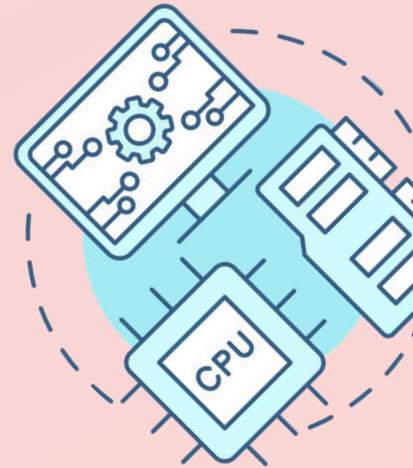


# コンピュータアーキテクチャ 演習 (4)

## Computer Architecture Exercise (4)

情報工学系 Berjab Nesrine

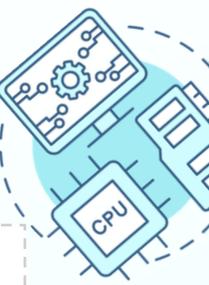


Computer Architecture support page :

<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

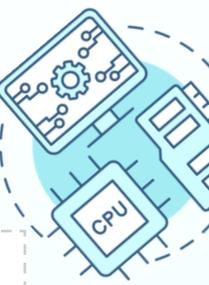
- 演習は **15:25~17:05** の時間で行う。**15:20** までに学術国際情報センター 3階、**情報工学系計算機室**に集合すること。**15:45** までに到着しない場合、欠席扱いになる。
- 最初の15分は課題の説明、その後は課題の進行とチェックポイントの確認を行う。演習ではACRi ルームを利用する。



## □ 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)



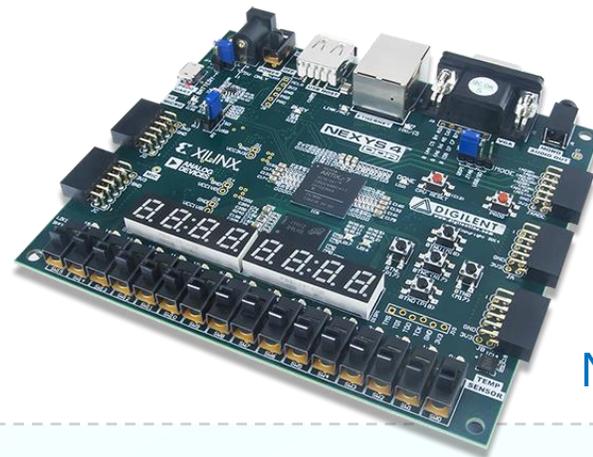
## □ 出席について

- 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。



ACRi ルームへようこそ！

📅 2024.09.24 🕒 2020.06.14

ようこそ。ACRi ルームは、100枚を超える FPGA ボードや [Alveo](#), [Versal](#) を含むサーバ計算機をリモートからアクセスして利用できる FPGA 利用環境です。

利用にはアカウントが必要です。[利用規約](#)と右カラムの利用説明をよく読んだ上で、[アカウントを申請](#)してください。提供された個人情報は[プライバシーポリシー](#)に従って管理・利用します。

【メンテナンス予告】ACRi ルームへのログイン時に最初に接続するサーバの更新を、9月24日 18:00 ごろに実施しました。SSH 接続時にエラーが出た場合は、`ssh-keygen -R gw.acri.c.titech.ac.jp` で既存の接続先情報を削除してください。(2024-09-24)

【復旧情報】ハードウェアトラブルにより稼働を停止していた as006 のサービスを再開しました。(2024-06-27)

ACRi ルームをより楽しむためのコンテンツとして、高位合成向けのプログラミングコンテストである [ACRi HLS Challenge](#) を開設しております。併せてご利用ください。チャレンジや高位合成に関する質問・コメントは [HLS Challenge についてのフォーラム](#) へどうぞ。

## 日別スケジュール

<前日 2024-10-08 \* 翌日> 移動 サーバ: 全て表示

サーバ	vs001	vs002	vs003	vs004	vs005	vs006	vs007	vs008
00:00	Close							
03:00	Open							
06:00	Open							
09:00	Open							
12:00	Open							
15:00	Open							
18:00	Open							
21:00	Open							



# 演習第五回の内容

## □目的:

### ➤ Project 5:

- 32 エントリ、ブロックサイズ1ワードのキャッシュのエントリ数を増やして、64 エントリに修正する。

### ➤ Project 6:

- MIG (Memory Interface Generator) を使って DRAM を動かし、読み出しと書き込みにかかる最大のサイクル数を求める。



# 演習第五回の内容 (Project 5)

## □目的:

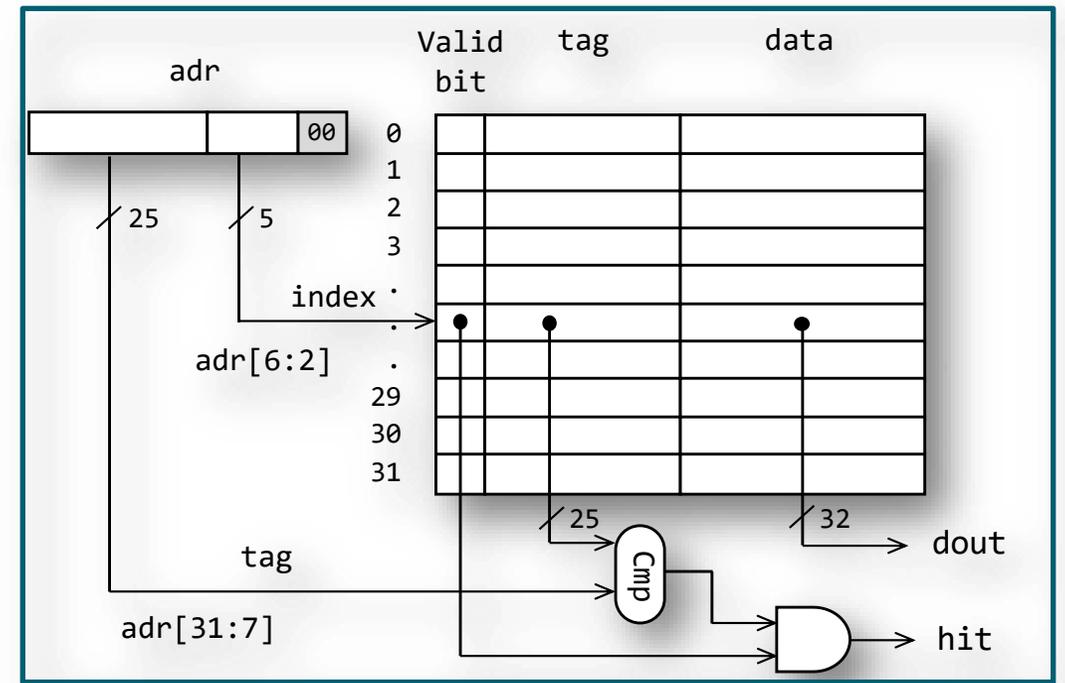
- ディレクトマッピング方式のキャッシュがどのように動作し、タグやインデックスを使ってデータが保存・アクセスされるかを理解する。
- キャッシュメモリのサイズやインデックスの幅を変更して、キャッシュエントリ数を調整する方法を理解する。
- キャッシュエントリ数を増やすことでキャッシュヒット・ミス率にどのような影響があるか、またシステム全体の性能にどう関わるかを確認する。



# ダイレクトマップ方式のキャッシュの基本

## □ダイレクトマップ方式のキャッシュの仕組み:

- キャッシュラインの位置はインデックスによって一意に定まる (直接写像される)。
- 主要な要素:
  - **index**: キャッシュ内の特定のラインを選択する。
  - **tag**: 正しいメモリブロックがそのラインに格納されているか識別するために使用される。
  - **valid bit**: そのキャッシュラインのデータが有効かどうかを示す。
  - **data**: 実際に格納されているデータ
    - この場合、32ビットのワード





# 現在のキャッシュコード (1/2)

- 1) キャッシュは32エントリの配列を使って実装されており、それぞれにバリッドビット、タグ、データを格納している。
- 2) 現在のコードでは、`adr[6:2]` の 5ビットをインデックスとして、キャッシュにアクセスする。
- 3) アドレスタグ (`adr[31:7]`) は格納されたタグと比較され、ヒットかミスかを判断する。

proc2.v

```
module m_cache_direct_mapped (  
    input wire w_clk,  
    input wire w_we,  
    input wire [31:0] w_adr,  
    input wire [31:0] w_wadr,  
    input wire [57:0] w_wd,  
    output wire w_hit,  
    output wire [31:0] w_dout  
);  
  
    reg [57:0] mem [0:31];  
    integer i; initial for (i=0; i<32; i=i+1) mem[i] = 0;  
  
    wire [4:0] w_index = w_adr[6:2];  
    wire w_v;  
    wire [24:0] w_tag;  
    assign {w_v, w_tag, w_dout} = mem[w_index];  
    assign w_hit = w_v & (w_adr[31:7]==w_tag);  
  
    always @(posedge w_clk) if (w_we) mem[w_index] <= w_wd;  
endmodule
```

(Source code available in /home/u\_nesrine/ca2024/src)



# 現在のキャッシュコード (2/2)

## □動作確認:

- `proc2.v`、`sample2.txt` を同じディレクトリにコピーする。
- シミュレーションにより、動作確認を行う。
- シミュレーションを実行すると以下の結果が出力される。

```
$ cd ~/ca2024/  
$ cp /home/u_nesrine/ca2024/src/proc2.v .  
$ cp /home/u_nesrine/ca2024/src/sample2.txt .  
$ /tools/cad/bin/verilator --binary -o simv proc2.v  
$ ./obj_dir/simv
```

```
simulation finished.  
cache hit          11643  
cache miss         13774  
total (hit+miss)   25417  
- proc_ex4.v:410: Verilog $finish  
- Simulation Report: Verilator 5.028 2024-08-21  
- Verilator: $finish at 3us; walltime 0.195 s; speed 21.203 us/s  
- Verilator: cpu 0.120 s on 1 threads; alloced 9 MB
```



# キャッシュサイズを64エントリにする

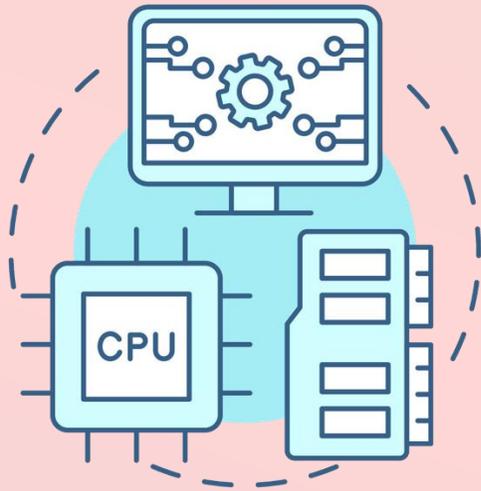
## □ 目標 :

- 32エントリと1ワードのブロックサイズを持つダイレクトマップ方式のキャッシュが与えられている。
- `proc2.v` の `m_cache_direct_mapped` モジュールを変更し、エントリ数を32から64に増やすこと。
- 変更が完了したら、シミュレーションを実行し、出力が以下と一致するか確認する。

```
simulation finished.  
cache hit          21559  
cache miss         3858  
total (hit+miss)   25417  
- proc_ex4.v:411: Verilog $finish  
- Simulation Report: Verilator 5.028 2024-08-21  
- Verilator: $finish at 3us; walltime 0.129 s; speed 23.262 us/s  
- Verilator: cpu 0.109 s on 1 threads; allocated 9 MB
```



Check Point 6



# Project 6



# 演習第五回の内容 (Project 6)

## □課題:

- DRAM の読み出しおよび書き込みにかかる最大のサイクル数を求める。
  - **目的** : DRAM の読み出しおよび書き込みにどのくらいのサイクル数がかかるかを知る。そして、DRAM のアクセスが遅いことを理解する。
  - **方法** : DRAM の読み出し（リード）および書き込み（ライト）を繰り返し、サイクル数を測定する。読み出しおよび書き込みのそれぞれについて、「測定したサイクル数」がそれまでの「最大のサイクル数」を上回った場合、「最大のサイクル数」を更新する。
  - **出力** : VIO で読み出しおよび書き込みにかかった最大のサイクル数を出力する。

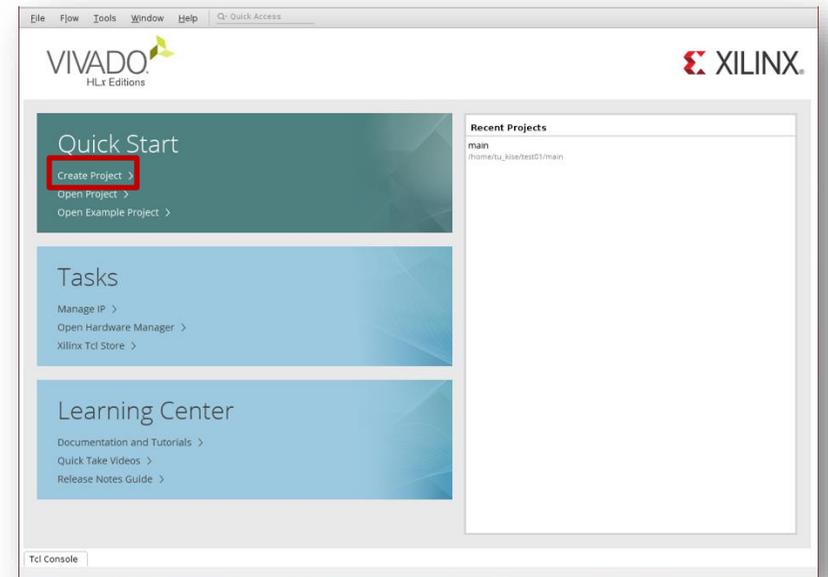


# 新しい Vivado Project を作る (1/2)

- ❑ 新しい Vivado Project 「project\_6」 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\_6” and location “/home/your\_username/ca2024” are selected.
  - Check “Create project subdirectory”.
- ❑ Click **Next**
- ❑ In **Default Part** window, select **Parts**, and write XC7A35TICSG324-1L.





# ステップ1:新しい Vivado Project を作る (2/2)

## ❑ 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 `main6.v`、`main6.xdc`と `arty_a7_mig.ucf` を作成したプロジェクトのディレクトリ `~/ca2024/project_6` にコピーする。

❑ Click [Add Sources](#), then select [Add or create design sources](#) and click [Next](#).

❑ In [Add or Create Design Sources](#) window, click [Add Files](#), select `main6.v` in `project_6` 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 `main6.xdc` in `project_6` directory, and click [OK](#).

❑ Click [Finish](#).

```
$ cd ~/ca2024/project_6
$ cp /home/u_nesrine/ca2024/src/main6.v .
$ cp /home/u_nesrine/ca2024/src/main6.xdc .
$ cp /home/u_nesrine/ca2024/src/arty_a7_mig.ucf .
```



# main6.v コードの説明 (1/3)

□ main6.v のコードは、DDR3 メモリに対する読み書き処理を行うモジュール。

□ DDR3 メモリとのインターフェースには、Xilinx のメモリインターフェース生成 (MIG) IP コアを使用している。

□ MIG コアに対して、書き込みコマンドと読み出しコマンドを送ることによって、メモリにデータを書き込んだり読み出したりする処理が実装されている。

□ また、メモリ初期化完了などの状態をLEDに表示する。

□ 入出力ポート:

□ w\_clk: 100MHz のクロック入力。

□ w\_led: 4ビットの LED 出力で、動作状態を表示する。

□ ddr3\_\* の各信号は DDR3 メモリとの接続用。

## main6.v

```
module m_main (  
    input wire w_clk,           // 100MHz clock signal  
    output wire [3:0] w_led,    // LED  
    //rest of code  
);
```

(Source code available in /home/u\_nesrine/ca2024/src)



# main6.v コードの説明 (2/3)

## □ レジスタと信号:

main6.v

```
//code  
reg [ `APP_ADDR_WIDTH-1 : 0 ] r_app_addr = 0;  
reg [ `APP_CMD__WIDTH-1 : 0 ] r_app_cmd = 0;  
//rest of code
```

(Source code available in /home/u\_nesrine/ca2024/src)

- いくつかのレジスタやワイヤが宣言されており、主に DDR3 メモリコントローラ (mig\_7series\_0) とのインターフェース用に使用される。
- r\_app\_addr と r\_app\_cmd はメモリ操作のためのアドレスとコマンドを指定するためのもの。
- r\_app\_en はコマンドが有効であること、r\_app\_wdf\_wren は書き込みが有効であることを示す。
- r\_app\_wdf\_data にはメモリへの書き込みデータである。
- r\_app\_wdf\_mask は書き込みデータがバイトごとに無効であるか否かを表す (0: 有効、1: 無効)。
- app\_rd\_data はメモリからの読み出しデータである。



# main6.v コードの説明 (3/3)

## □ メモリ操作のステートマシン

- `r_state` が書き込みと読み取り処理を管理するステートマシン。
- `init_calib_complete` が1になったときに、DRAMの初期化が完了する。
- **書き込みプロセス:**
  - ステート0で、メモリが準備完了 (`app_rdy`) で書き込みデータが受け入れ可能 (`app_wdf_rdy`) の場合、`r_app_en` と `r_app_wdf_wren` を1に設定し書き込み開始。
  - 開始後、ステート1に進み、書き込み完了と `r_app_addr` のインクリメントを実行。
- **読み出しプロセス:**
  - ステート3でメモリが準備完了の場合、`r_app_cmd` を `CMD_READ` に設定して読み取りを開始。
  - ステート4で、`app_rd_data_valid` が有効なときデータを読み出し、`r_sum` に加算し、アドレスをインクリメント。

### main6.v

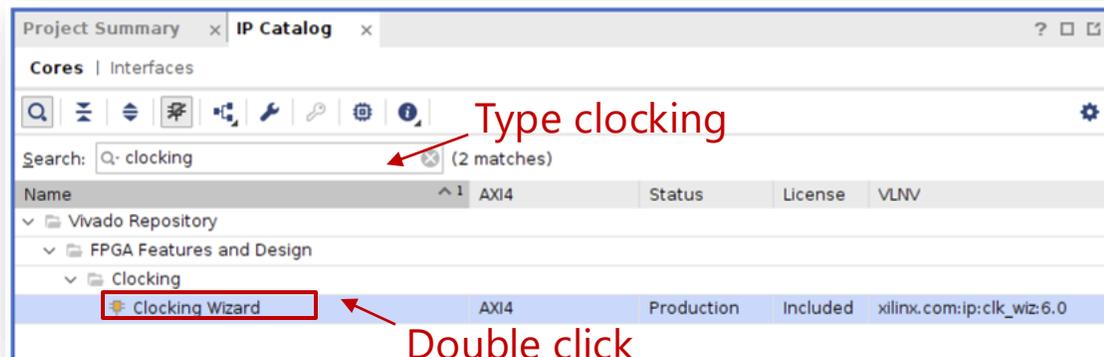
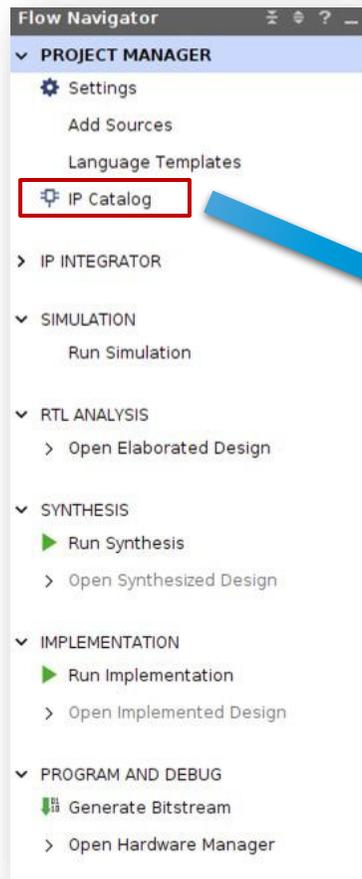
```
//code
reg [3:0] r_state = 0;
always @(posedge w_ui_clk) if (init_calib_complete) begin
    if (r_state==0 && app_rdy && app_wdf_rdy) begin ///// WRITE_1
        r_app_en           <= 1;
        r_app_wdf_wren     <= 1;
        r_app_cmd          <= `CMD_WRITE;
        r_state            <= 1;
    end
    //rest of code
end
```

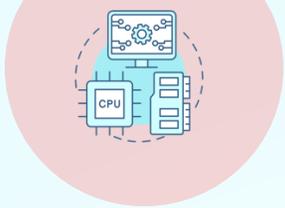
(Source code available in /home/u\_nesrine/ca2024/src)



# Clocking Wizard でclock を変化させる (1/2)

- ❑ Click IP Catalog
- ❑ Double click **Clocking Wizard** in IP Catalog window





# Clocking Wizard でclock を変化させる (2/2)

- Clocking Options**, Input Clock Information  
: 100.000 MHz Input Frequency
- Output Clocks**, clk\_out1 : 166.6 MHz Output Freq is Requested
- Output Clocks**, clk\_out2 : 200 MHz Output Freq is Requested
- Output Clocks**: disable reset, power\_down, input\_clk\_stopped, locked, clksbstopped

Component Name: clk\_wiz\_0

Clocking Options | Output Clocks | Port Renaming | MMCM Settings | Summary

**Clock Monitor**

Enable Clock Monitoring

**Primitive**

MMCM  PLL

**Clocking Features**

Frequency Synthesis  Minimize Power  
 Phase Alignment  Spread Spectrum  
 Dynamic Reconfig  Dynamic Phase Shift  
 Safe Clock Startup

**Jitter Optimization**

Balanced  
 Minimize Output Jitter  
 Maximize Input Jitter filtering

**Dynamic Reconfig Interface Options**

AXI4Lite  DRP  Phase Duty Cycle Config  Write DRP registers

**Input Clock Information**

Input Clock	Port Name	Input Frequency(MHz)	Jitter Options	Input jitter	Source	
<input checked="" type="checkbox"/> Primary	clk_in1	100.000	10.000 - 800.000	UI	0.010	Single ended clock capable pin
<input type="checkbox"/> Secondary	clk_in2	100.000	60.000 - 120.000		0.010	Single ended clock capable pin

OK Cancel

Component Name: clk\_wiz\_0

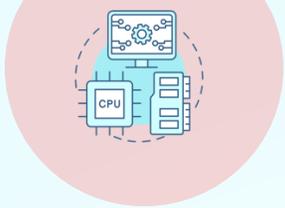
Clocking Options | **Output Clocks** | Port Renaming | MMCM Settings | Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives
		Requested	Actual	Requested	Actual	Requested	Actual	
<input checked="" type="checkbox"/> clk_out1	clk_out1	166.600	166.66667	0.000	0.000	50.000	50.0	BUFG
<input checked="" type="checkbox"/> clk_out2	clk_out2	200.000	200.00000	0.000	0.000	50.000	50.0	BUFG
<input type="checkbox"/> clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG

### Enable Optional Inputs / Outputs for MMCM/PLL

- reset  power\_down  input\_clk\_stopped  
 locked  clksbstopped



# VIO (Virtual Input/Output) の設定

## □ VIO のコンフィギュレーション:

1. Vivadoで IP Catalog を開き、vio を検索する。

2. 以下のようにVIOを設定する:

➤ Input Probe Count: 4

➤ Probe Width:

➤ In0: 28

➤ In1: 32

➤ In2: 8

➤ In3: 8

3. Click **Generate** and click **OK** if asked in Generate Output Products window.

The screenshot shows the Vivado IP Catalog interface. The search results for 'vio' are displayed, with 'VIO (Virtual Input/Output)' selected. A red box highlights the search term 'vio' and the selected IP. A red arrow points to the search term with the label 'Type vio'. Another red arrow points to the selected IP with the label 'Double click'. A blue arrow points from the selected IP to the configuration window. The configuration window shows the 'General Options' tab for 'VIO (Virtual Input/Output)'. The 'Input Probe Count' is set to 4, and the 'Output Probe Count' is set to 0. A red box highlights the 'Input Probe Count' field. A blue arrow points from the configuration window to the 'Generate Output Products' window. The 'Generate Output Products' window shows the 'General Options' tab for 'VIO (Virtual Input/Output)'. The 'Probe Port' and 'Probe Width' are configured as follows:

Probe Port	Probe Width [1 - 256]
PROBE_IN0	28
PROBE_IN1	32
PROBE_IN2	8
PROBE_IN3	8

A red box highlights the table content.



# MIG (Memory Interface Generator) の設定 (1/8)

- ❑ Xilinx Memory Interface Generator (MIG) という IP コアをベースに Xilinx FPGA で DRAM メモリを動かすこと
- ❑ MIG を生成するための IP コアの起動方法
  - Click IP Catalog
  - Double click Memory Interface Generator (MIG 7 Series) in IP Catalog window

The screenshot shows the Vivado IP Catalog window with the search bar containing 'mig' and '(1 match)'. The 'Memory Interface Generator (MIG 7 Series)' is selected. A blue arrow points to the 'Memory Interface Generator' configuration window on the right, which has a 'Next' button highlighted with a red arrow and the text 'Click next'.



# MIG (Memory Interface Generator) の設定 (2/8)

- ❑ MIG Output Options: check **Create Design**
- ❑ MIG Output Options, **Component Name**: mig\_7series\_0
- ❑ MIG Output Options, **Multi-Controller**: 1
- ❑ MIG Output Options, **AXI4 Interface**: uncheck

**Memory Interface Generator**

**AMD**  
**Vivado**  
ML Edition

**MIG Output Options**

**Create Design**  
Select this option to generate a memory controller. Generating a memory controller will create RTL, XDC, implementation and simulation files.

**Verify Pin Changes and Update Design**  
Selecting this feature verifies the modified XDC for a design already generated through MIG. This option will allow you to change the pin out and validate it instantly. It updates the input XDC file to be compatible with the current version of MIG. While updating the XDC it preserves the pin outs of the input XDC. This option will also generate the new design with the Component Name you selected in this page.

**Component Name**

Please specify the component name for the memory interface. The design directories will be generated under a directory with this name. Three directories will be created "example\_design", "user\_design" and "docs". The user\_design will contain the generated memory interface. The example\_design adds a simple example application connected to the generated memory interface.

Component Name

**Multi-Controller**

Up to maximum of 8 controllers with a combination of DDR3 SDRAM, QDRII+ SRAM or RLD RAM II can be generated. The number of controllers that can be accommodated may be limited by the data width and the number of banks available in device. Refer user guide for more information

Number of controllers

**AXI4 Interface**

Enables the AXI4 interface. AXI4 interface is supported only for DDR3 SDRAM and DDR2 SDRAM controllers with Verilog design entry.

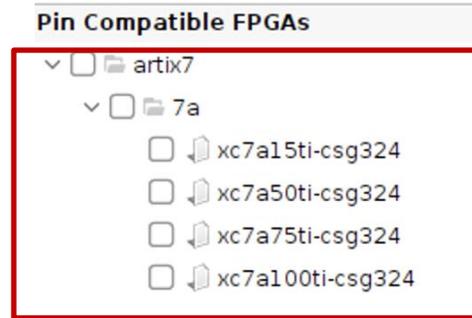
AXI4 Interface

User Guide < Back Next > Cancel

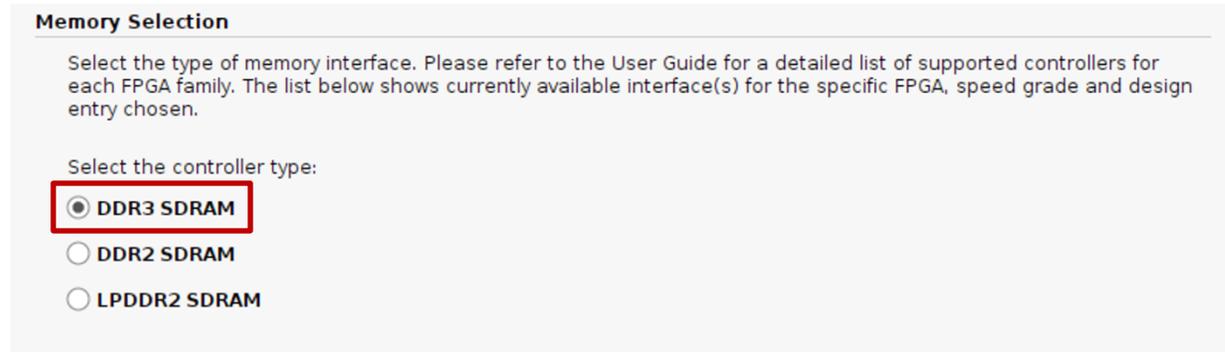


# MIG (Memory Interface Generator) の設定 (3/8)

Pin Compatible FPGAs: uncheck



Memory Selection: check DDR3 SDRAM





# MIG (Memory Interface Generator) の設定 (4/8)

- ❑ Options for Controller 0, **Clock Period:** 3000
- ❑ Options for Controller 0, **Memory Type:** MT41K128M16XX-15E
- ❑ Options for Controller 0, **Memory Type:** Create Custom Part
- ❑ Options for Controller 0, **Memory Voltage:** 1.35V
- ❑ Options for Controller 0, **Data Width:** 16
- ❑ Options for Controller 0, **Data Mask:** check
- ❑ Options for Controller 0, **Number of Bank Machines:** 4
- ❑ Options for Controller 0, **Ordering:** Strict

## Options for Controller 0 - DDR3 SDRAM

**Clock Period:** Choose the clock period for the desired frequency. The allowed period range(2500 - 3300) is a function of the selected FPGA part and FPGA speed grade. Refer to the User Guide for more information.

3,000 ps 333.33 MHz

**To achieve optimum resource utilization, maintain default clock period given by the tool or a value greater than default clock period. Please contact Xilinx Technical Support for further information**

**PHY to Controller Clock Ratio:** Select the PHY to Memory Controller clock ratio. The PHY operates at the Memory Clock Period chosen above. The controller operates at either 1/4 or 1/2 of the PHY rate. The selected Memory Clock Period will limit the choices.

4:1

**Memory Type:** Select the memory type. Type(s) marked with a warning symbol are not compatible with the frequency selection above.

Components

**Memory Part:** Select the memory part. Part(s) marked with a warning symbol are not compatible with the frequency selection above. Find an equivalent part or create a part using the "Create Custom Part" button if the part needed is not listed here. The "Create Custom Part" feature is not supported for RLD RAM II.

MT41K128M16XX-15E

Create Custom Part

**Memory Voltage:** Select the Voltage of the Memory part selected.

1.35V

**Data Width:** Select the Data Width. Parts marked with a warning symbol are not compatible with the frequency and memory part selected above.

16

**ECC:** MIG supports ECC for 72 bit data width configuration. To be able to select ECC, select a data width that has ECC supported.

Disabled

**Data Mask:** Enable or disable the generation of Data Mask (DM) pins using this check box. This option can be selectable only if the memory part selected has DM pins. Uncheck this box to not use data masks and save FPGA I/Os that are used for DM signals. ECC designs (DDR3 SDRAM, DDR2 SDRAM) will not use Data Mask.

**Number of Bank Machines:** This parameter defines the number of bank machines. A given bank machine manages a single DRAM bank at any given time.

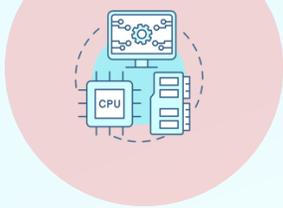
**Note:** Setting a lower value will result in lower resource utilization, but may effect controller efficiency for certain traffic patterns.

4

**ORDERING:** Normal mode allows the memory controller to reorder commands to the memory to obtain the highest possible efficiency. Strict mode forces the controller to execute commands in the exact order received.

Strict

**Memory Details:** 2Gb, x16, row:14, col:10, bank:3, data bits per strobe:8, with data mask, single rank, 1.35V,1.5V



# MIG (Memory Interface Generator) の設定 (5/8)

- Memory Options C0, **Input Clock**  
Period: 6000 ps
- Memory Options C0, **Read Burst Type and Length**: Sequential
- Memory Options C0, **Output Driver Impedance Control**: RZQ/6
- Memory Options C0, **RTT**: RZQ/6
- Memory Options C0, **Controller Chip Select Pin**: Enable
- Memory Options C0, **Memory Address Mapping Selection**: check below (BANK, ROW, COKUMN)

## Memory Options C0 - DDR3 SDRAM

**Input Clock Period:** Select the period for the PLL input clock (CLKIN). MIG determines the allowable input clock periods based on the Memory Clock Period entered above and the clocking guidelines listed in the User Guide. The generated design will use the selected Input Clock and Memory Clock Periods to generate the required PLL parameters. If the required input clock period is not available, the Memory Clock Period must be modified.

6000 ps (166.667 MHz)

Choose the Memory Options for the memory device. Memory Option selections are restricted to those supported by the controller. Consult the memory vendor data sheet for more information.

### Read Burst Type and Length

The burst type determines the data ordering within a burst. Consult the memory datasheet for more information. Burst length 8 is the only supported value.

Sequential

### Output Driver Impedance Control

Programmable impedance for the output buffer.

RZQ/6

### RTT (nominal) - On Die Termination (ODT)

Select the nominal value of ODT for the DQ, DQS/DQS# and DM signals on the component or DIMM interface. This must be set to RZQ/6 (40 ohms) for data rates at 1333 Mbps and above. In 2 slot DIMM configurations this value will be used for the unwritten slot during a write and will also be used for the unselected slot during a read. Use board level simulation to choose the optimum value.

RZQ/6

### Controller Chip Select Pin

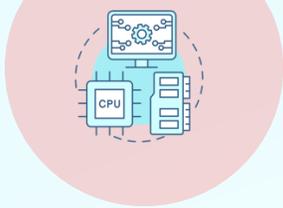
The Chip Select (CS#) pin can be tied low externally to save one pin in the address/command group when this selection is set to 'Disable'. Disable is only valid for single rank configurations.

Enable

### Memory Address Mapping Selection



ROW BANK COLUMN  
 BANK ROW COLUMN

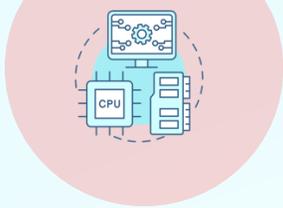


# MIG (Memory Interface Generator) の設定 (5/8)

- ❑ FPGA Options, **System Clock**: No Buffer
- ❑ FPGA Options, **Reference Clock**: No Buffer
- ❑ FPGA Options, **System Reset Polarity**: ACTIVE HIGH
- ❑ FPGA Options, **Debug Signals for Memory Controller**: OFF
- ❑ FPGA Options, **Internal Vref**: Check
- ❑ FPGA Options, **IO Power Reduction**: ON
- ❑ FPGA Options, **XADC Instantiation**: Enabled

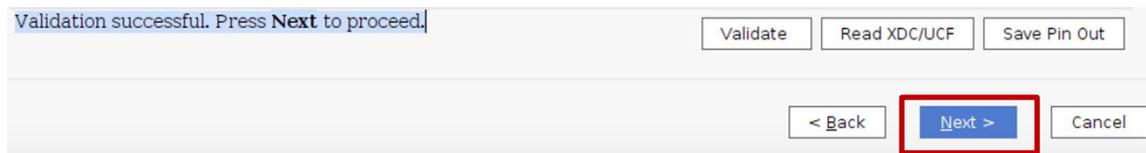
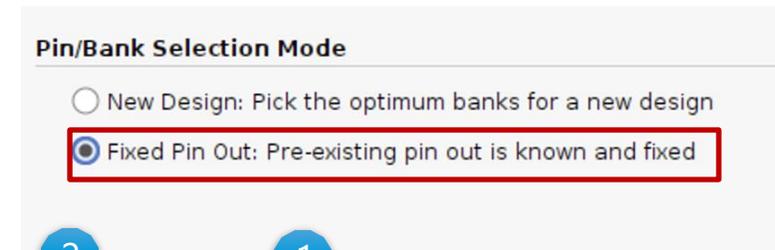
The screenshot displays the configuration interface for the Memory Interface Generator (MIG). The settings are as follows:

- System Clock:** No Buffer
- Reference Clock:** No Buffer
- System Reset Polarity:** ACTIVE HIGH
- Debug Signals Control:** Debug Signals for Memory Controller: OFF
- Sample Data Depth:** 1024
- Internal Vref:**
- IO Power Reduction:** ON
- XADC Instantiation:** Enabled



# MIG (Memory Interface Generator) の設定 (6/8)

- ❑ Extended FPGA Options, **Internal Termination Impedance: 50 Ohms**
- ❑ **Pin/Bank Selection Mode: check Fixed Pin Out: Pre-existing pin out is known and fixed**
- ❑ **Pin Selection: click Read XDC/UCF, and select a file named arty\_a7\_mig.ucf, and click Validate**
- ❑





# MIG (Memory Interface Generator) の設定 (7/8)

- ❑ System Signals Selection, **sys\_rst**: No connect
- ❑ System Signals Selection, **init\_calib\_complete**: No connect
- ❑ System Signals Selection, **tg\_compare\_error**: No connect

## System Signals Selection

Select the system pins below appropriately for the interface. Customization of these pins can also be made in the XDC after the design is generated. For more information see [UG586 Bank and Pin rules](#).

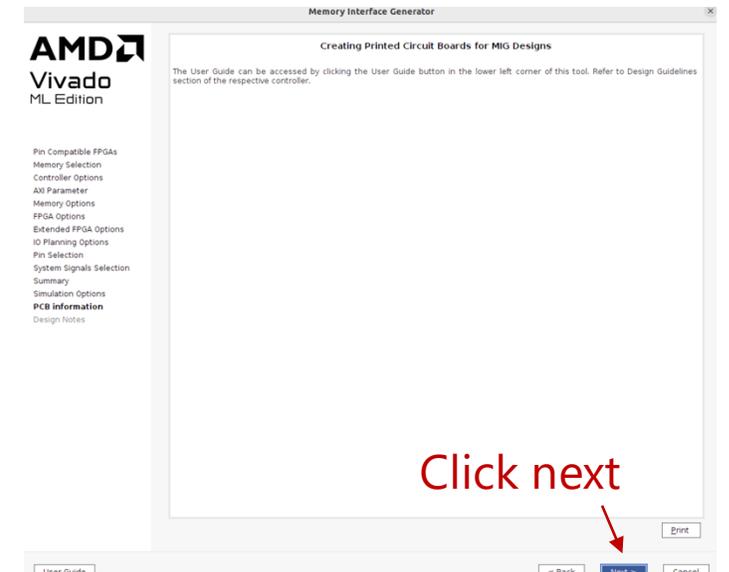
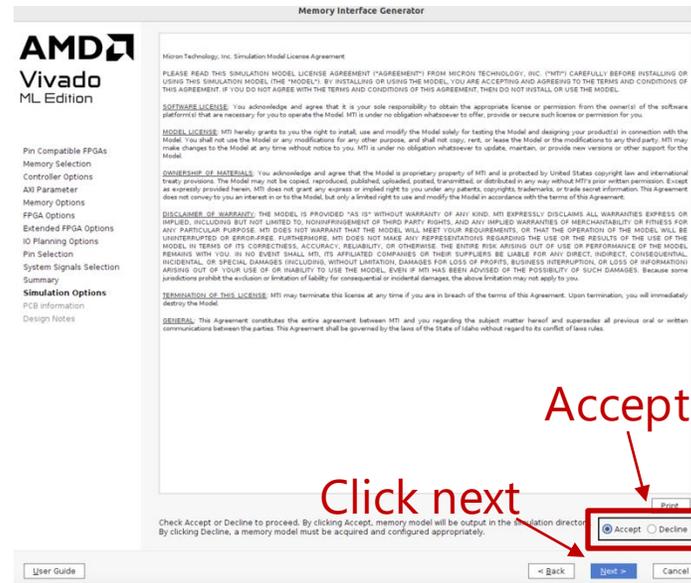
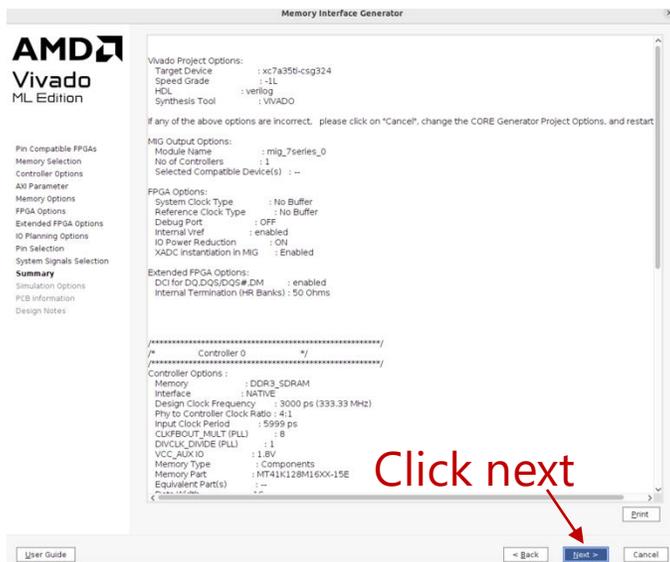
System Clock and Reference Clock pin selections will not be visible if the 'No Buffer' option was selected in the FPGA Options page.

## System Signals

These signals may be connected internally to other logic or brought out to a pin.

- **sys\_rst**: This input signal is used to reset the interface.
- **init\_calib\_complete**: This signal indicates that the interface has completed calibration and memory initialization and is ready for commands. LOC constraint will be generated in XDC for Example design only based on "Pin Number" selection below.
- **error**: This output signal indicates that the traffic generator in the Example Design has detected a data mismatch. This signal does not exist in the User Design.

Signal Name	Bank Number	Pin Number
sys_rst	Select Bank	No connect
init_calib_complete	Select Bank	No connect
tg_compare_error	Select Bank	No connect





# MIG (Memory Interface Generator) の設定 (8/8)

AMD Vivado ML Edition

Pin Compatible FPGAs  
Memory Selection  
Controller Options  
AXI Parameter  
Memory Options  
FPGA Options  
Extended FPGA Options  
IO Planning Options  
Pin Selection  
System Signals Selection  
Summary  
Simulation Options  
PCB Information  
Design Notes

DDR3 SDRAM Design for Artix-7 FPGAs

**Design Notes**

1. This design is tested with Vivado 2018.3 version
2. This design is simulated with Questa SIM 10.6c version, VCS N-2017.12-SP2 version, and IES 15.20.053 version
3. Components, RDIMMs, UDIMMs and SODIMMs are supported
4. If fly by delays are simulated, they must be limited to 1.2ns
5. Consult the Version Info for known limitations

**Key Enhancements for MIG 2.4 - 2015.4 release**

1. Updated Maximum supported design frequencies as per the 7 Series DC and AC Switching Characteristics data sheets

**Key Enhancements for MIG 2.3 - 2014.4 release**

1. Updated Maximum supported design frequencies as per the 7 Series DC and AC Switching Characteristics data sheets
2. DDR3 OCLK delay calibration enhancements

**Key Enhancements for MIG 2.2 - 2014.3 release**

1. Updated Maximum supported design frequencies as per the 7 Series DC and AC Switching Characteristics data sheets
2. Updated Maximum supported design frequencies according to (Xilinx Answer 59167)

**Key Enhancements for MIG 2.1 - 2014.2 release**

1. DDR3 Clocking Scheme changes
2. DDR3 read path calibration algorithm changes

**Key Enhancements for MIG 2.0 - 2014.1 release**

1. Extended IES and VCS support to Multi-Controller and Multi-Interface designs

User Guide < Back Generate Cancel



Generate Output Products

The following output products will be generated.

**Preview**

- mig\_7series\_0.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Structural Simulation
  - Implementation

**Synthesis Options**

Global

Out of context per IP

**Run Settings**

On local host: Number of jobs: 2

On remote hosts: Configure Hosts

Launch run on Cluster: Isf

Generate scripts only

Do not launch

? Apply Generate Skip



# DRAM からのリードとライトのレイテンシを求める

- DRAM の読み出しおよび書き込みにかかる最大のサイクル数を測定する。
- main6.v の m\_main モジュールを修正して、DRAM の読み出しと書き込みにかかる最大のサイクル数を求める。
  - DRAM の読み出しおよび書き込みにかかるサイクル数を測定する。
  - 最大のサイクル数を記録する。
  - VIO で確認する。

hw_vio_1				
Name	Value	Activity	Direction	VIO
> r_sum[31:0]	[U] 33554432		Input	hw_vio_1
> r_app_addr[27:0]	[H] 000_0000		Input	hw_vio_1
> r_ld_max_latency[7:0]	???		Input	hw_vio_1
> r_st_max_latency[7:0]	???		Input	hw_vio_1

## main6.v

```
//code

reg [9:0] r_ld_latency = 0; //register that stores the latency for a load operation
reg [9:0] r_ld_max_latency = 0; //maximum latency encountered during load operations
reg [9:0] r_st_latency = 0; //register that stores the latency for a store operation
reg [9:0] r_st_max_latency = 0; //maximum latency encountered during store operations

always @(posedge w_ui_clk) if (init_calib_complete) begin
    //Complete here
end

vio_0 vio0 (w_ui_clk, r_app_addr, r_sum, r_st_max_latency, r_ld_max_latency);

//rest of code
```

(Source code available in /home/u\_nesrine/ca2024/src)



Check Point 7



# References

- ❑ <https://www.acri.c.titech.ac.jp/wordpress/archives/6767>
- ❑ [https://github.com/kisek/fpga\\_arty\\_a7\\_dram](https://github.com/kisek/fpga_arty_a7_dram)