO GPIO0~GPIO11 → RasPiのGPIO 2~13 → gpio960~971
  - EMIO GPIO12~GPIO23 → RasPiのGPIO 16~27 → gpio972~983

# GPIO ドライバの操作方法①

/sys/class/gpioディレクトリの中身を確認してみます

```
root@zynqberry ~$ ls /sys/class/gpio
export gpiochip906 unexport
```

※906はMIO0

このexportへ使いたいGPIOの番号を書き出すと、そのGPIOを使うための仮想ファイルが用意されます。  
今回はGPIO2へのアクセスなので960番を書き出します。

```
root@zynqberry ~$ echo 960 > /sys/class/gpio/export
root@zynqberry ~$ ls /sys/class/gpio/
export gpio960 gpiochip906 unexport
```

新しくできた/sys/class/gpio/gpio960の中身は

```
root@zynqberry ~$ ls /sys/class/gpio/gpio960
active_low device direction edge power subsystem uevent value
```

# GPIO ドライバの操作方法②

仮想ファイルdirectionはGPIOの入出力の制御、  
仮想ファイルvalueはGPIOの値(1ならHigh、0ならLow)の制御に使います。  
まずGPIO 2を出力用にするため

```
root@zynqberry ~$ echo "out" > /sys/class/gpio/gpio960/direction
root@zynqberry ~$ cat /sys/class/gpio/gpio906/direction
out
```

この状態でvalueに1や0を書き出すと、GPIOに接続したLEDを点滅させることができます。

```
root@zynqberry ~$ echo 1 > /sys/class/gpio/gpio960/value
root@zynqberry ~$ echo 0 > /sys/class/gpio/gpio906/value
```

# /dev/memを使う方法

- GPIOのマップされているIOアドレス0xE000A000
- 1bitごとに1つのbitが対応している

オフセット	レジスタ名	機能
0x00000284	DIRM_2	GPIO入出力。0で入力、1で出力
0x00000288	OEN_2	出力イネーブル
0x00000048	DATA_2	GPIOピンへの出力値

例:0xE000A284のbit0を1にするとEMIO(0)=RasPiGPIO(2)が出力になる

# サンプルプロジェクト (リファレンスデザイン)

# リファレンスデザイン

- 現在、4種類のリファレンスデザインが利用可能

**Description** **Downloads** **Resources**

### Downloads

**Online Documentation:**

- [Trenz Electronic Wiki Documentation > ... > TE0726 Resources](#)

**Notes:**

- If you did not find the necessary documents, please send a request mail to Trenz Electronic Support ([support](#))

📁 **PCN** - Product change notifications

📁 **REV02** - PCB Revision: Schematics, AD, STEP, HW Design Files and more ...

📁 **REV03** - PCB Revision: Schematics, AD, STEP, HW Design Files and more ...

📁 **Reference\_Design** - FPGA Design Examples for different Vivado Versions

**Online Documentation:**

- [Trenz Electronic Wiki Documentation > ... > TE0726 Reference Designs](#)

**Notes:**

- If you did not find the necessary documents, please send a request mail to Trenz Electronic Support ([supp](#))

📁 **2015.4**

📁 **2016.2**

📁 **2016.4**

📁 **2017.1**

- 📁 **zynqberrydemo1** - ZynqBerry - Demo VIDEO/AUDIO Design with RPI video camera stream to monitor
- 📁 **zynqberrydemo2** - ZynqBerry - Demo VIDEO/AUDIO Design with Debian\_8.4 32Bit Example
- 📁 **zynqberrydemo3** - ZynqBerry - Demo VIDEO/AUDIO Design with Video and Audio Example

📁 **2017.4**

- 📁 **test\_board** - TE0726 Basic Linux Example

**zynqberrydemo1** - ZynqBerry - Demo VIDEO/AUDIO Design with RPI video camera stream to monitor

**zynqberrydemo2** - ZynqBerry - Demo VIDEO/AUDIO Design with Debian\_8.4 32Bit Example

※Debianのイメージ入り

**zynqberrydemo3** - ZynqBerry - Demo VIDEO/AUDIO Design with Video and Audio Example

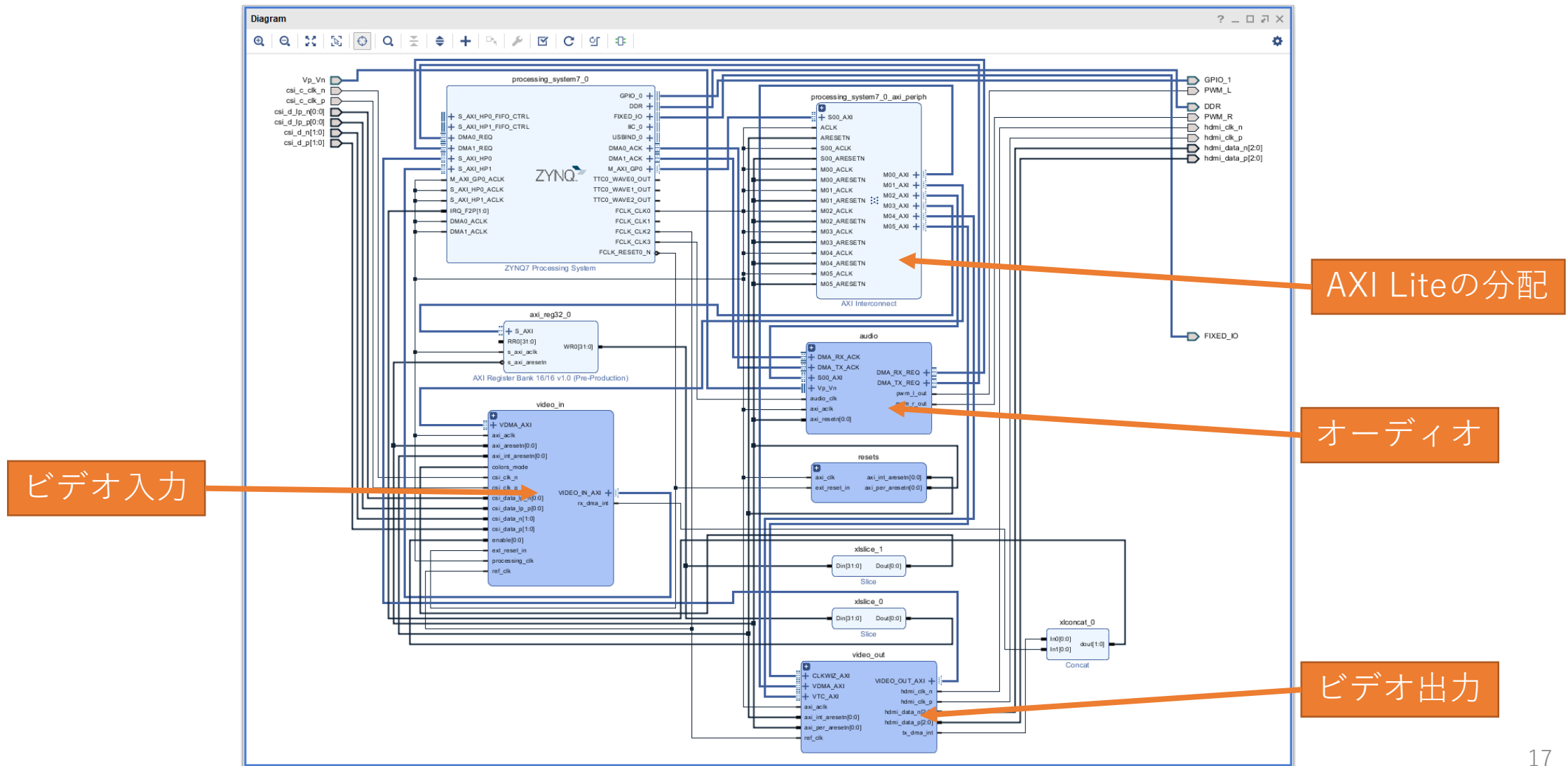
**test\_board** - TE0726 Basic Linux Example

本質的に  
同一

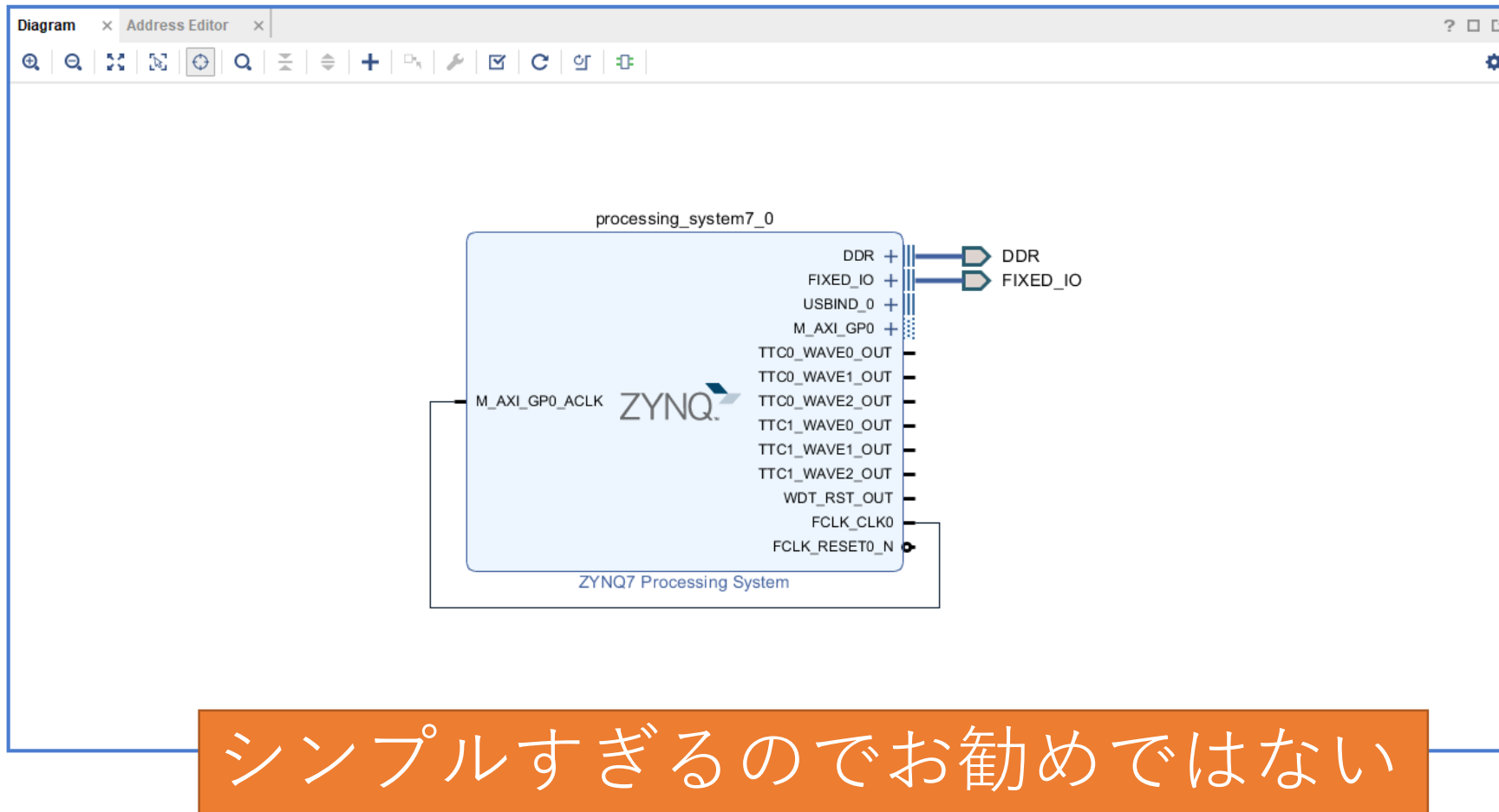
←シンプル



# サンプルプロジェクト 2017.1



# サンプルプロジェクト 2017.4



# バージョン間の相違(論理合成)

- Vivado2017.3以降では、MIOのEMIOから出る双方向ピンのImplementができない

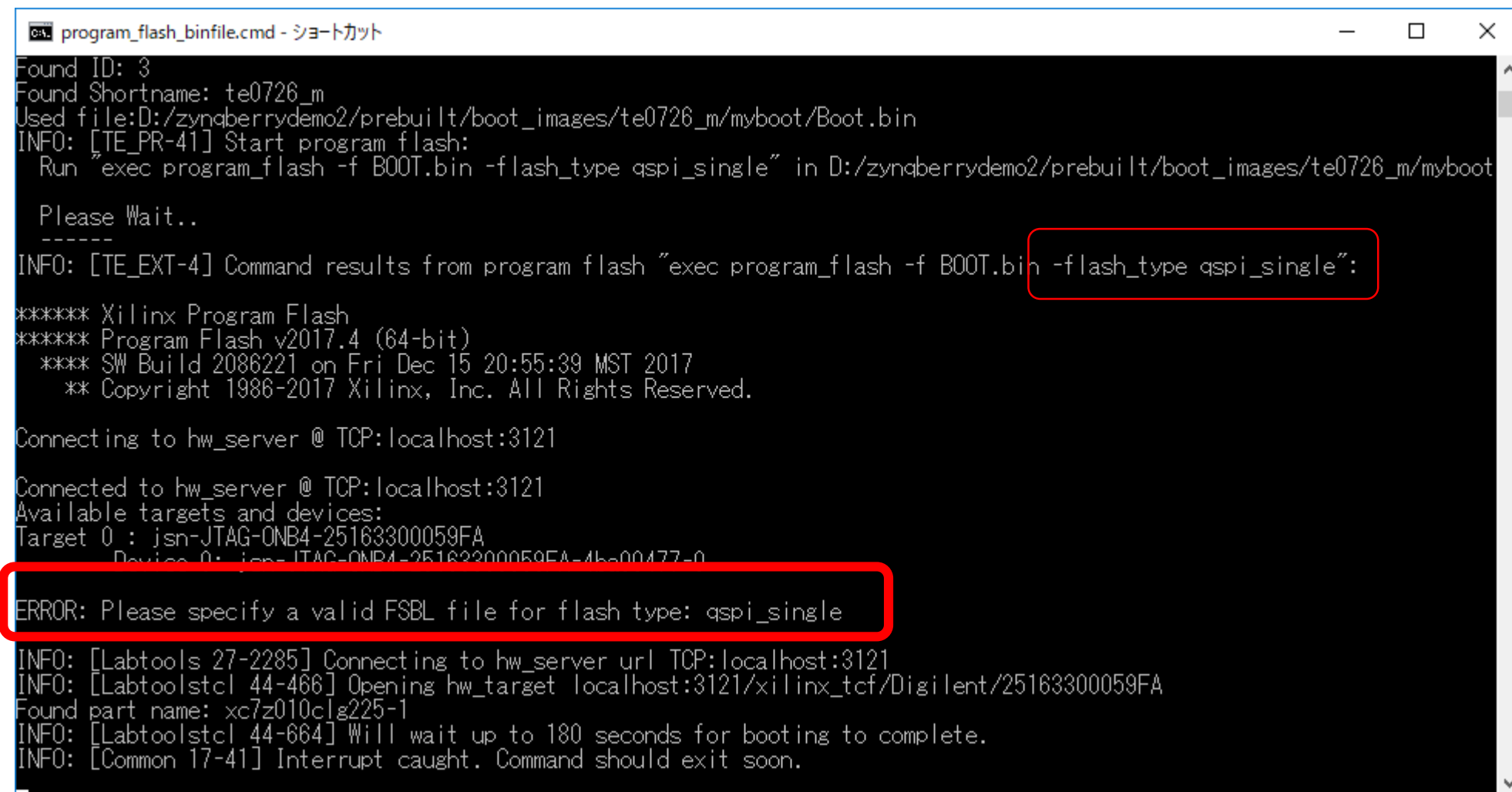


# バージョン間の相違(書き込み)

- Vivado 2017.4を使用する
  - プロジェクト2017.1のスクリプトでboot.binの書き込みが**できない**  
(qspiのFSBLかどうか確認せよ、と言われる)
  - プロジェクト2017.4のスクリプトでboot.binの書き込みはできる
- Vivado 2017.1または2を使用する
  - プロジェクト2017.1のスクリプトでboot.binの書き込みができる
  - プロジェクト2017.4のスクリプトでboot.binの書き込みはできる

書き込みに使うVivadoのバージョンは、Vivado 2017.1または2が良い

# 書き込み失敗のエラー画面



```
program_flash_binfile.cmd - ショートカット
Found ID: 3
Found Shortname: te0726_m
Used file:D:/zynqberrydemo2/prebuilt/boot_images/te0726_m/myboot/Boot.bin
INFO: [TE_PR-41] Start program flash:
  Run "exec program_flash -f BOOT.bin -flash_type qspi_single" in D:/zynqberrydemo2/prebuilt/boot_images/te0726_m/myboot
  Please Wait..
  -----
INFO: [TE_EXT-4] Command results from program flash "exec program_flash -f BOOT.bin -flash_type qspi_single":

***** Xilinx Program Flash
***** Program Flash v2017.4 (64-bit)
***** SW Build 2086221 on Fri Dec 15 20:55:39 MST 2017
***** Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.

Connecting to hw_server @ TCP:localhost:3121
Connected to hw_server @ TCP:localhost:3121
Available targets and devices:
Target 0 : jsn-JTAG-ONB4-25163300059FA
Device 0 : jsp-JTAG-ONB4-25163300059FA-4b00477-0

ERROR: Please specify a valid FSBL file for flash type: qspi_single

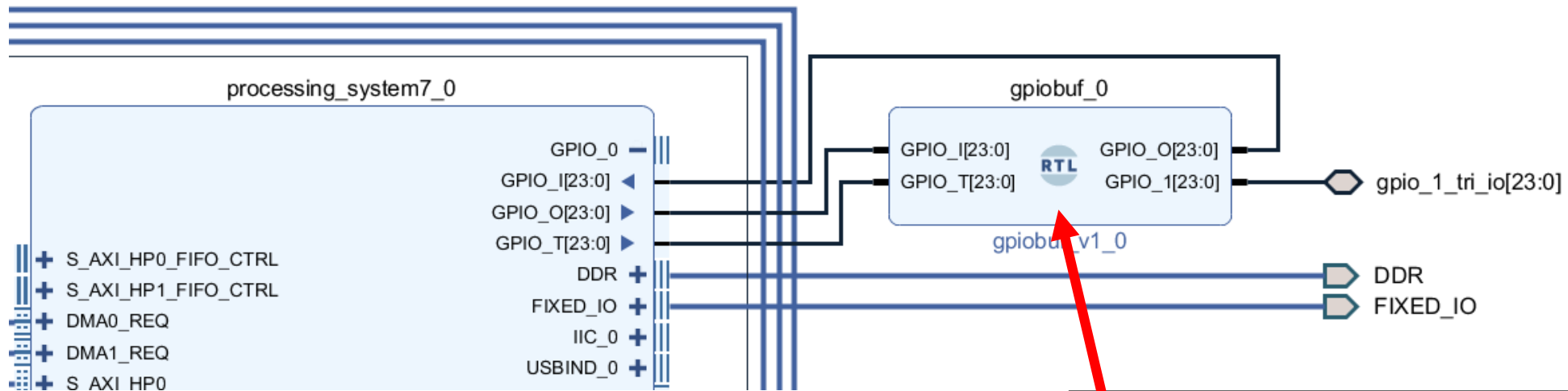
INFO: [Labtools 27-2285] Connecting to hw_server url TCP:localhost:3121
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/25163300059FA
Found part name: xc7z010clg225-1
INFO: [Labtoolstcl 44-664] Will wait up to 180 seconds for booting to complete.
INFO: [Common 17-41] Interrupt caught. Command should exit soon.
```

# バージョン間の相違(まとめ)

サンプルプロジェクト のバージョン	機能	Vivado 2017.1~2	Vivado 2017.4	Debianイメージ
2017.1 zynqberrydemo2	リッチ ◎ (HDMI,PWM Audio, CSIカメラ,etc…)	論理合成○ 書き込み○	論理合成× 書き込み×	あり
2017.4 test_board	シンプル × (何もない)	論理合成? 書き込み○	論理合成○ 書き込み○	なし

# Vivado 2017.3以降でzynqberrydemo2を使う方法

- 3ステートバッファを明示的に入れる



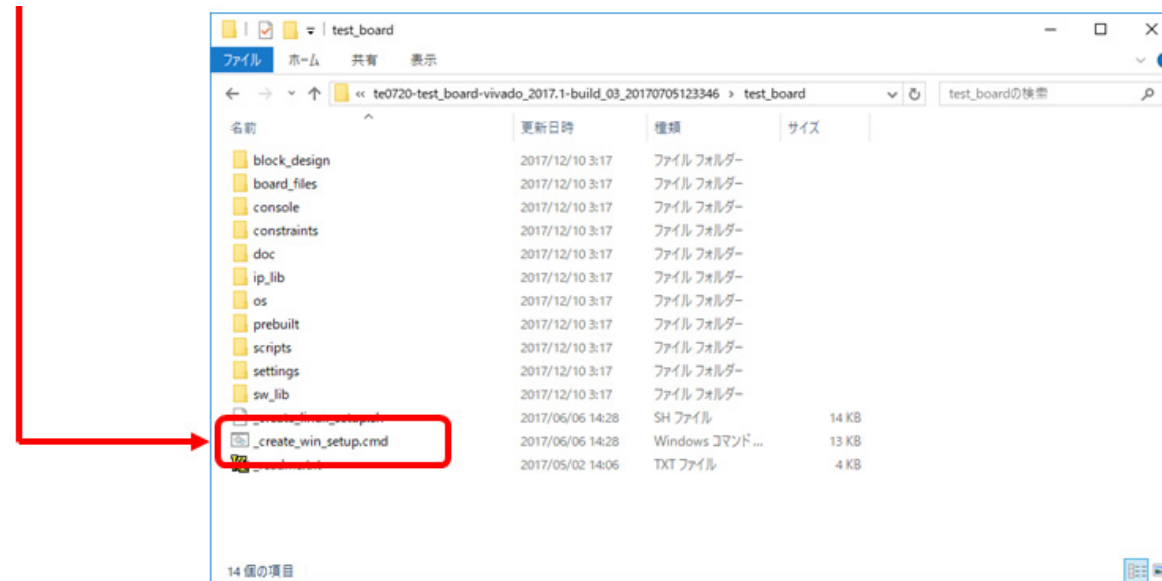
```
GPIO_LOOP : FOR i in 0 to 23 generate
  IOBUF_INST : IOBUF port map (
    IO => GPIO_1(i),
    O => GPIO_O(i),
    I => GPIO_I(i),
    T => GPIO_T(i)
  );
end generate;
```

# Trenzスクリプト



# Trenzスクリプトの使い方

- Trenzスクリプトとは
  - Vivadoのプロジェクトの生成や、ビルドや書き込みを行うためのバッチファイルとTCLスクリプト
  - プロジェクトのフォルダの中にある\_create\_win\_setup.cmdを実行



# Trenzプロジェクトの生成

- DOSプロンプトが現れるので、「1」を押して、Enterを押します。
















```
C:\WINDOWS\system32\cmd.exe

D:\te0726\test_board>set local
-----Set design paths-----
-- Run Design with: _create_win_setup
-- Use Design Path: D:\te0726\test_board\
-----
-----TE Reference Design-----
-----
-- (c) Go to CMD-File Generation (Manual setup)
-- (d) Go to Documentation (Web Documentation)
-- (x) Exit Batch (nothing is done!)
-- (0) Create minimum setup of CMD-Files and exit Batch
-- (1) Create maximum setup of CMD-Files and exit Batch
-----
Select (ex.: '0' for min setup):
```

```
C:\WINDOWS\system32\cmd.exe

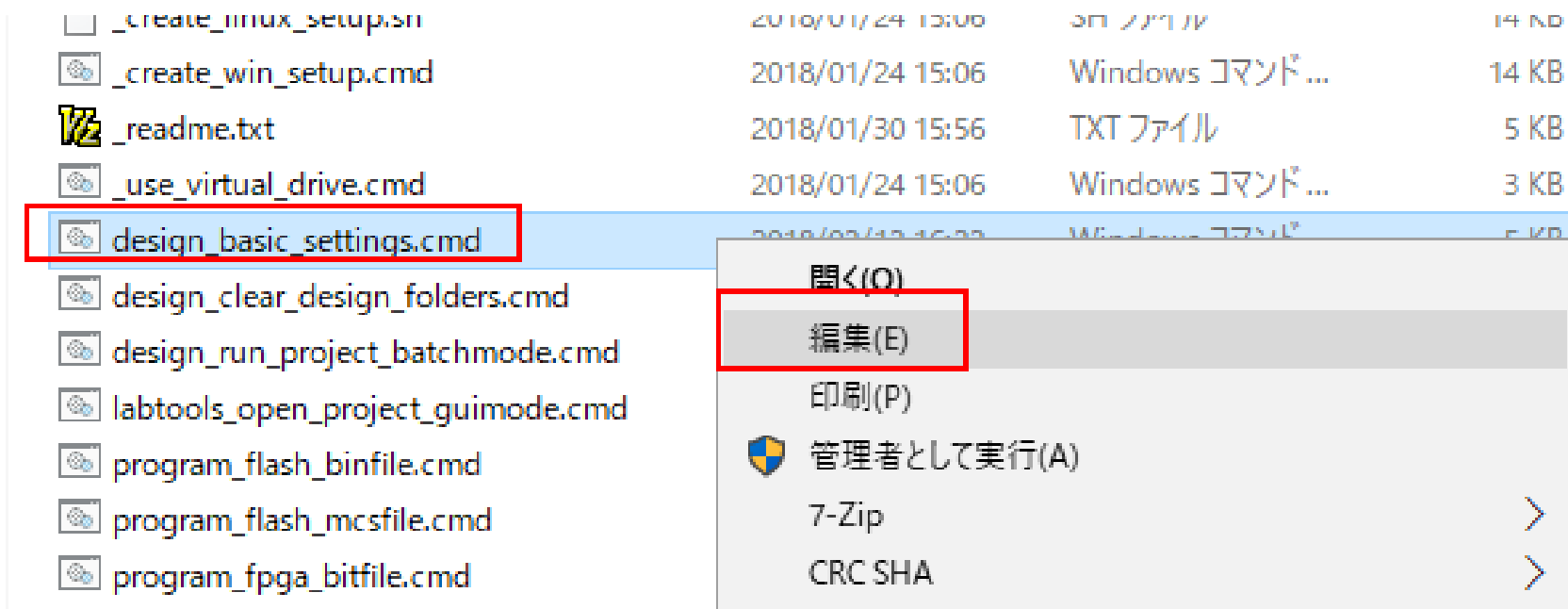
Select (ex.: '0' for min setup):1
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
1 個のファイルをコピーしました。
-----Minimal Setup-----
--- 0. (optional) use "_use_virtual_drive.cm", see Xilinx "AR# 52787"
--- 1. Open "design_basic_settings.cmd" with text editor
--- 2. Set Xilinx Installation path, default: @set XILDIR=C:/Xilinx
--- 3. Set the Board Part you bought, example: @set PARTNUMBER=te0726-3m
    For available names see: ./board_files/TExxx_board_files.csv
--- 4. Save "design_basic_settings.cmd"
--- 5. To create vivado project, execute: ./vivado_create_project_gui_mode.cmd
    Use Trenez Electronic Wiki for more information:
    https://wiki.trenz-electronic.de/display/PD/Project+Delivery
-----Finished-----
続行するには何かキーを押してください . . .
```

# 各種コマンドファイルが生成される

 _create_linux_setup.sh	2018/01/24 15:06	SH ファイル	14 KB
 _create_win_setup.cmd	2018/01/24 15:06	Windows コマンド...	14 KB
 _readme.txt	2018/01/30 15:56	TXT ファイル	5 KB
 _use_virtual_drive.cmd	2018/01/24 15:06	Windows コマンド...	3 KB
 design_basic_settings.cmd	2018/02/12 16:22	Windows コマンド...	5 KB
 design_clear_design_folders.cmd	2018/02/12 16:22	Windows コマンド...	6 KB
 design_run_project_batchmode.cmd	2018/02/12 16:22	Windows コマンド...	5 KB
 labtools_open_project_guiemode.cmd	2018/02/12 16:22	Windows コマンド...	5 KB
 program_flash_binfile.cmd	2018/02/12 16:22	Windows コマンド...	6 KB
 program_flash_mcsfile.cmd	2018/02/12 16:22	Windows コマンド...	6 KB
 program_fpga_bitfile.cmd	2018/02/12 16:22	Windows コマンド...	6 KB
 sdk_create_prebuilt_project_guiemode.cmd	2018/02/12 16:22	Windows コマンド...	6 KB
 vivado_create_project_batchmode.cmd	2018/02/12 16:22	Windows コマンド...	6 KB
 vivado_create_project_guiemode.cmd	2018/02/12 16:22	Windows コマンド...	6 KB
 vivado_open_existing_project_guiemode.c...	2018/02/12 16:22	Windows コマンド...	5 KB

# スクリプトのカスタマイズ

- design\_basic\_settings.cmdを右クリックし、編集



# 最低限編集すべき箇所

- XILINXディレクトリへのパス
- Vivadoのバージョン
- ボードの型番
- zsys\_bd.tcl内のバージョン情報

```
@REM -SUSoC (optional used for programming only): %XILDIR%\SUSoC%\VIV
① @set XILDIR=C:/Xilinx
@REM -Attention: These scripts support only the predefined Vivado Ver
② @set VIVADO_VERSION=2017.1
@REM -----
@REM Set Board part number of the project which should be created
@REM -Available Numbers: (you can use ID, PRODID, BOARDNAME or SHORTNAM
@REM -variable is used for project creation and programming
@REM -Example TE0726 Module :
@REM -USE ID | USE PRODID | Use BOARDNAME
③ @REM -@set PARTNUMBER=1 | @set PARTNUMBER=te0726-3m | @set PARTNUMBER=
@set PARTNUMBER=LAST_ID
@REM -----
```

使用するボード	PARTNUMBERに設定する値
TE0720-3-1CF(A) (Gigazee)	5
TE0720-3-2IF(A) (Gigazee)	1
TE0726-03M (Zynqberry)	3

# block\_designのバージョン

- 2017.1のプロジェクトファイルをVivado2017.1以外で使う場合は以下の変更が必要
  - block\_designフォルダにzsys\_bd.tclというファイルの25行目  
`set scripts_vivado_version 2017.1`

```
19 variable script_folder↓
20 set script_folder [_tcl::get_script_folder]↓
21 ↓
22 #####↓
23 # Check if script is running in correct Vivado version.↓
24 #####↓
25 set scripts_vivado_version 2017.1↓
26 set current_vivado_version [version -short]↓
27 ↓
28 if { [string first $scripts_vivado_version $current_vivado_version] == -1 } {↓
29     puts ""↓
```

# よく使うスクリプト

スクリプト名	機能
<b>design_run_project_batchmode.cmd</b>	FPGAのビルドを行い、FSBLとU-Bootを結合してboot.binを生成する
design_clear_design_folders.cmd	v_logとVivadoプロジェクトを削除
vivado_create_project_gui mode.cmd	Vivadoプロジェクトの生成
<b>program_flash_binfile.cmd</b>	生成されたboot.binを書き込み

## 基本的な流れ

1. vivado\_create\_project\_gui mode.cmd . . . プロジェクト作成
2. design\_clear\_design\_folders.cmd . . . プロジェクトクリーンアップ
3. design\_run\_project\_batchmode.cmd . . . ビルド
4. program\_flash\_binfile.cmd . . . 書き込み

# 最初の書き込み

- ZynqberryはSPI ROMからしか起動できない(SDブート不可)
  - 書き込みにはTrenzスクリプトのprogram\_flash\_binfile.cmdを用いる
- 準備
  - design\_basic\_settings.cmdを開く
  - @set PARTNUMBER=3に設定する
  - @set SWAPP=NAを@set SWAPP=u-bootにする
- 書き込み
  - prebuilt/boot\_images/**m**/**u-boot**/Boot.bin が書き込まれる

PARTNUMBERで決まる値

SWAPPで決まる値

2017.4プロジェクトの場合



# 書き込み中のように

```
C:\WINDOWS\system32\cmd.exe
Start Flash Programming with BIN File
Found ID: 3
Found Shortname: m
Used file:D:/te0726/test_board/prebuilt/boot_images/m/u-boot/Boot.bin
INFO: [TE_PR-41] Start program flash:
  Run "exec program_flash -f BOOT.bin -fsbl D:/te0726/test_board/prebuilt/software/m/zynq_fsbl_flash.elf -flash_type qspi_single" in D:/te0726/test_board/prebuilt/boot_images/m/u-boot
  Please Wait..
-----
INFO: [TE_EXT-4] Command results from program flash "exec program_flash -f BOOT.bin -fsbl D:/te0726/test_board/prebuilt/software/m/zynq_fsbl_flash.elf -flash_type qspi_single":

***** Xilinx Program Flash
***** Program Flash v2017.4 (64-bit)
***** SW Build 2086221 on Fri Dec 15 20:55:39 MST 2017
***** Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.

Connecting to hw_server @ TCP:localhost:3121
WARNING: Failed to connect to hw_server at TCP:localhost:3121
Attempting to launch hw_server at TCP:localhost:3121
Connected to hw_server @ TCP:localhost:3121
```

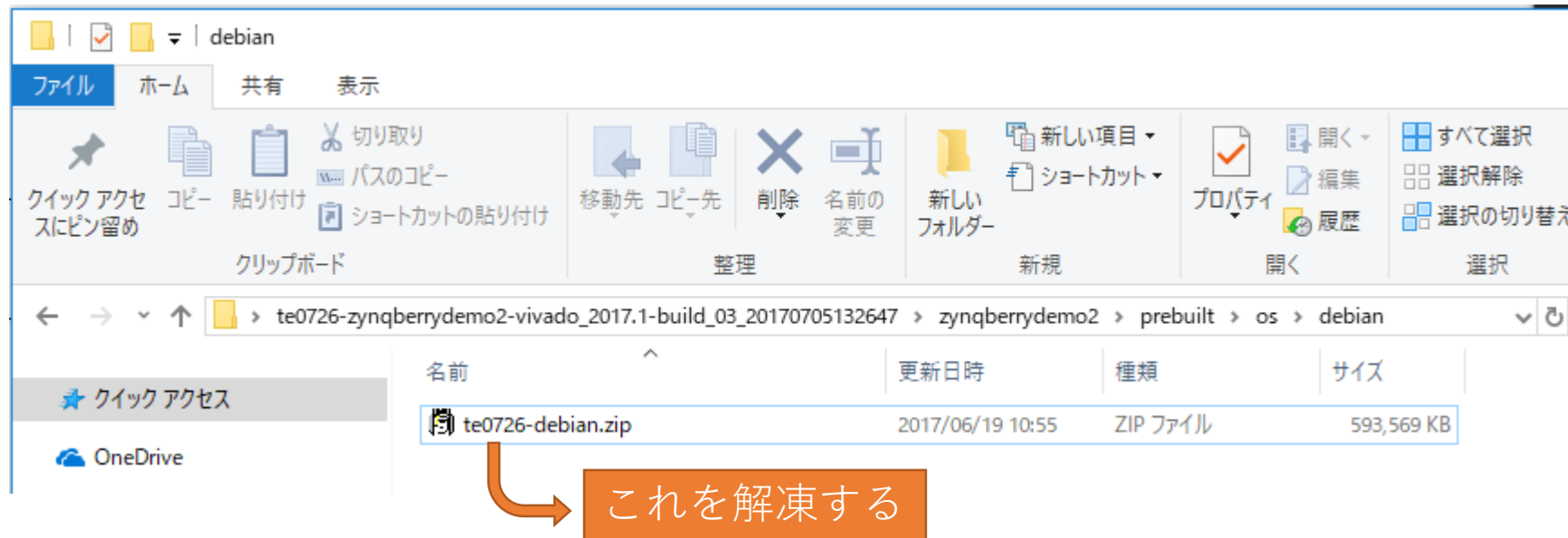
```
C:\WINDOWS\system32\cmd.exe
Performing Erase Operation...
Erase Operation successful.
INFO: [Xicom 50-44] Elapsed time = 5 sec.
Performing Program Operation...
0%...20%...40%...60%...80%...100%
Program Operation successful.
INFO: [Xicom 50-44] Elapsed time = 4 sec.

Flash Operation Successful
-----
INFO: [Labtools 27-2285] Connecting to hw_server url TCP:localhost:3121
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/25163300059FA
Found part name: xc7z010clg225-1
INFO: [Labtoolstcl 44-664] Will wait up to 180 seconds for booting to complete.
INFO: [Labtools 27-2278] Zynq reset successful
INFO: [TE_PR-1] Reboot Device is done (Note successfully software reboot depends also on devices design).
INFO: [Labtools 27-1434] Device xc7z010 (JTAG device index = 1) is programmed with a design that has no supported debug core(s) in it.
WARNING: [Labtools 27-3361] The debug hub core was not detected.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active.
```

# Trenz公式Debian Linux

# Trenz製Debian Linuxの所在

- リファレンスデザイン2017.1版のzynqberry\_demo2のprebuildをダウンロード
- zynqberrydemo2¥prebuilt¥os¥debianにある

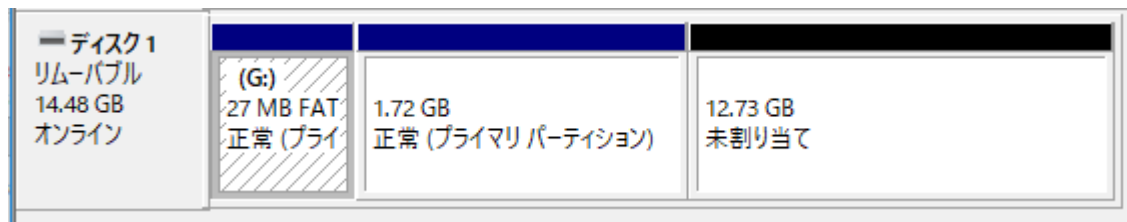
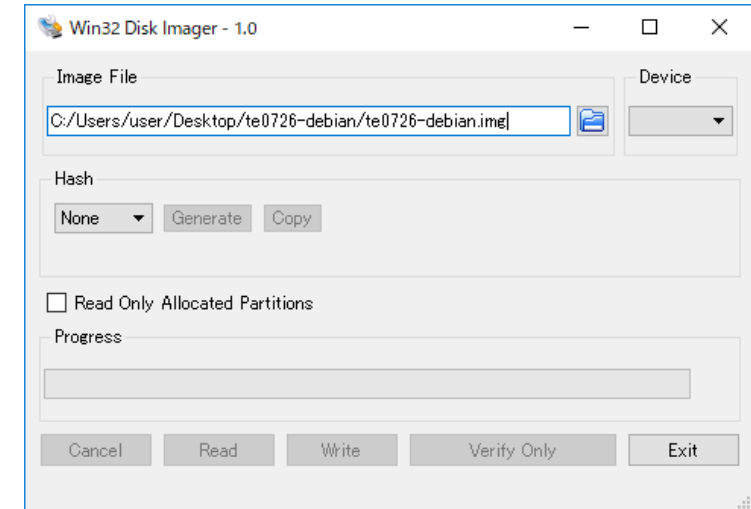


# Debianのimgファイルを書き込む

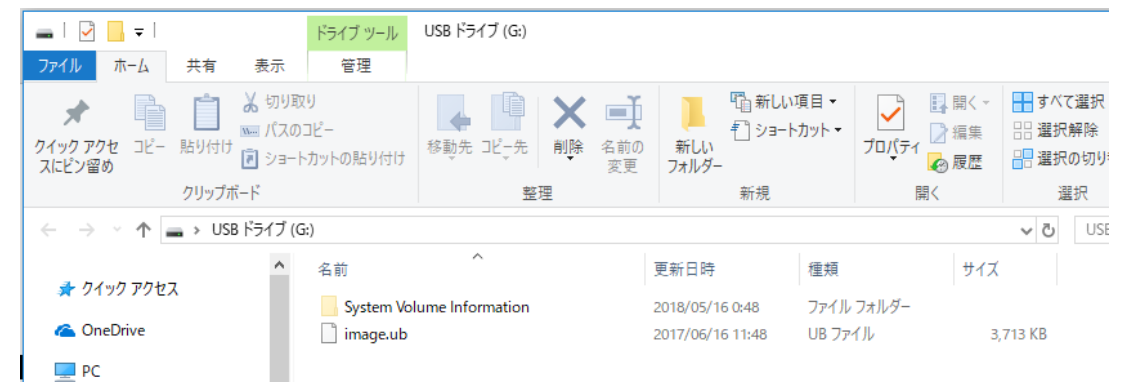
- Win32DiskImagerで書き込む



1.8GBある

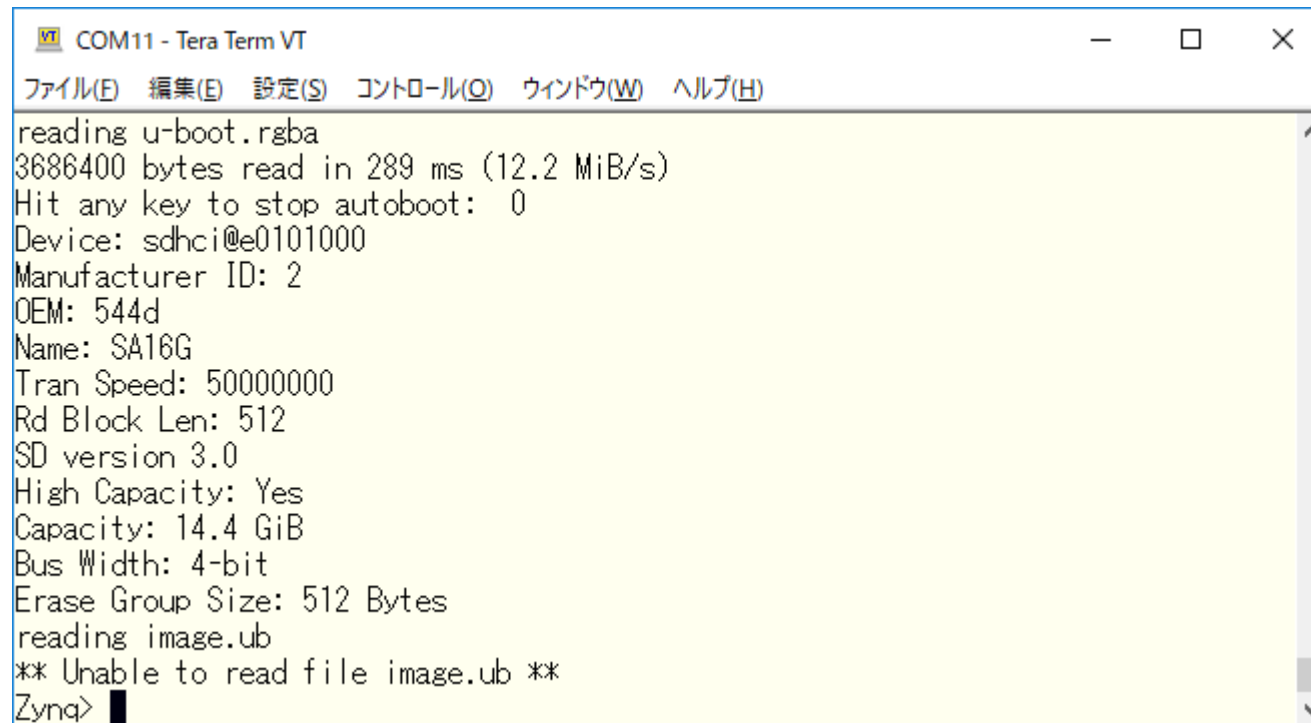


FATのパーティションには  
uimage.ubのみ



# Linuxの起動1

- Trenzサンプルデザインのprebuildにあるboot.binを書き込むと、カーネルイメージ(image.ub)が要求される



```
COM11 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
reading u-boot.rgba
3686400 bytes read in 289 ms (12.2 MiB/s)
Hit any key to stop autoboot: 0
Device: sdhci@e0101000
Manufacturer ID: 2
OEM: 544d
Name: SA16G
Tran Speed: 50000000
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 14.4 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
reading image.ub
** Unable to read file image.ub **
Zynq>
```

# Linuxの起動2

- zynqberrydemo2¥prebuilt¥os¥debianにあるdebianのイメージを書き込んだSDカードがあれば、起動する

```
COM11 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
Capacity: 14.5 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
reading image.ub
3801436 bytes read in 221 ms (16.4 MiB/s)
## Loading kernel from FIT Image at 10000000 ...
Using 'conf@2' configuration
Verifying Hash Integrity ... OK
Trying 'kernel@0' kernel subimage
  Description: Linux Kernel
  Type: Kernel Image
  Compression: uncompressed
  Data Start: 0x100000d4
  Data Size: 3777912 Bytes = 3.6 MiB
  Architecture: ARM
  OS: Linux
  Load Address: 0x00008000
  Entry Point: 0x00008000
  Hash algo: sha1
  Hash value: 664c9e07855bcb142e085538593f861886155dec
Verifying Hash Integrity ... sha1+ OK
## Loading fdt from FIT Image at 10000000 ...
Using 'conf@2' configuration
Trying 'fdt@0' fdt subimage
  Description: Flattened Device Tree blob
  Type: Flat Device Tree
  Compression: uncompressed
  Data Start: 0x1039a740
  Data Size: 22231 Bytes = 21.7 KiB
  Architecture: ARM
  Hash algo: sha1
  Hash value: 4c40b59b32531fca144f2965ac25422d4b7794fe
Verifying Hash Integrity ... sha1+ OK
Booting using the fdt blob at 0x1039a740
Loading Kernel Image ... OK

COM11 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
[ OK ] Reached target Timers.
[ OK ] Reached target Basic System.
Starting Regular background program processing daemon...
[ OK ] Started Regular background program processing daemon.
Starting OpenBSD Secure Shell server...
[ OK ] Started OpenBSD Secure Shell server.
Starting /etc/rc.local Compatibility...
Starting Login Service...
Starting LSB: Brings up/down network automatically...
Starting LSB: Start NTP daemon...
Starting D-Bus System Message Bus...
[ OK ] Started D-Bus System Message Bus.
smc95xx 1-1.1:1.0 eth0: hardware isn't capable of remote wakeup
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
Starting System Logging Service...
Starting Permit User Sessions...
[ OK ] Started /etc/rc.local Compatibility.
[ OK ] Started LSB: Start NTP daemon.
[ OK ] Started Permit User Sessions.
[ OK ] Started LSB: Brings up/down network automatically.
[ OK ] Started System Logging Service.
Starting Getty on tty1...
[ OK ] Started Getty on tty1.
Starting Serial Getty on ttyPS0...
[ OK ] Started Serial Getty on ttyPS0.
[ OK ] Reached target Login Prompts.
[ OK ] Started Login Service.
[ OK ] Reached target Multi-User System.
[ OK ] Reached target Graphical Interface.
Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

Debian GNU/Linux 8 zynq ttyPS0
zynq login: █
```

ユーザ名はroot  
パスワードはroot

# デスクトップの起動



startxでデスクトップが起動

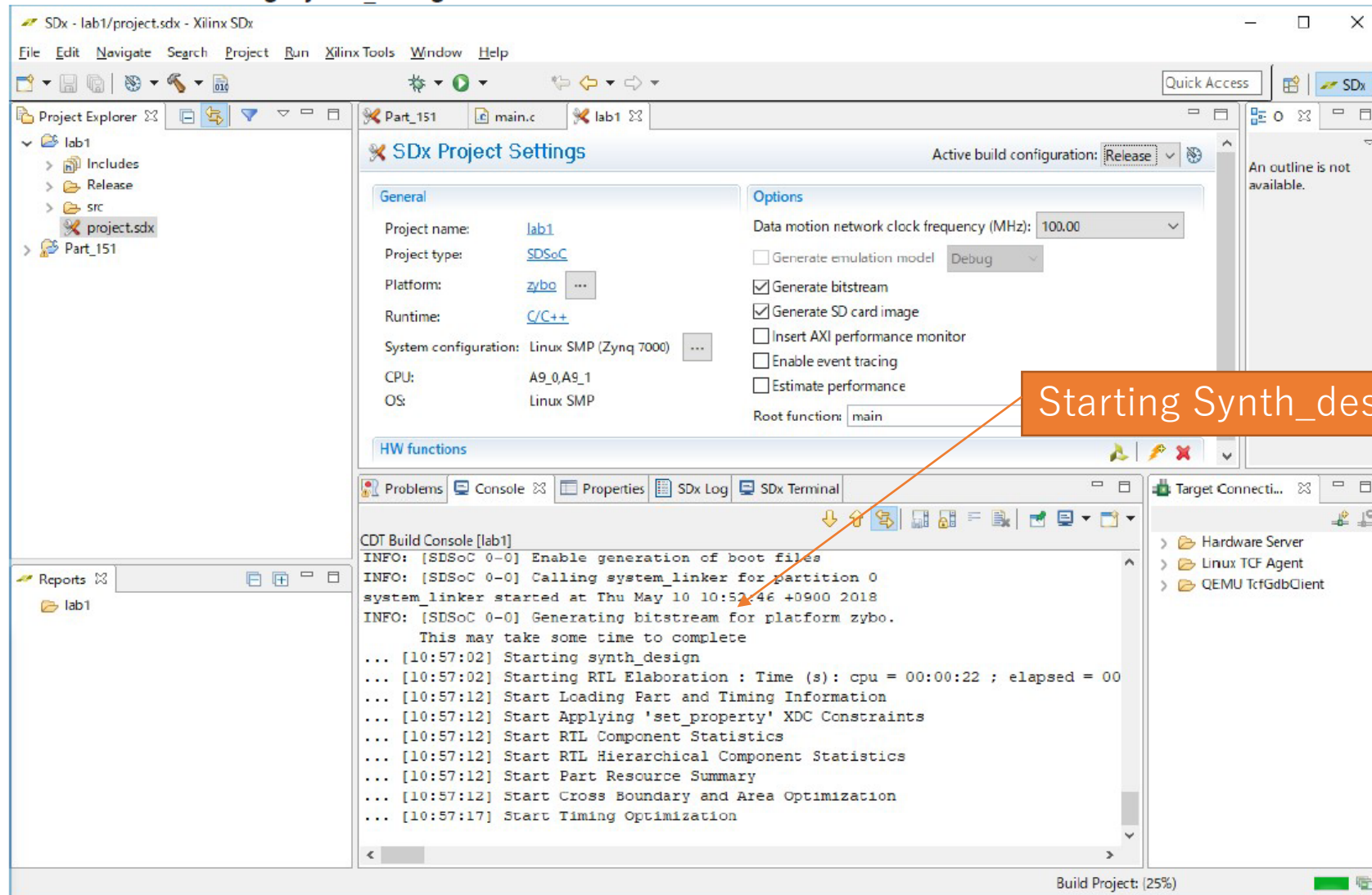
# SDSoC



# SDSoCについて

- VivadoとHLSとXSDKを統合したものがSDx (SDSoC)
- SDxをダウンロードすると両方インストールされる
- VivadoではRTLから書くが、SDxはソフトウェアを書いて、ハードウェア化したい部分を指定する
- 必ずBoard Platform ファイルが必要

# SDSoCの動作画面



# Zynqberry と SDSoc

- ZynqberryのSDSoC対応は2016.2で止まっている
  - 2016.3からSDSocとSDAccelを統合したSDxに切り替わり、Xilinx的にはSDx一本でいく方針?のようなので2016.2からファイル構成や作法、必要ファイルなども大きく変わっていると思われます。

2015.4  
2016.2

- te0726\_m\_demo1 - ZynqBerry - Demo VIDEO/AUDIO Design with RPI video camera stream to monitor
- te0726\_m\_demo2 - ZynqBerry - Demo VIDEO/AUDIO Design with Debian\_8.4
- te0726\_m\_demo3 - ZynqBerry - Demo VIDEO/AUDIO Design with Video and Audio Example
- te0726\_m\_sdsoc - TE0726-M SDSoc**
- test\_board - TE0726 Test Board

2016.4

- test\_board - TE0726 Zynq PS

2017.1

- zynqberrydemo1 - ZynqBerry - Demo VIDEO/AUDIO Design with RPI video camera stream to monitor
- zynqberrydemo2 - ZynqBerry - Demo VIDEO/AUDIO Design with Debian\_8.4 32Bit Example
- zynqberrydemo3 - ZynqBerry - Demo VIDEO/AUDIO Design with Video and Audio Example

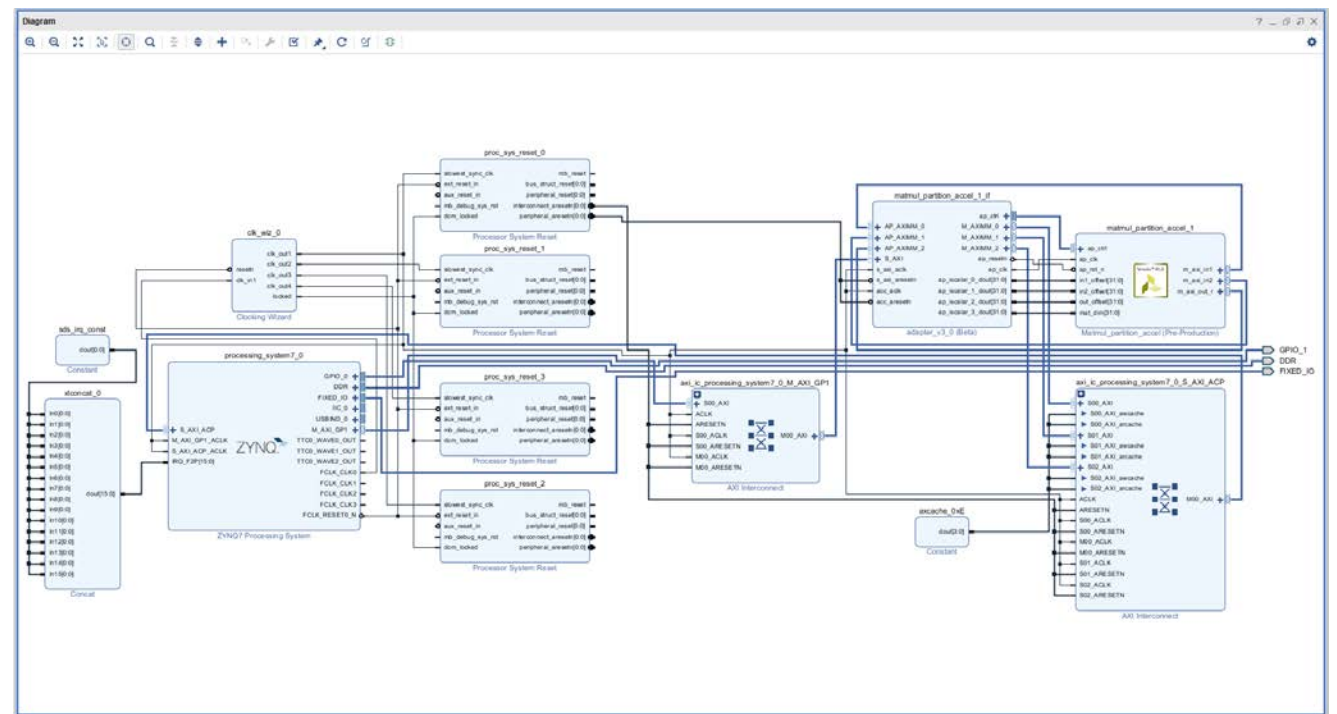
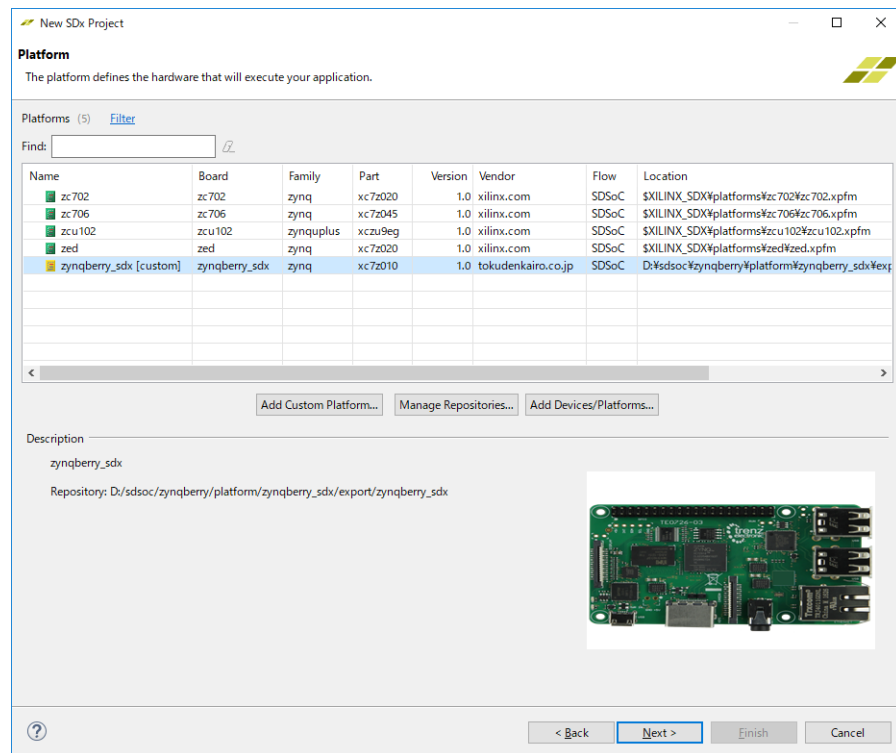
2017.4

- test\_board - TE0726 Zynq PS

最後のSDSoC対応版

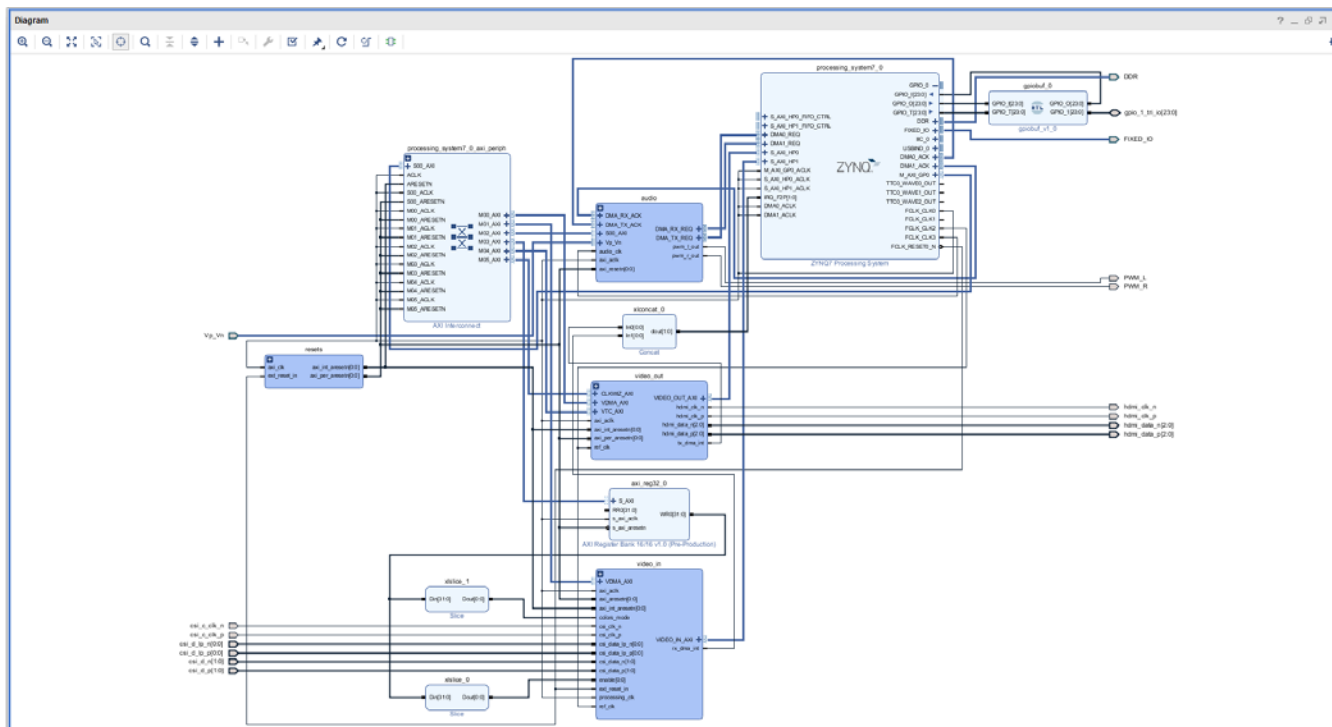
# カスタムプラットフォームの作成

- 自分でZynqberry用のプラットフォームを作成
- 一応、C関数のハードウェア化までは確認

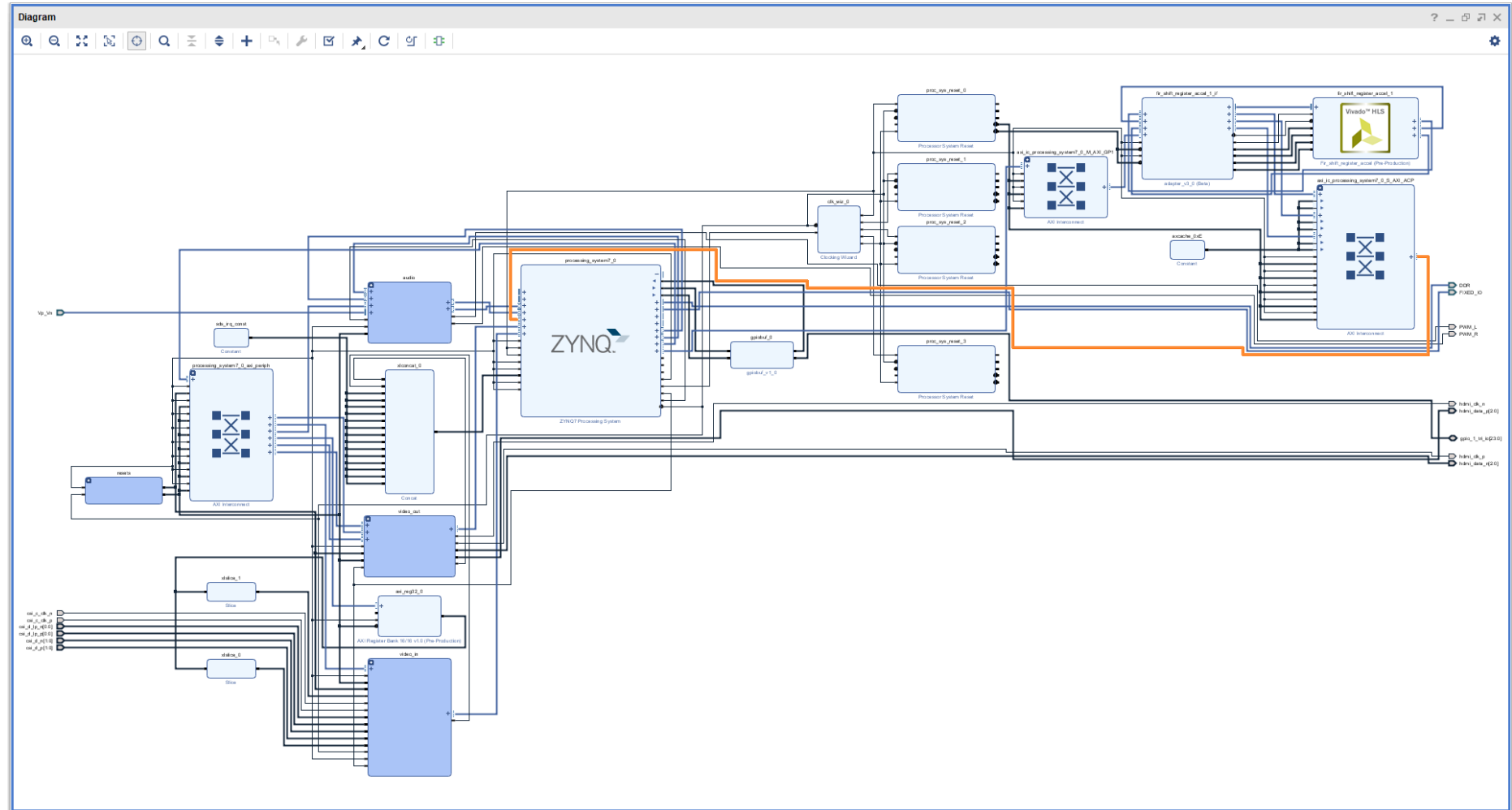


より、リッチなプラットフォームへ

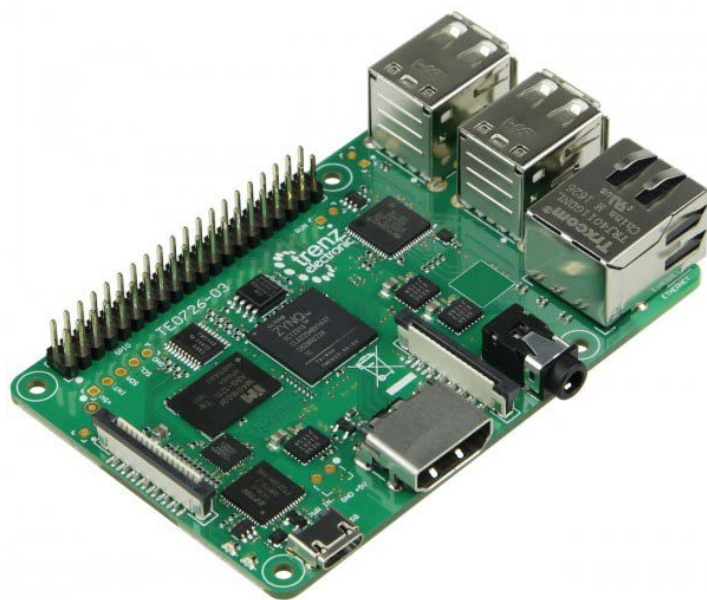
- zynqberrydemo2をベースにVivado 2018へ移行し、プラットフォームファイルを生成中



# SDSoCの回路を組み込み



こういうことがやりたかったんじゃないかな？



```
root@zynqberry $ heavy_process
```

```
int main () {
```

```
...
```

```
hw_func1();
```

```
hw_func2();
```

```
...
```

```
}
```

HW化された関数

FPGAをアクセラレータとして使えるRaspberry Pi互換のボード  
動的にHWを再構成しながら重い処理をハードでこなす



# 行列計算プログラムをHW化

- SDSocのサンプルにあるarray partition
- Zynqberry(XC7Z010)には入りきらなかった

The screenshot displays the Xilinx IDE interface. The top pane shows the 'SDx Build Console' with several error messages related to resource constraints. The bottom pane shows the 'Utilization Estimates' window with a table of resource usage.

**SDx Build Console [testapp2, Debug]**

```
====>The following messages were generated while processing D:/sdsoc/zynqberry/platform/testapp2/Debug/_sds/p0/_vpl/ipi\imp\imp.r  
ERROR: [VPL 30-640] Place Check : This design requires more Slice LUTs cells than are available in the target device. This design  
ERROR: [VPL 30-640] Place Check : This design requires more LUT as Logic cells than are available in the target device. This des  
ERROR: [VPL 30-640] Place Check : This design requires more RAMB18 and RAMB36/FIFO cells than are available in the target device  
ERROR: [VPL 30-640] Place Check : This design requires more CARRY4 cells than are available in the target device. This design re  
ERROR: [VPL 30-99] Placer failed with error: 'IO Clock Placer stopped due to earlier errors. Implementation Feasibility check fa  
Please review all ERROR and WARNING messages during placement to understand the cause for failure.  
-----  
----- failed: Placer could not place all instances -----
```

**Utilization Estimates**

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	2699
FIFO	-	-	-	-
Instance	6	260	12261	4990
Memory	136	-	0	0
Multiplexer	-	-	-	2283
Register	0	-	7633	96
<b>Total</b>	<b>142</b>	<b>260</b>	<b>19894</b>	<b>10068</b>
Available	120	80	35200	17600
Utilization (%)	<b>118</b>	<b>325</b>	56	57

Detail

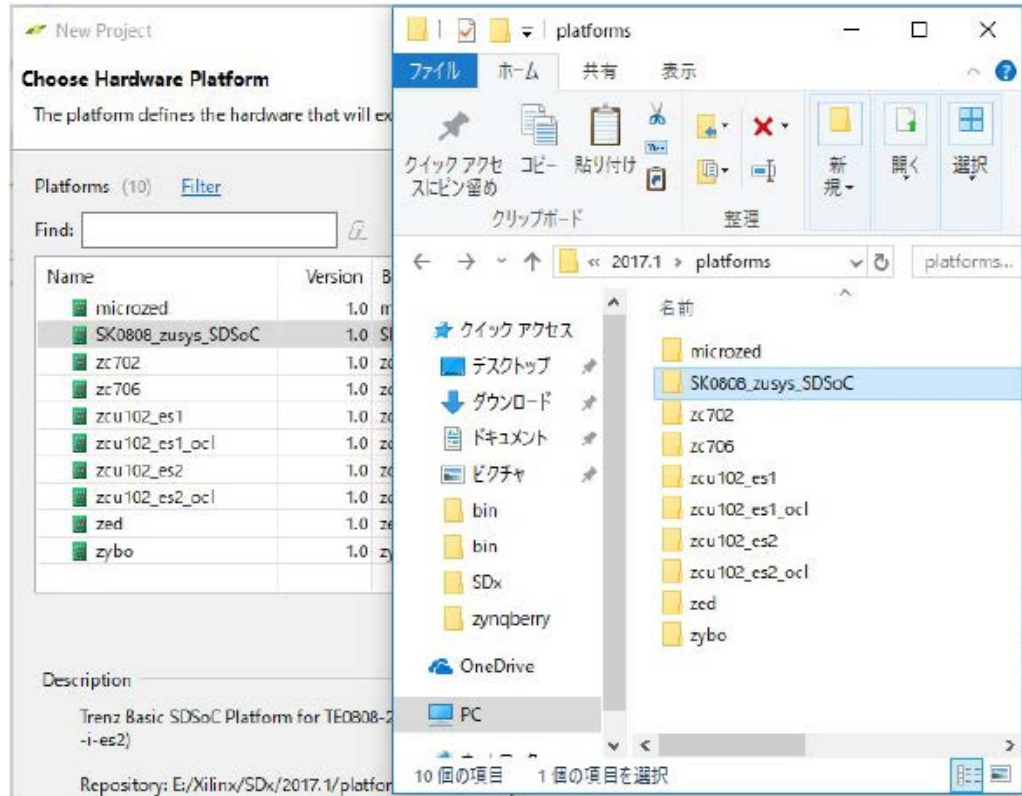
Synthesis | Performance | Resource | Performance Profile | Resource Profile | Dataflow

より大きなFPGAが必要



# 他のTrenzボードで試す

- TE0808 (UltraScale+ SOM)はSDx 2017.1のサンプルあり



- プロジェクトの生成がうまくいっていない？
- おそらく、platformの情報が足りない
- 2016.3, 2016.4, 2017.1, 2018.1で試した限りだと微妙に構成が変わりつつあってこのPlatformの記述や構成が互換がまったくなさそうある。

# 様々な書き込み方

# ① xdevcfgを使う方法

- FPGA(PL)の動的な書き換えが可能

`cat bitstream.bit > /dev/xdevcfg`

- sambaとかでファイル共有しておいて、  
bitstreamを転送して、xdevcfgで書き換えると非常に楽
- /etc/rc.localの中で任意のbitstreamをロードさせることが可能
- PSの設定（クロックなど）が変わる場合に対応できない

## ② Linux上からROMを書き換える方法

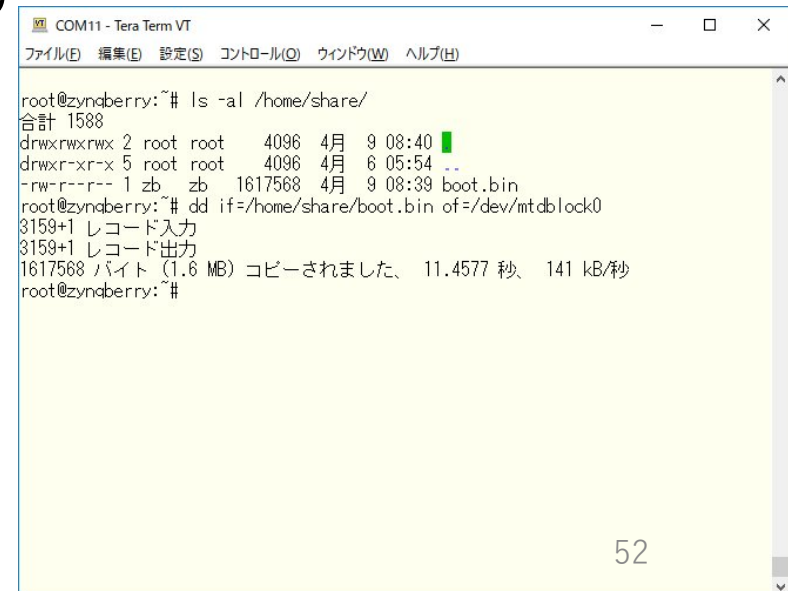
- LinuxがSPI ROMを認識していれば可能
- /dev/mtdblock0 デバイスをddで操作することで、SPI ROMの内容を生で読み書きする

- 書き込み

dd if=/home/share/boot.bin of=/dev/mtdblock0

- 読み出し

dd if=/dev/mtdblock0 of=spirom.bin  
hexdump -Cv spirom.bin -n 512



```
COM11 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)

root@zynberry:~# ls -al /home/share/
合計 1588
drwxrwxrwx 2 root root 4096 4月 9 08:40
drwxr-xr-x 5 root root 4096 4月 6 05:54
-rw-r--r-- 1 zb zb 1617568 4月 9 08:39 boot.bin
root@zynberry:~# dd if=/home/share/boot.bin of=/dev/mtdblock0
3159+1 レコード入力
3159+1 レコード出力
1617568 バイト (1.6 MB) コピーされました、 11.4577 秒、 141 kB/秒
root@zynberry:~#
```

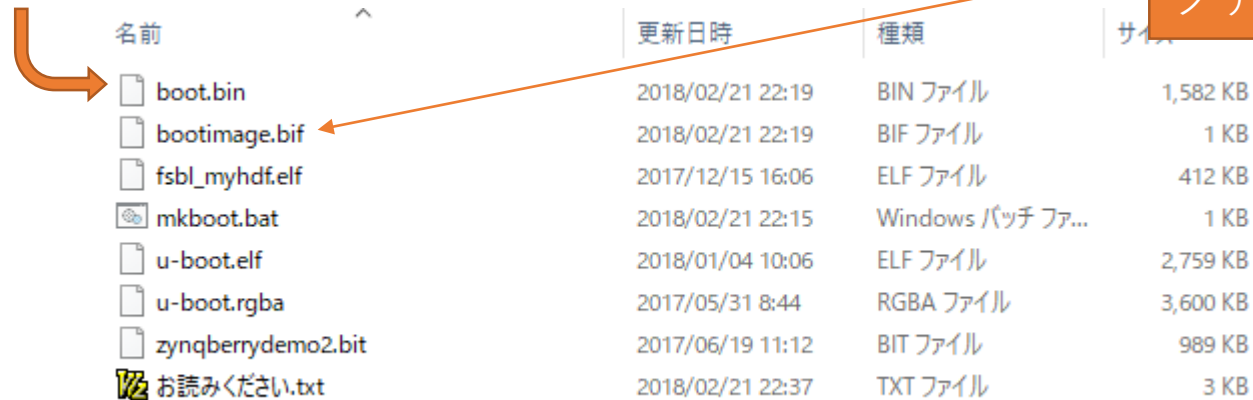
# 作成したFPGAとソフトのROM化

- PLの設定が変わるような場合はROM化しなければならない
- FSBL + bitstream + u-boot → boot.binを生成する
- 特設サイトのファイルを使うのが一番楽

## Zynqberry用のboot.binを作るためのバッチファイル

最終的にはこれが一番重要です

- [zynqberry\\_bootbin.zip](#) Zynqberry用のBoot.binを作るためのファイルです (2.1MBytes)



名前	更新日時	種類	サイズ
boot.bin	2018/02/21 22:19	BIN ファイル	1,582 KB
bootimage.bif	2018/02/21 22:19	BIF ファイル	1 KB
fsbl_myhdf.elf	2017/12/15 16:06	ELF ファイル	412 KB
mkboot.bat	2018/02/21 22:15	Windows バッチ ファ...	1 KB
u-boot.elf	2018/01/04 10:06	ELF ファイル	2,759 KB
u-boot.rgba	2017/05/31 8:44	RGBA ファイル	3,600 KB
zynqberrydemo2.bit	2017/06/19 11:12	BIT ファイル	989 KB
お読みください.txt	2018/02/21 22:37	TXT ファイル	3 KB

bifファイルに結合する  
ファイルが書いてある

# 自分で作ったboot.binの書き込み方

- Vivado 2017.1~2とXSDK2017. 1~2を用意する
- 2017.1~2のTrenzプロジェクトを用意する
- design\_basic\_settings.cmdを以下のように設定する
  - VIVADO\_VERSION=2017.2
  - PARTNUMBER=3
  - SWAPP=myboot
- 生成したboot.binをprebuilt¥boot\_images¥te0726\_m¥mybootに置く
- program\_flash\_binfile.cmdで書き込み開始

# boot.binとLinuxの起動の組み合わせ

- Trenz製のboot.bin
  - petalinuxで作られている
  - prebuildに初めから入っているもの
    - Trenz製のdebian(ulmage.ub)を起動する専用SPI ROMの扱いなどが無いので、いろいろ不便
- 自主製作のboot.bin
  - これから述べる手順でオープンな手順で作ったもの
    - 普通のLinuxカーネル(ulmage)を起動  
Ubuntu 14.10

# まとめ

- MIOは不用意に動かしてはならない
- GPIOを動かすには /sys/class/gpio または /dev/mem
- リファレンスデザインは2017.1および2017.4。  
2017.1がリッチなデザインで、こちらを推奨
- 公式Debianは2017.1のzynqberry demo2の中にある
- スクリプトを使ってプロジェクトを生成する
- ROMの書き込みもスクリプトで行う
- SDSoCはこれから使えるようにする



# 課題

- zynqberry\_demo2のboot.binを書き込めるようにしてください
- Linuxが起動するようにしてください  
(Trenz Debial Linuxでも、特電Ubuntuでも可)
- zynqberry\_bootbin.zipの中のboot.binを書き込んでください
- GPIO2にLEDをつないで、チカチカさせてください
  - やり方は問いません