Ver. 2024-10-08

コンピュータアーキテクチャ 演習 (1)
Computer Architecture Exercise (1)

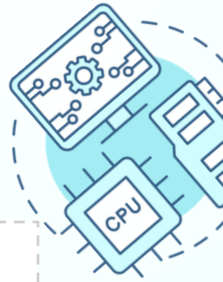
情報工学系　Berjab Nesrine

Computer Architecture support page :

https://www.arch.cs.titech.ac.jp/lecture/CA/

2024年度（令和6年）版

コンピュータアーキテクチャ 演習の注意点 (1)



❑ 連絡について

- 連絡は Slack を使用する。登録がまだの場合は速やかに行うこと。招待メールが来ていない場合は、教員あるいはTAにmアドレスを伝え再送要求すること。

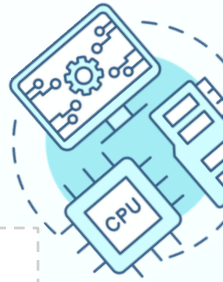
❑ 演習について

- 演習は 15:25～17:05 の時間で行う。15:20 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合すること。15:45 までに到着しない場合、欠席扱いになる。
- 最初の15分は課題の説明、その後は課題の進行とチェックポイントの確認を行う。演習ではACRi ルームを利用する。

❑ グループ作業

- 3～4人のグループを作成し、グループ内で情報を共有しながら演習を進める。問題が発生した場合、まずグループ内で相談し、それでも解決しない場合は TA や教員に質問すること。

コンピュータアーキテクチャ 演習の注意点 (2)

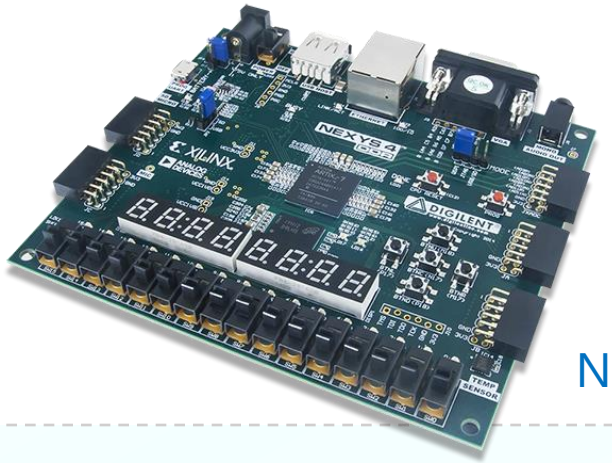


❑ 出席について

- 演習には出席点があるため、全ての授業に休まず参加すること。チェックポイントの図が演習スライドに示されている箇所で、作業の確認を受ける。全てのチェックポイントをクリアすることを目指す。

❑ 演習時間外について

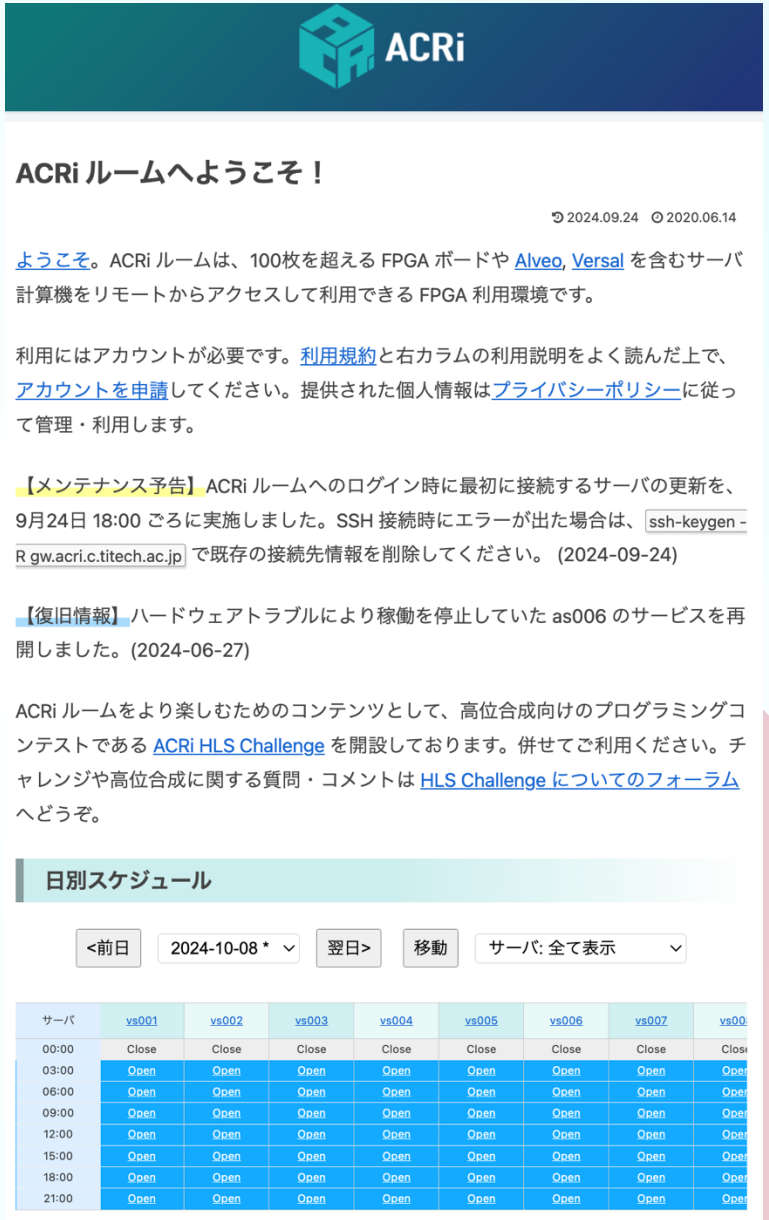
- 演習時間外にも手元の FPGA ボードや ACRi ルームを利用できる。
- 手元のFPGA ボードの貸出も可能なので、独自のハードウェア設計に挑戦してみよう！



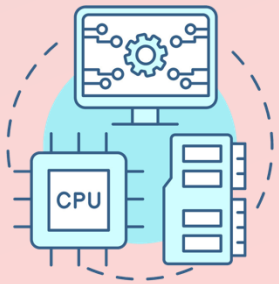
Nexys4 DDR Artix-7 FPGAボード

【重要】 ACRiルームのサーバの予約

- ACRiルームのアカウントを使って，次のURLからログインする。
  https://gw.acri.c.titech.ac.jp/wp
- アカウントがなければ、今、次のURLのページを参考にアカウントを申請すること。
  https://gw.acri.c.titech.ac.jp/wp/manual/apply-for-account
  - ACRiルームでは、次の2種類のアカウントを使用している。
    1. Web予約システム用アカウント
    2. Linuxサーバログイン用アカウント
  - 2つのアカウントで同じ「ユーザアカウント名」を使っているが、別々の「パスワード」が設定されているので注意すること。
- 「予約ページトップ」から，vs0xx~vs7xxで始まるサーバで演習の日の15:00～18:00の枠を予約すること。



Slack の登録をする（５分）
必要であれば ACRi のアカウント作成



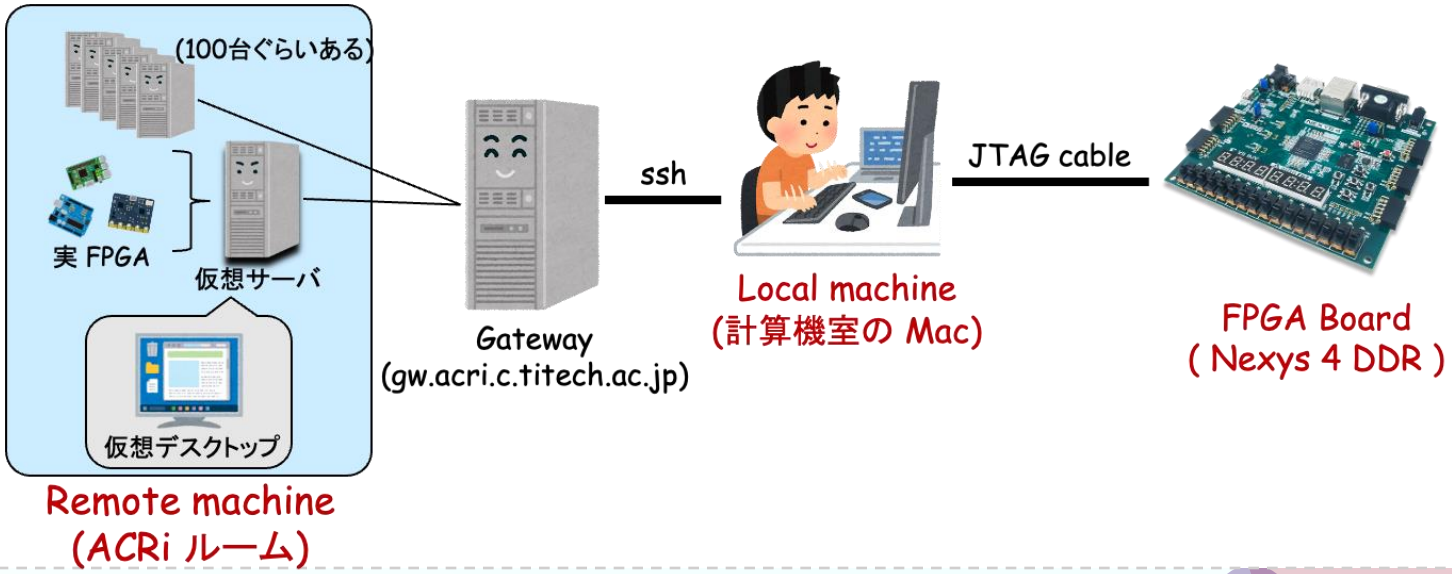
TAの自己紹介、クラス分けと担任の確認（５分）

	担当Staff（Slackアカウント名）
クラス A	Yuji Yamada (TA_classA)
クラス B	Noriaki Shimooka (TA_classB)
クラス C	Aoba Fujino (TA_classC)

演習第一回の内容 (1/2)

❑ Project 1

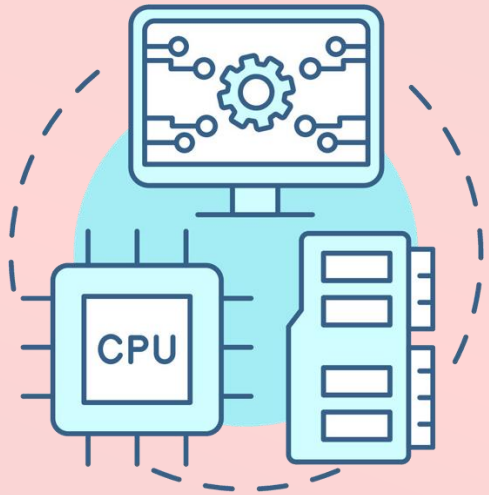
- ネットワーク経由でFPGAボードにコンフィギュレーションを行うことを体験する。
- Vivadoを使い、リモートから手元のFPGAにビットストリームを書き込むプロセスを学ぶ。
- Verilogで実装した論理回路を、FPGAの実機で動作させる一連の手順を経験する。



演習第一回の内容 (2/2)

❑ Project 2

- UARTの基本的な使い方を理解することを目指す。
- シリアル通信による送信回路と受信回路の仕組みを理解する。
- 両方の回路を用いてシリアル通信を実装するデザインを作成する。



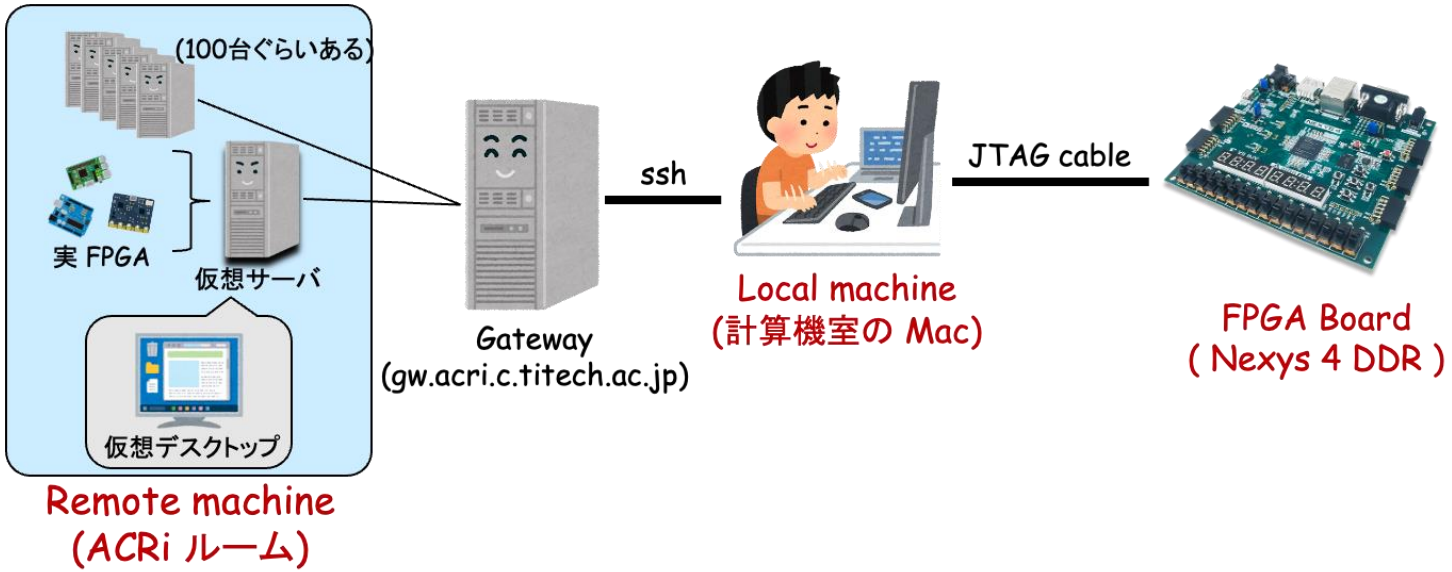
Project 1

グループに分かれて作業

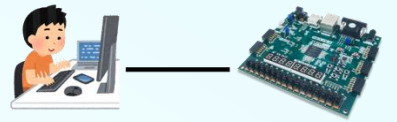
演習第一回の内容 (1/2)

❑ Project 1

- ネットワーク経由でFPGAボードにコンフィギュレーションを行うことを体験する。
- Vivadoを使い、リモートから手元のFPGAにビットストリームを書き込むプロセスを学ぶ。
- Verilogで実装した論理回路を、FPGAの実機で動作させる一連の手順を経験する。



Local machine をセットアップする



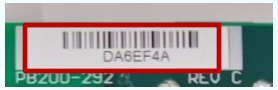
❑ 目的

- Nexys 4 DDR FPGA ボードと Mac を連携させ、正しく動作させるためにローカルマシンの設定を行う。

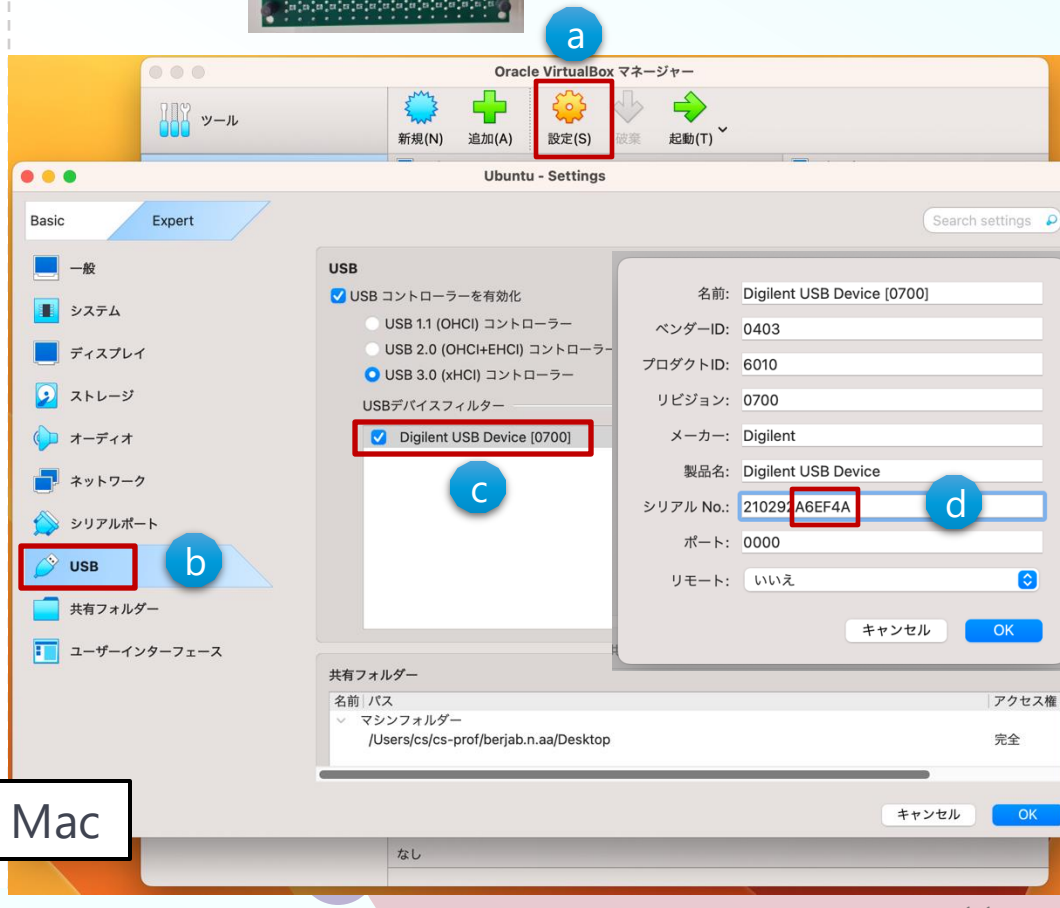
❑ Steps

Power switch on/off

1. FPGAボードの電源をオンにする。
2. 計算機室のMac上で VirtualBox を立ち上げる。
3. シリアル番号の確認
   - ➢ a 設定 (S) → b USB → c Digilent USB Device [700]
   - ➢ d FPGAボード裏のシリアル番号と異なる場合は、VirtualBoxで番号を編集する。
4. VMを起動。

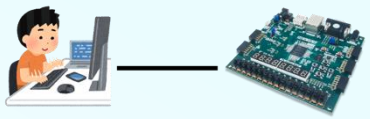


FPGA ボードの裏側
→シリアル番号：AE6EF4A



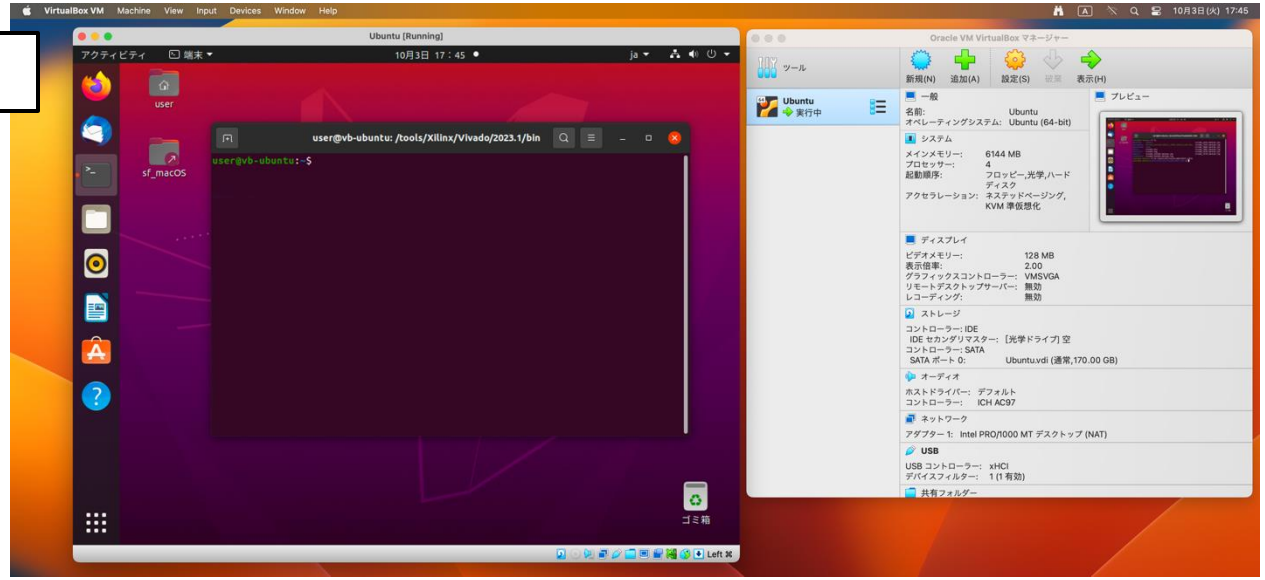
Local Mac

Remote Configuration (Local machine)



- VMのターミナルを立ち上げる。

Local Ubuntu VM



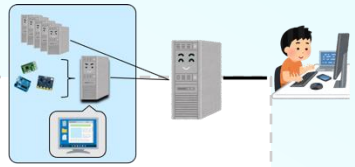
- ターミナルで次のコマンドを入力し，Vivado を起動する.
  - 「Vivado 2023.2」を利用する。

```
$ cd /tools/Xilinx/Vivado/2023.1/bin
$ ./hw_server
```

Local Ubuntu VM
ターミナル

- 「./hw_server」 は、FPGA ボードと Vivado が通信できるようにするのコマンド ライン。

Remote machine をセットアップする (1/2)



- ACRiルームの計算機にリモートデスクトップで接続する。
- リモートデスクトップ接続には、「Microsoft Remote Desktop 10」を利用する。

1. Mac のターミナルを起動したら、以下のコマンドを入力する。

```
$ ssh -f -N -L 13389:<使用するサーバ名>:3389 <ユーザアカウント名>@gw.acri.c.titech.ac.jp
```

Local Mac ターミナル

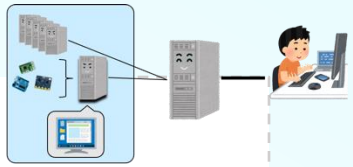
→ <>で括ってある部分については自分が使うユーザ名やサーバ名 (vs001など) を入力する

2. パスワードの入力を求められるのでパスワードを入力する。

→ ACRiルームのLinuxサーバーにログインするためのアカウントのパスワードを入力する。

3. 次に Microsoft Remote Desktop 10 を起動する。
4. ウィンドウの中央にある青い「add PC」ボタンをクリックする。
5. リモート PC の設定画面が出てきたら「PC name」に localhost:13389と入力する。
6. 「Add User Account」を選択するとアカウントの追加ウィンドウが出る。
7. 「Add」をクリックしてリモート PC の設定を終了。
8. リモート PC をダブルクリックすることで FPGA 利用環境へ接続する。

Remote machine をセットアップする(2/2)



❑ 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。

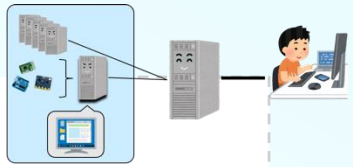
❑ ターミナルで次のコマンドを入力し，Vivado を起動する.

- 「Vivado 2023.2」を利用する。

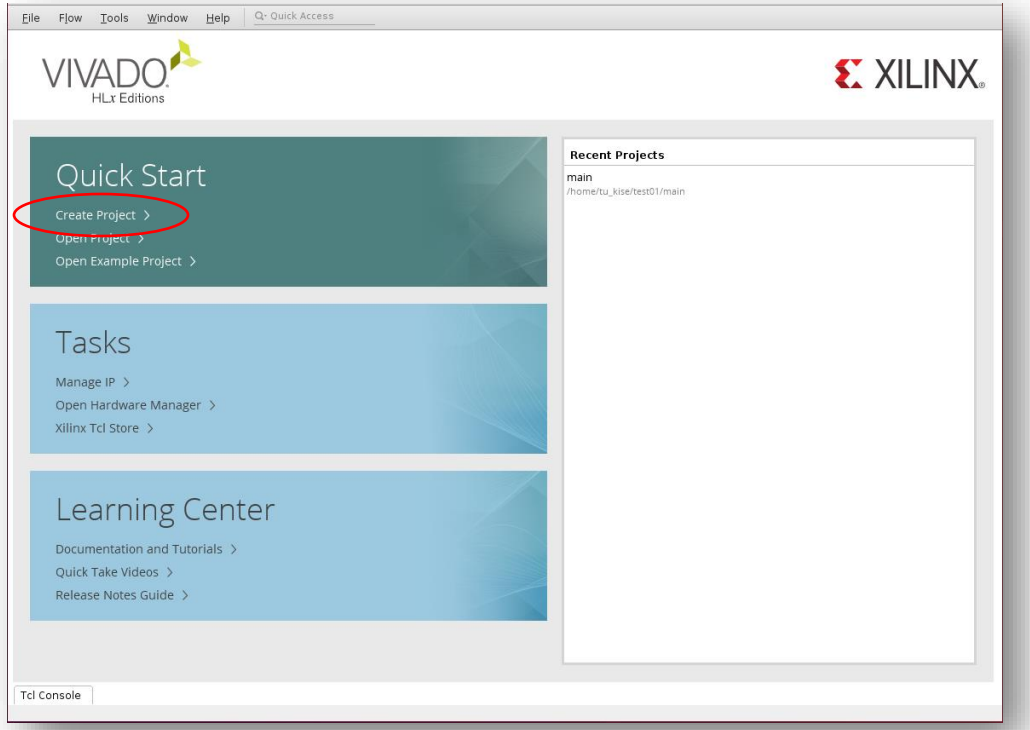
```
$ source /tools/Xilinx/Vivado/2023.2/settings64.sh
$ vivado &
```

Remote Ubuntu
ターミナル

Create a new Vivado project (1/2)

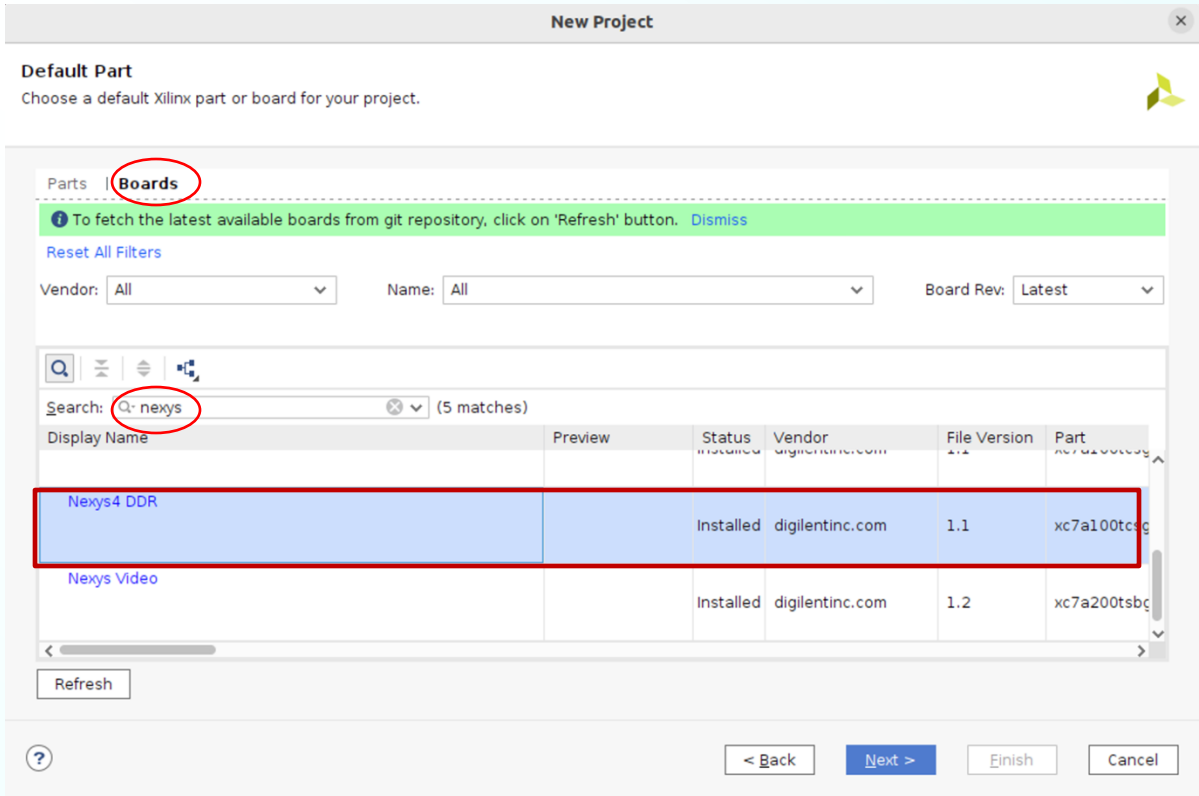


- Select Create Project, Click Next
- Project name “project_1” and location “/home/your_username/ca2024” are selected.
  - Check “Create project subdirectory”.
- Click Next

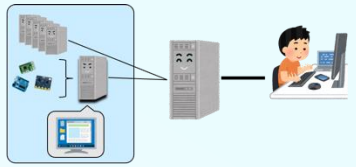


Create a new Vivado project (2/2)

- 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 Boards, and write nexys.
- Select Nexys4 DDR and click Next.
- Confirm the summary in New Project Summary window and click Finish.



Source codeをコピーする

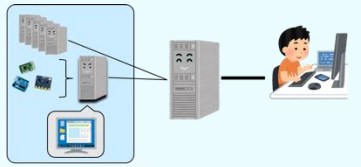


- ターミナルで，ファイルをコピーする。
- /home/u_nesrine/ca2024/ に保存されている blink.v と blink.xdc を，作成したプロジェクトのディレクトリ ~/ca2024/project_1 にコピーする。

```
$ cd ~/ca2024/project_1
$ cp /home/u_nesrine/ca2024/src/blink.v .
$ cp /home/u_nesrine/ca2024/src/blink.xdc .
```

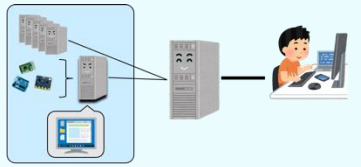
Remote Ubuntu ターミナル

Bitstream file generation



- Click Add Sources, then select Add or create design sources and click Next.
  - In Add or Create Design Sources window, click Add Files, select blink.v in project_1 directory, and click OK.
  - Click Finish.
- Click Add Sources, then select Add or create constraints and click Next.
  - In Add or Create Constraints window, click Add Files, select blink.xdc in project_1 directory, and click OK.
  - Click Finish.
- Click Generate Bitstream, click Yes, click OK, and wait.

Local machineでリモートポートフォワーディングを行う

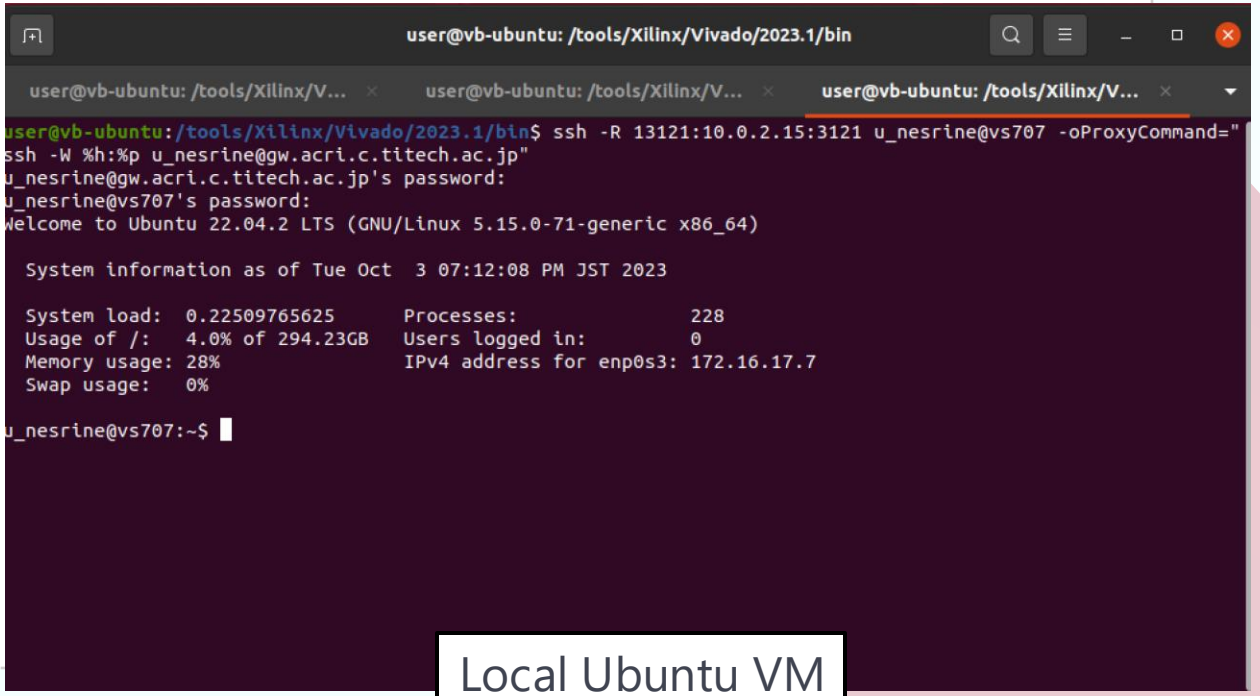


- Local machineのVM上でもう一つターミナルを立ち上げる。
- ターミナルで次のコマンドを入力する。

```
$ ssh -R 13121:<Local Machine IP Address>:3121 <ACRiユーザアカウント名>@<使用するサーバ名> -oProxyCommand="ssh -W %h:%p <ACRi username>@gw.acri.c.titech.ac.jp"
```

Local Ubuntu VM ターミナル

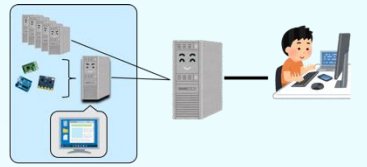
→ <Local Machine IP address>: 10.0.2.15 にする。



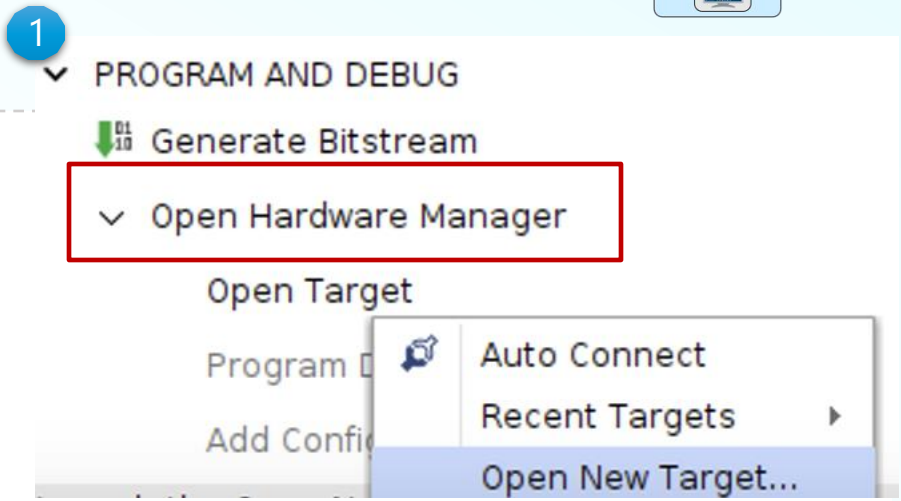
- このコマンドは、ゲートウェイサーバ（gw.acri.c.titech.ac.jp）を経由してリモートサーバに接続し、リバースポートフォワーディングを設定するもの。これにより、リモートサーバがローカルマシンの指定したポート（3121）にアクセスできるようになる。

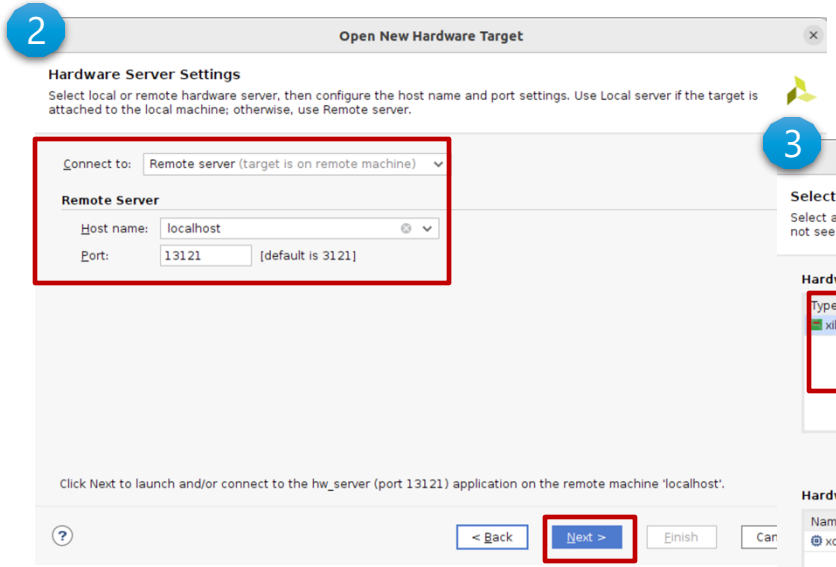
Local Ubuntu VM ターミナル

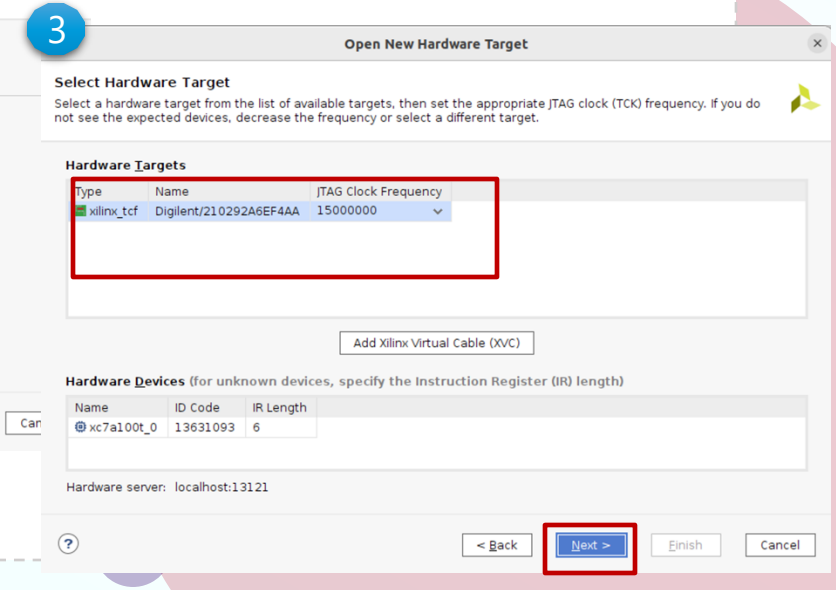
FPGA configuration



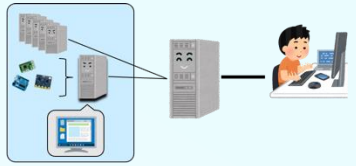
- Hardware Manager を開く。
- Open Target → Open New Target を押して以下のように入力を進めていく。
- 「Connect to:」を Remote server に変更する。
- 「Host name」に localhost と入力する。
- 「Port」に 13121 と入力する。
- Click Next and then Finish。
- Then click Program device。



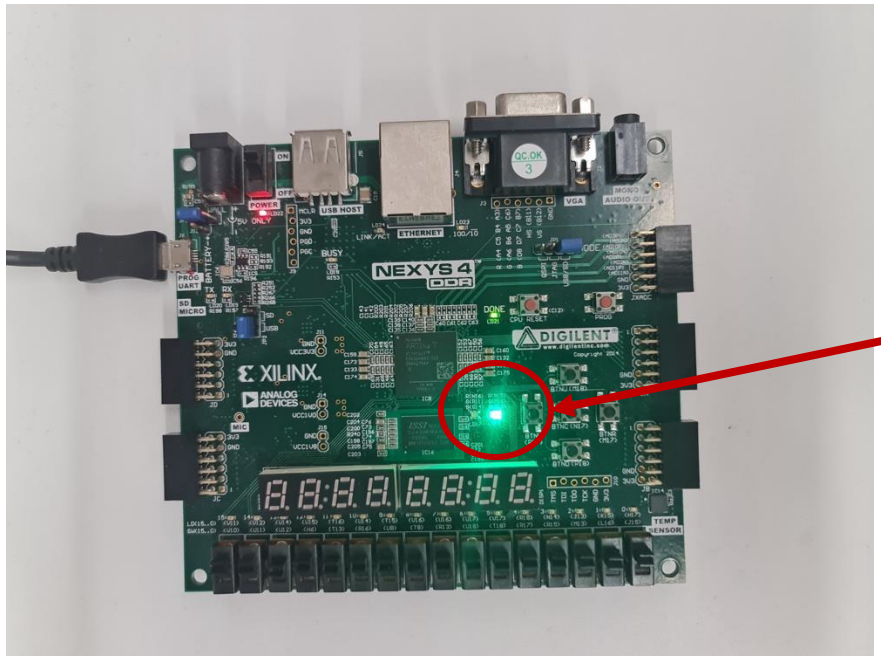




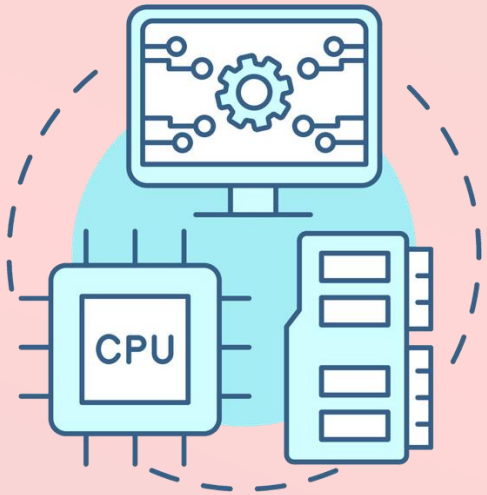
リモートコンフィグレーションの確認



❑ 正しく動作している手元の FPGA ボードを担当TAに確認してもらう。



Blinking LED
（点滅する）



Project 2

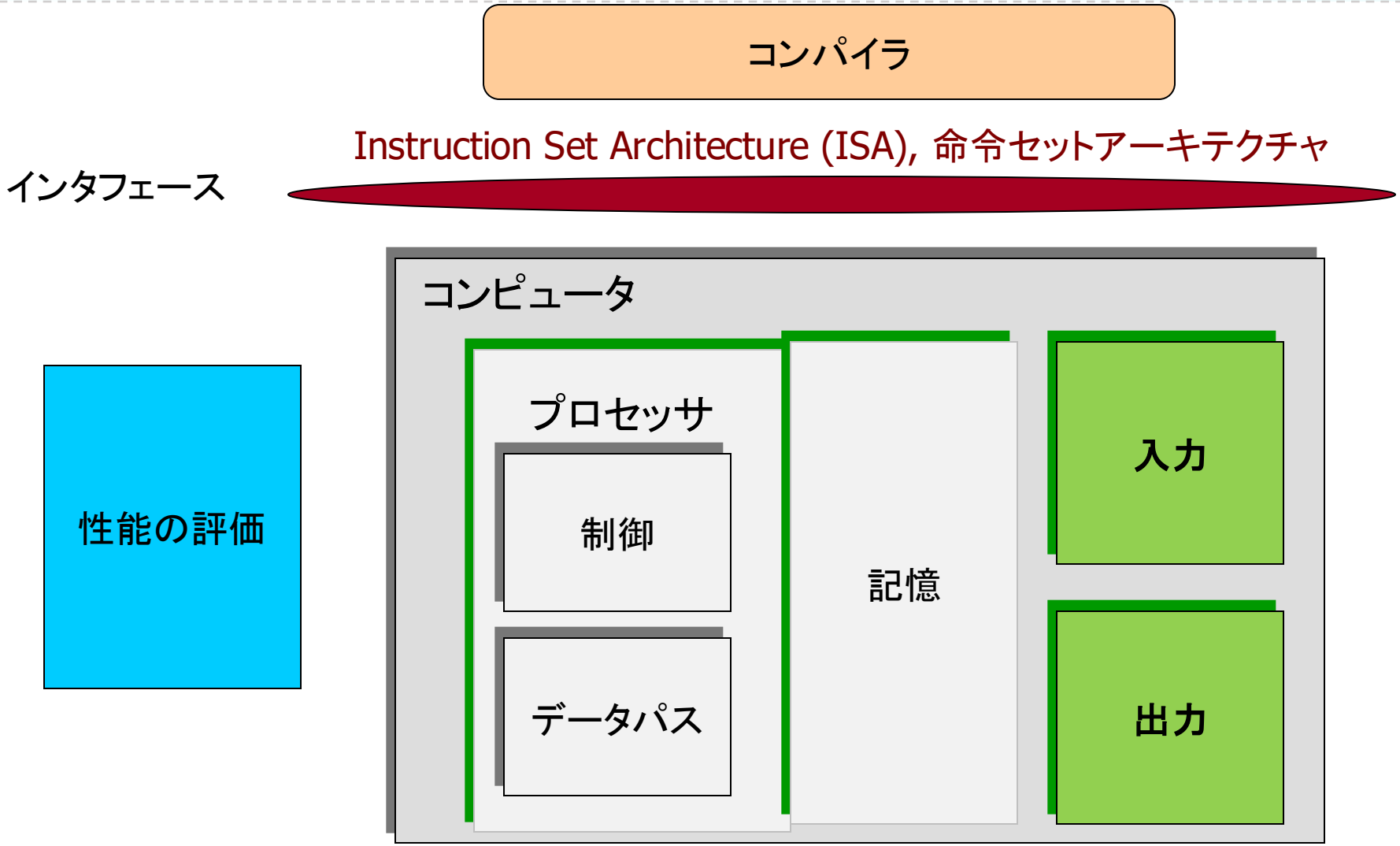
手元のFPGAはこれ
以降使わない！

演習第一回の内容

❑ Project 2

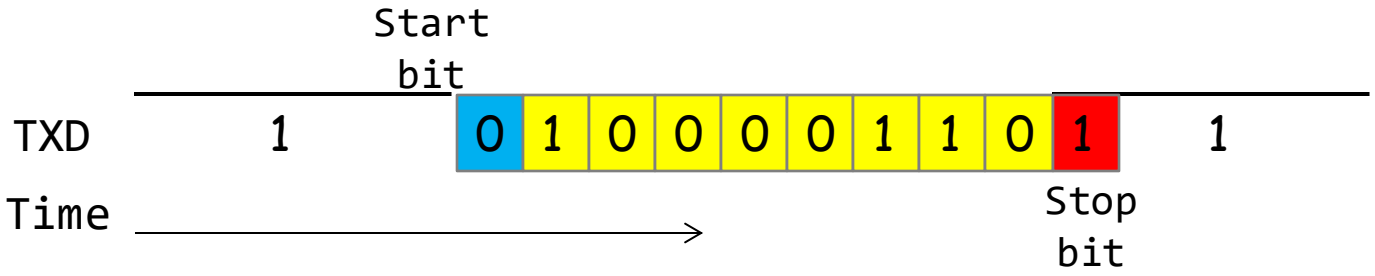
- UARTの基本的な使い方を理解することを目指す。
- シリアル通信による送信回路と受信回路の仕組みを理解する。
- 両方の回路を用いてシリアル通信を実装するデザインを作成する。

コンピュータの古典的な要素



調歩同期方式によるシリアル信号の変換

- 調歩同期方式によるシリアル信号をパラレル信号に変換したり、その逆方向の変換をおこなう集積回路をUARTと呼ぶ。8ビット（1バイト）単位でデータを送信・受信する。
- UARTを用いることで、FPGAとコンピュータの間でのお手軽なデータ通信が可能。
- 例えば，‘a’ という文字を送信する場合，**’a’** は **8’h61**, 8’b01100001 （次スライドのASCII Tableを参照）なので，下図のタイミングで送信線TXDを制御する。
  - データが送信されるまで送信線TXD を1とする。
  - まず、青色で示した0 （これをスタートビットと呼ぶ）を送信することで，データ送信の開始を明示。
  - 次に、黄色で示した様に送信したいデータ 8’b01100001 の最下位ビットから順番に送信する。
  - 最後に，赤色で示した1（これをストップビットと呼ぶ）を送信する。
- 1ビットを送受信するための時間間隔は送信側と受信側で同じレートを用いる。これをボー・レート (baud) と呼ぶ。例えば，1000 baud であれば，１ビット送信の間隔は 1msec となる。



ASCII Table

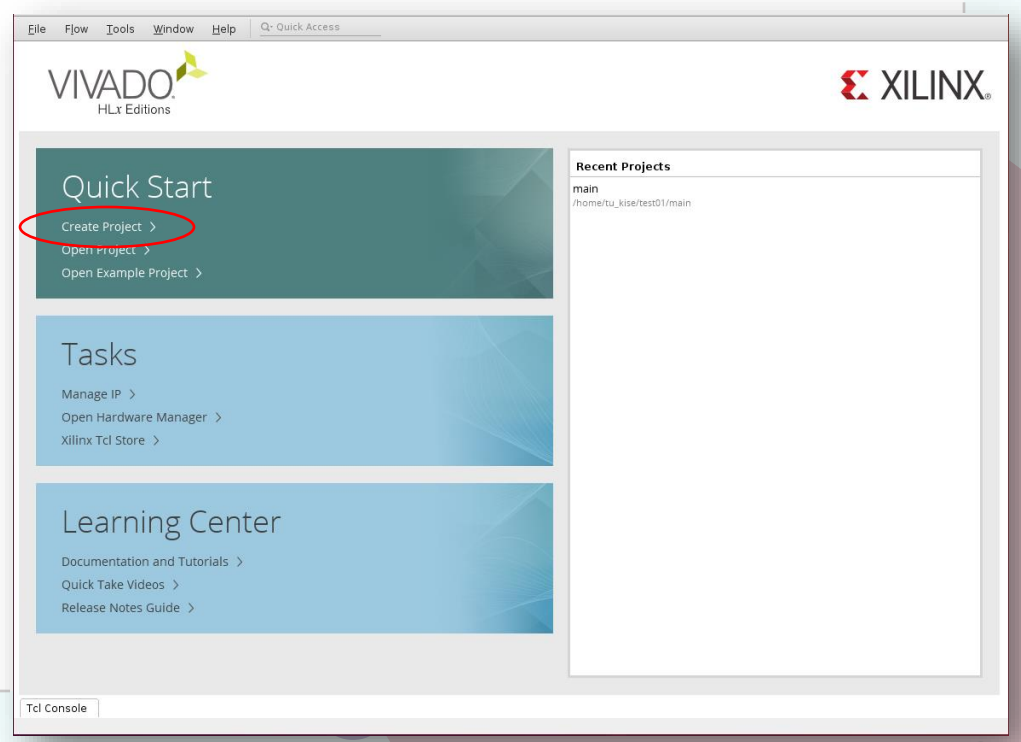
Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

演習1 project 2 環境情報 (1/2)

- 新しいVivado Project「project_2」を作る。
- ターミナルで次のコマンドを入力し，Vivado を起動する.
  - 「Vivado 2024.1」を利用する。

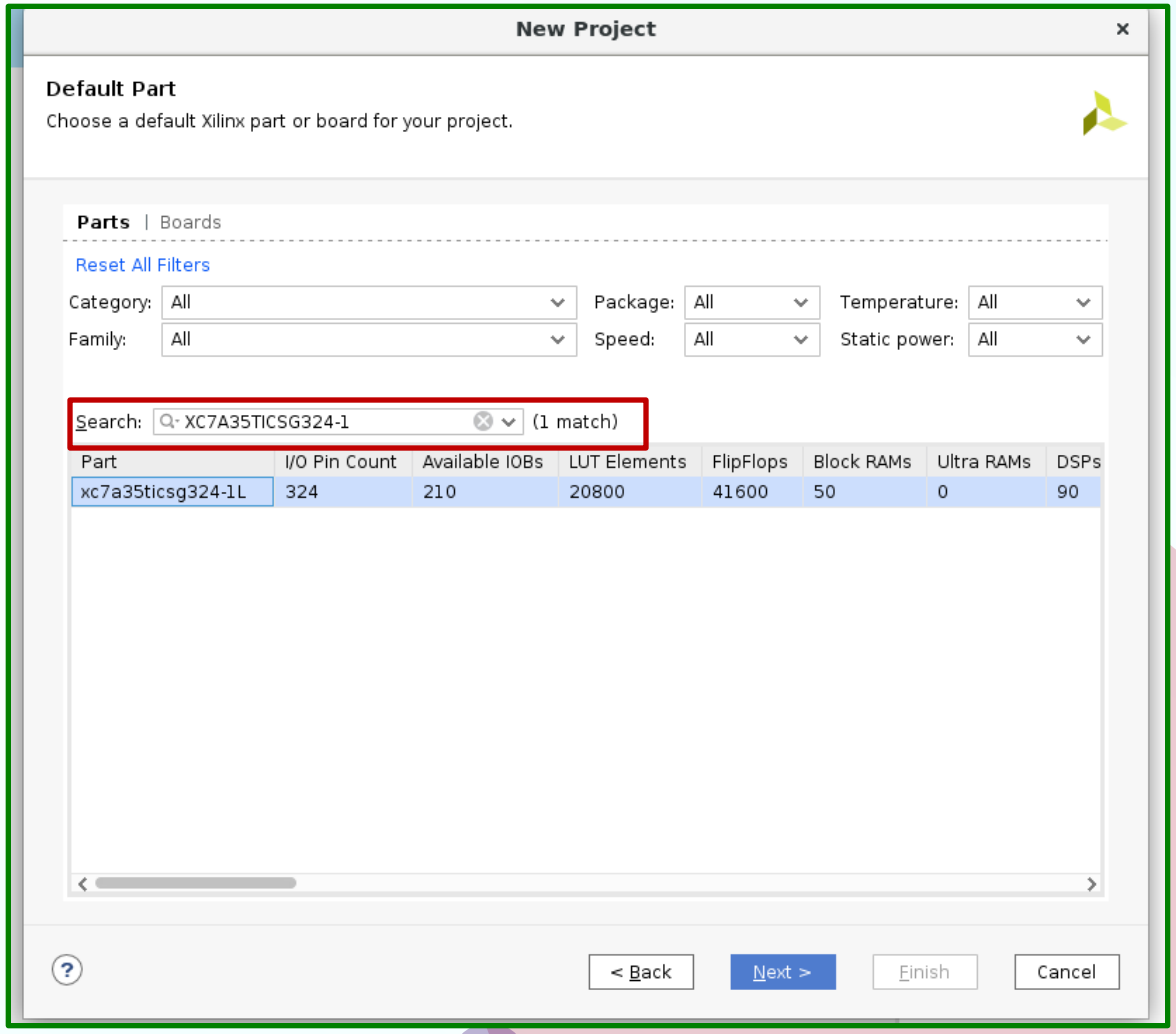
```
$ source /tools/Xilinx/Vivado/2024.1/settings64.sh
$ vivado &
```

- Select Create Project, Click Next
- Project name “project_2” and location “/home/your_username/ca2024” are selected.
  - Check “Create project subdirectory”.
- Click Next



演習1 project 2 環境情報 (2/2)

- 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 Parts, and write XC7A35TICSG324-1L.
- Select XC7A35TICSG324-1L and click Next.
- Confirm the summary in New Project Summary window and click Finish.



Source codeをコピーする

- ターミナルで，ファイルをコピーする。
- /home/u_nesrine/ca2024/src/ に保存されている main2.v と main2.xdc を，作成したプロジェクトのディレクトリ ~/ca2024/project_2 にコピーする。

```
$ cd ~/ca2024/project_2
$ cp /home/u_nesrine/ca2024/src/main2.v .
$ cp /home/u_nesrine/ca2024/src/main2.xdc .
```

Bitstream file generation

- Click Add Sources, then select Add or create design sources and click Next.
  - In Add or Create Design Sources window, click Add Files, select main2.v in project_2 directory, and click OK.
  - Click Finish.
- Click Add Sources, then select Add or create constraints and click Next.
  - In Add or Create Constraints window, click Add Files, select main2.xdc in project_2 directory, and click OK.
  - Click Finish.
- Click Generate Bitstream, click Yes, click OK, and wait.

シリアル通信による送信回路 m_UartTx

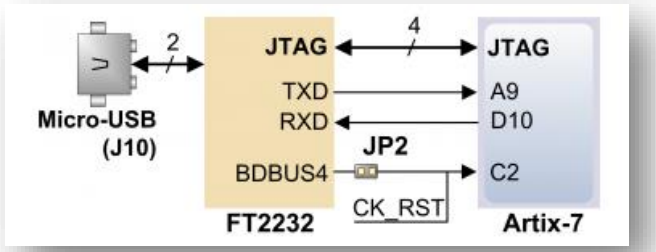
- m_uart_tx/rx モジュールは 100MHz、1 Mbaudの速度でUARTの送受信を行うもの。FPGAがPCから8ビットのデータを受信して、それを返送する仕組みになっている。
- トップのモジュール m_main では, UART受信機でPCからデータを待ち続ける。
  - 文字が受信されると、r_char が更新され、それがUART送信機を使ってPCに送信される。文字が受信されない場合、デフォルトでASCII文字 'a' を送信する。

main2.v (Source code available in /home/u_nesrine/ca2024/src)

```
module m_main (
    input  wire  w_clk,       // 100MHz clock signal
    input  wire  w_uart_rx,   // UART rx, data line from PC -> FPGA
    output wire  w_uart_tx,   // UART tx, data line from FPGA -> PC
    output wire  [3:0] w_led  // LED
);
    reg [31:0] r_cnt = 0;
    always @(posedge w_clk) r_cnt <= r_cnt + 1;
    assign w_led = r_cnt[26:23];

    reg r_we = 0;
    always @(posedge w_clk) r_we <= (r_cnt[27:0]==0);

    wire w_en;
    wire [7:0] w_char;
    reg  [7:0] r_char = 8'h61;  // 8'h61 for 'a'
    m_uart_rx m1 (w_clk, w_uart_rx, w_char, w_en);
    always @(posedge w_clk) if (w_en) r_char <= w_char;

    m_uart_tx m2 (w_clk, r_we, r_char, w_uart_tx);
endmodule
```

Inside main2.xdc

- XDC (Xilinx Design Constraint) ファイルとは：Verilog上のワイヤ名とFPGAの物理的なピンの対応付けや、FPGA内で用いるクロック信号の周波数などを記述する『制約ファイル』
- このプロジェクトで用いる XDC (Xilinx Design Constraint) ファイル
  - FPGAの出力信号が w_txd （これはコンピュータの入力信号）
  - FPGAの入力信号が w_rxd （これはコンピュータの出力信号）

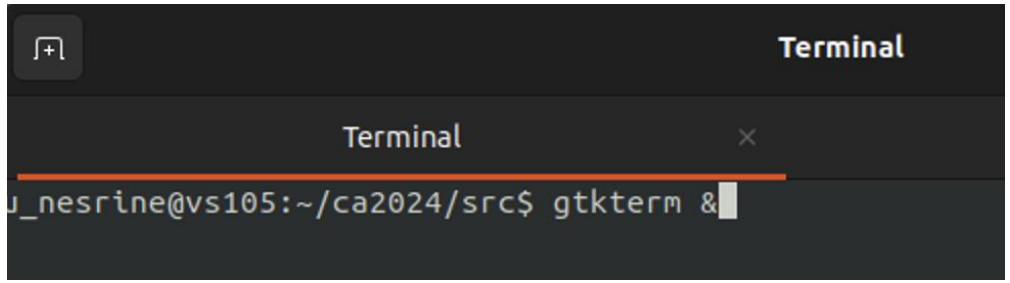


main2.xdc (Source code available in /home/u_nesrine/ca2024/src)

```
##############################################################################
set_property -dict { PACKAGE_PIN E3  IOSTANDARD LVCMOS33} [get_ports { w_clk }];
create_clock -add -name sys_clk -period 10.00 [get_ports { w_clk }];

##############################################################################
set_property -dict { PACKAGE_PIN H5  IOSTANDARD LVCMOS33} [get_ports { w_led[0] }];
set_property -dict { PACKAGE_PIN J5  IOSTANDARD LVCMOS33} [get_ports { w_led[1] }];
set_property -dict { PACKAGE_PIN T9  IOSTANDARD LVCMOS33} [get_ports { w_led[2] }];
set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33} [get_ports { w_led[3] }];

##############################################################################
set_property -dict { PACKAGE_PIN A9  IOSTANDARD LVCMOS33} [get_ports { w_rxd }];
set_property -dict { PACKAGE_PIN D10 IOSTANDARD LVCMOS33} [get_ports { w_txd }];
##############################################################################
```

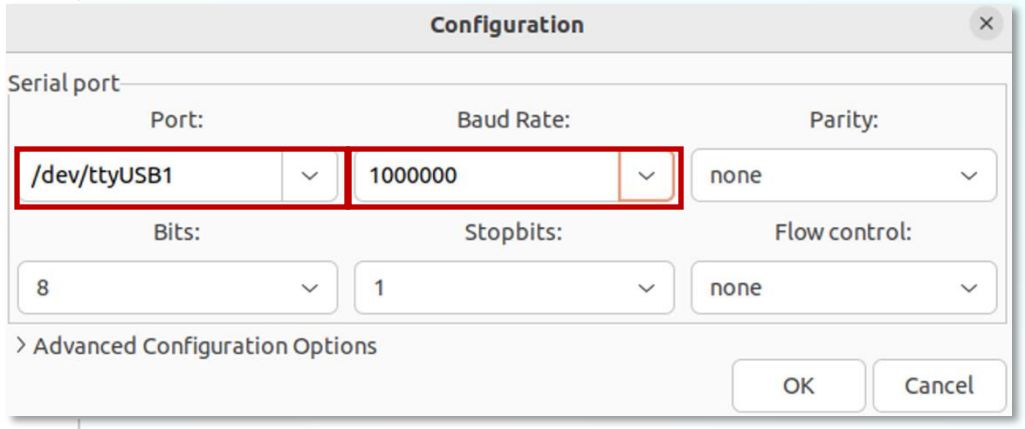
GtkTerm を利用する

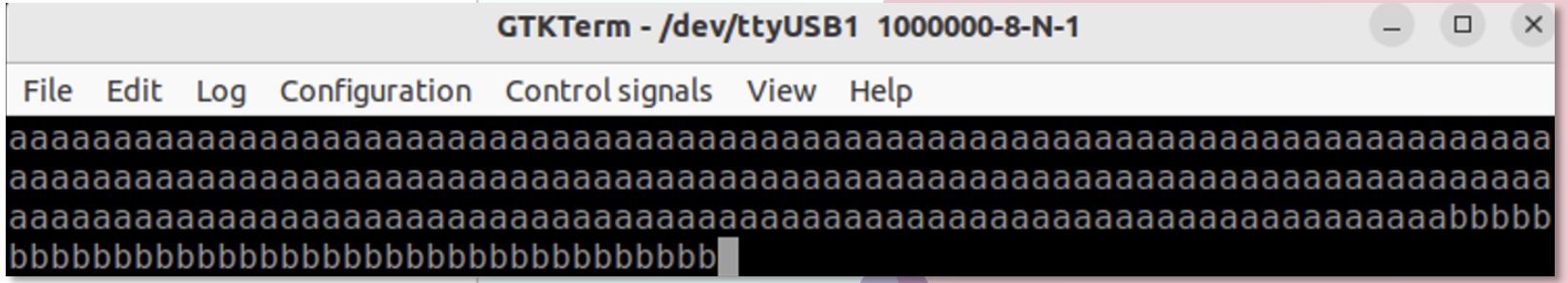
- 「リモート デスクトップ接続」で ACRiルームにログインする。
- コマンド *gtkterm &* で GtkTerm を起動する。



GtkTerm の設定

- Configuration から Port を選択する。
  - Port として /dev/ttyUSB1 を選択する。
  - Baud Rate に 1000000 を入力して、1Mbaud とする。
  - OK ボタンを押す。
- GtkTerm に約2.5秒に1回の間隔で a が表示される。キー入力するとその文字が表示されるようになる (図では"b")」





Check Point 2 の課題

- トップのモジュール m_mainを変更して、入力された小文字を大文字に変換して出力するようにする。

- 正しく動作している画面を担当TAに確認してもらう。