2023年度（令和5年）版

Ver. 2023-10-06

Course number: CSC.T363

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

Nesrine BERJAB, Department of Computer Science

berjab_at_c.titech.ac.jp

Computer Architecture support page: https://www.arch.cs.titech.ac.jp/lecture/CA/

コンピュータアーキテクチャ 演習(Exercise)の注意点

- 連絡手段は Slack を利用します.
  - 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 (Teaching Assistant) に伝えてください.
- 演習は 15:25～17:05 です.
  - 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 に集まってください.
  - 20分以上(15:45までに)遅刻したら欠席扱いになります.
  - 前半は課題の説明(15分程度)で, 後半は課題の解決とチェックポイントの確認.
- 演習は手元の FPGA ボードと 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 TA 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.

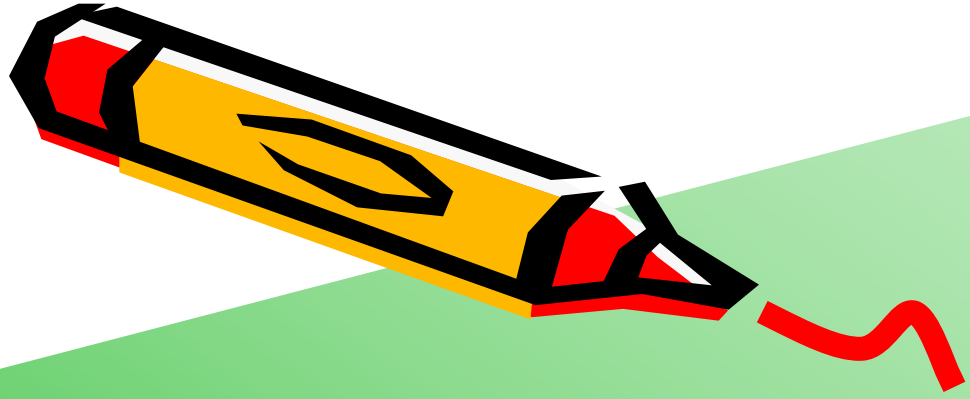
- 演習時間以外も手元の FPGA ボードと ACRi ルームを利用できます.
  - 手元の FPGA ボードを借りることができます.
  - 独自のハードウェア設計などに挑戦しましょう.

【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

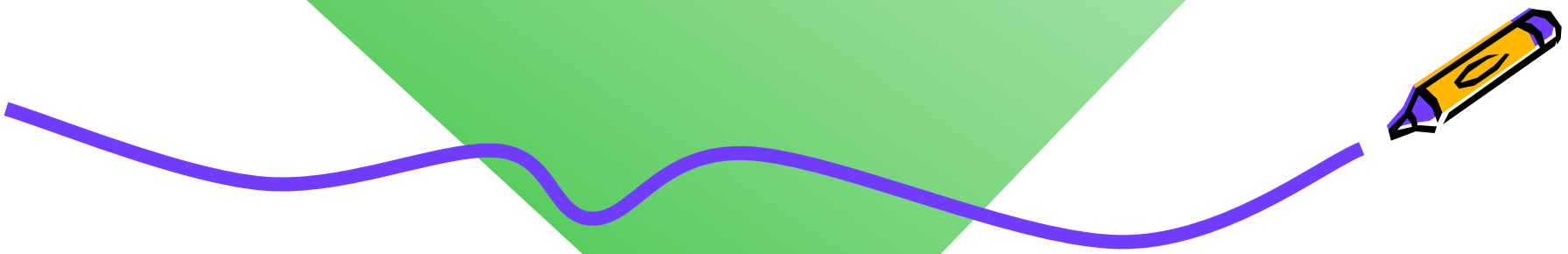
- アカウントがなければ、今、次のURLのページを参考にアカウントを申請すること
  - https://gw.acri.c.titech.ac.jp/wp/manual/apply-for-account

- 「予約ページトップ」から，vs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 の登録をします（5分）

必要であればACRiのアカウント作成

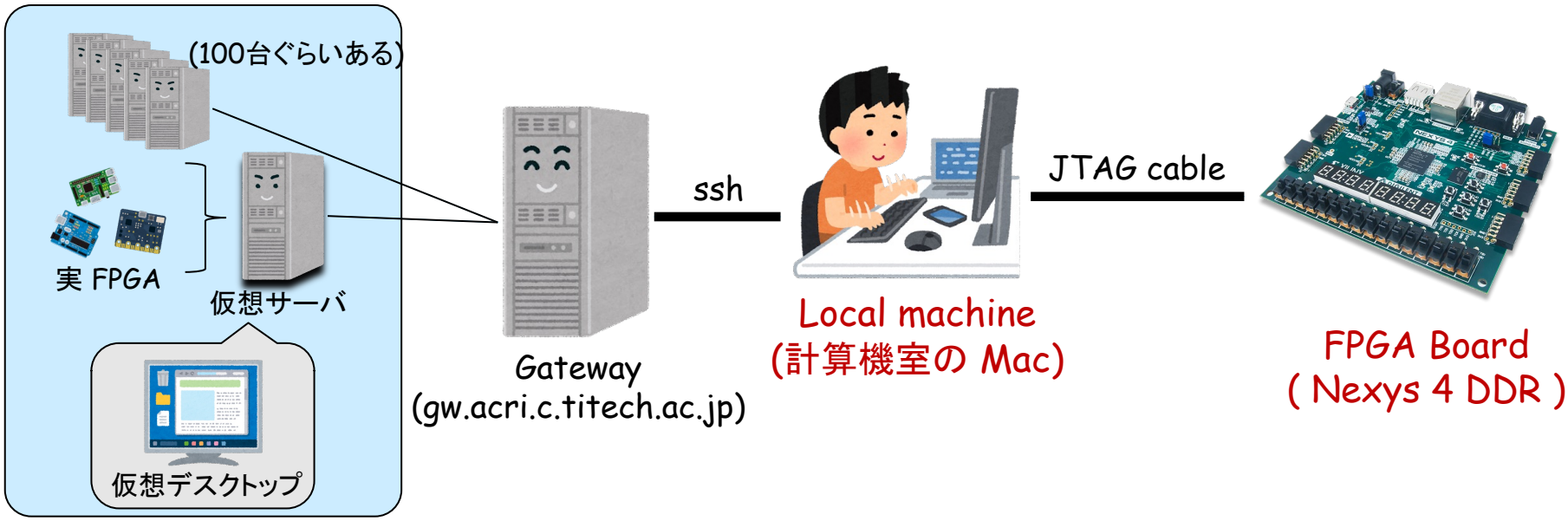


TAの自己紹介、クラス分けと担任の確認（5分）

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

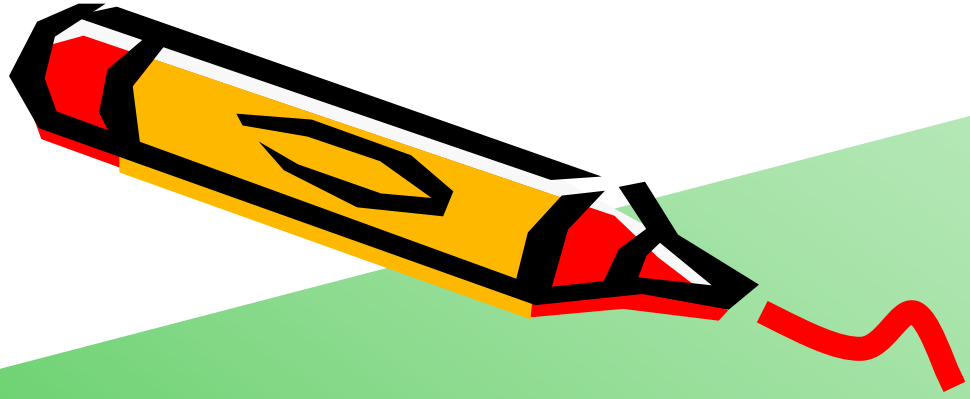
Exercise (1)

- Project_20
  - Vivado で FPGA をリモートコンフィグレーションのやり方を理解する.
    - ACRi ルームの Vivado から手元の FPGA ボード (Nexys 4 DDR) に Bitstream を書き込む.



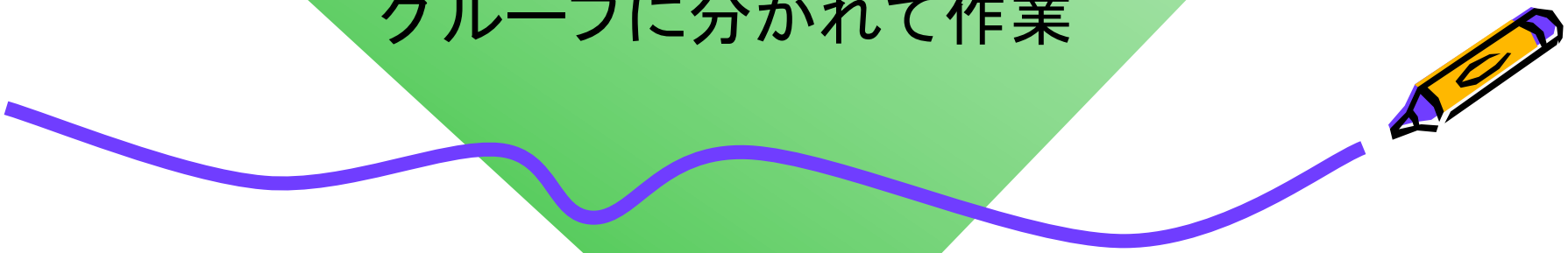
Exercise (1)

- Project_21
  - UARTの使い方を理解する.
  - シリアル通信による送信回路を理解する.
  - シリアル通信による受信回路を理解する.
  - 送信回路と受信回路を用いたデザインを実装する.

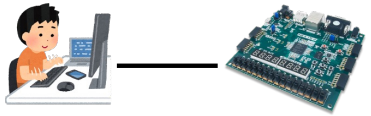


Project_20

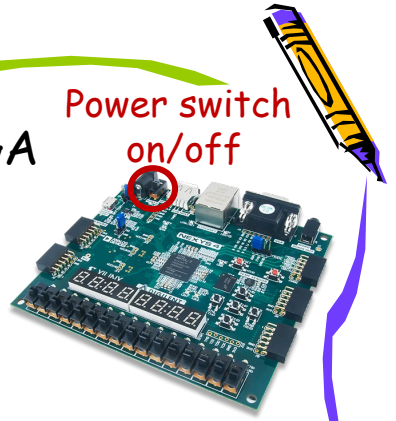
グループに分かれて作業



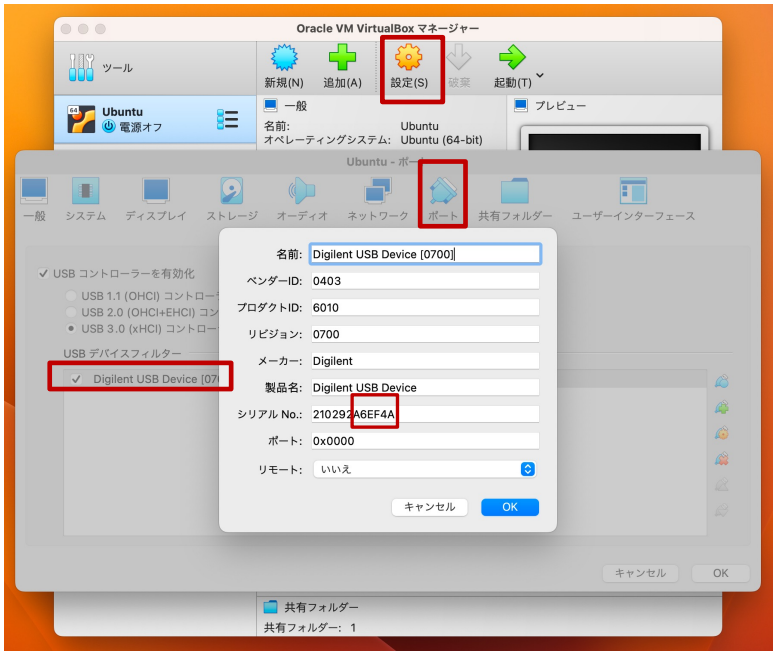
Local machine をセットアップする



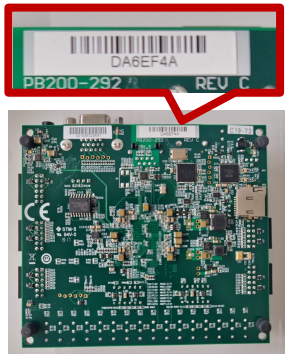
- 計算計算機室の Nexys 4 DDR は電源が入っていないため、利用時は FPGA ボード上にある電源スイッチを ON にする必要がある.
- 計算機室の Mac で VirtualBox が立ち上げる.



- VirtualBox マネージャー で Mac に接続されている Nexys 4 DDR のシリアル No. を確認
  - 設定(S)→ ポート → Digilent USB Device [700]
  - VirtualBox に設定されている USB デバイスのシリアル No. が実際に Mac に接続されている Nexys 4 DDR のシリアル No. とは異なる場合:
    - VirtualBox の USB 設定でデバイスの登録を一度解除し、再びデバイスを登録し直す必要がある.

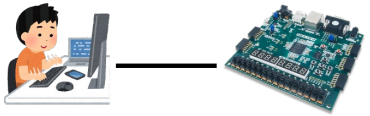


- VMを起動する

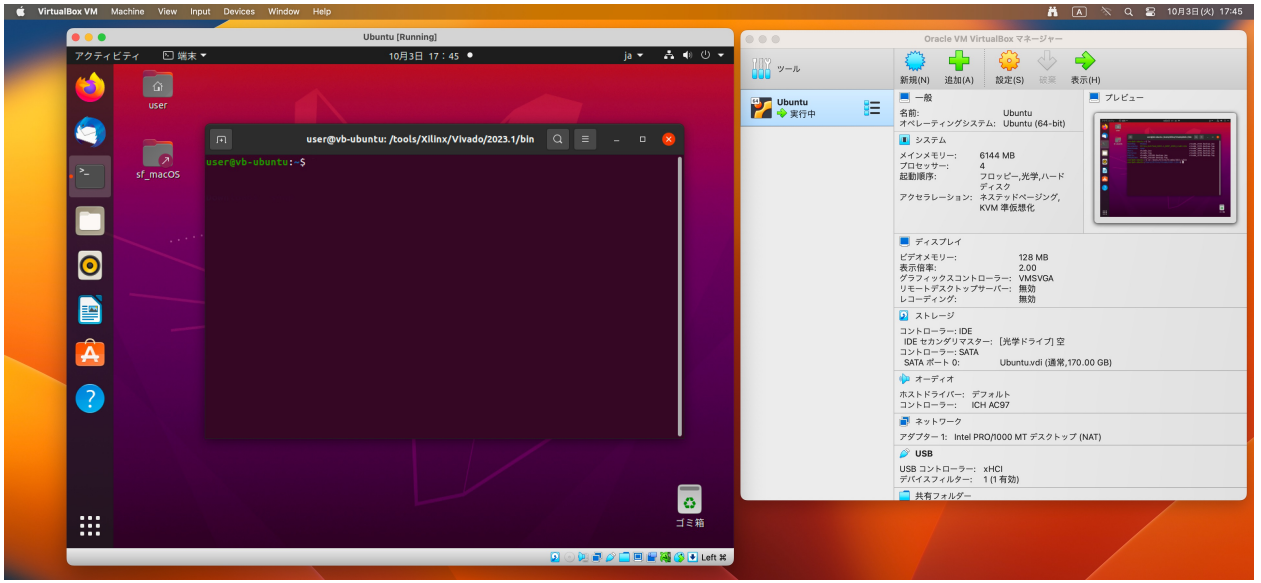


FPGA ボード の裏側

Remote Configuration (Local machine)



- VM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 2023.1 の hw_server を立ち上げる.

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

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.
- macOS にはリモートデスクトップクライアントが標準でインストールされていないため、Microsoft 社の「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
```

→ <>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.
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)

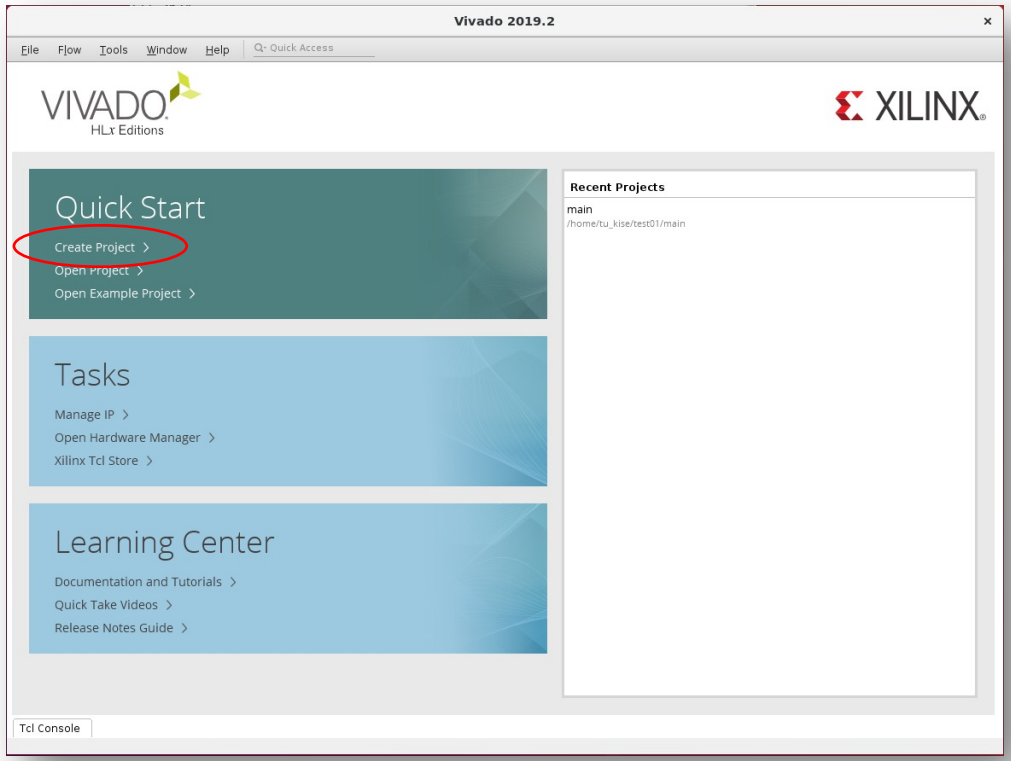
- リモート デスクトップ接続した Ubuntu のターミナルを立ち上げる.

- ターミナルで次のコマンドを入力し，Vivado を起動する.
  - 「Vivado 2022.2」を利用する.

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

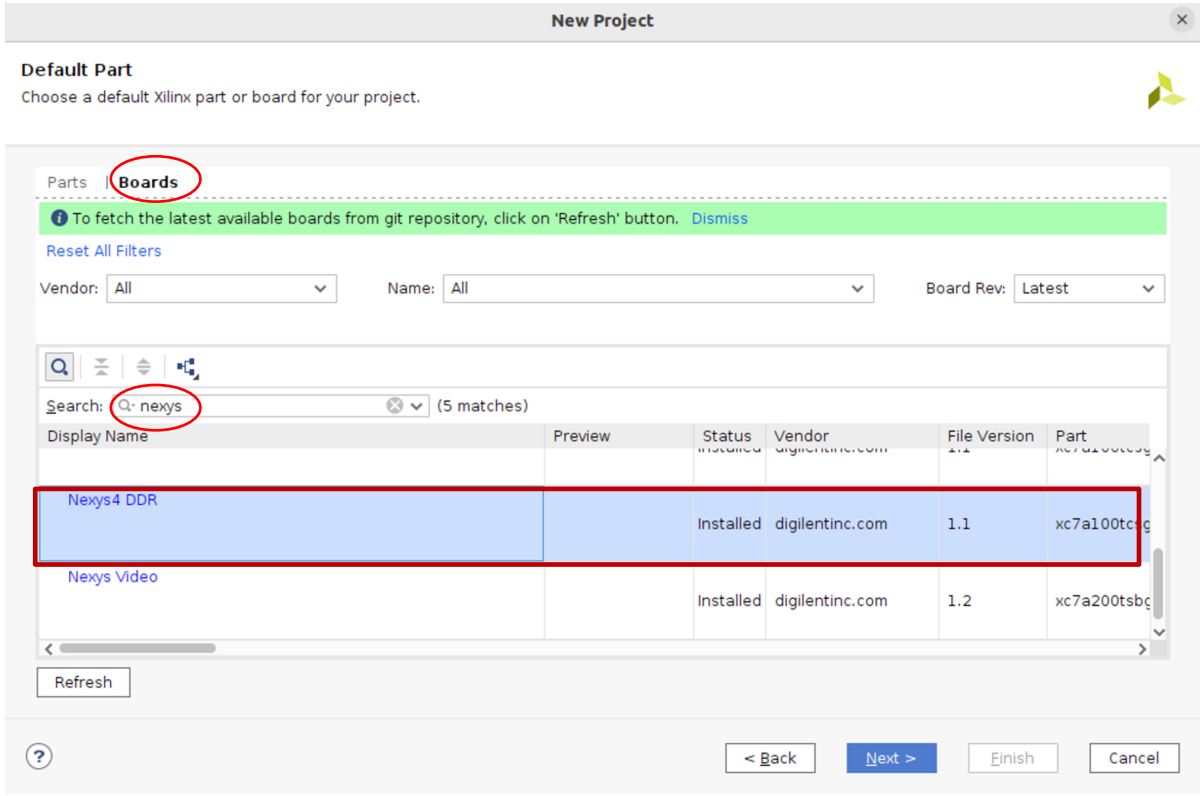
Create a new Vivado project (1/3)

- Select Create Project, Click **Next**
- Project name "**project_20**" and location "/home/your_username/ca" are selected.
  - Check "Create project subdirectory".
- Click **Next**

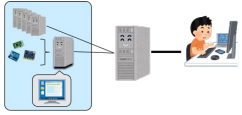


Create a new Vivado project (2/3)

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



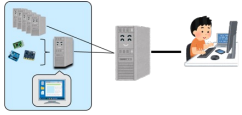
Bitstream file generation and FPGA configuration



- ターミナルで，ファイルをコピーする.
- /home/u_nesrine/ca/2023/ 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 ~/ca/project_20 にコピーする.

```
$ cp /home/u_nesrine/ca/2023/src/blink.v .
$ cp /home/u_nesrine/ca/2023/src/bink.xdc .
```

Bitstream file generation and FPGA configuration



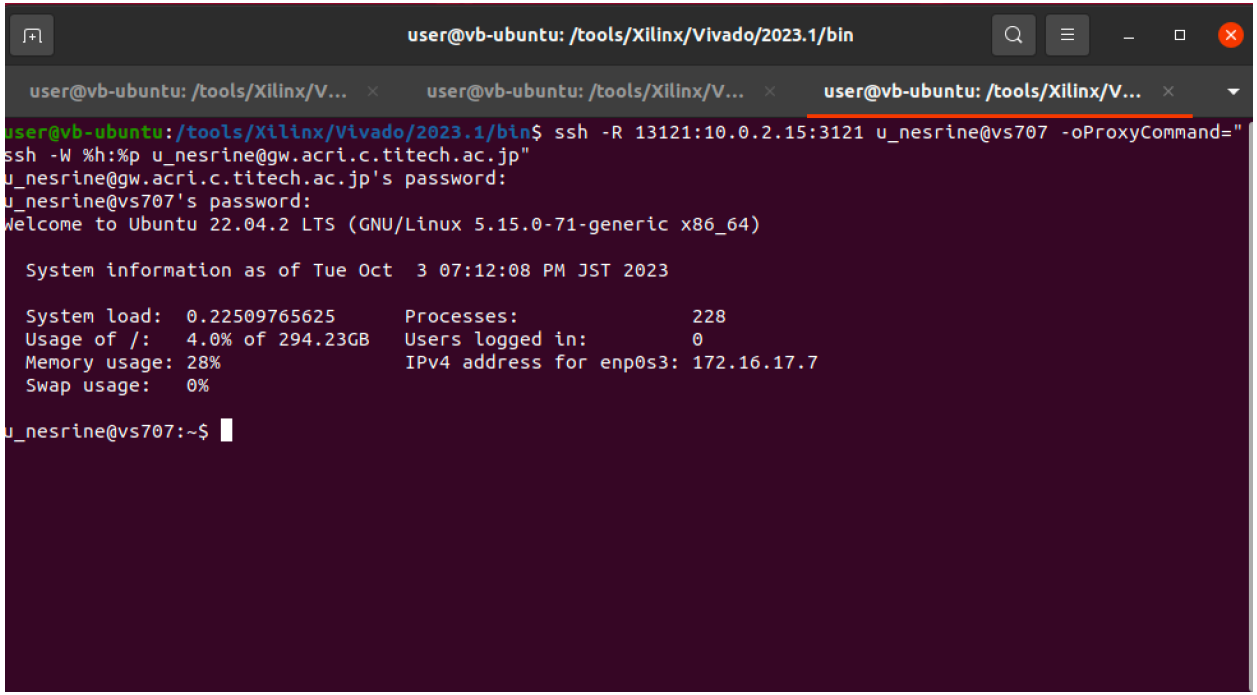
- Click **Add Sources**, then select **Add or create design sources** and click **Next**.
  - In **Add or Create Design Sources** window, click **Add Files**, select **blink.v** in project_20 directory, and click **OK**.
  - Click **Finish**.
- Click **Add Sources**, then select **Add or create constraints** and click **Next**.
  - In **Add or Create Design Sources** window, click **Add Files**, select **blink.xdc** in project_20 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.

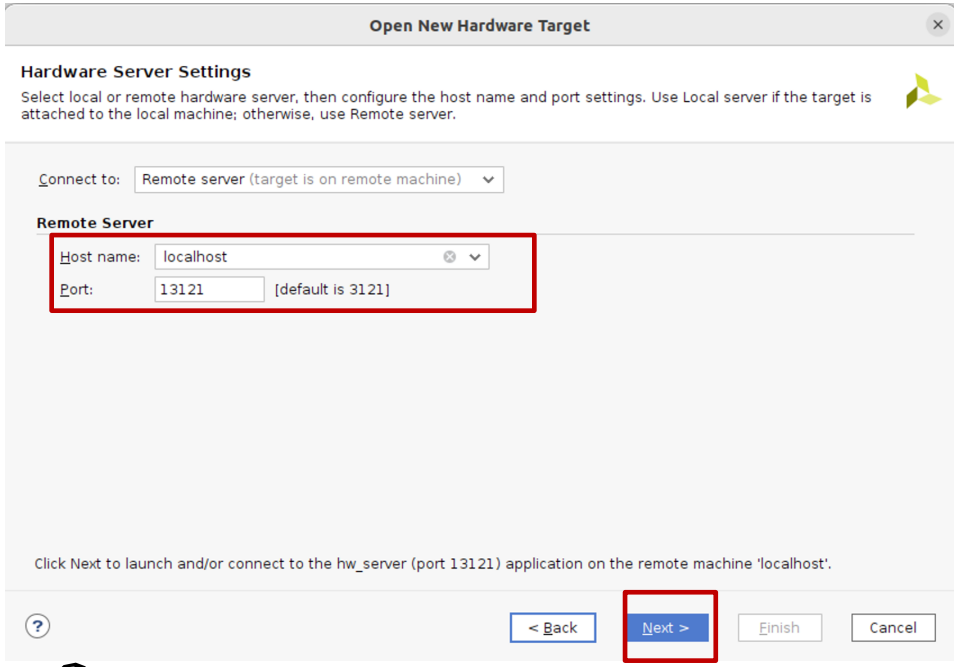
```
$ ssh -R 13121:<Local Machine IP Address>:3121 <ACRiユーザアカウント名>@<使用するサーバ名>: -oProxyCommand="ssh -W %h:%p <ACRi username>@gw.acri.c.titech.ac.jp"
```

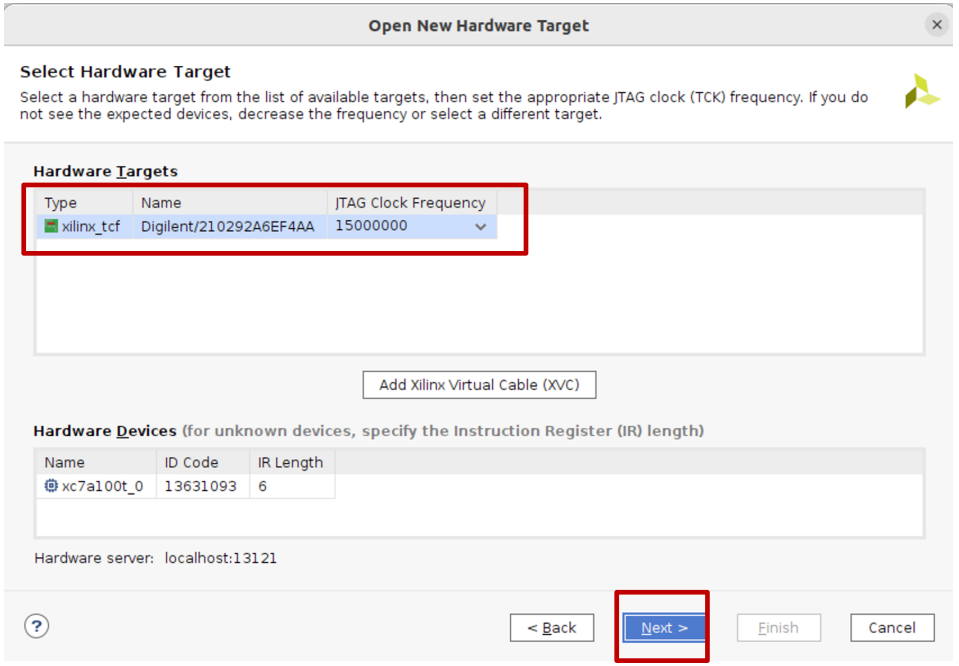
⟶ <Local Machine IP>: 10.0.2.15 にする.



Bitstream file generation and FPGA configuration

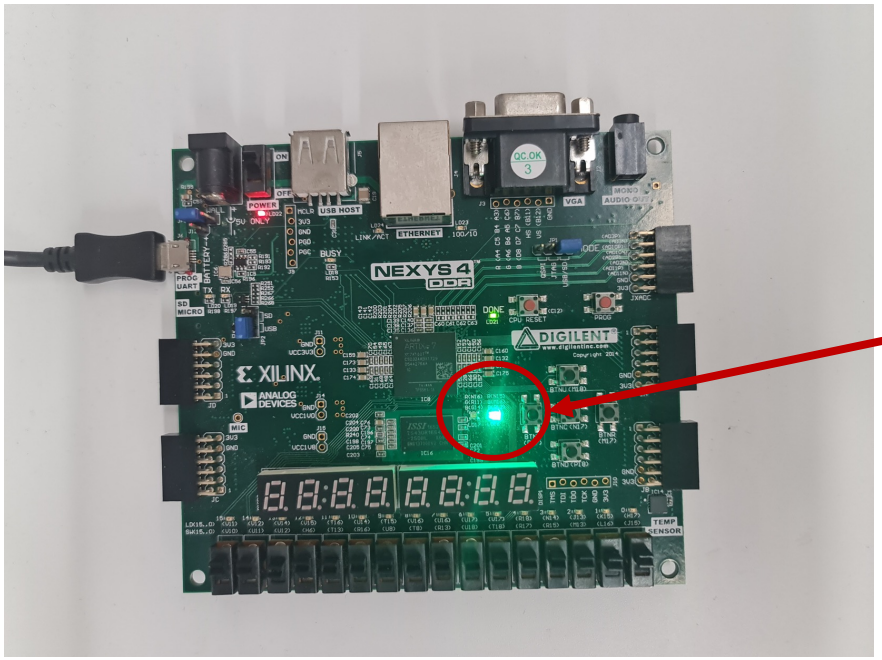
- Hardware Manager を開く
- Open Target を押して以下のように入力を進めていく
- 「Host name」に localhost と入力する.
- 「Port」に 13121 と入力する.
- Click Next and then Finish
- Then click Program device

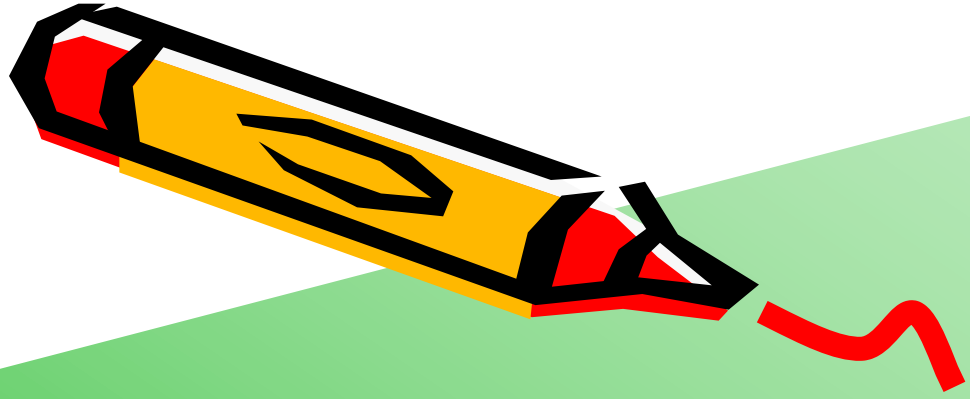




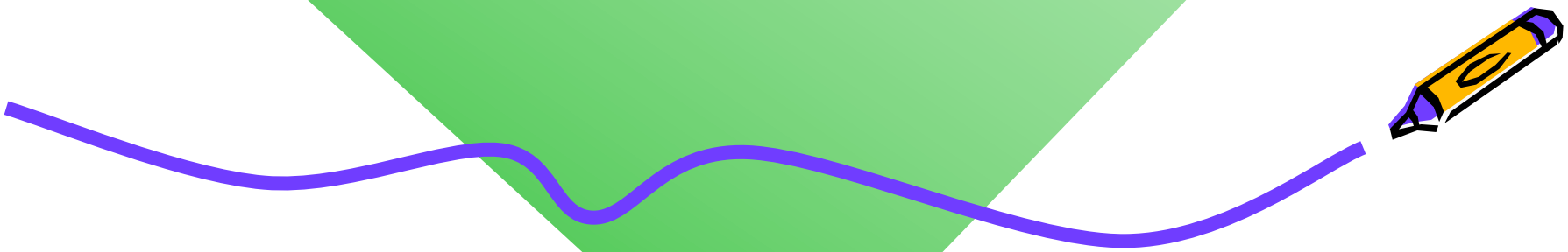
リモートコンフィグレーションの確認

- 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.





Project_21



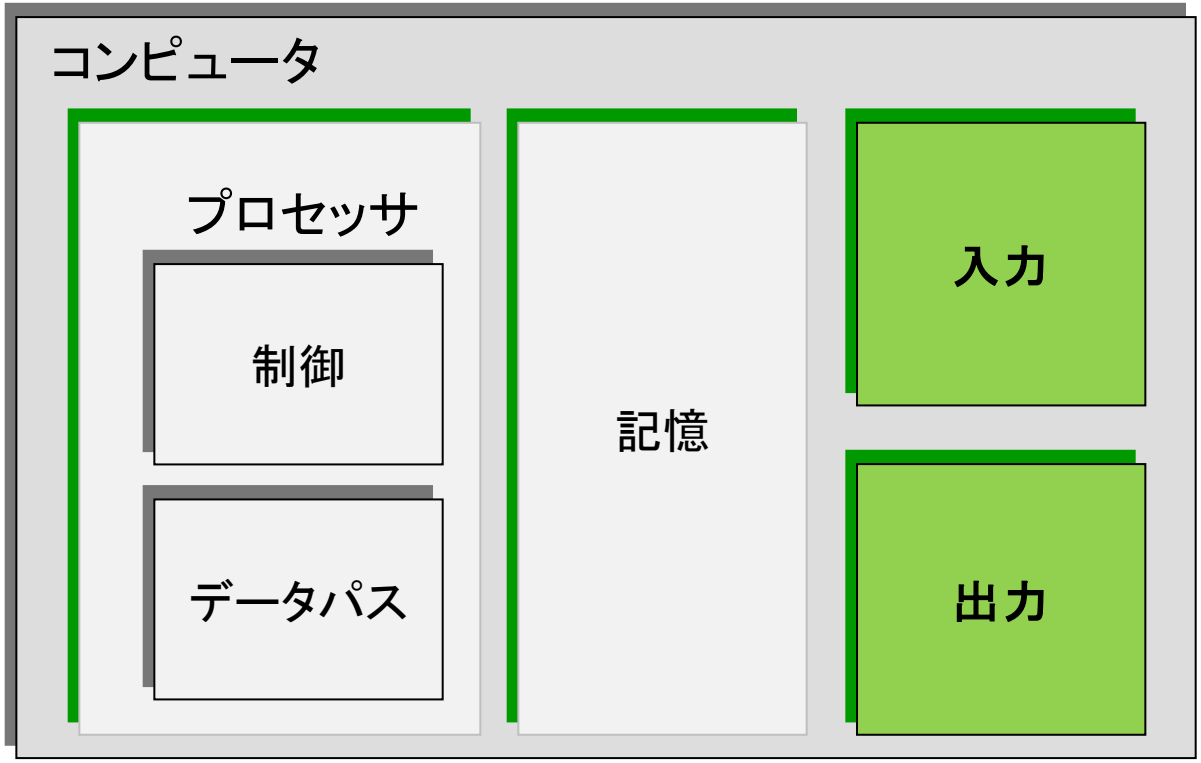
コンピュータの古典的な要素

コンパイラ

インタフェース

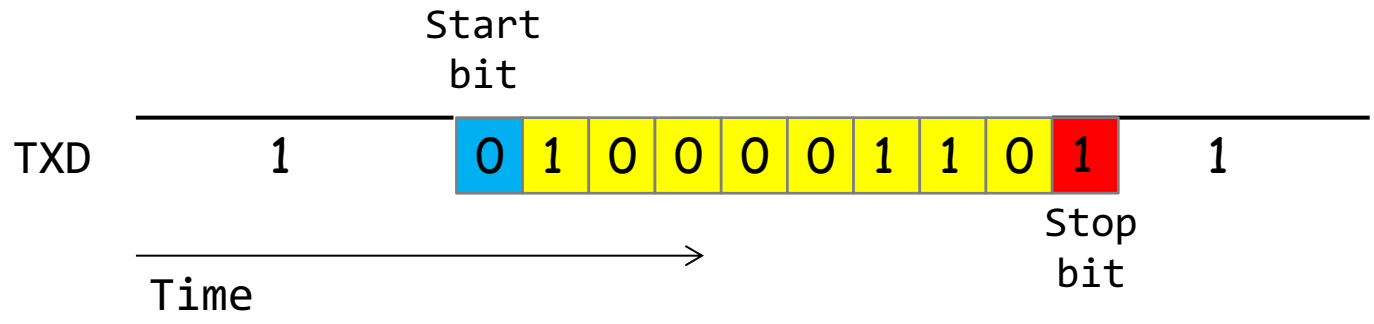
Instruction Set Architecture (ISA), 命令セットアーキテクチャ

性能の評価



UART (Universal Asynchronous Receiver/Transmitter)

- 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．

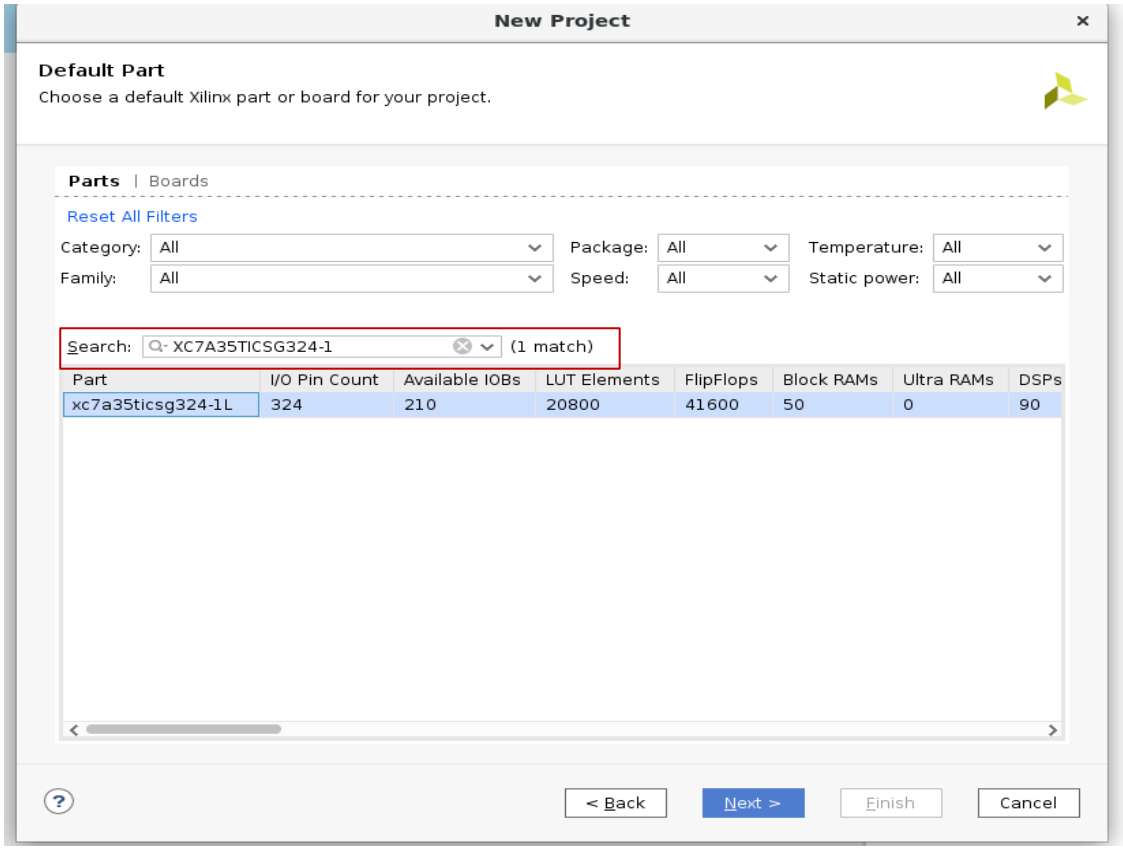


ASCII Table

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	

演習環境情報

- 新しVivado Project「project_21」を作る.
- Vivado
  - 2022.01のバージョンを利用
- 搭載するFPGA
  - XC7A35TICSG324-1L



シリアル通信による送信回路 m_UartTx

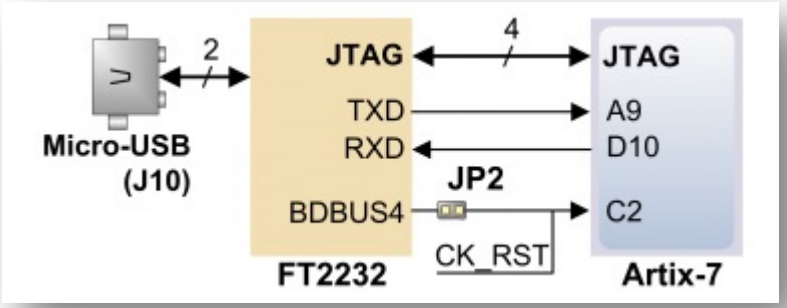
- システムクロック 50MHz，1Mbaud を想定する送信回路
- トップのモジュール m_main では，2秒に1回の頻度で，文字 a を送信する．

code201.v

```
/*************************************************************************************/
/* code201.v                For CSC.T363 Computer Architecture, Archlab TOKYO TECH */
/*************************************************************************************/
`timescale 1ns/100ps
`default_nettype none
/*************************************************************************************/
module m_main (w_clk100, w_txd);
   input  wire w_clk100;
   output wire w_txd;
      reg r_clk = 0;
   always@(posedge w_clk100) r_clk <= ~r_clk; // 50MHz clock signal
      reg [31:0]  r_cnt = 1;
   always@(posedge r_clk) r_cnt <= (r_cnt>(100_000_000 - 1)) ? 0 : r_cnt+1;
   m_UartTx m_UartTx0(r_clk, 8'h61, (r_cnt==0), w_txd);
endmodule
/*************************************************************************************/
module m_UartTx (w_clk, w_din, w_we, w_txd);
   input wire       w_clk, w_we;
   input wire [7:0] w_din;
   output wire      w_txd;
   reg [8:0]        r_buf = 9'b111111111;
   reg [7:0]        r_wait= 0;
   always@(posedge w_clk) begin
      r_wait <= (w_we) ? 0 : (r_wait>=49) ? 0 : r_wait + 1;
      r_buf  <= (w_we) ? {w_din, 1'b0} : (r_wait>=49) ? {1'b1, r_buf[8:1]} : r_buf;
   end
   assign w_txd = r_buf[0];
endmodule
/*************************************************************************************/
```

Inside main20.xdc

- 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）

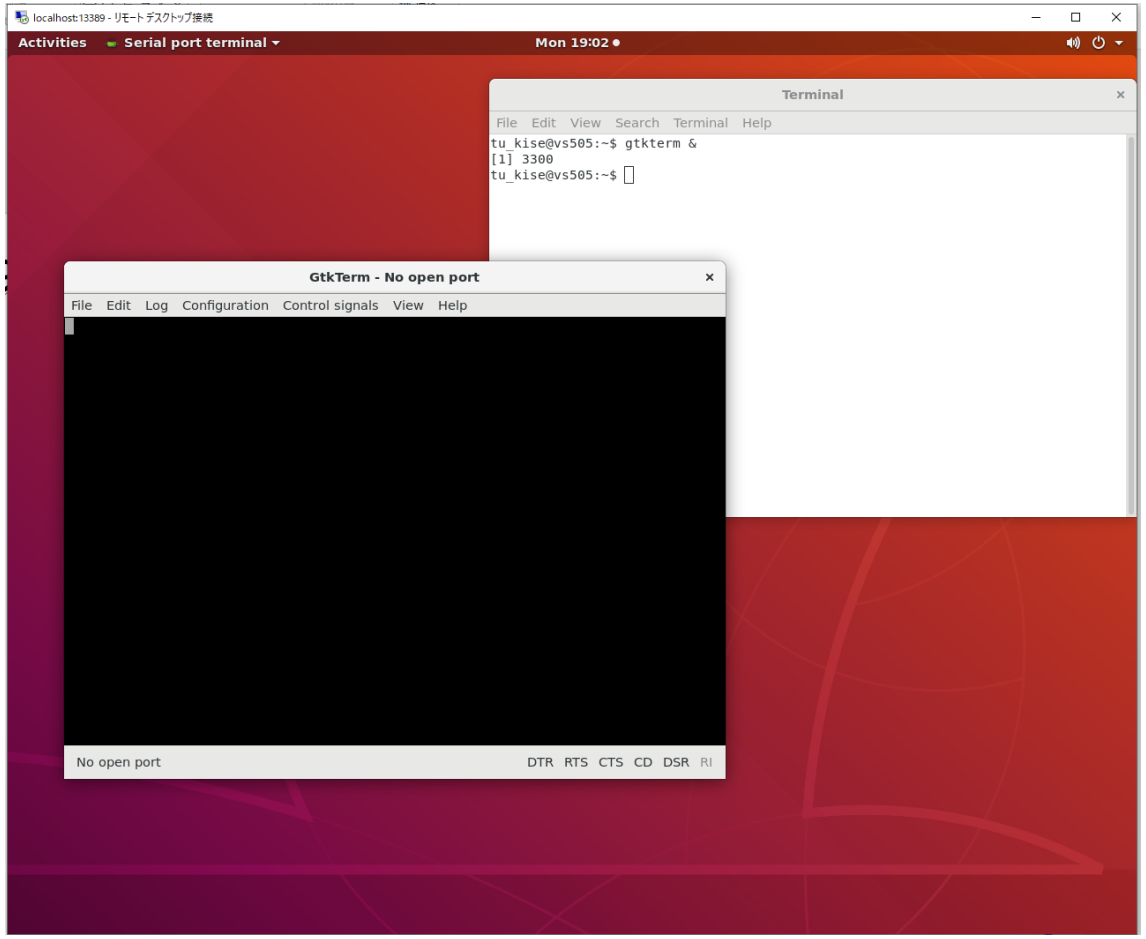


main20.xdc

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design] //bitファイル圧縮
set_property BITSTREAM.CONFIG.CONFIGRATE 50  [current_design] //書き込み高速化
set_property CONFIG_MODE SPIx4               [current_design] //書き込み高速化

set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports { w_clk100 }];
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 }];
```

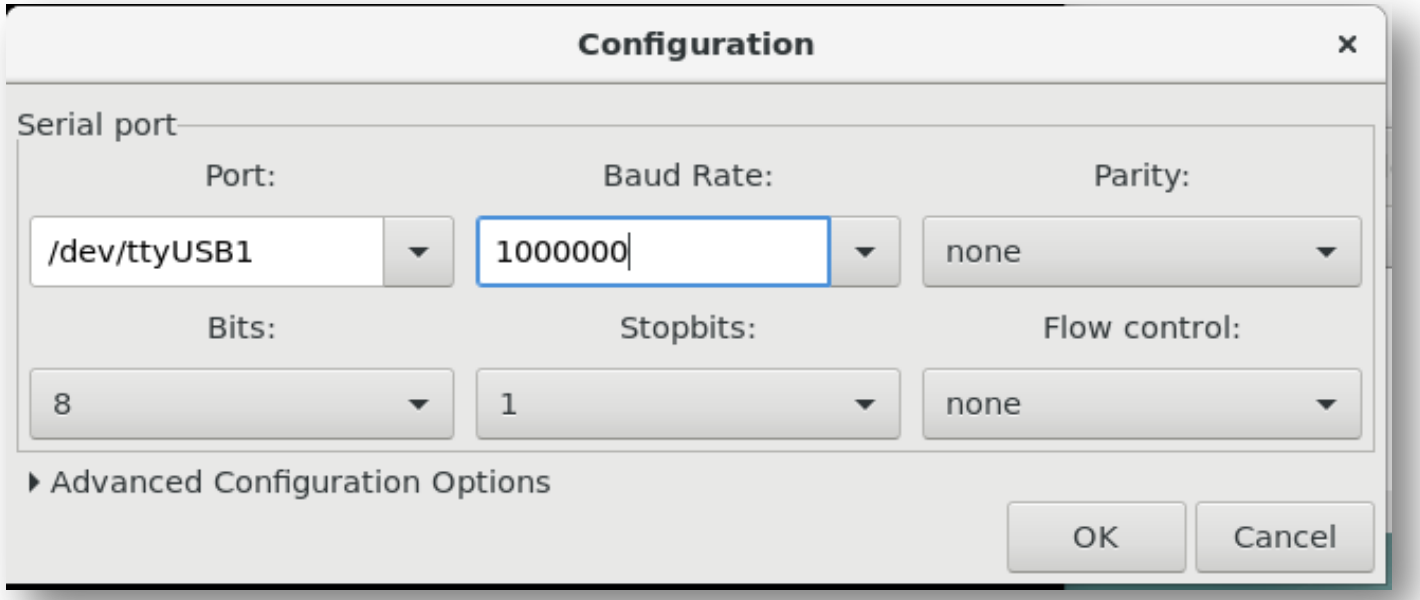
GtkTerm を利用する

- 「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. Baud rate を変更する場合には, この値を適切に修正すること.
    - gtkterm –p /dev/ttyUSB1 –s 1000000 &
  - OK ボタンを押す.



Vivado でビットファイルを生成してコンフィギュレーション

- Vivado のプロジェクトを作成する.
  - 設計ファイル code201.v
  - 制約ファイル main20.xdc
  - として, ビットストリームファイルを生成して, コンフィギュレーションする.
- GtkTerm に 2秒に1回の間隔で a が表示される.

シリアル通信による受信回路 m_UartRx

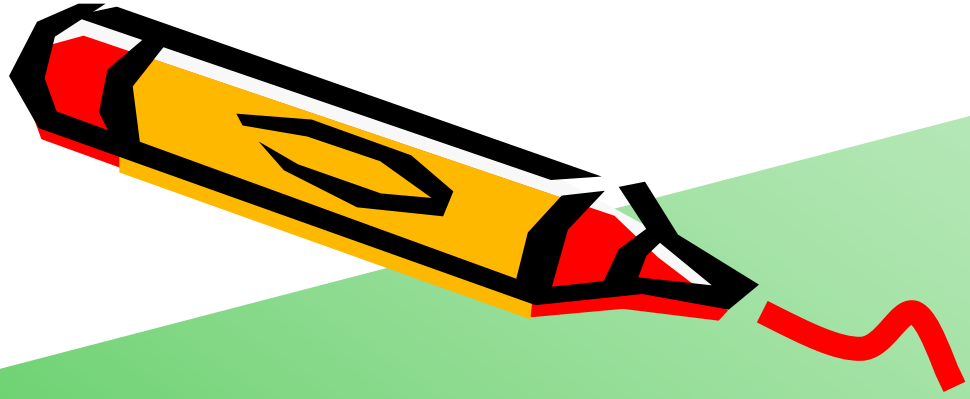
- システムクロック 50MHz，1Mbaud を想定する受信回路.
- 受信した8ビットのデータを r_dout に出力し，そのことを伝えるために r_en を1にする.

code202.v

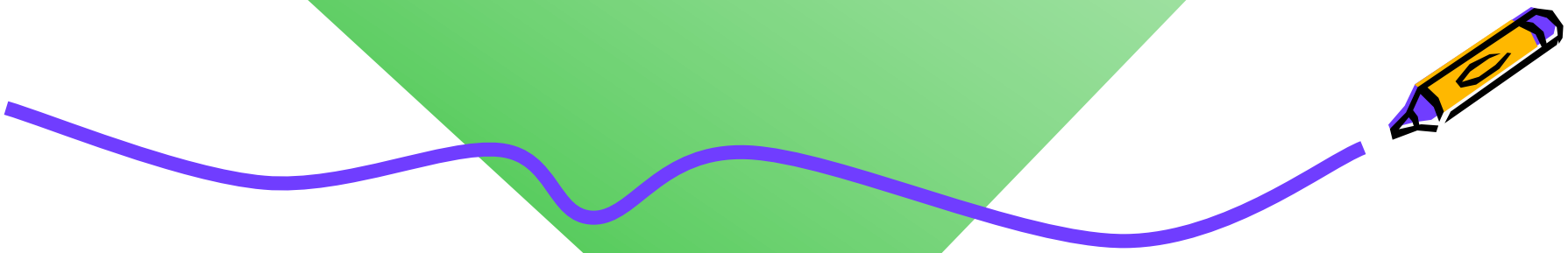
```
/*************************************************************************************/
module m_UartRx (w_clk, w_rxd, w_dout, r_en);
   input  wire       w_clk, w_rxd;
   output wire [7:0] w_dout;
   output reg        r_en = 0;
   reg [2:0] r_detect_cnt = 0; /* to detect the start bit */
   always @(posedge w_clk) r_detect_cnt <= (w_rxd) ? 0 : r_detect_cnt + 1;
   wire w_detected = (r_detect_cnt>2);
   reg       r_busy = 0;
   reg [3:0] r_cnt  = 0;
   reg [7:0] r_wait = 0;
   always@(posedge w_clk) r_wait <= (r_busy==0) ? 0 : (r_wait>=49) ? 0 : r_wait + 1;
   reg [8:0] r_data = 0;
   always@(posedge w_clk) begin
      if (r_busy==0) begin
         {r_data, r_cnt, r_en} <= 0;
         if(w_detected) r_busy <= 1;
      end
      else if (r_wait>= 49) begin
         r_cnt  <= r_cnt + 1;
         r_data <= {w_rxd, r_data[8:1]};
         if (r_cnt==8) begin r_en <= 1; r_busy <= 0; end
      end
   end
   assign w_dout = r_data[7:0];
endmodule
/*************************************************************************************/
```

送信回路と受信回路を用いたデザインの設計と実装

- 50MHz のシステムクロック，1Mbaud (1,000,000 baud) の UARTで，「受信した文字」をそのまま送信するモジュール m_main を設計・実装して，FPGAで動作を確認する．
  - 先に示した m_UartTx と m_UartRx をそのまま使う．
  - 正しく動作している画面を担当TAに確認してもらう．



References



References (1/2)

- Computer Architecture support page
  - http://www.arch.cs.titech.ac.jp/lecture/CA/
- Computer Logic Design support page
  - http://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/

References (2/2)

- Xilinx Vivado Design Suite
  - https://japan.xilinx.com/products/design-tools/vivado.html
- Digilent Arty A7-35 A7: FPGA Trainer Board
  - https://reference.digilentinc.com/reference/programmable-logic/arty-a7/start
- Digilent Nexys 4 DDR Atrix-7 FPGA
  - https://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/
- Verilog HDL
  - https://ja.wikipedia.org/wiki/Verilog