2023年度版

Ver. 2023-04-11a

Course number: CSC.T341

コンピュータ論理設計 演習(1)
Computer Logic Design Exercise(1)

情報工学系　荒堀喜貴

Yoshitaka ARAHORI, Department of Computer Science

arahori_at_c.titech.ac.jp

Computer Logic Design support page https://www.arch.cs.titech.ac.jp/lecture/CLD/

重要

- 演習は 8:50～10:30 です．8:45までに学術国際情報センター3階 情報工学系計算機室 に集まってください．
- ACRiルームのサイトで，演習の日の 6:00～9:00 と 9:00～12:00 の時間帯を予約をしてください．
- vs から始まるサーバを選択して予約すること．予約する2つの時間帯で同じサーバを予約すること．
  - 各マシンの負荷を下げるために，仮想マシンの名前の最後の2文字が12～15は使わない．具体的には vs001～vs011, vs101～vs111, vs201～vs211, vs301～vs311, vs401～vs411, vs501～vs511, vs601～vs610 から選ぶこと．
  - https://gw.acri.c.titech.ac.jp/wp/

ACRiルームのサーバー計算機

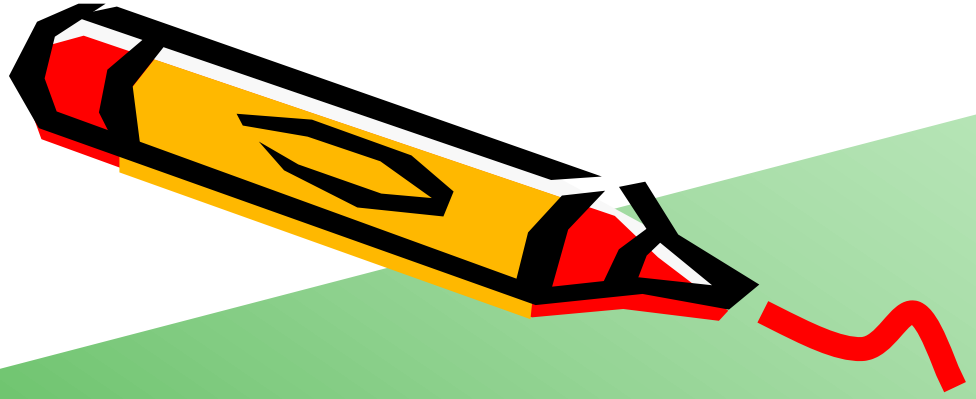
各マシンの負荷を下げるために，仮想マシンの名前の最後の2文字が12～15は使わない．
具体的には vs001～vs011, vs101～vs111, vs201～vs211, vs301～vs311, vs401～vs411, vs501～vs511, vs601～vs610 から選ぶこと．

vs001～vs015　vs101～vs115　vs201～vs215　vs301～vs315　vs401～vs415　vs501～vs515　vs601～vs610

コンピュータ論理設計 演習(Exercise)の注意点

- 演習はACRiルームを利用します。
- 3～4人のグループを作成します．そのグループ内で情報を共有しながら演習を進めてください．
- 問題はグループ内で相談して解決する，あるいは，担当のTA(Teaching Assistant)や教員に質問してください．
- 演習には出席点があります．休まずにきちんと出席しましょう．
- 演習スライドにチェックポイントの図がある場所は，作業を確認してもらう場所です．すべてのチェックポイントをクリアしましょう．

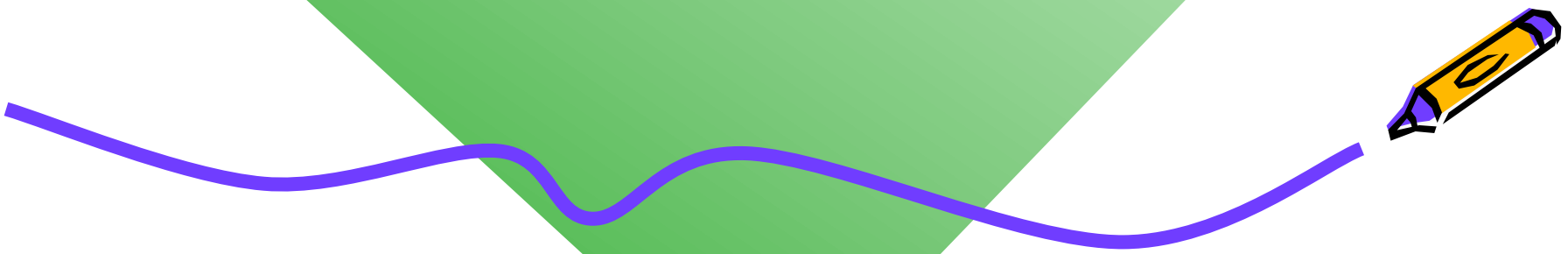
- 演習時間でなくてもACRiルームを利用できます．現在は，1日に4枠（3時間 x 4枠 = 12時間）を利用できます。独自のハードウェア設計などに挑戦しましょう．



Slack の登録をします．（5分）

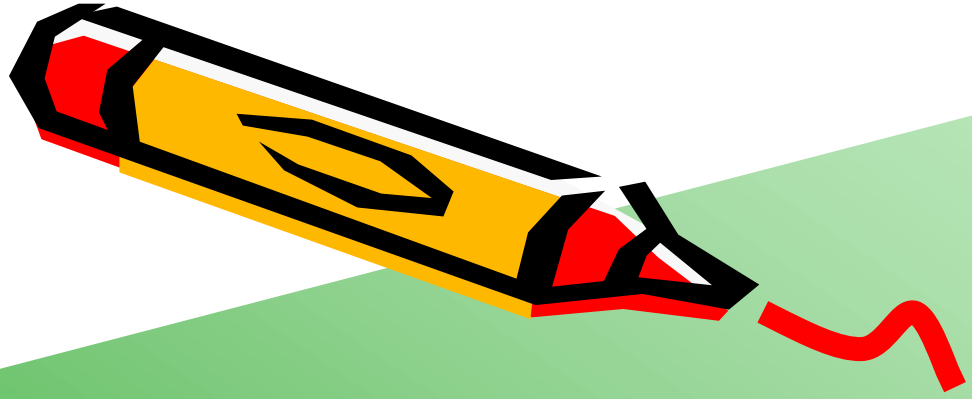
必要であればACRiのアカウント作成

まだ Slack に登録されていない学生は，mアドレスを教員あるいはTAに伝えてください．



ACRiルームのユーザー登録

- 論理回路理論でACRiルームのアカウントを持っていれば，必要ありません．
- まだユーザー登録していなければ，今，次のサイトで新規ユーザーの登録をしてください．
  - https://gw.acri.c.titech.ac.jp/wp/
- 登録時のメールアドレスには m.titech.ac.jp を用いること．

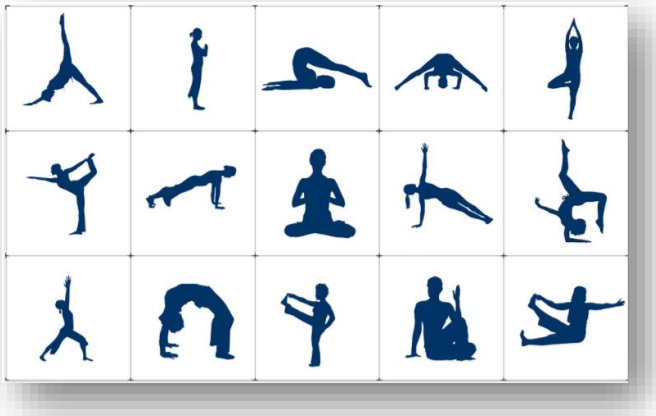


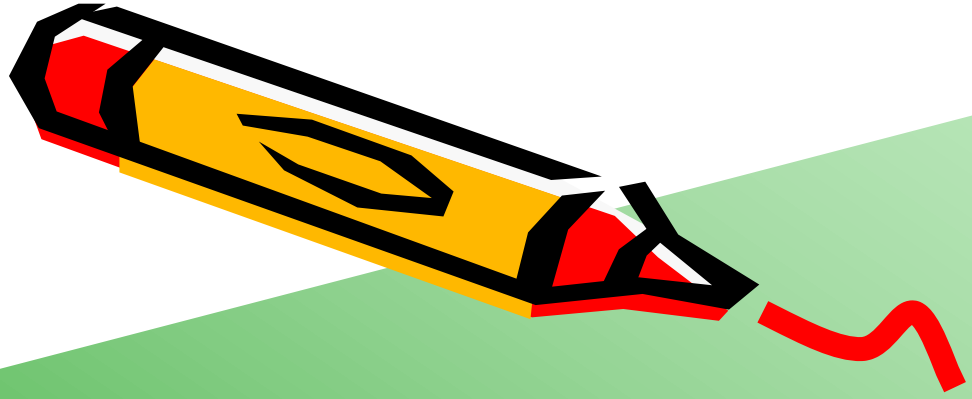
グループと担任の確認（5分）

クラスA 荒堀先生 Arahori
クラスB 篠田 Shinoda
クラスC 下岡 Shimooka
クラスD 山田 Yamada

Exercise(1)

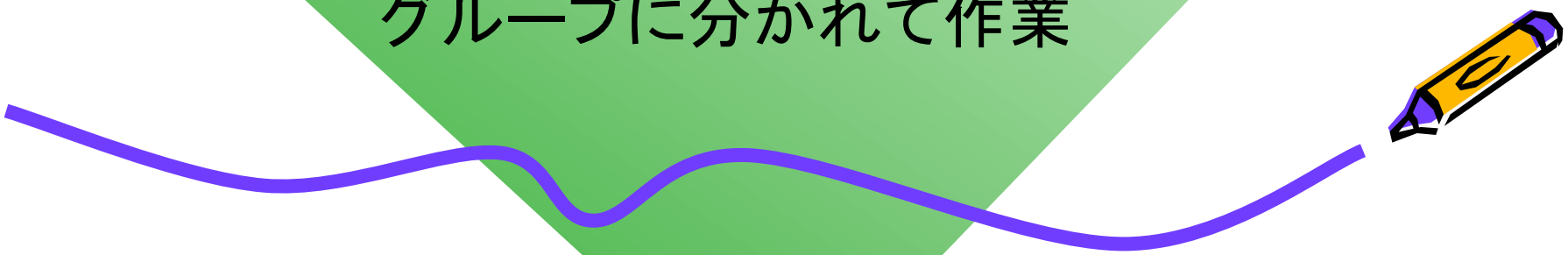
- Project_0
  - ACRiルームの計算機にリモートデスクトップで接続する.
  - FPGAに接続できることを確認する.
- Project_1
  - FPGAをコンフィギュレーションするプロセスを理解する.
  - VIOを用いてFPGAの動作をリモートで確認する方法を理解する.
  - Verilog HDL を編集して，4ビットカウンタの回路を実装して，動作を確認する.





Project_0

グループに分かれて作業



ACRiルームの利用に関して

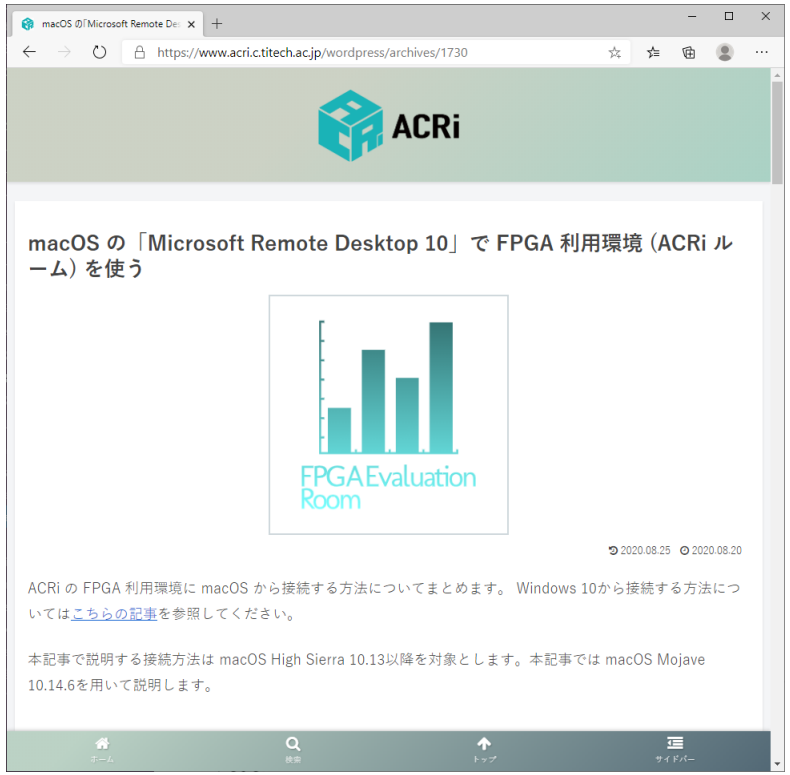
- ssh を除いて、ACRiルームでログインしている仮想マシン（vs001 など）から外部にアクセスすることはできません.
- 仮想マシン（vs001 など）では，ウェブブラウザを起動しないでください.

Windows 10, Windows 11 を利用している場合

- 次のブログ記事を参考に接続する。
- Windows 10 の「リモート デスクトップ接続」で FPGA 利用環境 (ACRi ルーム) を使う
  - https://www.acri.c.titech.ac.jp/wordpress/archives/283

macOS を利用している場合

- 次のブログ記事を参考に接続する。
- macOS の「Microsoft Remote Desktop 10」で FPGA 利用環境 (ACRi ルーム) を使う
  - https://www.acri.c.titech.ac.jp/wordpress/archives/1730
- **Microsoft Remote Desktop** has already been installed on your Mac environment, so refrain from trying to install it again.
- (Mac MS Remote Desktop) "User account: Add User Account..." cannot be used in your environment.



Ubuntu でターミナルとVivado を立ち上げる

- リモート デスクトップ接続した Ubuntu のターミナルを立ち上げる.

左上に表示された Activities という部分が、Windows で言うところのスタートメニューになります。クリックすると画面が暗くなり、いくつかのショートカットが表示されます。まずはここからターミナルを起動しましょう。中央の検索窓に「terminal」ないし「term」と入力すると、ターミナルが見つかりますので、クリックして起動します。

ターミナルを起動するときの様子。

- ターミナルで次のコマンドを入力し，FPGA開発用のソフトウェア Vivado を起動する.
  - 「Vivado 2022.2」を利用する.

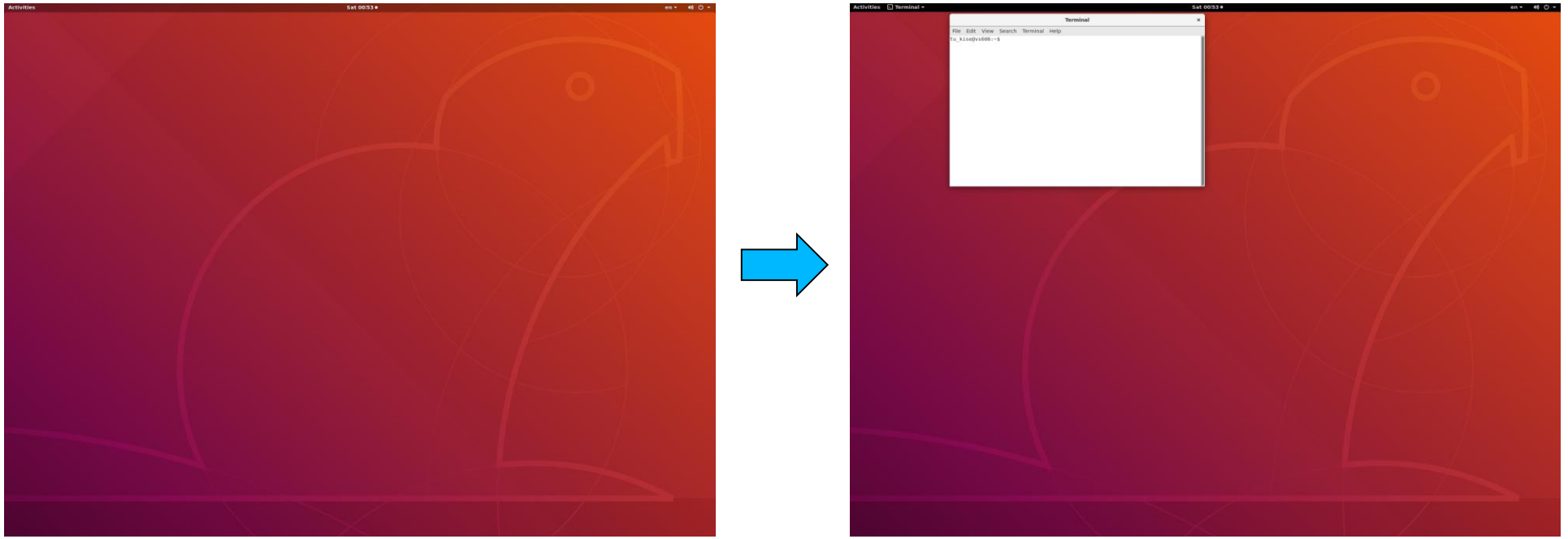
```
$ source /tools/Xilinx/Vivado/2022.2/settings64.sh
$ vivado &
```

- ターミナルの起動，Vivado の起動に関する詳細は、次の記事を参考にすること.

  https://gw.acri.c.titech.ac.jp/wp/manual/vivado-vitis

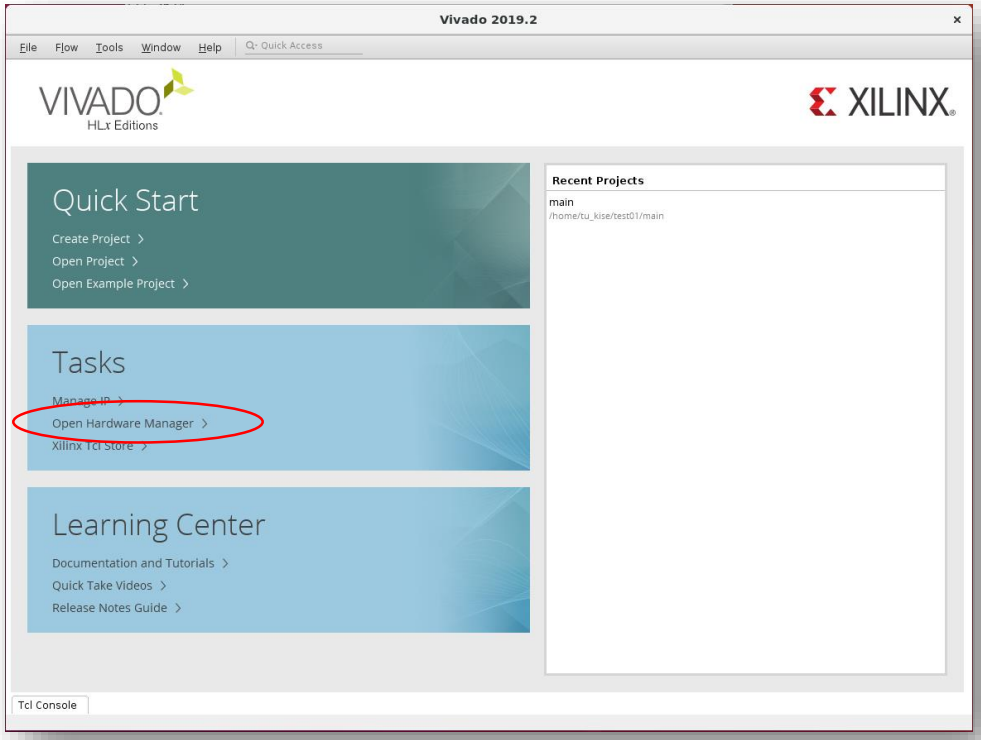
補足：Ubuntu でターミナルを立ち上げる別の方法

- Ubuntu のデスクトップで Ctrl + Alt + T のキーを同時に押すとターミナルが立ち上がる.
- Mac のリモートデスクトップで開いている Ubuntu では control + option + T のキーを同時に押すとターミナルが立ち上がる.

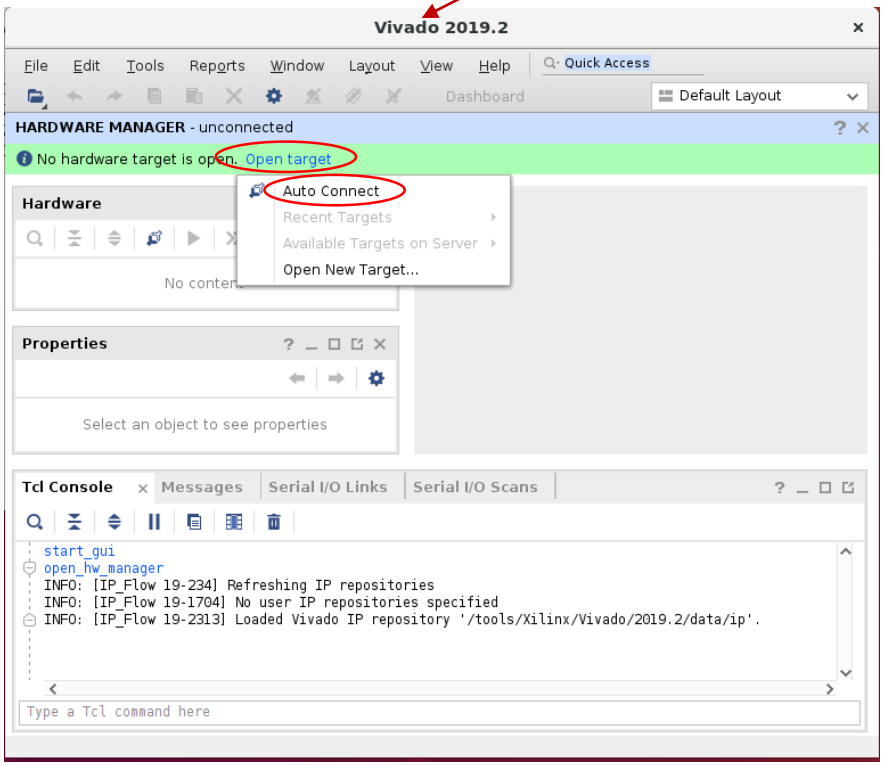


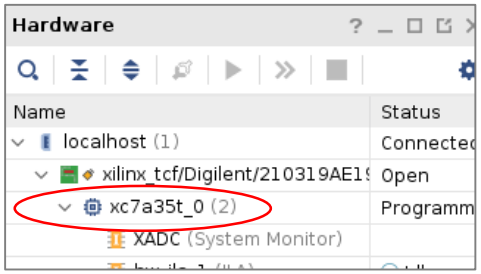
FPGAに接続できることを確認

- Invoke Vivado 2022.2 by typing "vivado &"
- Select **Open Hardware Manager**
- Click **Open target** and select **Auto Connect**
- **xc7a35t** will appear
- Please exit Vivado



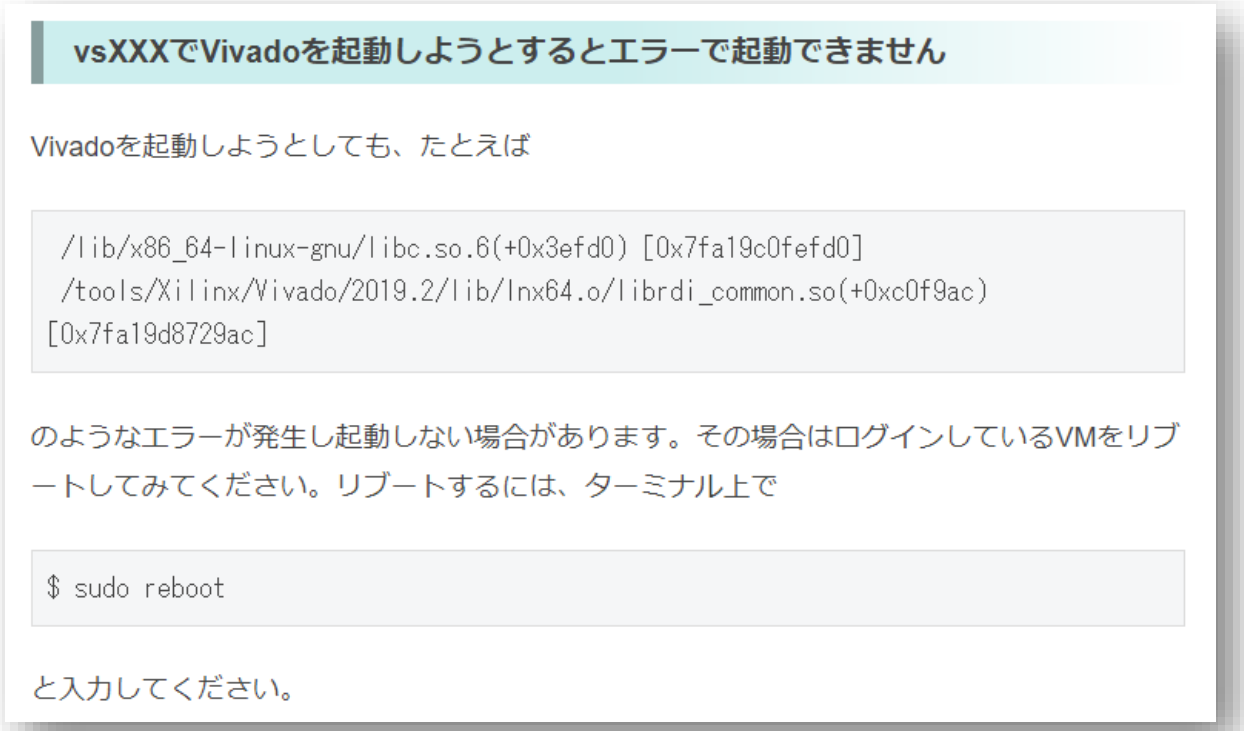
This should be Vivado 2022.2





FPGAに接続できない場合

- ターミナルから，予約したサーバを再起動して，再挑戦.
  - sudo reboot
- ACRi ルームに関するよくある質問 (FAQ) が助けてくれる？
  - https://gw.acri.c.titech.ac.jp/wp/manual/faq



vsXXXでVivadoを起動しようとするとエラーで起動できません

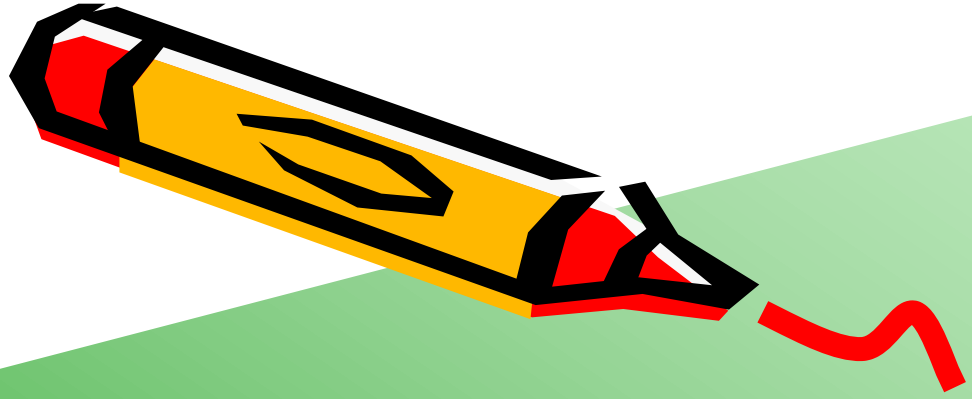
Vivadoを起動しようとしても、たとえば

```
 /lib/x86_64-linux-gnu/libc.so.6(+0x3efd0) [0x7fa19c0fefd0]
 /tools/Xilinx/Vivado/2019.2/lib/lnx64.o/librdi_common.so(+0xc0f9ac)
[0x7fa19d8729ac]
```

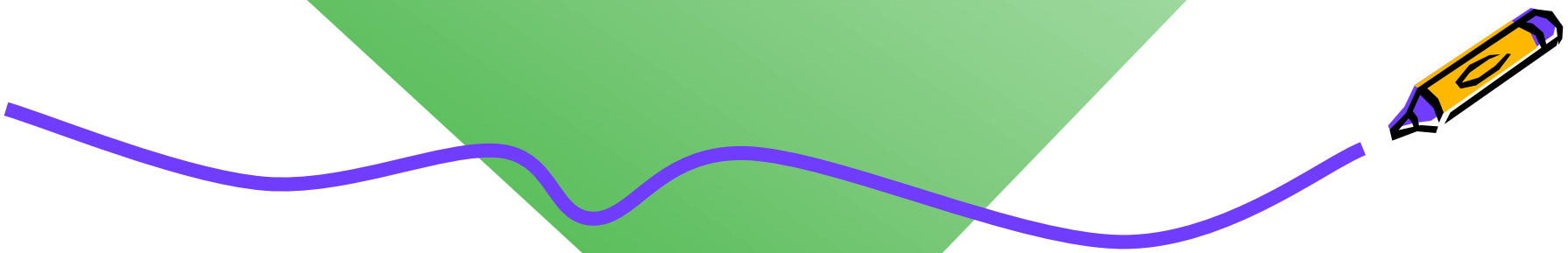
のようなエラーが発生し起動しない場合があります。その場合はログインしているVMをリブートしてみてください。リブートするには、ターミナル上で

```
$ sudo reboot
```

と入力してください。



Project_1



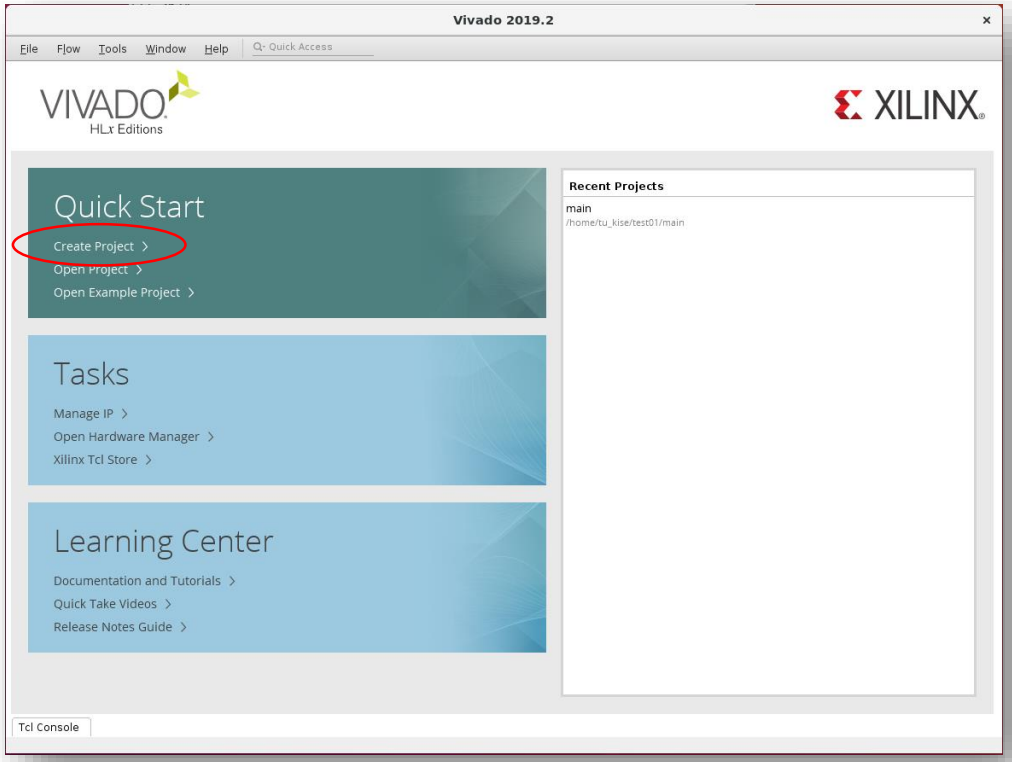
Ubuntu でターミナルとVivado を立ち上げる

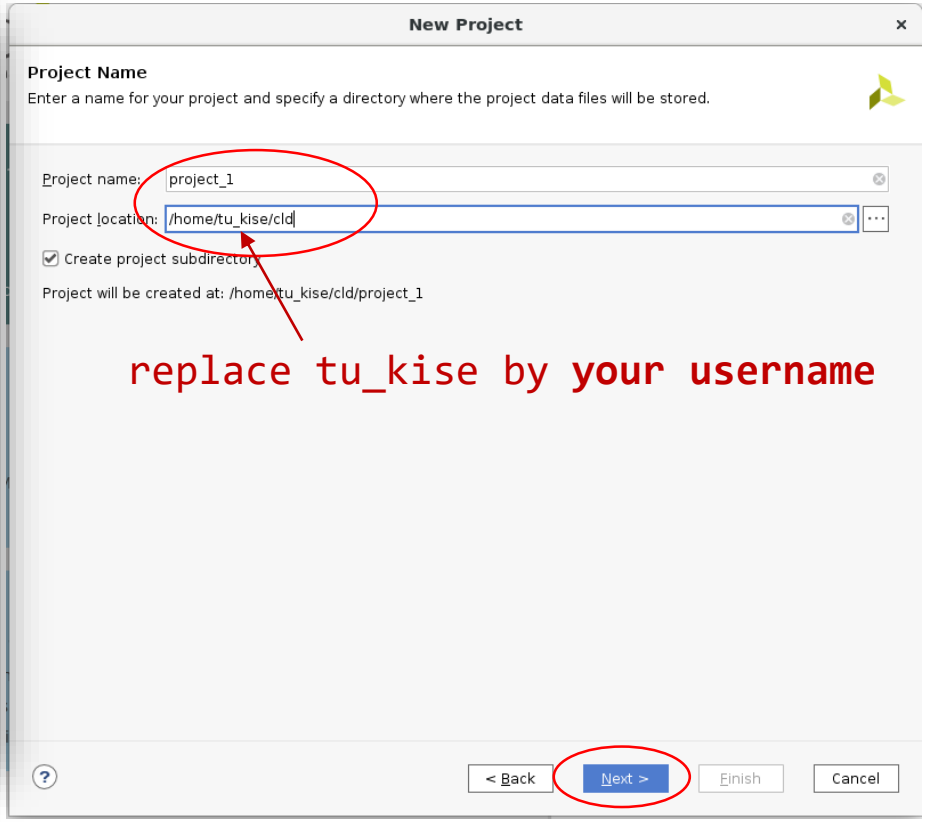
- 次の記事を参考にターミナルを起動して，Vivadoを起動する．
- サーバで Vivado と Vitis (または SDK) を使用する
  - https://gw.acri.c.titech.ac.jp/wp/manual/vivado-vitis
  - 「Vivado 2022.2 」を利用する．

Create a new project (1/3)

- Invoke Vivado 2022.2 by typing "vivado &"
- Select Create Project, Click **Next**
- Project name "project_1" and location "/home/username/cld" are selected.
  - Check "Create project subdirectory".

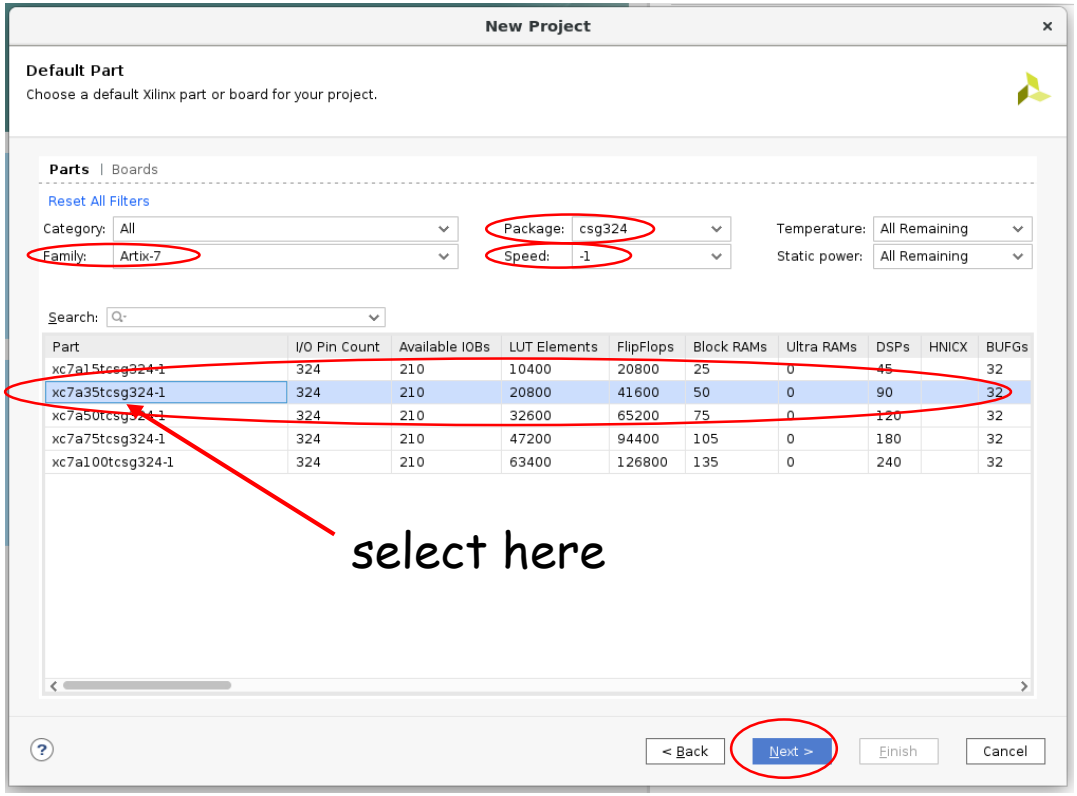
```
$ source /tools/Xilinx/Vivado/2022.2/settings64.sh
$ vivado &
```





Create a new project (2/3)

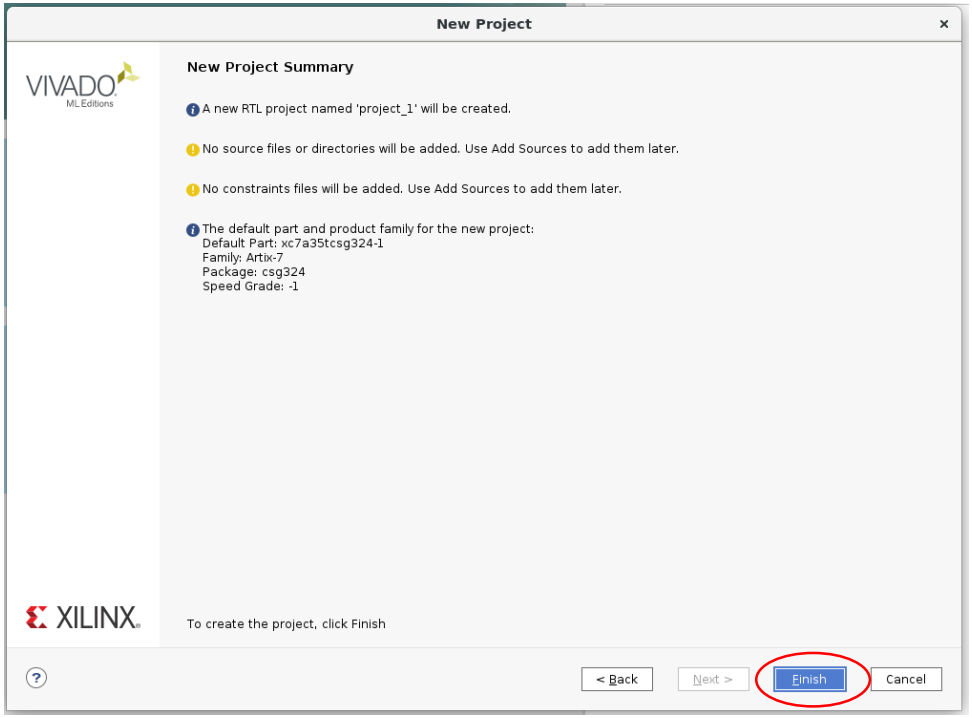
- In Project Type window, select **RTL project** and click **Next**.
- In **Add Sources** window, click Next.
- In **Add Constraints (optional)** window, click Next.
- In **Default Part** window, select **Artix-7** in Family, select **csg324** in Package, select **-1** in Speed. Then select **xc7a35tcsg324-1** and click **Next**.

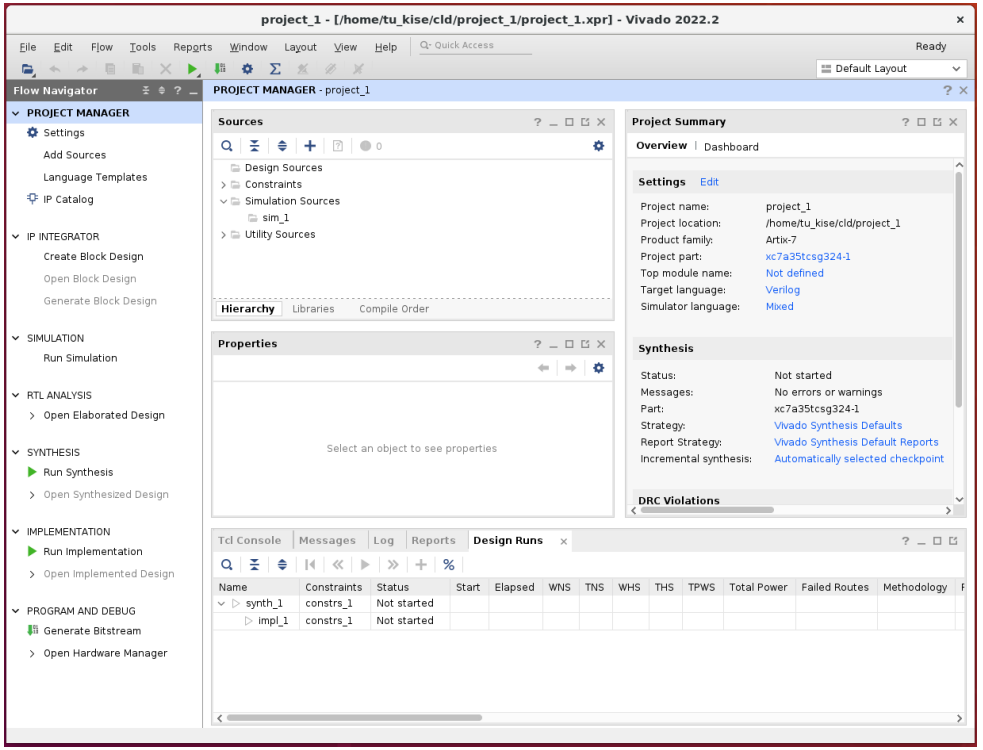


Arty A7に搭載されている
FPGAの型番は
xc7a35tcsg324-1 です.

Create a new project (3/3)

- Confirm summary in **New Project Summary** window, and click **Finish**.





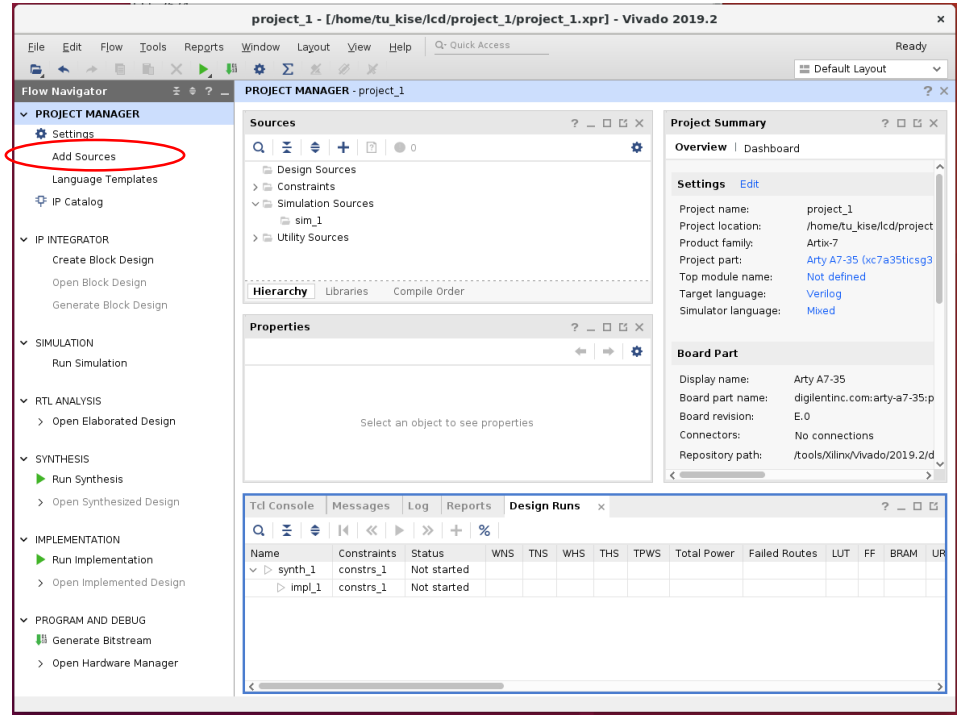
Bitfile generation and FPGA configuration (1/7)

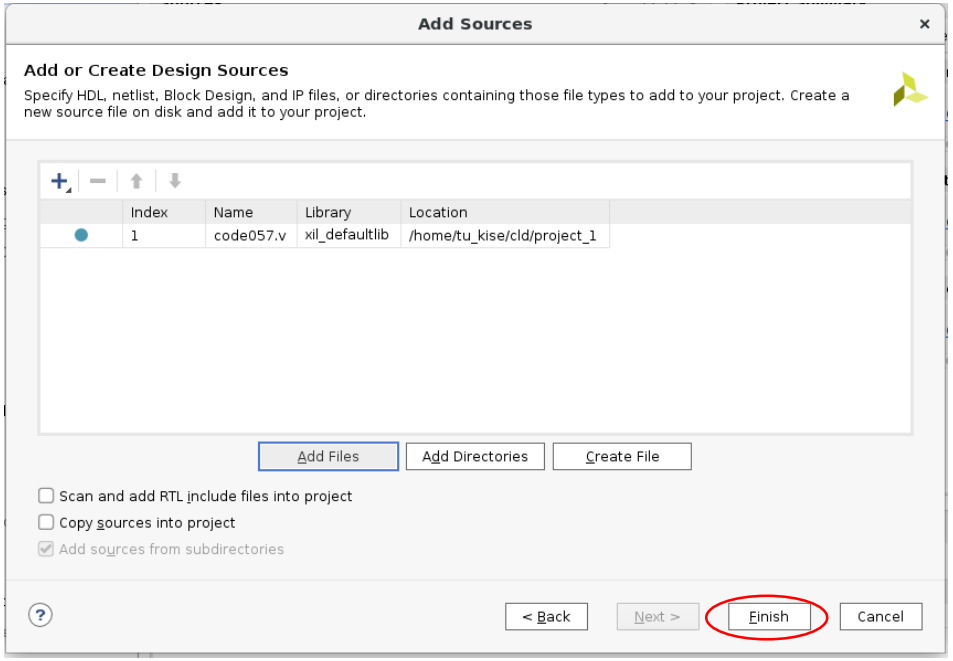
- ターミナルで，ファイルをコピーする．
- /home/tu_kise/cld/2023/ に保存されている code057.v と main11.xdc を，作成したプロジェクトのディレクトリ ~/cld/project_1 にコピーする．
- /home/tu_kise は automount のディレクトリなので，アクセスしないとファイルが見えない．tabキーによる補完がうまく動作しないことがあるので注意する．
- 最後の ls コマンドで，code057.v と main11.xdc が表示されることを確認．

```
$ ls /home/tu_kise
$ cd ~/cld/project_1
$ cp /home/tu_kise/cld/2023/code057.v .
$ cp /home/tu_kise/cld/2023/main11.xdc .
$ ls
code057.v   project_1.cache  project_1.ip_user_file   project_1.xpr
main11.xdc  project_1.hw     project_1.sim
```

Bitfile generation and FPGA configuration (2/7)

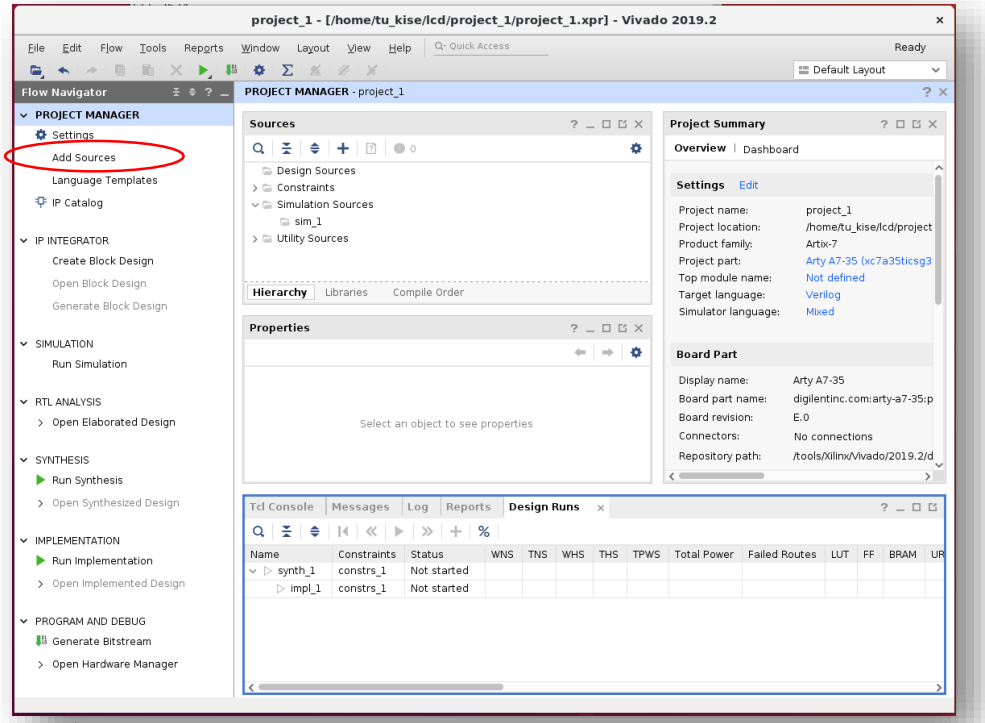
- Click **Add Sources**, then select **Add or create design sources** and click **Next**.
- In **Add or Create Design Sources** window, click **Add Files**, select **code057.v** in project_1 directory, and click **OK**.
- Click **Finish**.

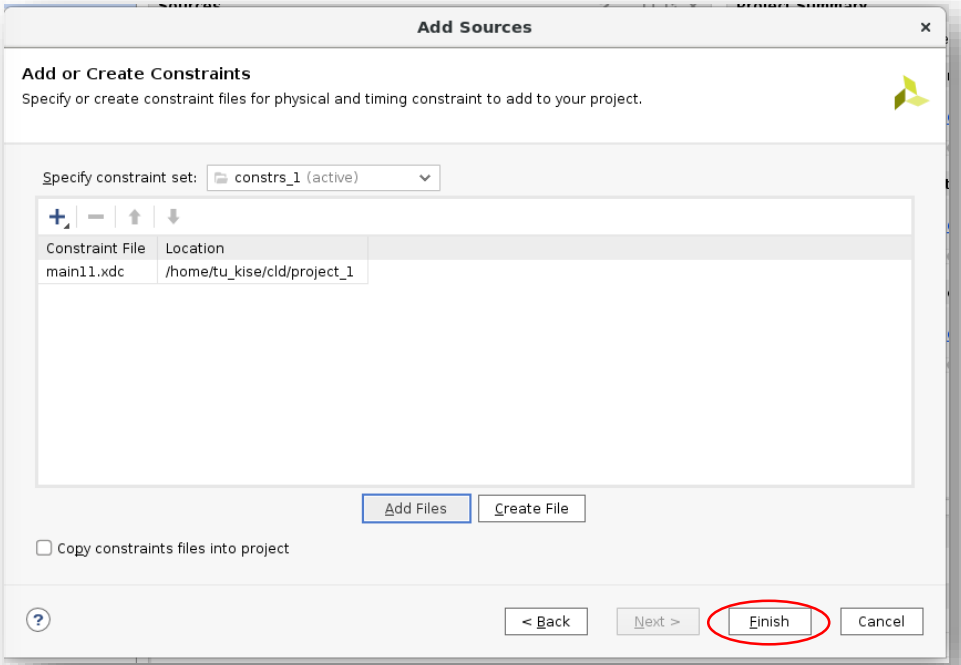




Bitfile generation and FPGA configuration (3/7)

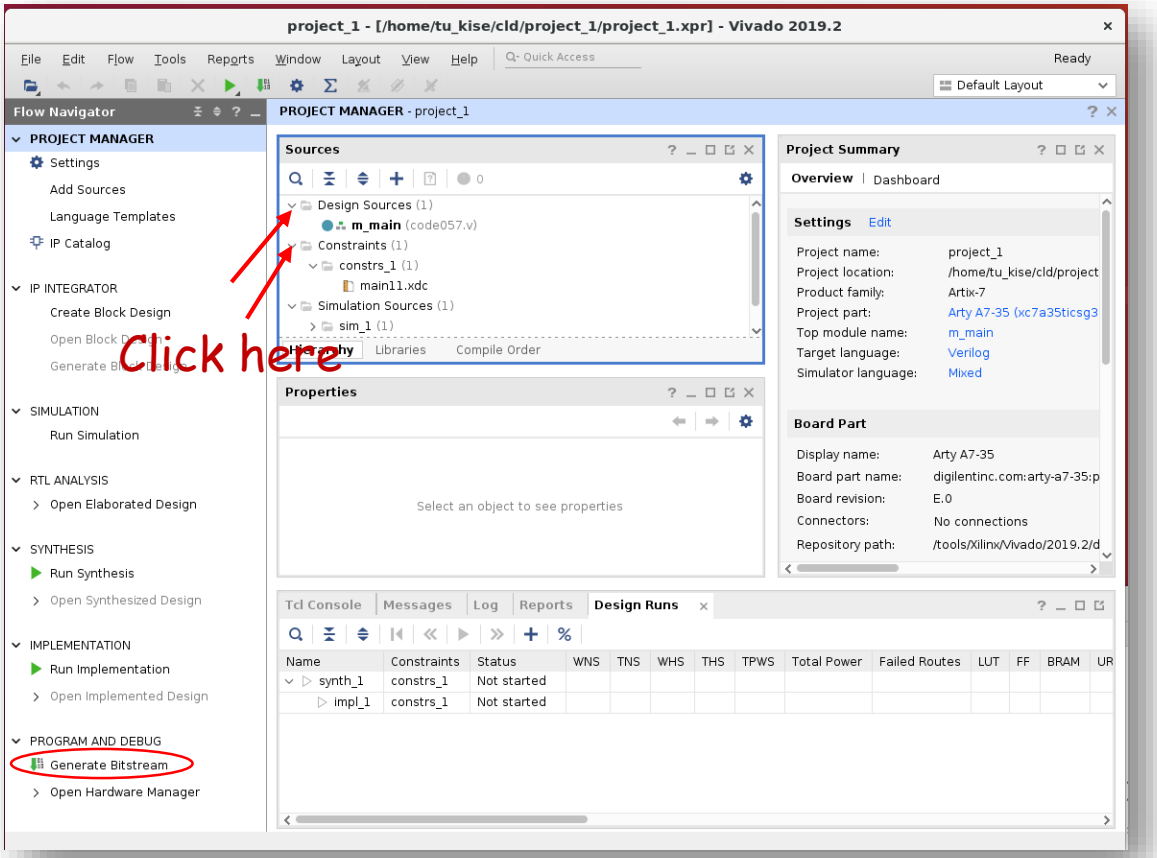
- Click **Add Sources**, then select **Add or create constraints** and click **Next**.
- In **Add or Create Design Sources** window, click **Add Files**, select **main11.xdc** in project_1 directory, and click **OK**.
- Click **Finish**.

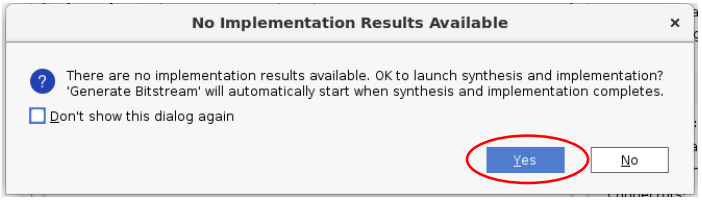


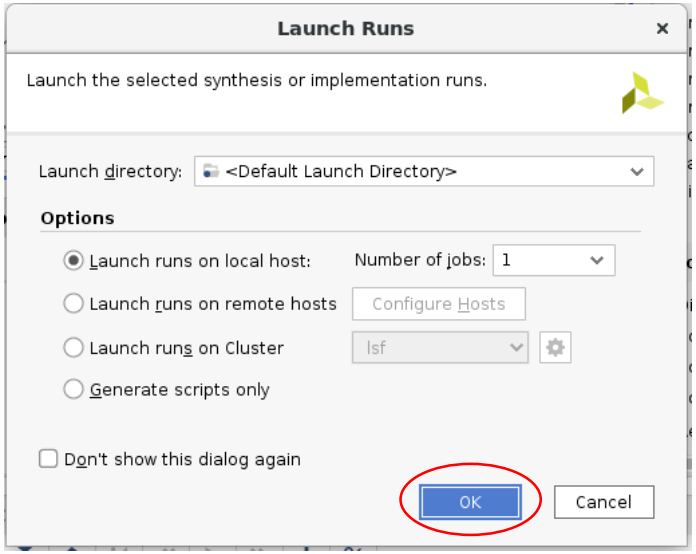


Bitfile generation and FPGA configuration (4/7)

- Click Design Sources then you will see **code057.v** in the project.
- Click Constraints then you will see **main11.xdc** in the project.
- Click **Generate Bitstream**, click **Yes**, click **OK**, and wait around two minutes.

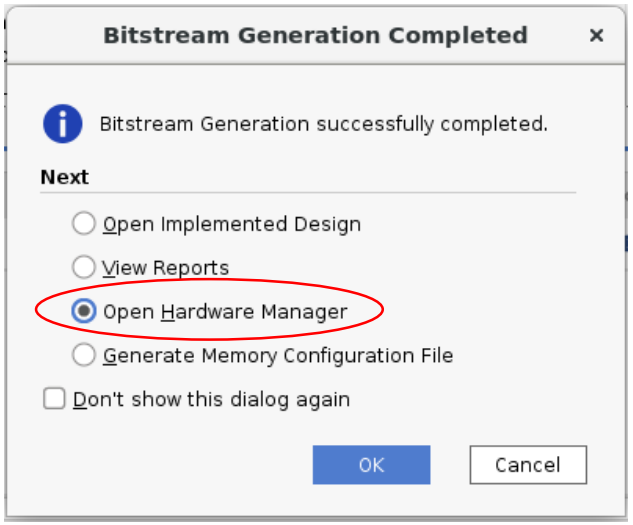


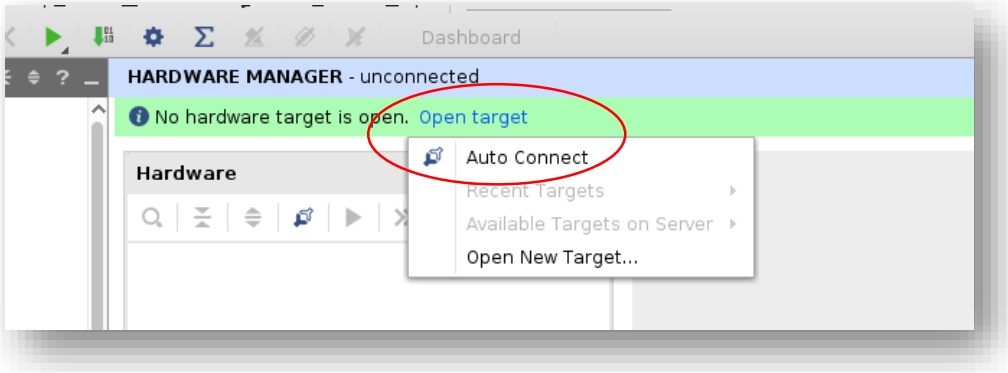


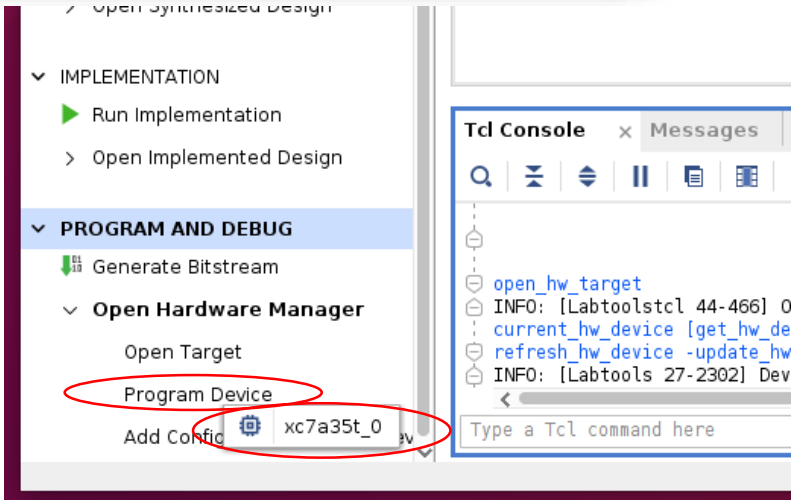


Bitfile generation and FPGA configuration (5/7)

- Select **Open Hardware Manager**, and click **OK**.
- Click **Open Target** in the green bar and select **Auto Connect**.
- Click **Program Device** and select **xc7a35t_0**.

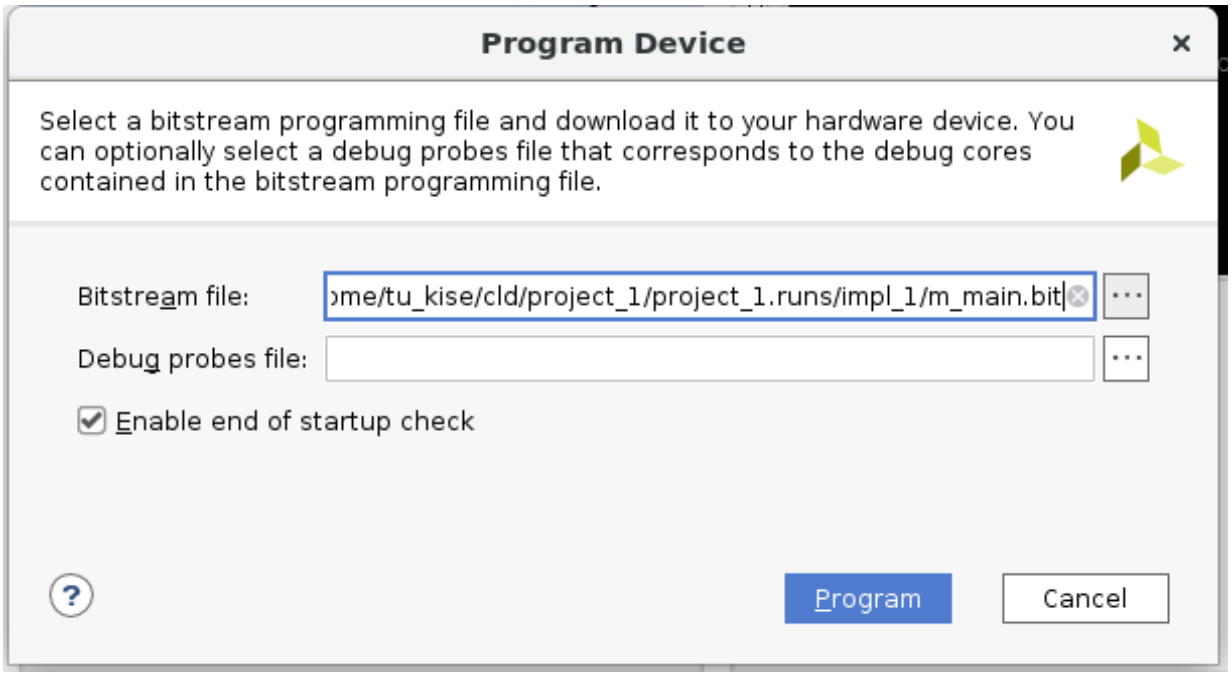






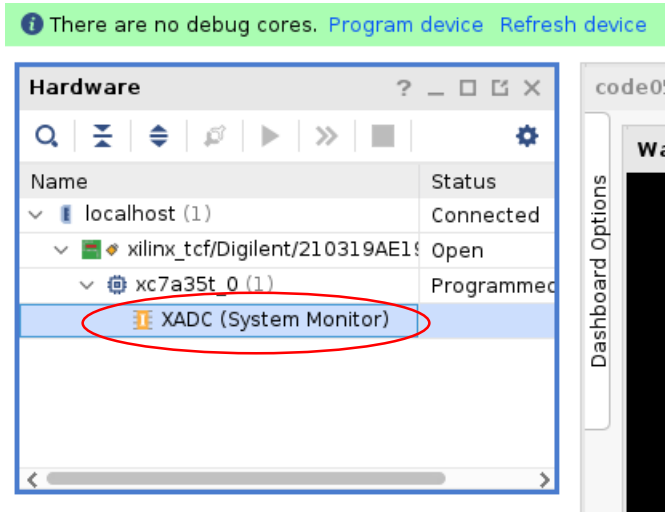
Bitfile generation and FPGA configuration (6/7)

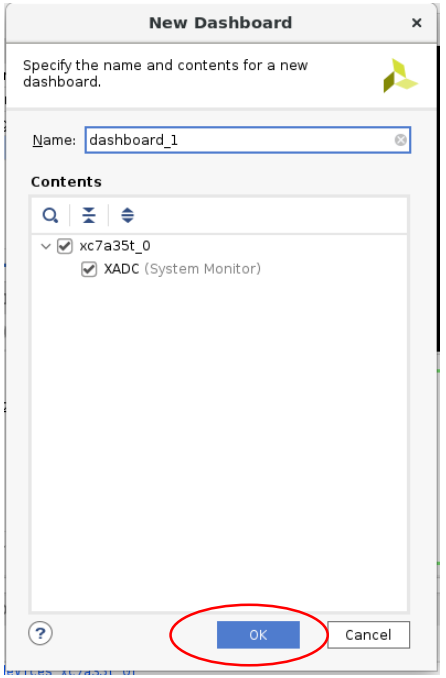
- In **Program Device** window, select m_main.bit in project_1.runs/impl_1 directory (please use the default setting).
- Click **Program**.
- Well done! Your FPGA board will be running.

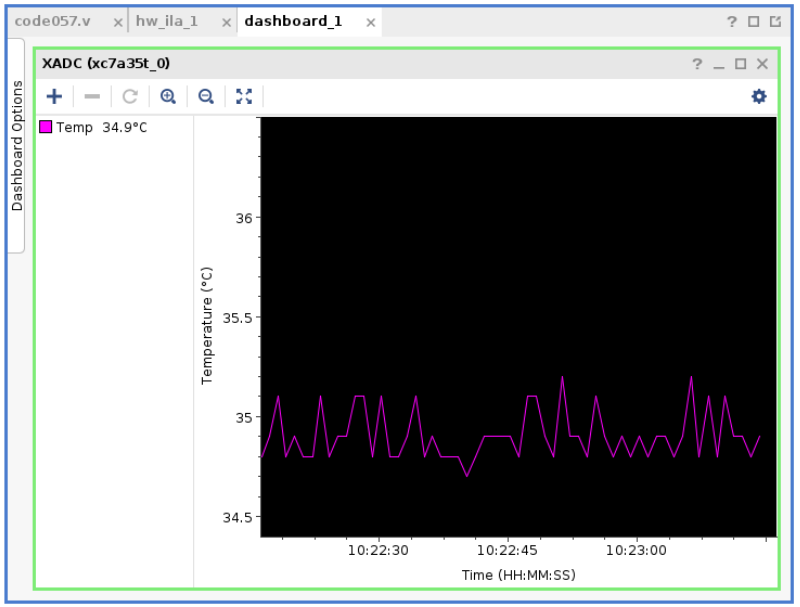


Bitfile generation and FPGA configuration (7/7)

- Double click XADC (System Monitor)
- Clock OK in New Dashboard window
- You can see the temperature of the FPGA chip



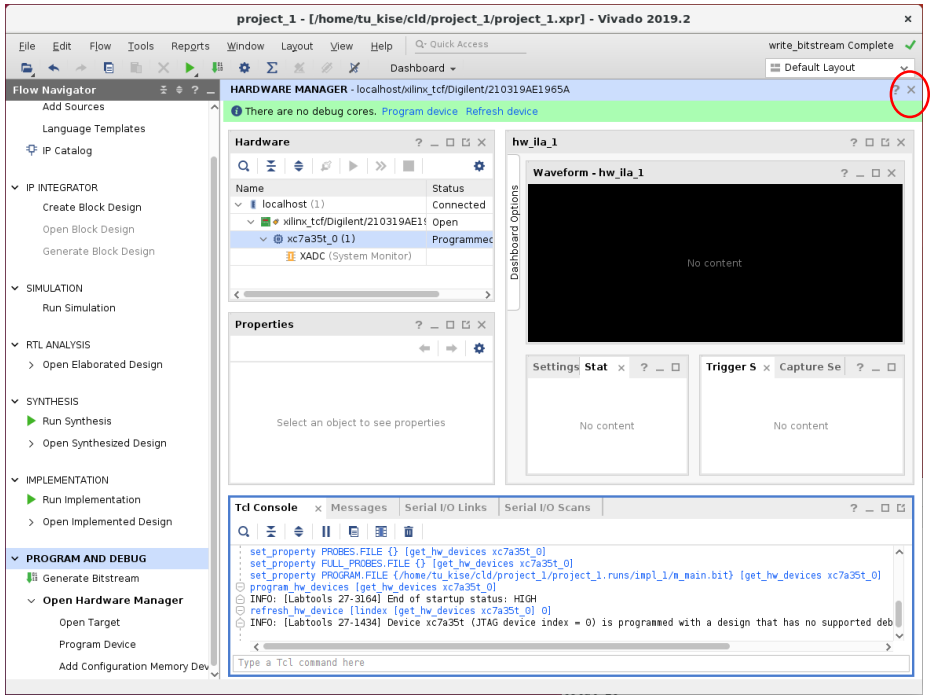




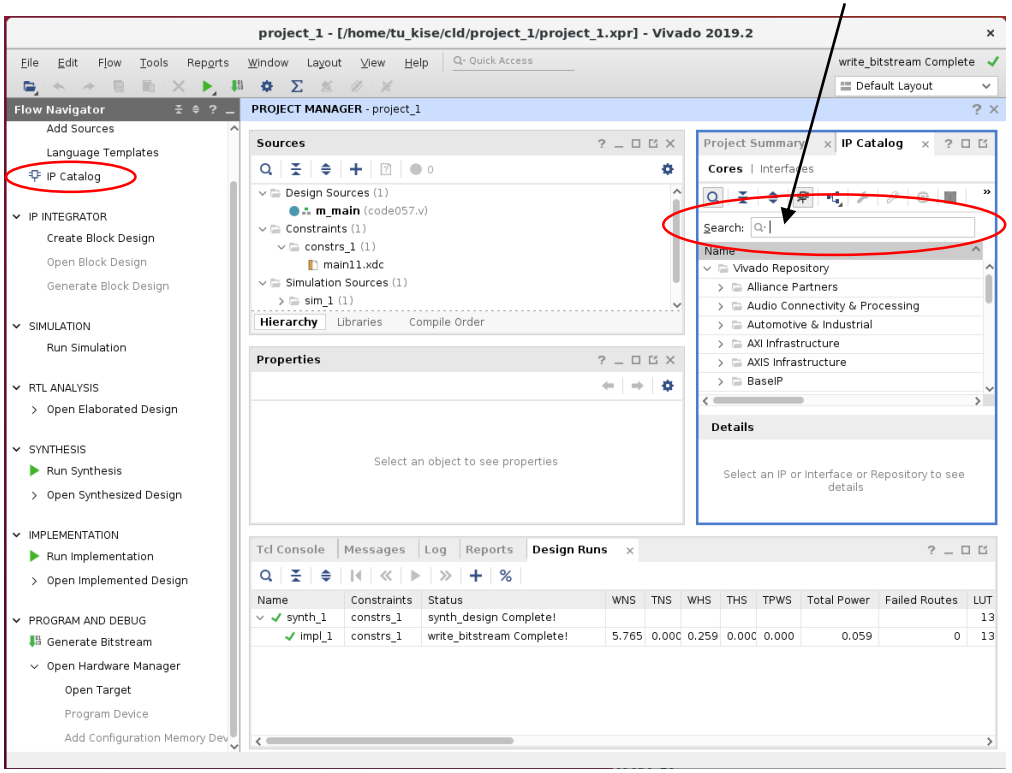
ここまでが，「FPGAをコンフィギュレーションするプロセスを理解する．」

VIOを用いてFPGAの動作をリモートで確認 (1/6)

- Click **X** and click **OK** in **Confirm Close** window to close HARDWARE MANAGER.
- Click **IP Catalog**, and type **vio** in **Search** area to use VIO (Virtual Input/Output)

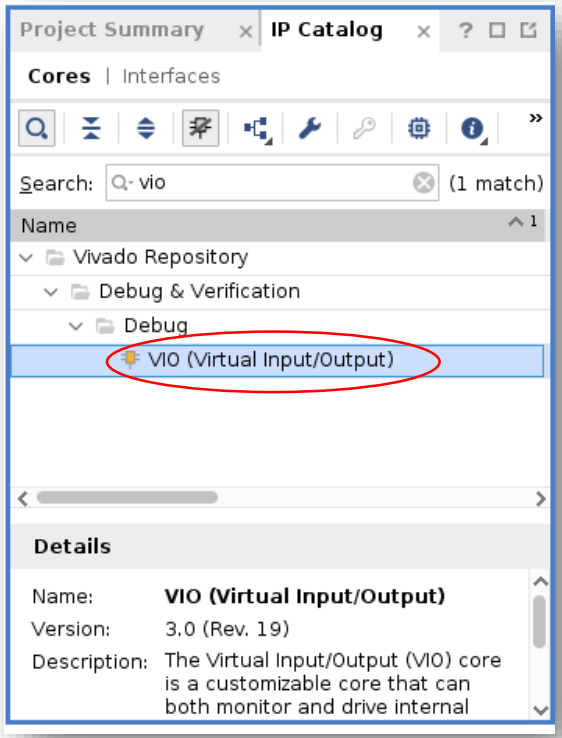


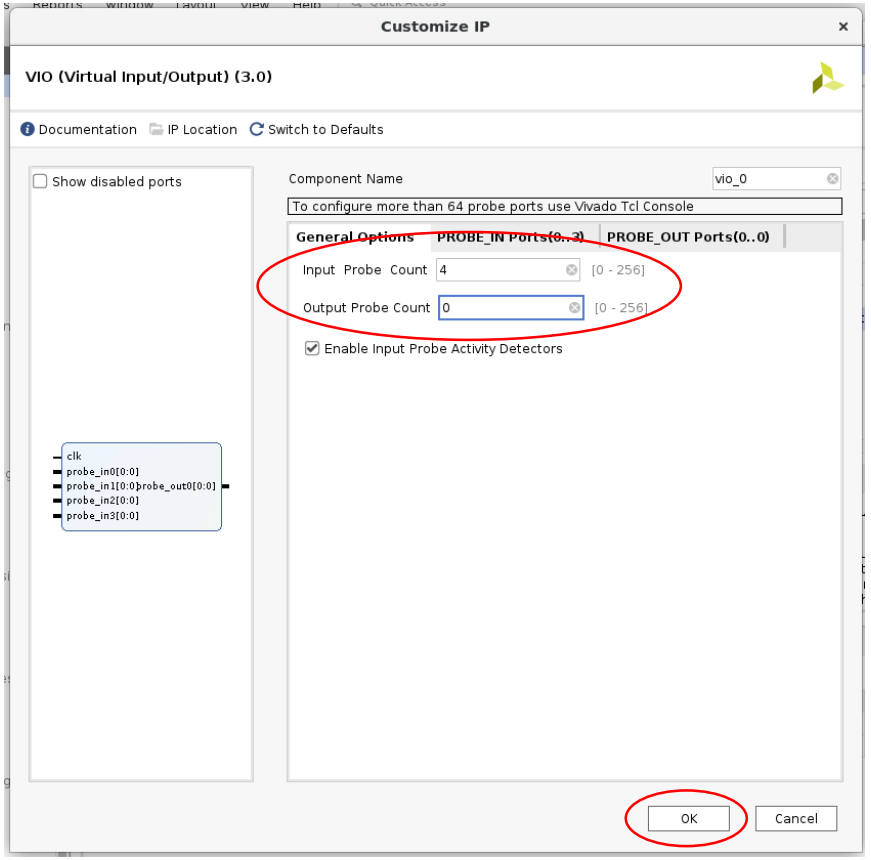
Type vio



VIOを用いてFPGAの動作をリモートで確認 (2/6)

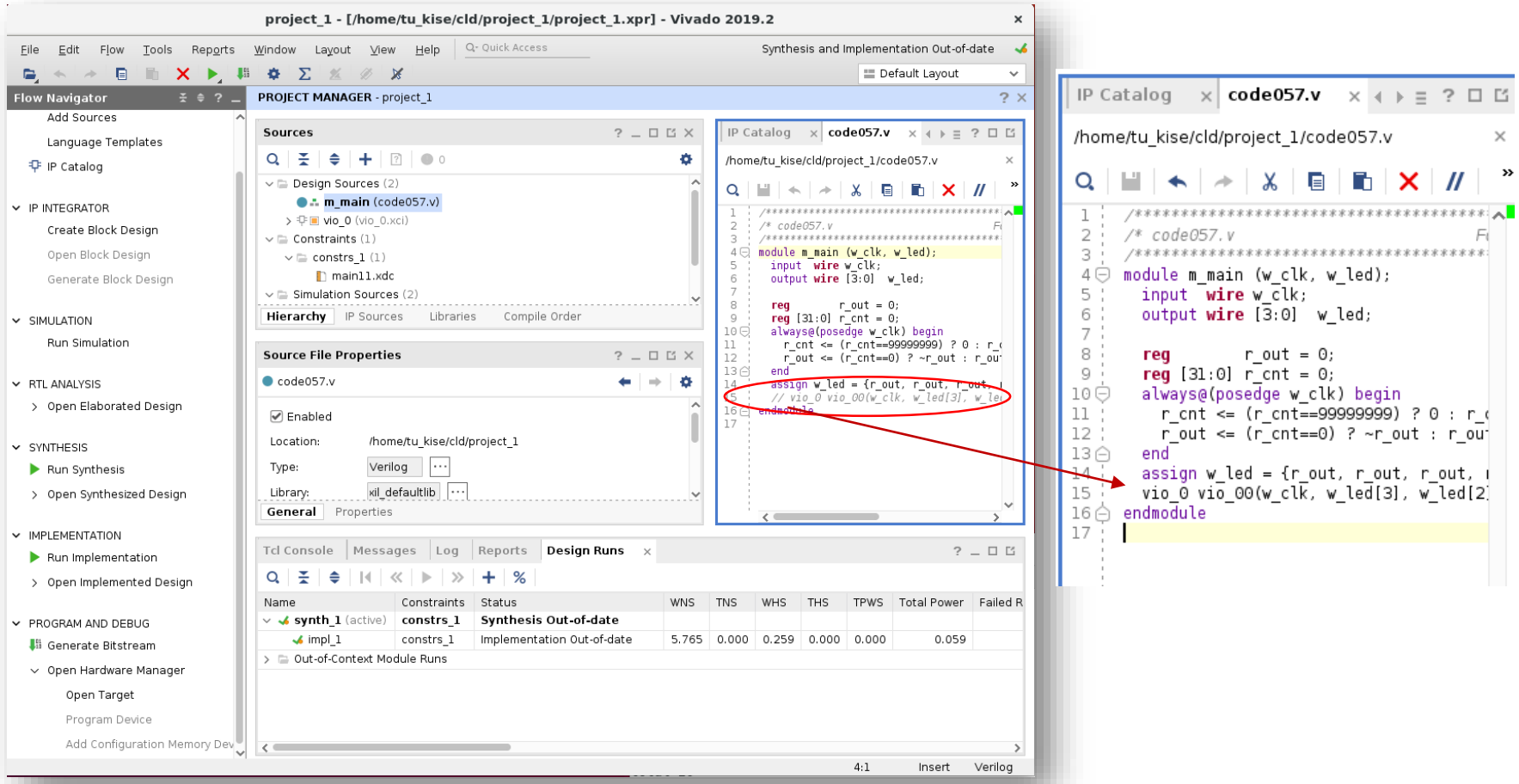
- Double click **VIO (Virtual Input/Output)**.
- In **VIO** window, set **4** for Input Probe Count, set **0** for Output Probe Count, and click **OK**.
- Click **Generate**, and click **OK** if asked in **Generate Output Products** window.





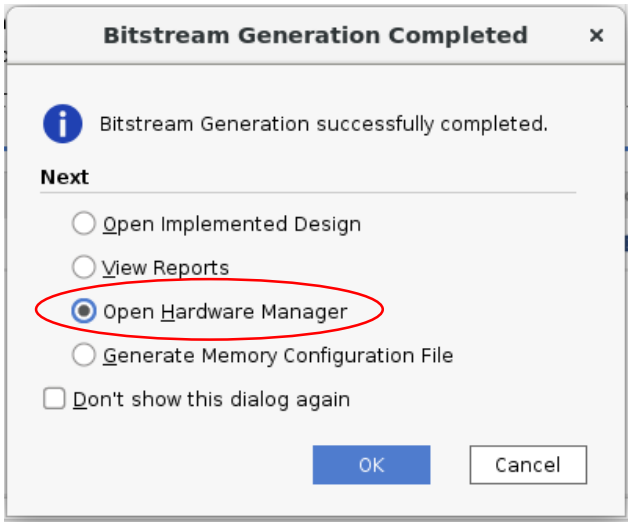
VIOを用いてFPGAの動作をリモートで確認 (3/6)

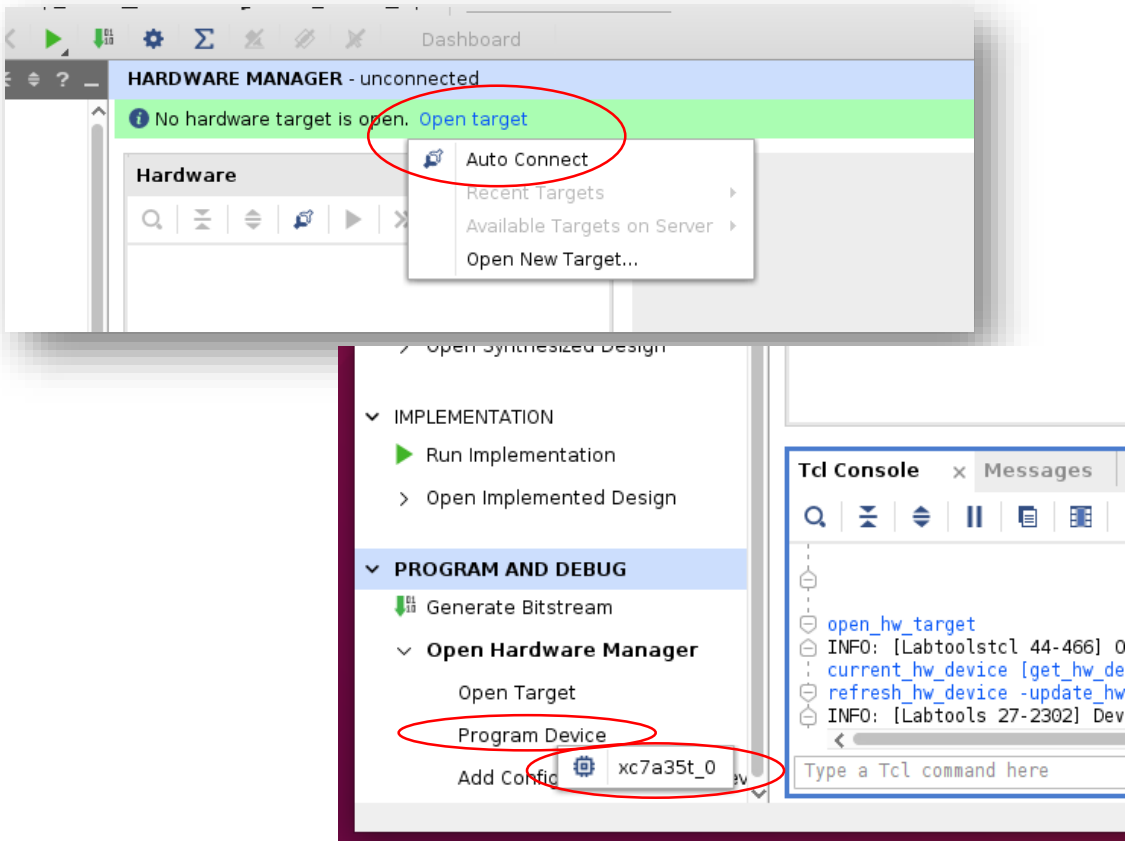
- You see vio_0 in your **Design Sources**
- Double click **m_main** to edit code057.v.
- 15行目のコメントアウトを取り除き(// を削除する), Ctrl + s で保存する。



VIOを用いてFPGAの動作をリモートで確認 (4/6)

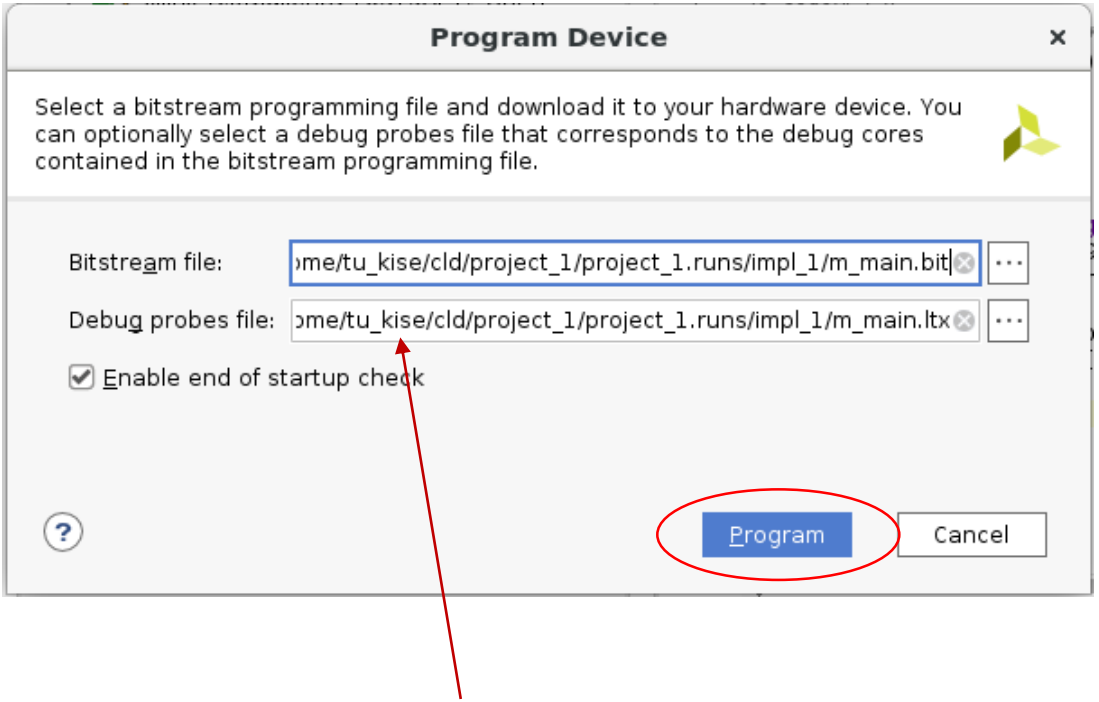
- Click **Generate Bitstream**, click **Yes**, click **OK**, and wait about three minutes.
- Select **Open Hardware Manager**, and click **OK**.
- Click **Open Target** in the green bar and select **Auto Connect**.
- Click **Program Device** and select **xc7a35t_0**.





VIOを用いてFPGAの動作をリモートで確認 (5/6)

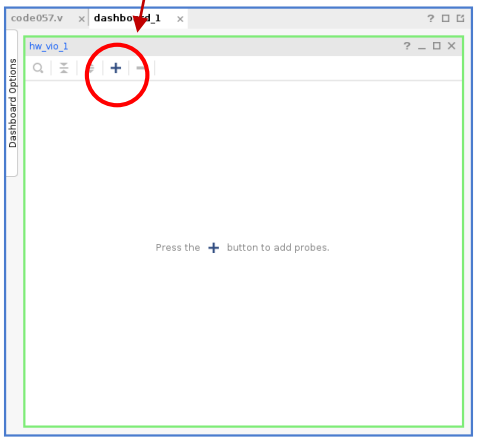
- In **Program Device** window, Bitstream file and Debug probes file will be set automatically.
- Click **Program** and configure FPGA.

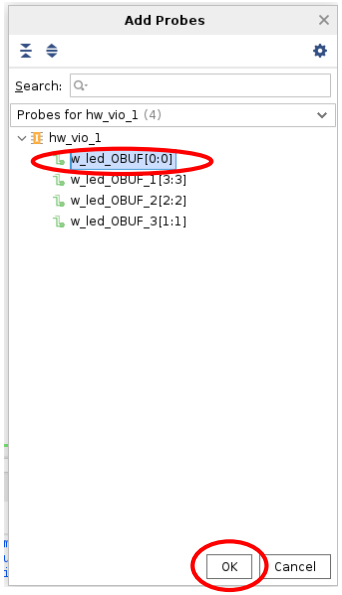


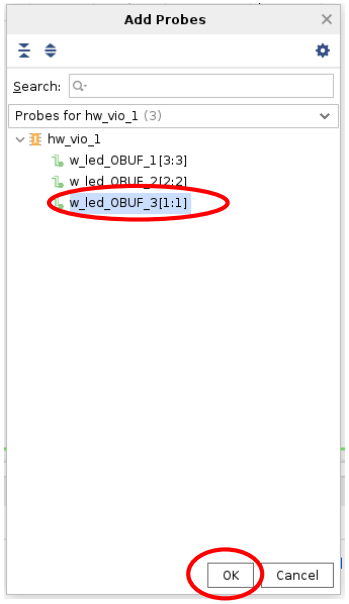
自動的にVIO等のデバッガを制御するためのDebug probes fileが指定される.

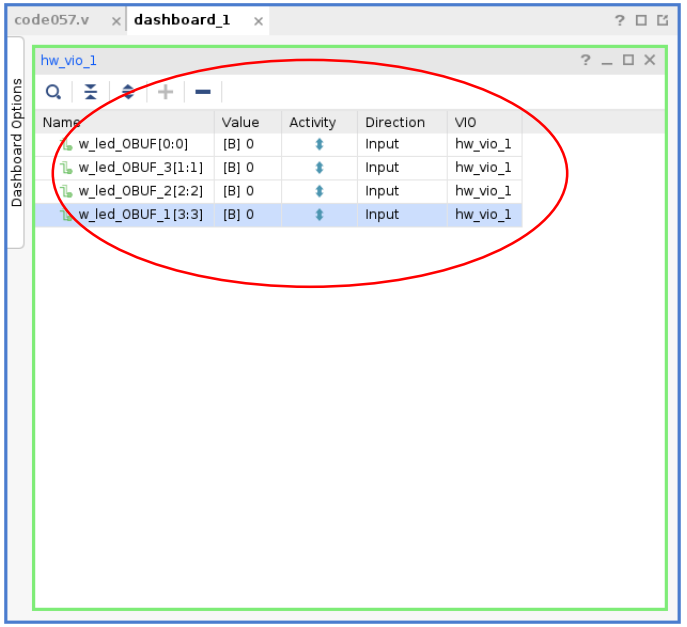
VIOを用いてFPGAの動作をリモートで確認 (6/6)

- Click + button in **hw_vio_1** window. Select w_led_OBUF[0:0] by clicking it. Then click OK.
- Click + button. Select w_led_OBUF3[1:1] by clicking it. Then click OK.
- Click + button. Select w_led_OBUF2[2:2] by clicking it. Then click OK.
- Click + button. Select w_led_OBUF1[3:3] by clicking it. Then click OK.
- You will see the values of four LEDs change every second!









Virtual Input/Output (VIO)

- Virtual Input/Output (VIO) について次の連載を参照すること.
- FPGA をもっと活用するために IP コアを使ってみよう
  - https://www.acri.c.titech.ac.jp/wordpress/archives/40

Code057.v

- w_clk は 100MHz のクロック信号
- 32ビットのレジスタ r_cnt を，毎サイクル、インクリメントする．ただし，値が99,999,999 の時には0に初期化される．（つまり，1秒毎に初期化される）
- 100,000,000 サイクル毎（1秒）に，1ビットのレジスタ r_out の値を反転する．

```
/*********************************************************************/
/* code057.v                          For CSC.T341 CLD Archlab TOKYO TECH */
/*********************************************************************/
module m_main (w_clk, w_led);
  input  wire w_clk;
  output wire [3:0]  w_led;

  reg        r_out = 0;
  reg [31:0] r_cnt = 0;
  always@(posedge w_clk) begin
    r_cnt <= (r_cnt==99999999) ? 0 : r_cnt +1;
    r_out <= (r_cnt==0) ? ~r_out : r_out;
  end
  assign w_led = {r_out, r_out, r_out, r_out};
  // vio_0 vio_00(w_clk, w_led[3], w_led[2], w_led[1], w_led[0]);
endmodule
```

```
module m_main (w_clk, w_led);
  input  wire w_clk;
  output wire [3:0]  w_led;

  reg        r_out = 0;
  reg [31:0] r_cnt = 0;
  always@(posedge w_clk) begin
    r_cnt <= (r_cnt==99999999) ? 0 : r_cnt +1;
    r_out <= (r_cnt==0) ? ~r_out : r_out;
  end
  assign w_led[0] = r_out;
  assign w_led[1] = r_out;
  assign w_led[2] = r_out;
  assign w_led[3] = r_out;
  // vio_0 vio_00(w_clk, w_led[3], w_led[2], w_led[1], w_led[0]);
endmodule
```

ビット連結 { } を用いた左のコードと，ビット毎に接続 assign する右のコードは等価．

補足：Code057.v

- w_clk は 100MHz のクロック信号
- 32ビットのレジスタ r_cnt を，毎サイクル、インクリメントする．ただし，値が99,999,999 の時には0に初期化される．（つまり，1秒毎に初期化される）
- 100,000,000 サイクル毎（1秒）に，1ビットのレジスタ r_out の値を反転する．

- 1KHz のクロック信号は 1,000 Hz と同じ．
- 1KByte のメモリは 1024 Byte と同じ．

- 1MHz のクロック信号は 1000 x 1000 = 1,000,000 Hz と同じ．
- 1MByte のメモリは 1024 x 1024 = 1,048,576 Byte と同じ．

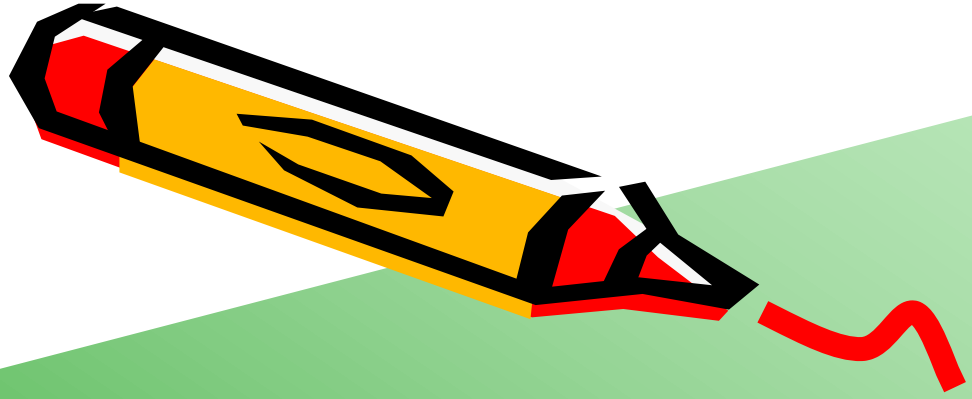
- 100MHz のクロック信号は 100 x 1000 x 1000 = 100,000,000 Hz と同じ．

演習

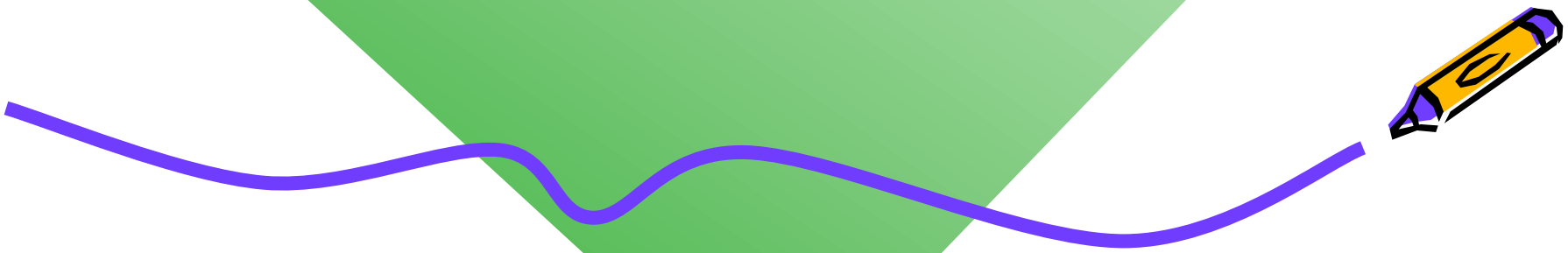
- Verilog HDL を編集して、4ビットカウンタの回路を実装して、動作を確認する。1秒毎に、0, 1, 2, 3, …, 15, 0, 1, 2, … と4ビットの値が変化するハードウェアを実装する。
  - code057.v を編集して、4ビットのワイヤ w_led が1秒毎に1インクリメントされるように修正する。
  - Bitfileを生成して、FPGAをコンフィギュレーションする。
  - VIOを使い、1秒間隔で4ビットの値がインクリメントすることを確認する。

- 正しい動作を確認したら、担当の教員あるいはTAに確認してもらうこと.

Check Point 1 (CP1)



References



References

- Computer Logic Design support page
  - https://www.arch.cs.titech.ac.jp/lecture/CLD/
- ACRi Room
  - https://gw.acri.c.titech.ac.jp
- ACRi Blog
  - https://www.acri.c.titech.ac.jp/wordpress/
- 情報工学系計算機室
  - http://www.csc.titech.ac.jp/
- Xilinx Vivado Design Suite
  - https://japan.xilinx.com/products/design-tools/vivado.html
- Digilent Arty A7-35T
  - https://reference.digilentinc.com/reference/programmable-logic/arty-a7/start
- Verilog HDL
  - https://ja.wikipedia.org/wiki/Verilog