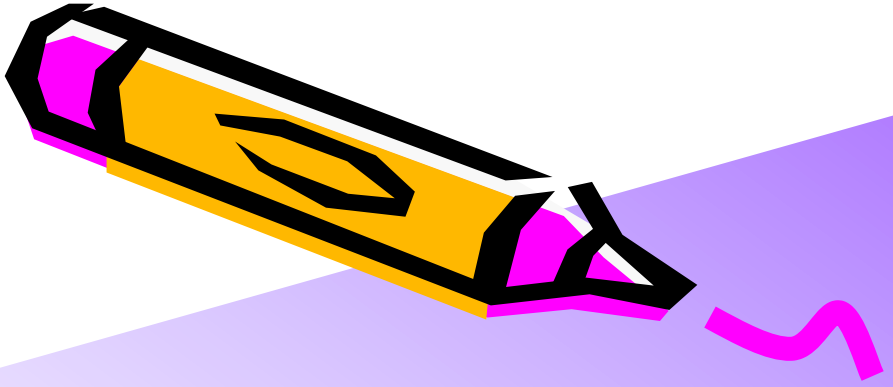
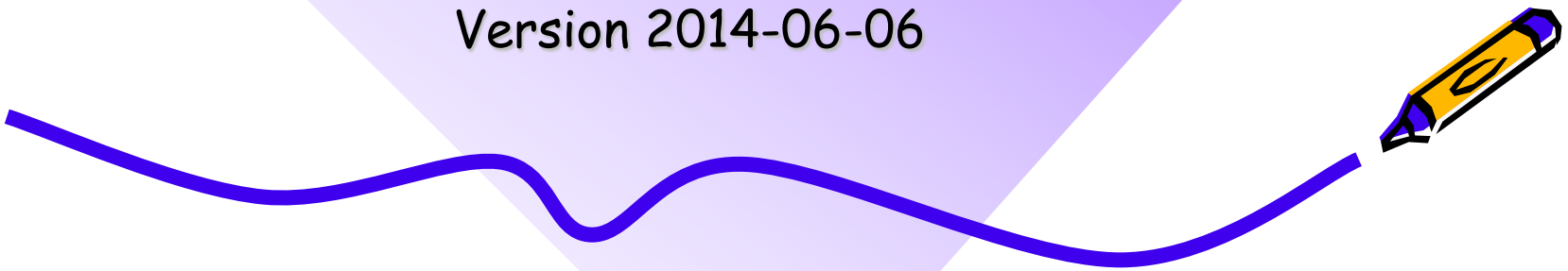


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

General Guide and Contest Rule

コンテスト実行委員会コアチーム

Version 2014-06-06



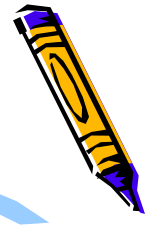
このドキュメント

- このドキュメントは，第2回ARC/CPSY/RECONF高性能コンピュータシステム設計コンテストの概要とルールを説明するものです．

- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

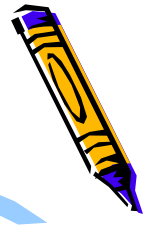
ドキュメントガイド



- 本ファイルには以下のドキュメントが含まれます
  - P30 General Guide and Contest Rule
  - P32 Reference Design Test Manual (ATLYS board)
  - P40 Design Guide for Altera DE-2 board
  - P42 Reference Design Test Manual (DE2 board)
  - P33 Architecture of Reference Design Processor
  - P34 DRAM Memory Interface
  - P35 MIPS Cross Compiler Setup Manual
  - P36 SDK Setup Manual
  - P37 Application Specification of Processor Design Category
  - P51 Application Specification of Computer System Design Category
  - 【後ほど作成予定】 Design Data Submission

- これらのドキュメントは、第1回コンテストから大部分を流用しています
  - 参考）第１回IPSJ-ARC高性能プロセッサ設計コンテストのWEBサイト
    - http://www.arch.cs.titech.ac.jp/contest/

コンテストの趣旨



- 今後よりいっそう高度化・多様化が進む，コンピュータシステムの基盤技術を支える研究者・技術者の育成と交流・ 研究開発の成果検証を目的に，コンピュータシステムの設計を競うコンテストを開催します．コンテスト参加者には，複数のアプリケーション課題に対して処理 性能を高める事を目標として，FPGAボード上で動作するコンピュータシステムを予め設計してもらいます．

- 当日はコンテスト参加者の設計したコンピュータ システムを持ち寄り，実際に会場で動作させることで性能を競います．プロセッサ技術・アーキテクチャ技術・リコンフィギャラブル技術や関連技術について， 最先端の研究開発の成果をアピールする機会となることを期待します．

コンテストの重要日程

項目	日程
参加登録の開始	2014年 6月6日（金）
参加登録の締切	7月25日（金）
予選デザインと予稿の原稿（1～4ページ）の提出	8月4日（月） 昼12時
予選結果の公表（決勝進出チームの決定）	8月11日（月）
FIT2014デザインコンテスト予稿集の原稿 （1～4ページ）の提出	8月22日（木）
FIT2014のイベント企画にて デザインのポスター発表と決勝戦	9月5日（金）

コンテストルール　1/5

1. 実施形態はチーム制です．1～4人で1チームを構成して参加してください．
2. 競技部門には2種類があります．選択して参加してください．
   - プロセッサ設計部門
     - プロセッサアーキテクチャを如何に高性能にするかを競う部門。基本的なアプリケーションで性能を競う。
     - 第１回プロセッサ設計コンテストとほぼ同じ内容ですが、以下の点が異なります
       - ホストPCとのインターフェイス（シリアル通信　→　UDP/IP・シリアル）
       - XilinxとALTERAのFPGAボードのいずれでも参加可能
   - コンピュータシステム設計部門
     - 応用に近いアプリケーションで、複数種類の処理の組合せでシステムトータルの性能を競う。

コンテストルール　2/5

3. プロセッサ設計部門の競技内容
   - FPGAボードは，Digilent社のATLYSボード(Xilinx) もしくは、ALTERA DE2-115ボードを用います．
   - アプリケーションプログラムは次の4つです．
     - 310_sort　　：整数のソーティング
     - 320_mm　　：行列積，要素は整数
     - 330_stencil　　：ステンシル計算，要素は整数
     - 340_spath　　：最短経路問題
   - コンテストの参加者は，次のデザインを提出してください．
     - 1種類のFPGAの回路情報（ファイル名は System.bit としてください）
     - 4種類の実行バイナリコード（256KBのバイナリファイルを4種類）
   - FPGAに実装されたプロセッサの処理時間を競います．
     - 実行委員が提供するデータ（256KB）を用いて，アプリケーションプログラムの処理時間を測定し，スコアを計算します．
   - ホストとのデータ転送には、イーサネット(100Mbps)・シリアル（1Mbps）変換の小型ボードを用います（後述）．

コンテストルール　3/5

4. コンピュータシステム設計部門の競技内容
   - FPGAボードは，Digilent社のATLYSボード(Xilinx)もしくは、ALTERA DE2-115ボードを用います．**その他の参加可能なボードは【検討中】です．**
   - 課題となるアプリケーションプログラムは以下のものです．
     - 400_oflow　　：オプティカルフロー処理
   - コンテストの参加者は，次のデザインを提出してください．
     - 1種類のFPGAの回路情報（ファイル名は System.bit としてください）
     - 1種類のデータ（512KBのバイナリファイル）
       （形式は自由・命令コードであっても良い）
   - FPGAに実装されたプロセッサの処理時間を競います．
     - 実行委員が提供する画像データ（JPEG様圧縮形式・別途説明、64KBx8フレーム）を用いて，オプティカルフロー処理プログラムの処理時間を測定し，スコアを計算します．

コンテストルール　4/5

5. 予選におけるスコアの計算方法と予選通過の条件
   - プロセッサ設計部門
     - それぞれのアプリケーションについて，実行時間の速いチームから順に，次の点数を与えます．1位 10点，2位 7点，3位 5点，4位 4点
     - 4種類のアプリケーションで獲得した点数の合計がスコアとなります．
     - 4種類のすべてのアプリケーションが規定時間（2分とします）内に正しい結果を出力し，スコアが上位のチームが決勝に進出します．
   - コンピュータシステム設計部門
     - 課題アプリケーションの処理時間に応じて順位・スコアをきめ、スコアが上位のチームが決勝に進出します．
   - ただし，参加チーム数等の状況により，点数を与える順位を調整することがあります．
   - 予稿の品質があまりに低い場合は予選を通過しないことがあります．
6. 決勝におけるスコアの計算方法

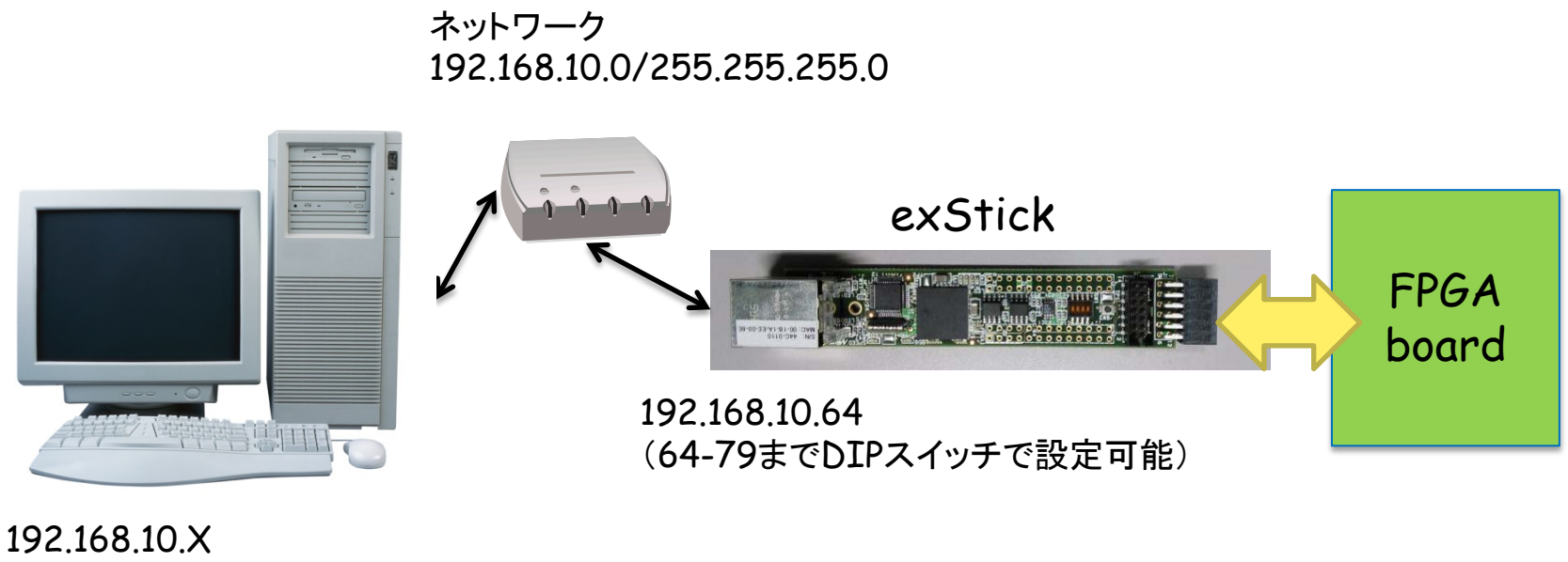
   【後程公開】

コンテストルール　5/5

7. 実行時間の計測方法
   - ホストからのUDP/IP(Ethernet)通信にて512KBのデータ（256KBのプログラムと256KBのアプリケーションデータ）の送信開始時刻T1から，計算結果を受け取り，最後にENDという文字列を受け取るまでの時刻T2を実行時間(execution time) として計測します．すなわち，execution time = T2 – T1 です．
   - UDP/IPの通信は100Mbpsで行う事とします．
8. 実行結果の検証
   - 設計した回路は，実行委員会が提供するツールキットと全く同じ結果を出力（通信にてホスト計算機に送信）する必要があります．また，計算結果を出力して，最後には END という文字列を出力してください．
     すなわち，実行を開始してから，ホスト計算機が受信する END という文字までの全ての文字列がツールキットと等しくなるように回路を設計してください．

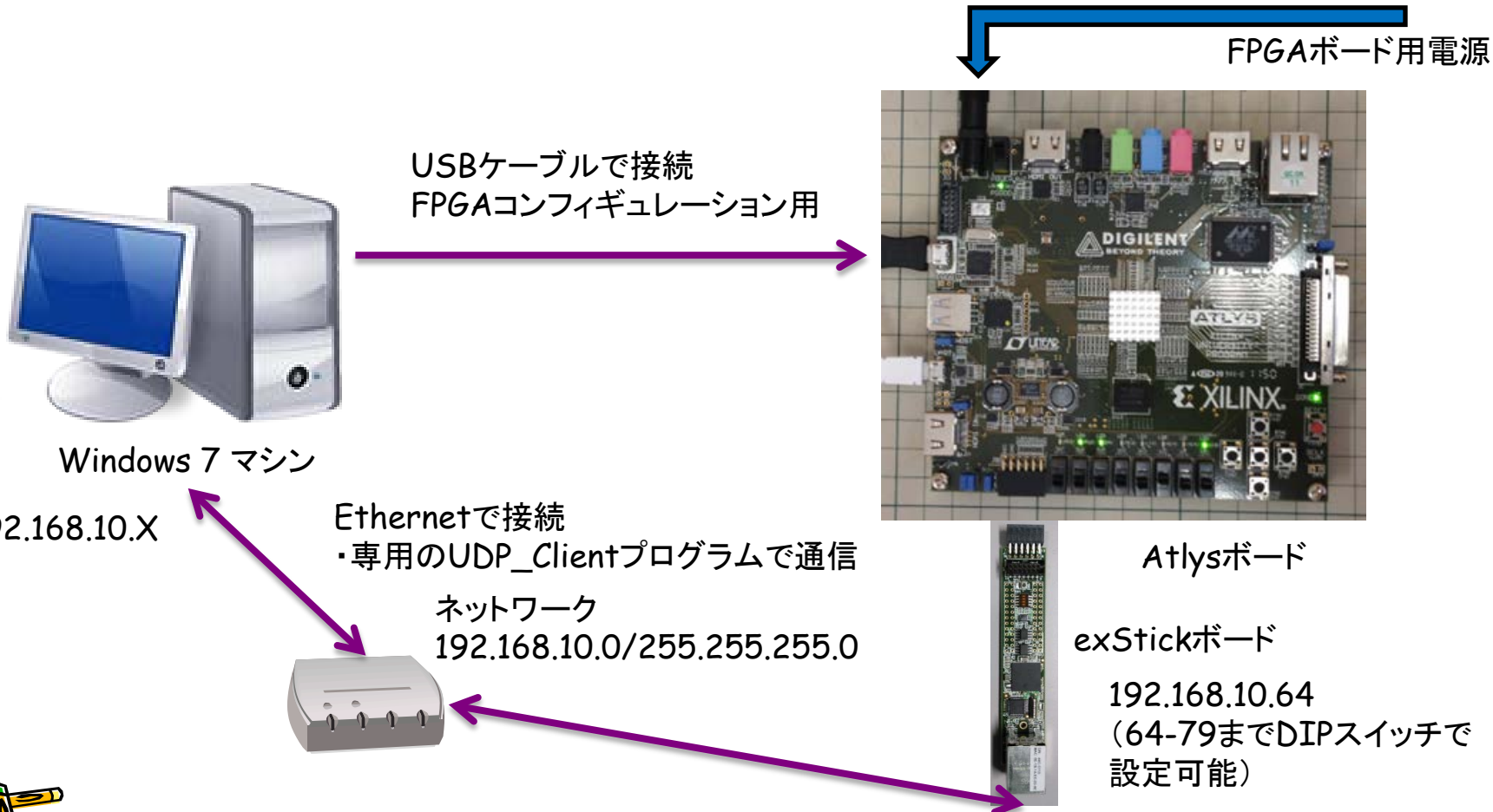
ハードウェアセットアップ

- PCとFPGAボードを、小型ボード(exStick)を介して接続します
  - イーサネット(100Mbps)・シリアル(1Mbps)変換



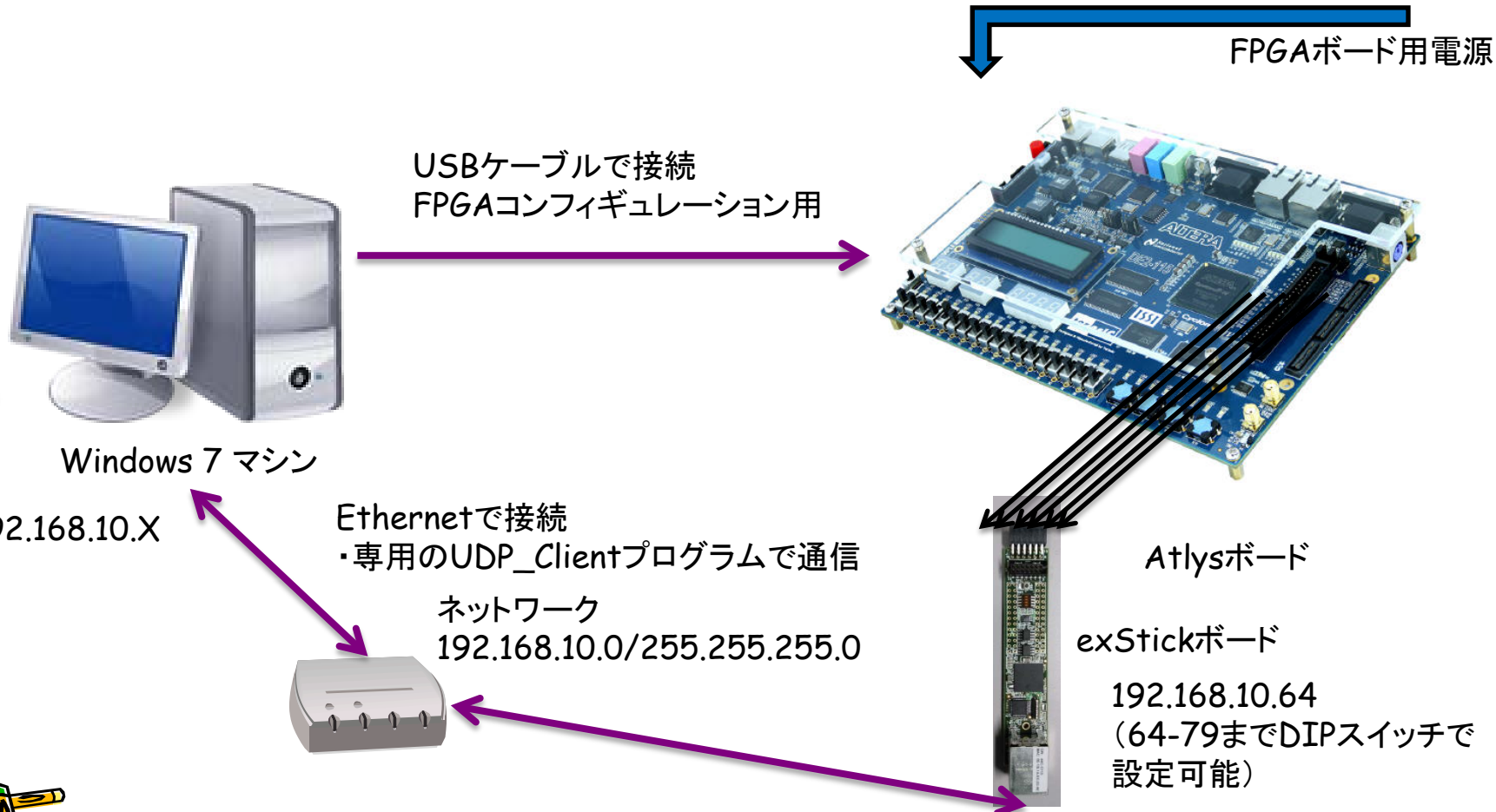
参考）ATLYSボードの開発環境セットアップ

- Windows 7 マシン と Atlysボード を USBケーブル で接続
- FPGAボード用電源 を使ってAtlysボードに電源供給

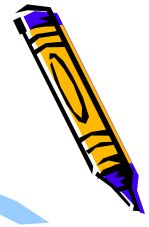


参考）DE2-115ボードの開発環境セットアップ

- Windows 7 マシン と DE2-115ボード を USBケーブル で接続
- FPGAボード用電源 を使ってDE2-115ボードに電源供給



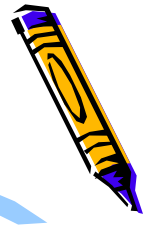
設計を始めましょう



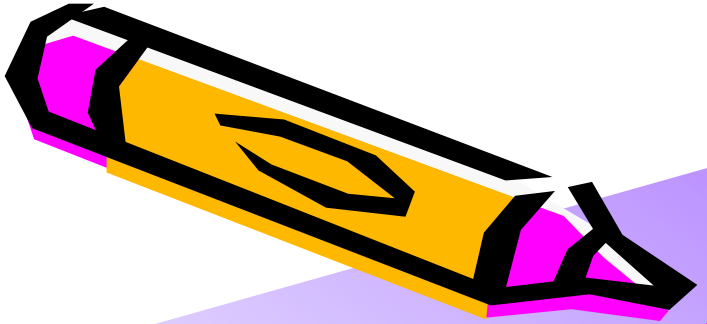
- Atlysボード，DE2-115ボード，USBケーブルなどの必要なハードウェアが揃っていない場合
  - コンテストホームページを参照の上，必要なハードウェアを揃えてください．
  - Atlysボードは，貸し出し出来る可能性があります．希望者はお問い合わせください． contest_support@virgo.is.utsunomiya-u.ac.jp

- Atlysボード，DE2-115ボード，USBケーブルなどの必要なハードウェアが揃っている場合
  - ドキュメント **“P32 Reference Design Test Manual”** を参考に，リファレンスデザインの動作テストをおこなってください．

配付ライセンスについて



- FPGAデザインサンプルはGPLです.
- ソフトウェア開発環境(SDK), ドキュメント類は修正BSDライセンスです.
- 詳細は各パッケージ・ディレクトリに含まれるライセンスファイルを参照してください.

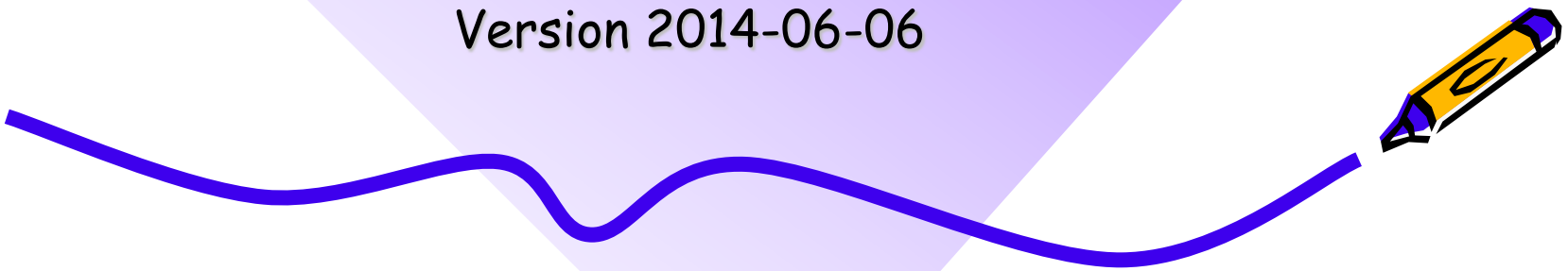


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

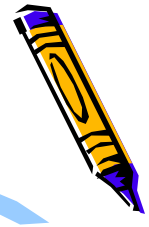
Reference Design Test Manual

コンテスト実行委員会コアチーム

Version 2014-06-06



このドキュメント

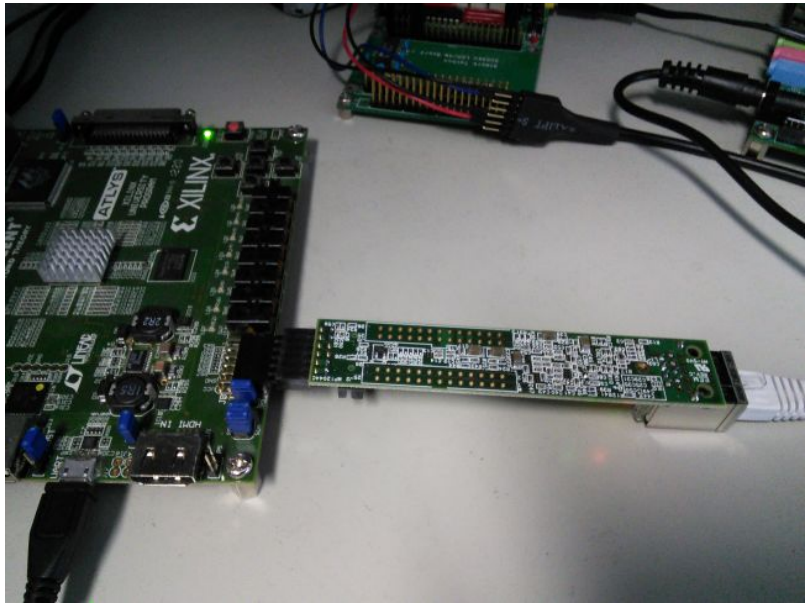


- このドキュメントは，ツールキットに含まれるリファレンスデザインの概要とテスト方法を説明するものです．

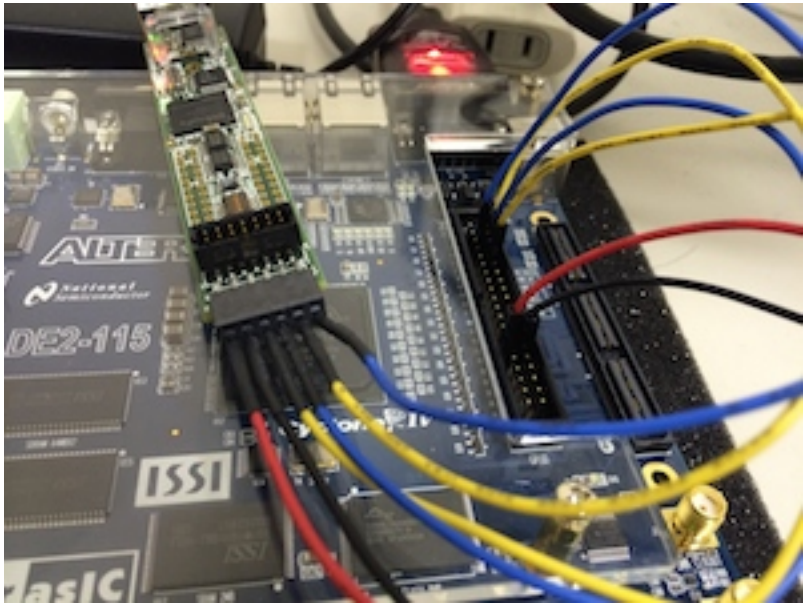
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

exStickBridgeの接続例



Xilinx Atlysボード



ALTERA DE2ボード

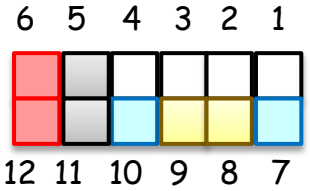
裏返して、両方が長いピンヘッダで差すとちょうどPMODの同じピン同士が接続されます。

接続ケーブルを用いて1本ずつ接続します

exStickBridgeの接続方法

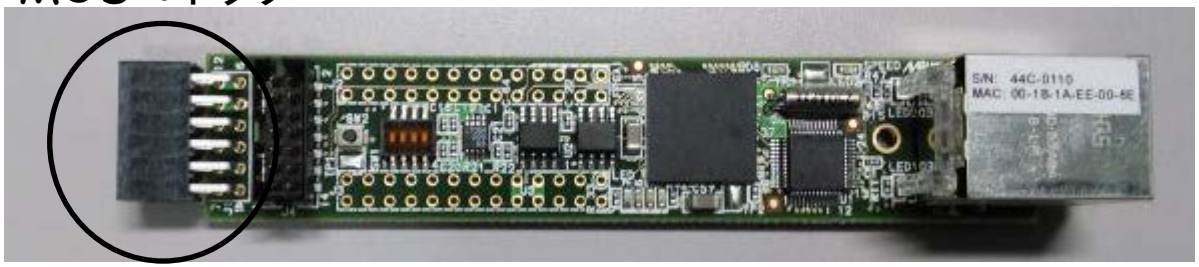
- 入出力ピン
  - exStickのPMODコネクタ
    - P7: UART-TX (out)
    - P8: UART-RX (in)
    - P9: RST_X (in)
    - P10: INIT_FPGA_X (out)

P10はFPGAボードをリセットする信号です。
リファレンスデザインは、exStickを接続して使用する必要があります。



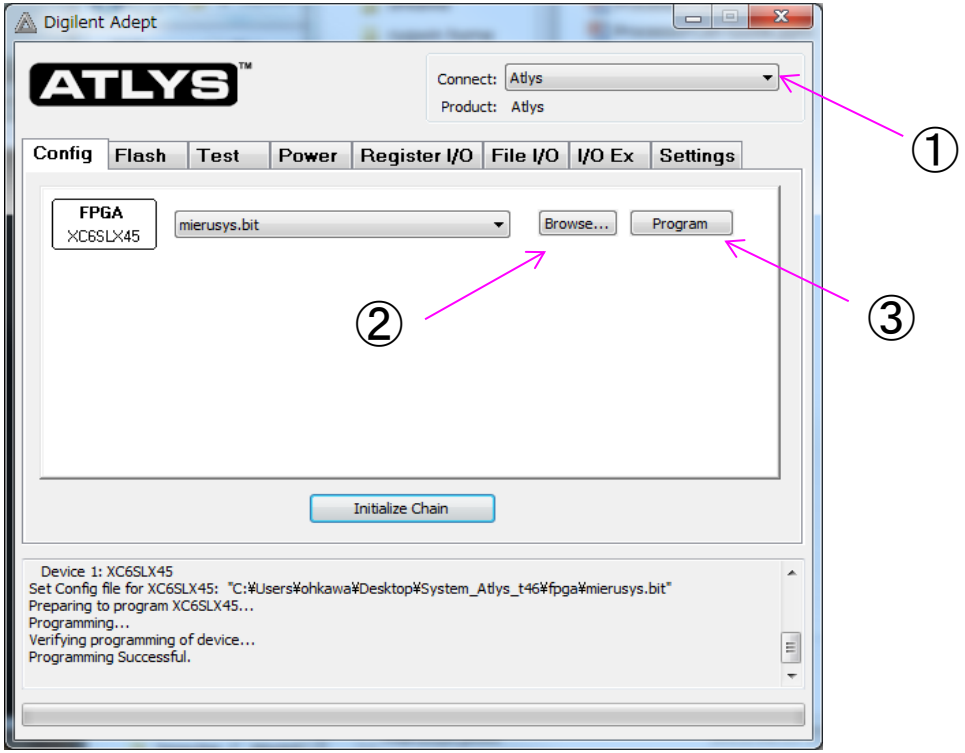
P5,P11: GND
P6,P12: VDD(3.3V)

PMODコネクタ



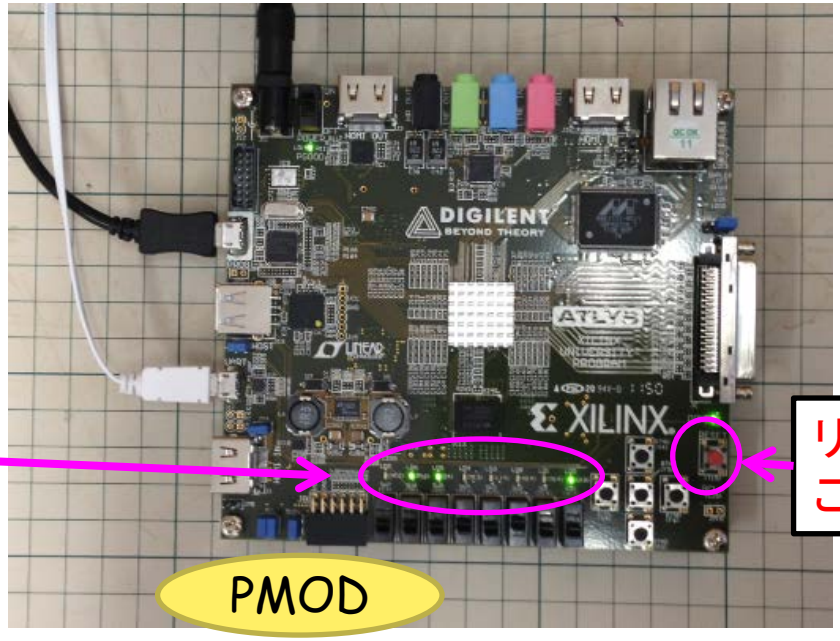
Atlysボード
FPGAのコンフィギュレーション

- コンテストホームページから，プロセッサを含むリファレンスデザインの回路データ System_Atlys_t63 をダウンロードしてください．
- System_Atlys_t63 をAdeptで書き込みます
  - ①Atlysボードが認識されていることを確認し、
    ②Browseで先程のbitファイルを選択して、
    ③Programで書き込みます



Atlysボード
サンプルアプリケーションの実行(1)

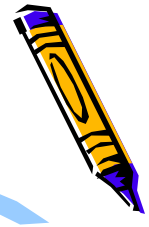
- Adeptで回路データ(bit)を書き込むと、LD0 が点灯，LD7 が点滅します
  - LEDの意味は本資料末尾の「参考：LEDの説明」をご覧ください．
- これで，PMODのUARTポートが，プログラムデータを受信するための待ち受け状態になります．
  - リセットボタンを押すと，FPGAを初期状態に戻すことが出来ます．



LEDはこちら
LD0：点灯
LD7：点滅
で正常動作です

リセットボタンは
こちら

exStickBridgeを用いた リファレンスデザインの動作検証 （1）



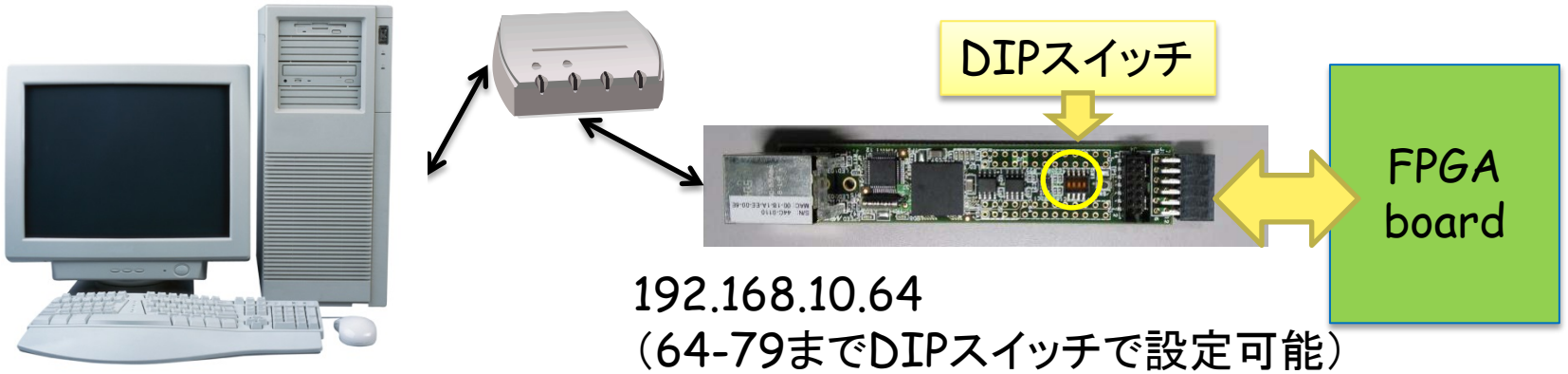
- 以下のファイルを用いて、exStickBridgeの動作検証を行います。
  - exStickBridge_v05.bit.zip
  - System_Atlys_t63.bit.zip
  - UDP_Client_v01.jar
  - DesignCon_SDK.1.0.2.tgz

- 上記ファイルに対応するプロジェクトファイルは以下の通りです
  - exStickBridge_v05.zip (Xilinx ISE14.7プロジェクト)
  - System_Atlys_t63.zip (Xilinx ISE14.7プロジェクト)
  - UDP_Client_v01.zip (Eclipseプロジェクト)

exStickBridgeを用いた リファレンスデザインの動作検証 （2）

- 動作検証のためのネットワーク環境
  - Pingコマンドで応答を確認可能
  - 動作確認済みのスイッチについては別表を参照

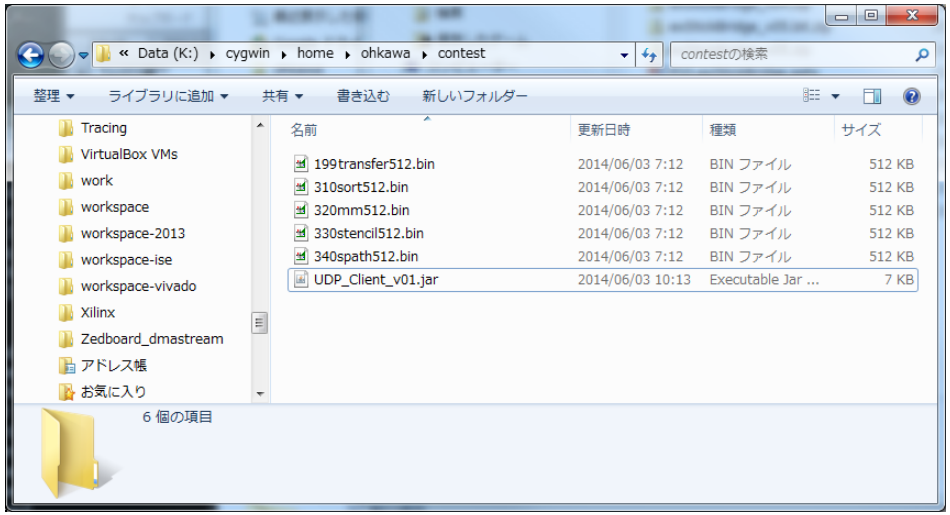
ネットワーク
192.168.10.0/255.255.255.0



exStickBridgeを用いた リファレンスデザインの動作検証 （3）

- DesignCon_SDK1.0.2.zipを展開します
- binディレクトリにある以下のファイルを、UDP_Client_v01.jarと同じディレクトリに置きます
  - 310sort512.bin, 320mm512.bin, 330stencil512.bin, 340spath512.bin
- 以下のコマンドで、4つのアプリを順次実行します。

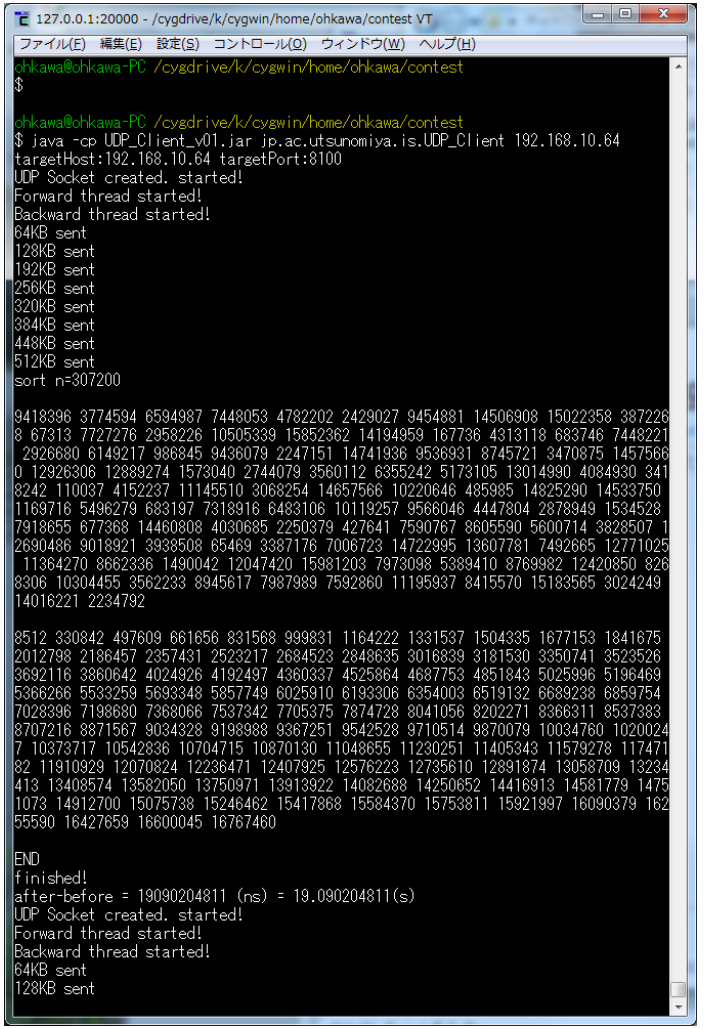
```
% java -cp UDP_Client_v01.jar jp.ac.utsunomiya.is.UDP_Client 192.168.10.64
```



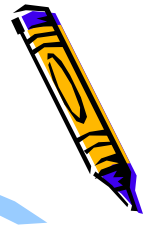
exStickBridgeを用いた リファレンスデザインの動作検証 （4）

- 実行
  - データ転送は8秒程度
  - 4つのアプリが順次実行される
  - アプリ開始時にFPGAボードが毎回リセットされる（リセットのために16バイトのUDPパケットを送信）
- 以下のログファイルが出力
  - ToUDP: 送信したデータ
  - FromUDP: 受信したデータ
- UDP通信のエラーが無いか確認するには、199transfer512.binを使用可能
  - FPGA上のプログラムが、データ領域のデータ全てを、PCに送ります
  - つまり199transfer512.binの後半256KB(199transfer.dat)とFromUDP.binが一致するはず

```
% java -cp UDP_Client_v01.jar
jp.ac.utsunomiya.is.UDP_Client 192.168.10.64
199transfer512.bin
```



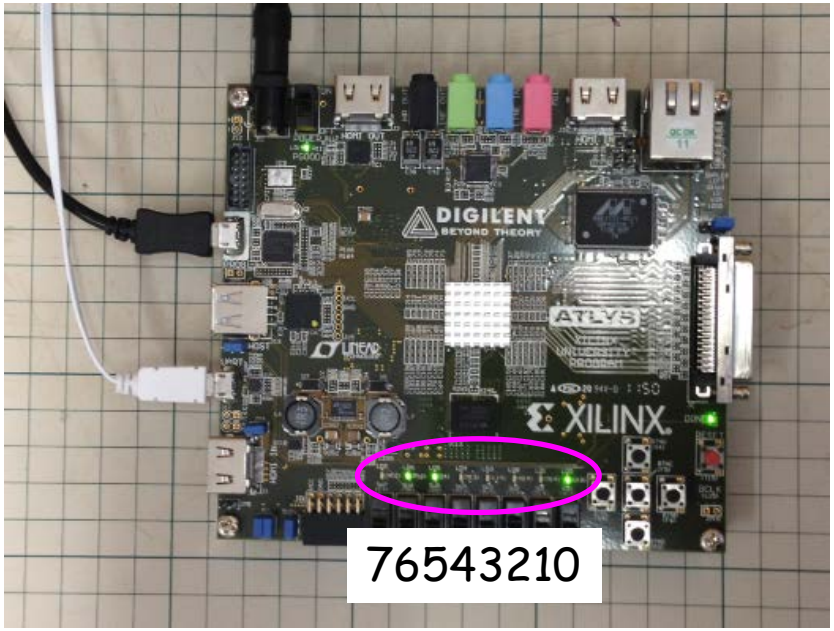
動作確認済みスイッチ一覧



- BUFFALO BSL-WS-G2116M
  - http://buffalo.jp/product/wired-lan/switch/bsl-ws-g21m/
- （追加しましょう）

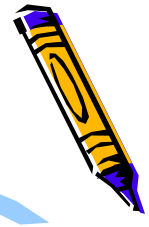
参考:LEDの説明

- リファレンスデザインにおける下図の赤枠で囲んだ各LEDは以下の状態を示します.



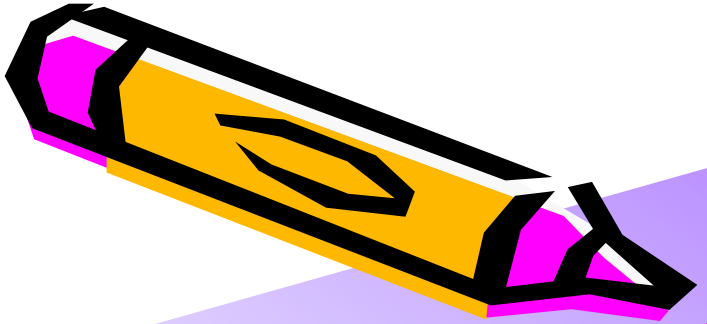
0: メモリイメージの転送完了
1: DRAMのキャリブレーションが完了
2: シリアル通信中(受信)
3: シリアル通信中(送信)
4: DRAMがbusy状態
5: プロセッサの命令デコードエラー
6: プロセッサのメモリアライメントエラー
7: heartbeat(一定間隔で点滅)

参考:LEDの説明　詳細



- ツールキットに含まれる Verilog HDL 記述を示します.
- rtl/MieruSys.vに、LEDへの出力の内容が記述されています
  - LD7 (ULED[7]):　カウンタの25ビット目なので、約33Mクロックで点滅
  - LD0(ULED[0]):　リセット時転送、イメージの転送が完了すると消えます

```
always @(posedge CLK) ULED[7] <= cnt_t[25];
always @(posedge CLK) ULED[6] <= CORE_STAT[1];      // Processor Memory Alignment Error
always @(posedge CLK) ULED[5] <= CORE_STAT[0];      // Processor Decode Error
always @(posedge CLK) ULED[4] <= mem_busy;          // DRAM is working
always @(posedge CLK) ULED[3] <= ~TXD;              // Uart TXD
always @(posedge CLK) ULED[2] <= ~RXD;              // Uart RXD
always @(posedge CLK) ULED[1] <= ~calib_done;       // DRAM calibration done
always @(posedge CLK) ULED[0] <= ~INIT_DONE;        // MEMORY IMAGE transfer is done
```

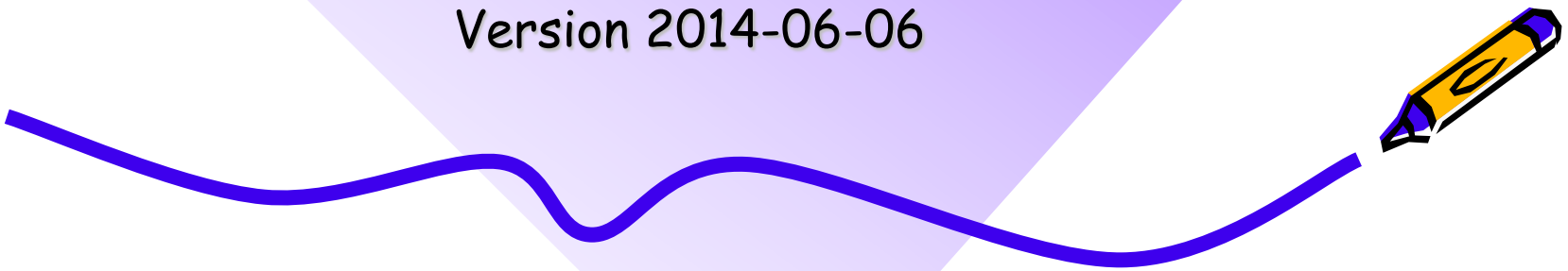


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

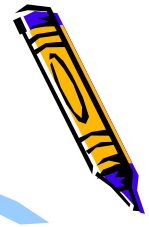
Design Guide for Altera DE-2 board

コンテスト実行委員会コアチーム

Version 2014-06-06



このドキュメント

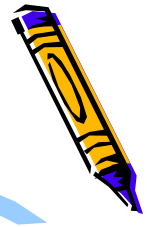


- このドキュメントは，第2回ARC/CPSY/RECONF高性能コンピュータシステム設計コンテストにおいてAltera DE2-115 開発ボードを使用する場合に必要となる情報を提供する事を目的とします．

- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

ドキュメントガイド



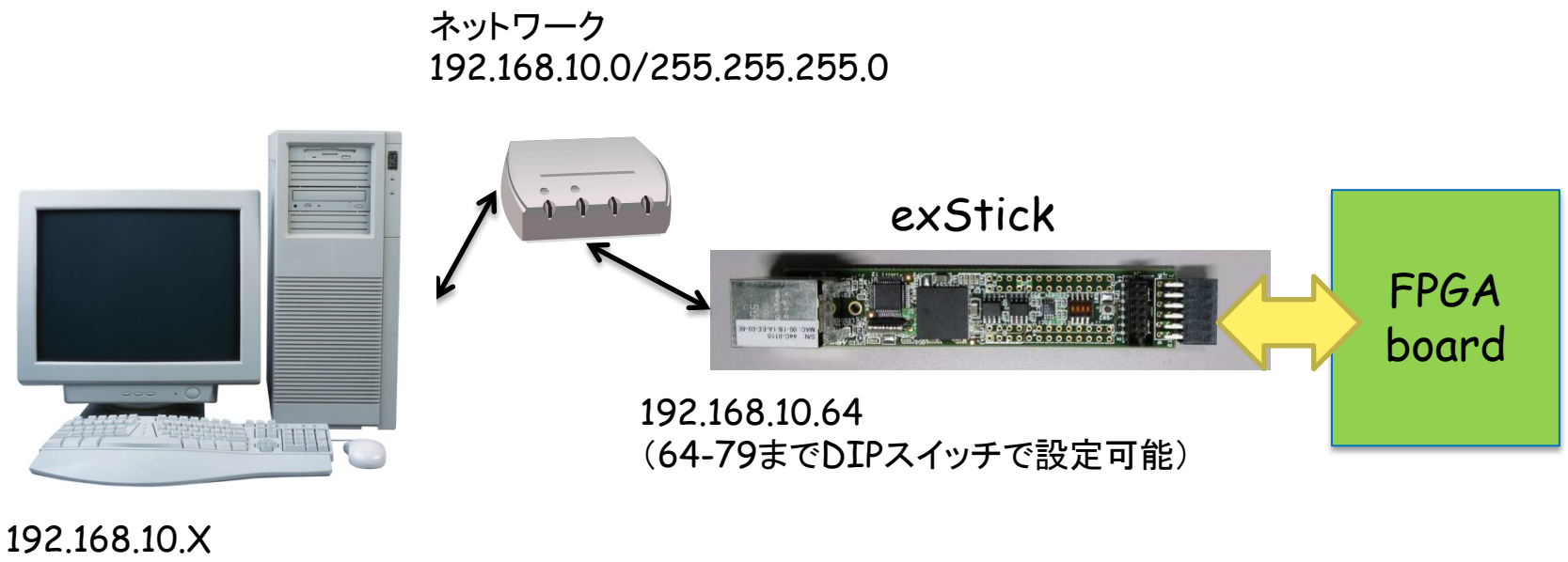
- 本ドキュメントを読む前に，本コンテストの「General Guide and Contest Rule」を熟読されている事を前提とします．

- DE2-115ボードの仕様について下記のドキュメントを参考にして下さい
  - DE2_115_User_Manual.pdf（DE2-115ボードに付属のCDにあります）
- Alteraのツールの基本的な使用方法等については，最低限下記のドキュメントの内容を理解されていると仮定してます．
  - Introduction to the Altera Qsys System Integration Tool
  - Making Qsys Components

  （AlteraのWebサイトの「トレーニング」->「ユニバーシティプログラム」をクリックして現れるサイトの左側にあるEducational Materials -> Computer Organization ->Tutorialsからダウンロードできます）

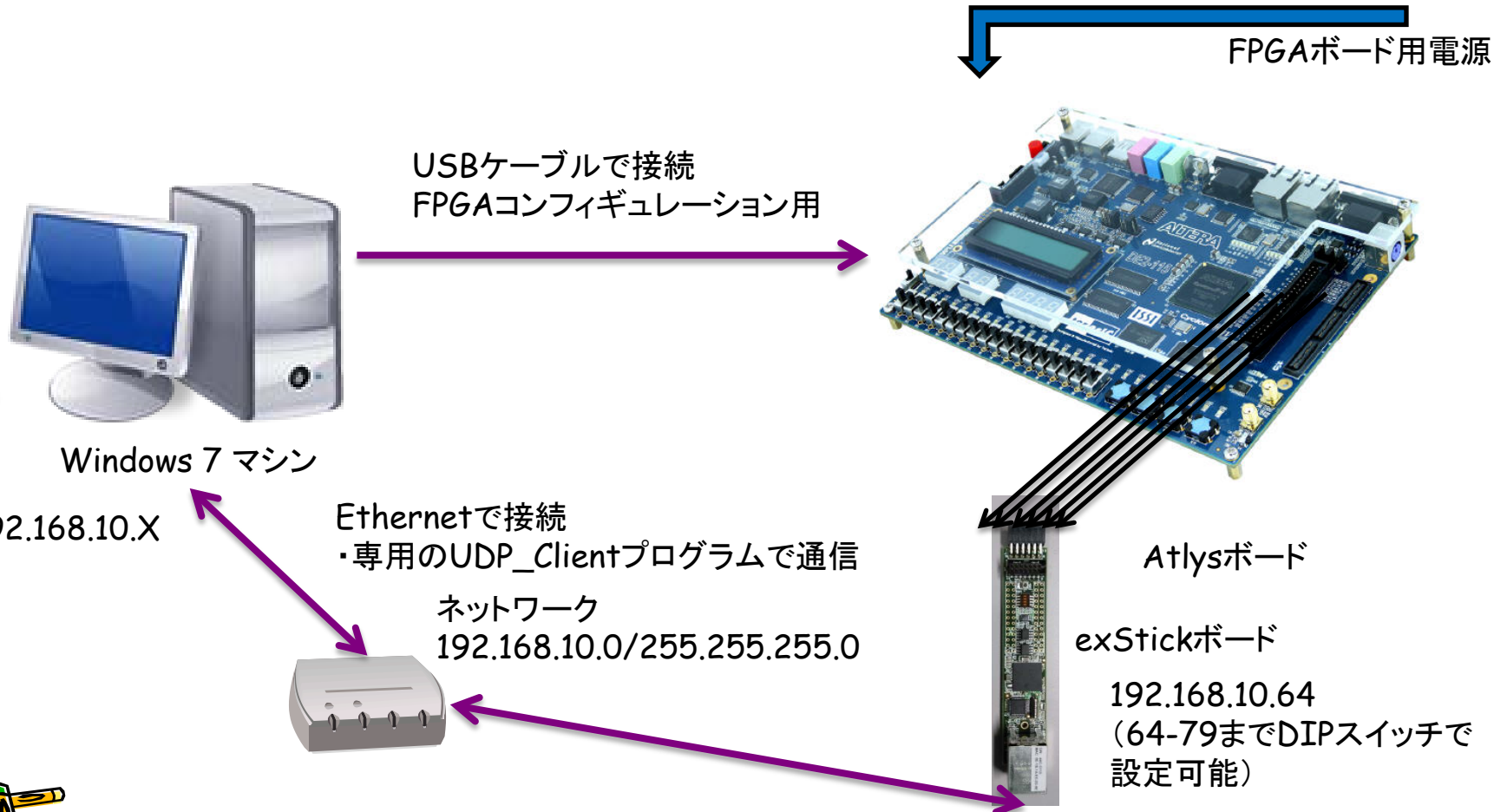
ハードウェアセットアップ

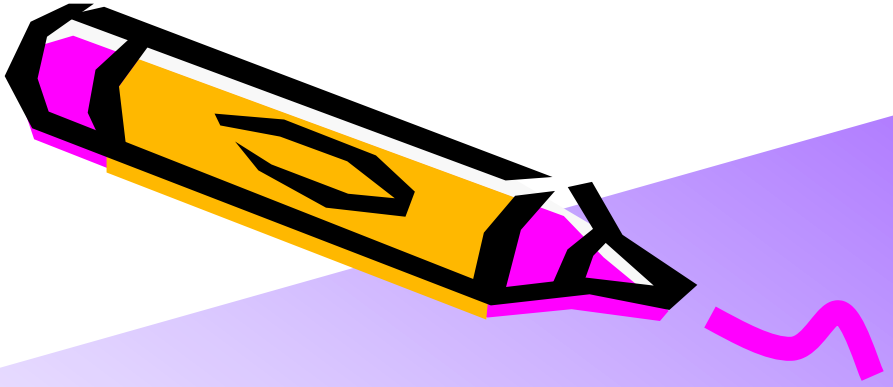
- PCとFPGAボードを、小型ボード(exStick)を介して接続します
  - イーサネット(100Mbps)・シリアル(1Mbps)変換



DE2-115ボードの開発環境セットアップ

- Windows 7 マシン と DE2-115ボード を USBケーブル で接続
- FPGAボード用電源 を使ってDE2-115ボードに電源供給



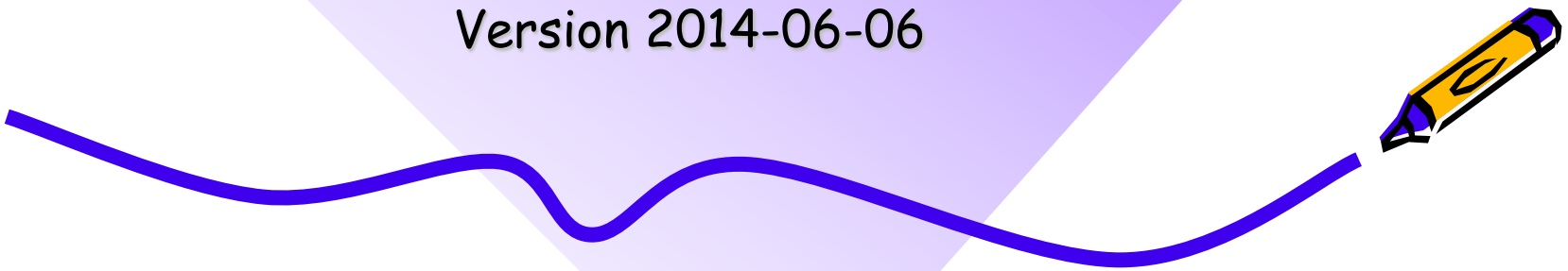


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

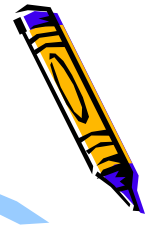
Reference Design Test Manual

コンテスト実行委員会コアチーム

Version 2014-06-06



このドキュメント

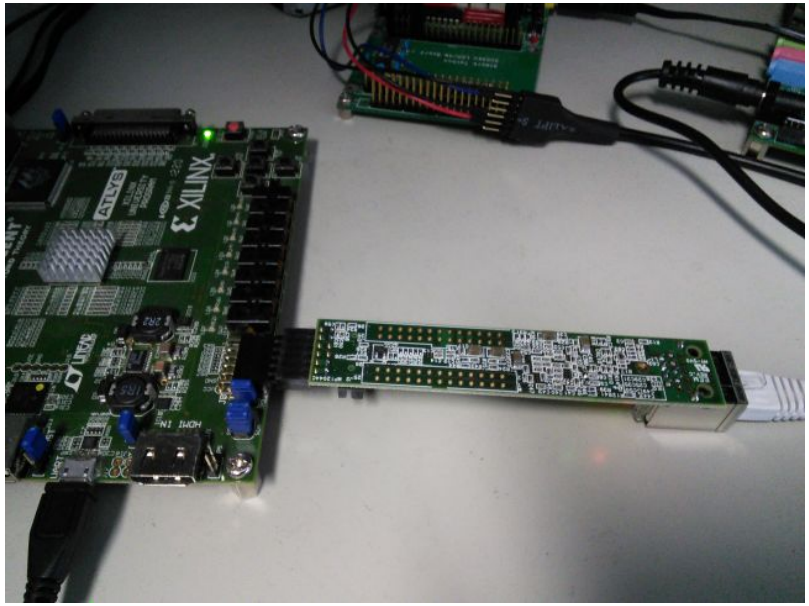


- このドキュメントは，Altera DE2-115ボードを用いた場合の，ツールキットに含まれるリファレンスデザインの概要とテスト方法を説明するものです．

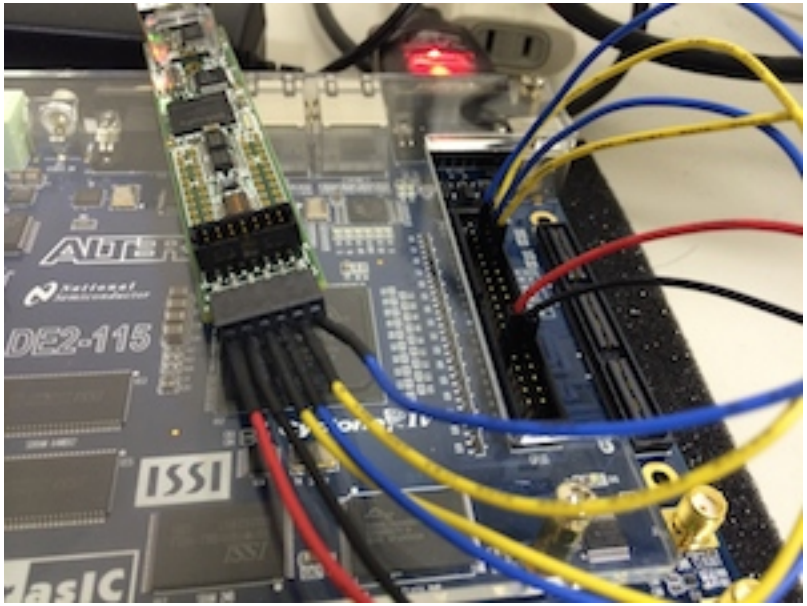
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

exStickBridgeの接続例



Xilinx Atlysボード



ALTERA DE2ボード

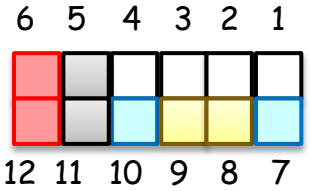
裏返して、両方が長いピンヘッダで差すとちょうどPMODの同じピン同士が接続されます。

接続ケーブルを用いて1本ずつ接続します

exStickBridgeの接続方法

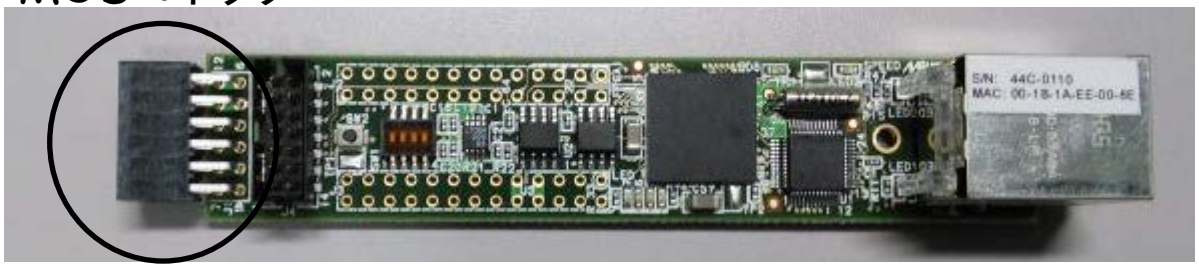
- 入出力ピン
  - exStickのPMODコネクタ
    - P7: UART-TX (out)
    - P8: UART-RX (in)
    - P9: RST_X (in)
    - P10: INIT_FPGA_X (out)

P10はFPGAボードをリセットする信号です。
リファレンスデザインは、exStickを接続して使用する必要があります。



P5,P11: GND
P6,P12: VDD(3.3V)

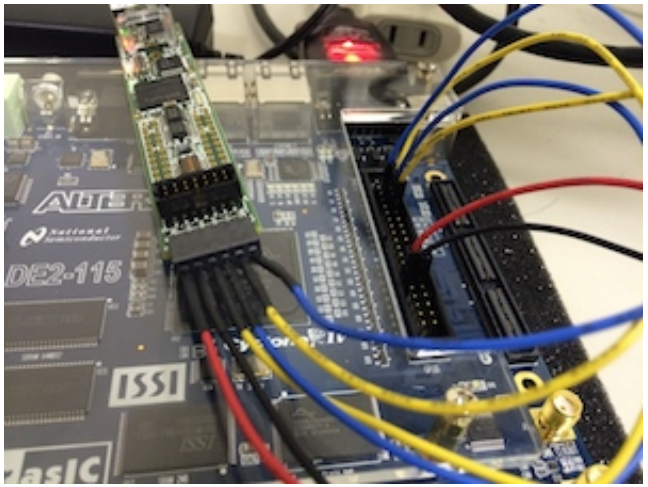
PMODコネクタ



exStickBridgeとDE2-115ボードの接続方法

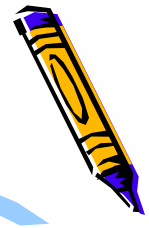
- ジャンパー線を用意してexStickBridgeとDE2-115ボードのGPIOと接続して下さい
- ピンの接続については本ドキュメントでは以下の表の用にします

exStickBridge	DE2-115
VDD(Pmod12番)	GPIOの3.3V
GND(Pmod11番)	GPIOのGND
INIT_FPGA_X(Pmod10番)	GPIO[9]
RST_X(Pmod9番)	GPIO[7]
UART-RX (Pmod8番)	GPIO[5]
UART-TX (Pmod7番)	GPIO[3]



ジャンパー線による接続例

exStickBridgeを用いた リファレンスデザインの動作検証に必要なファイル

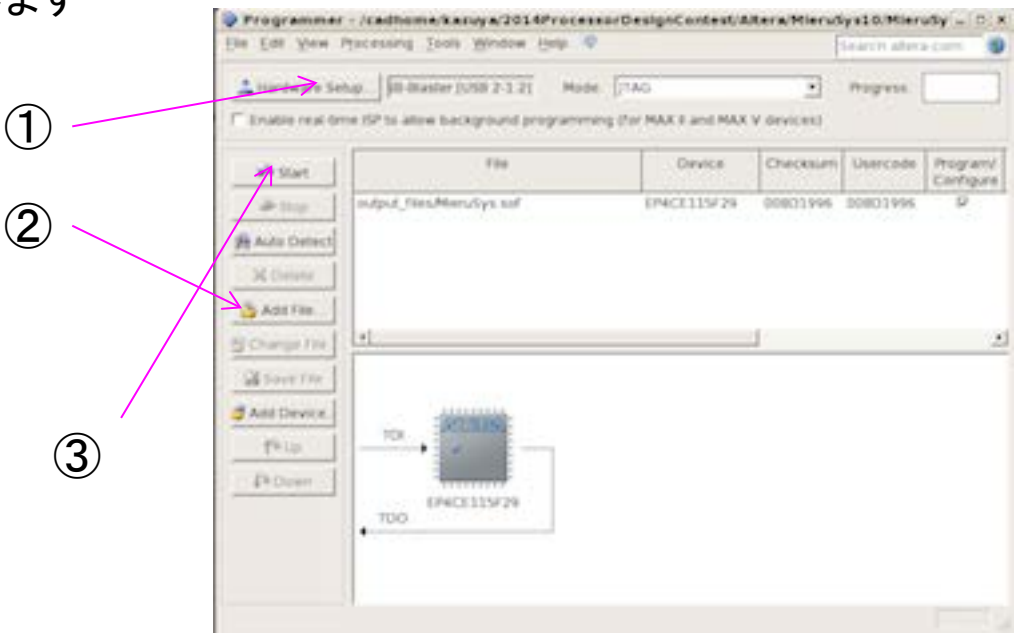


- 以下のファイルを用いて、exStickBridgeの動作検証を行います。
  - exStickBridge_v05.bit.zip
  - MieruSys10.sof.zip
  - UDP_Client_v01.jar
  - DesignCon_SDK.1.1.1.tgz

- 上記ファイルに対応するプロジェクトファイルは以下の通りです
  - exStickBridge_v05.zip (Xilinx ISE14.7プロジェクト)
  - MieruSys10.zip (Aletra Quartus 13.1プロジェクト)
  - UDP_Client_v01.zip (Eclipseプロジェクト)

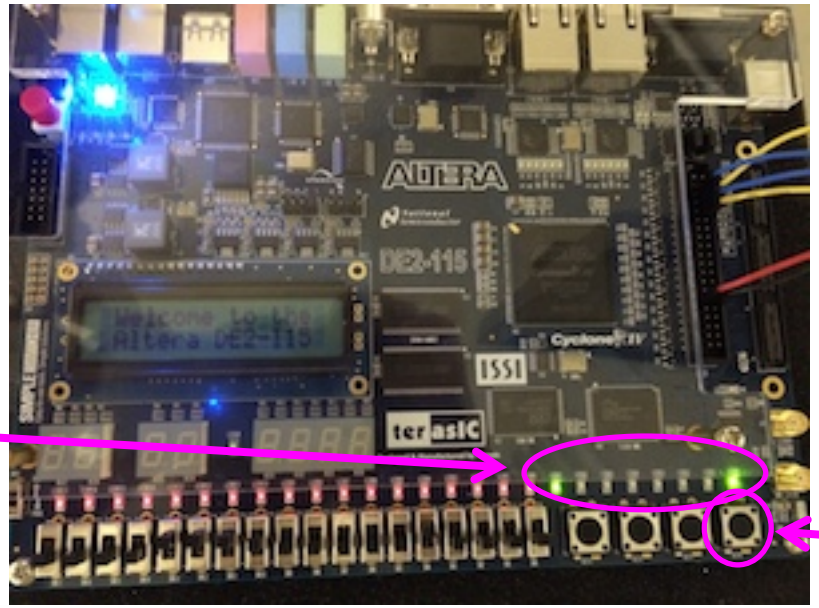
DE2-115ボード
FPGAのコンフィギュレーション

- コンテストホームページから，プロセッサを含むリファレンスデザインの回路データMieruSys10.sof.zipをダウンロードしてください．
（解凍するとMieruSys.sofというファイルになります）
- De2-115ボードの電源をいれます．
- MieruSys.sofをQuartusから起動できるProgrammerで書き込みます
  - ①Hardware SetupでUSB Blasterが選択されている事を確認
  - ②Add fileでMieruSys10.sofを選択して追加します
  - ③Startで書き込みます



DE2-115ボード
サンプルアプリケーションの実行(1)

- Programmerで回路データ(sof)を書き込むと、LEDG0 が点灯，LEDG7 が点滅します
  - LEDの意味は「参考:LEDの説明」をご覧ください.
- これで，PMODのUARTポートが，プログラムデータを受信するための待ち受け状態になります.
  - リセットボタンを押すと，FPGAを初期状態に戻すことが出来ます.

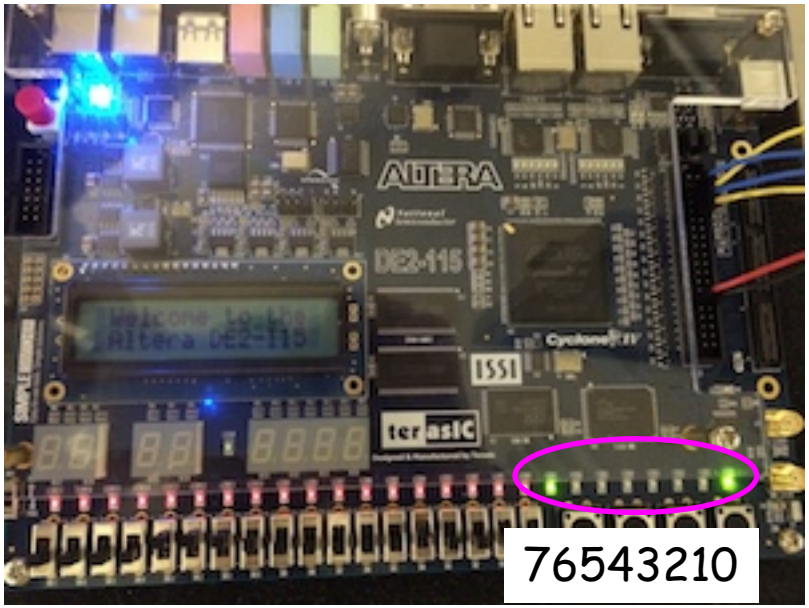


LEDはこちら
LD0:点灯
LD7:点滅
で正常動作です

リセットボタンはこちら

参考:LEDの説明

- リファレンスデザインにおける下図の赤枠で囲んだ各LEDは以下の状態を示します.

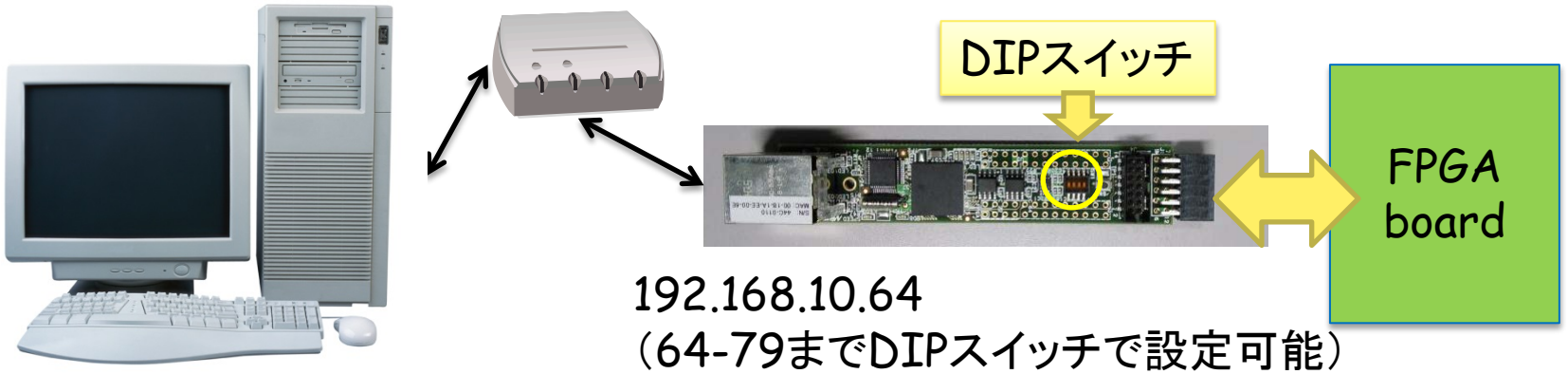


0: メモリイメージの転送完了
1: 常に0
2: シリアル通信中(受信)
3: シリアル通信中(送信)
4: DRAMがbusy状態
5: プロセッサの命令デコードエラー
6: プロセッサのメモリアライメントエラー
7: heartbeat(一定間隔で点滅)

exStickBridgeを用いた リファレンスデザインの動作検証 （1）

- 動作検証のためのネットワーク環境
  - Pingコマンドで応答を確認可能
  - 動作確認済みのスイッチについては別表を参照

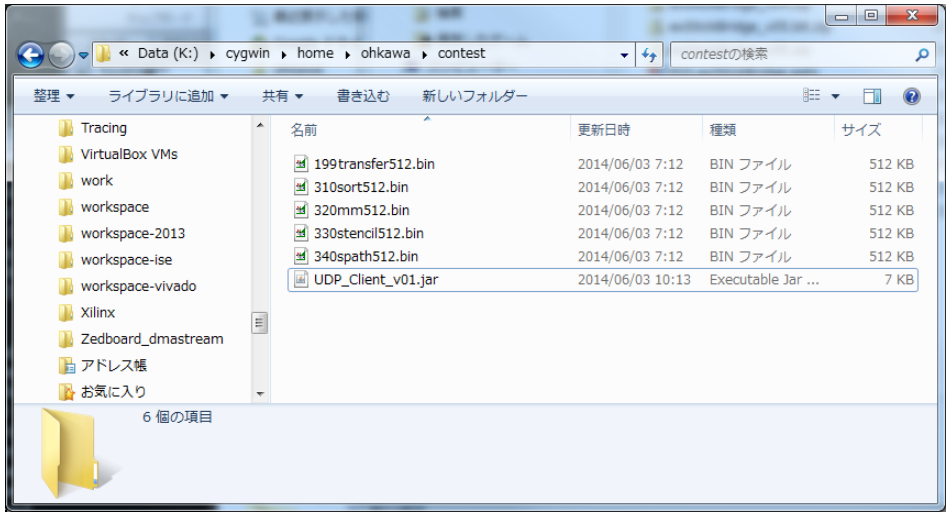
ネットワーク
192.168.10.0/255.255.255.0



exStickBridgeを用いた リファレンスデザインの動作検証 （2）

- DesignCon_SDK1.1.1.zipを展開します
- bin/alteraディレクトリにある以下のファイルを、UDP_Client_v01.jarと同じディレクトリに置きます
  - 310sort512.bin, 320mm512.bin, 330stencil512.bin, 340spath512.bin
- 以下のコマンドで、4つのアプリを順次実行します。

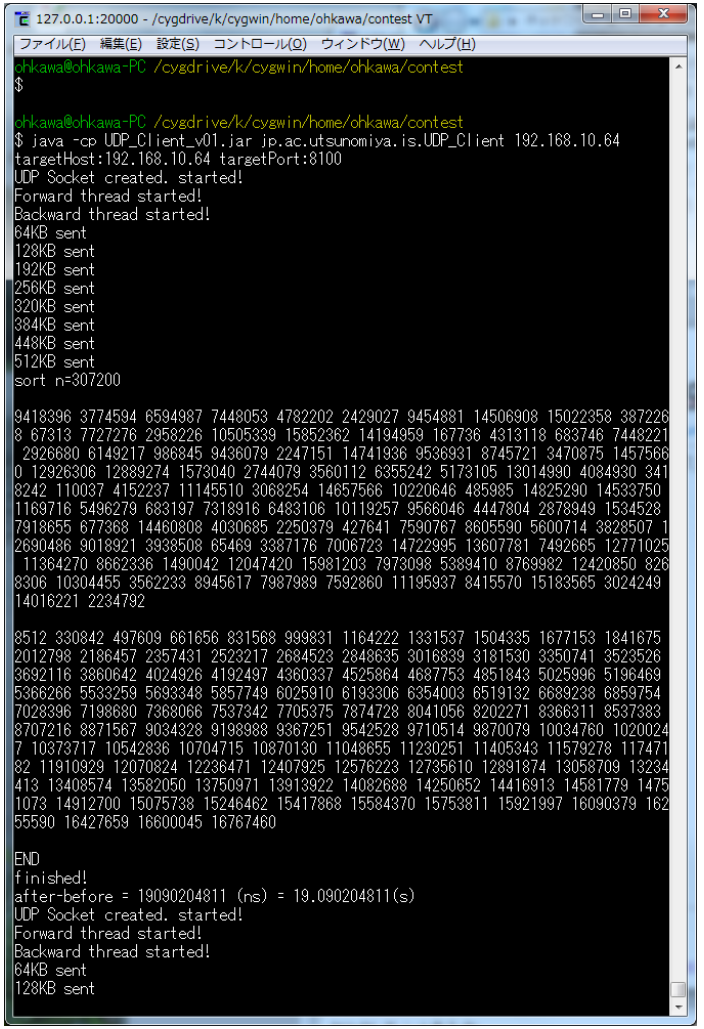
```
% java -cp UDP_Client_v01.jar jp.ac.utsunomiya.is.UDP_Client 192.168.10.64
```

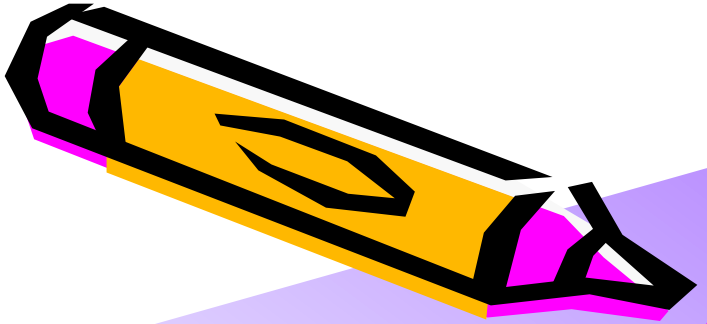


exStickBridgeを用いた リファレンスデザインの動作検証 （3）

- 実行
  - データ転送は8秒程度
  - 4つのアプリが順次実行される
  - アプリ開始時にFPGAボードが毎回リセットされる（リセットのために16バイトのUDPパケットを送信）
- 以下のログファイルが出力
  - ToUDP: 送信したデータ
  - FromUDP: 受信したデータ
- UDP通信のエラーが無いか確認するには、199transfer512.binを使用可能
  - FPGA上のプログラムが、データ領域のデータ全てを、PCに送ります
  - つまり199transfer512.binの後半256KB(199transfer.dat)とFromUDP.binが一致するはず

```
% java -cp UDP_Client_v01.jar
jp.ac.utsunomiya.is.UDP_Client 192.168.10.64
199transfer512.bin
```



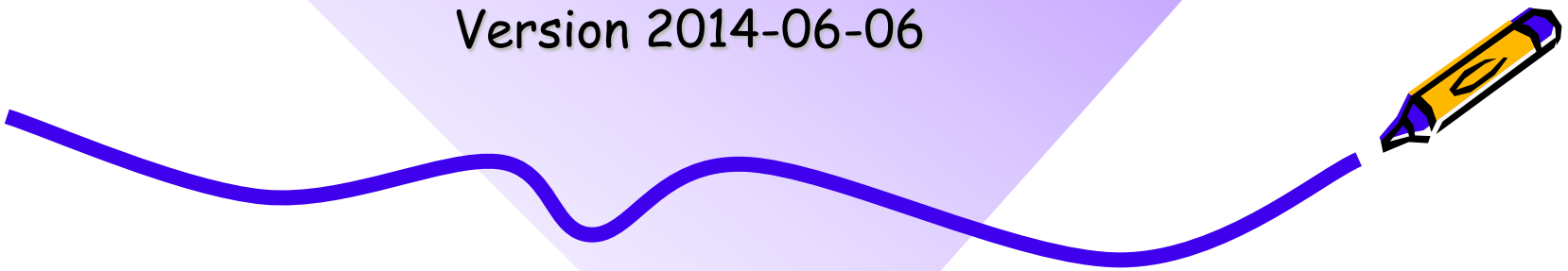


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

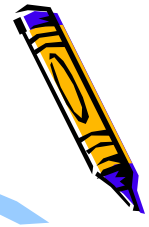
Architecture of Reference Design Processor

コンテスト実行委員会コアチーム

Version 2014-06-06



このドキュメント

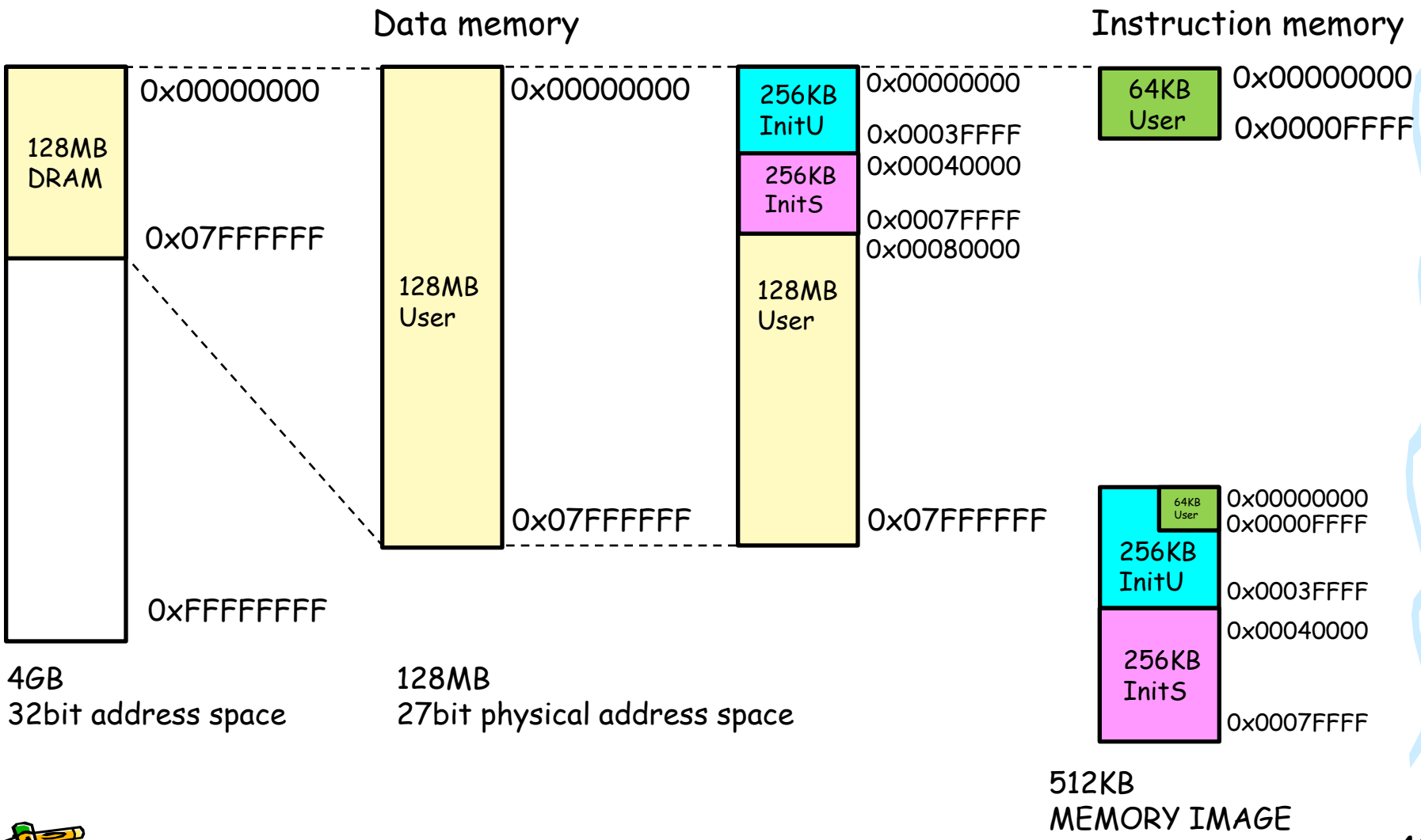


- このドキュメントでは，リファレンスデザインに含まれるプロセッサの構成（アーキテクチャ）について説明します．
- また，Xilinx ISEを用いて，リファレンスデザインの回路ファイル（bitファイル）を生成する方法を示します．

- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

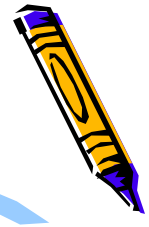
Memory Map of ToolKit Ver.1 Reference Design



Memory Map of ToolKit Ver.1 Reference Design

- Atlysボードには，128MBのDRAMが搭載されており，これを自由に用いることができます．
- 512KBのデータをシリアル通信でFPGAに送信します．
- 512KBのデータは256KBの InitU，256KBのInitS により構成されます．
  - InitUは参加者が提供するデータで，プロセッサが実行するプログラムなどを格納してください．
  - InitSは実行委員会が提供するデータで，アプリケーションのための入力パラメータや入力データが格納されます．
- リファレンスデザインでは，受信した512KBのデータをDRAMの0番アドレスから順に格納します．
  - すなわち，0x00000000 ～ 0x0007FFFF に，512KBのデータが格納されます．
- リファレンスデザインでは，プロセッサが実行する命令を格納するために，64KBの命令メモリを実装しています．
  - 512KBのデータの先頭の64KBのみが，この命令メモリに格納されます．

Memory Mapped I/O of the Reference Design



- FPGAからexStickBridgeへの出力は，シリアル通信を用います．
- リファレンスデザインのプロセッサは，アドレス 0 にストアすると，そのデータがシリアル通信で送信されます．
  - 例えば，Cのアプリケーションプログラムで次の記述をおこなうと A がターミナルに表示されます．

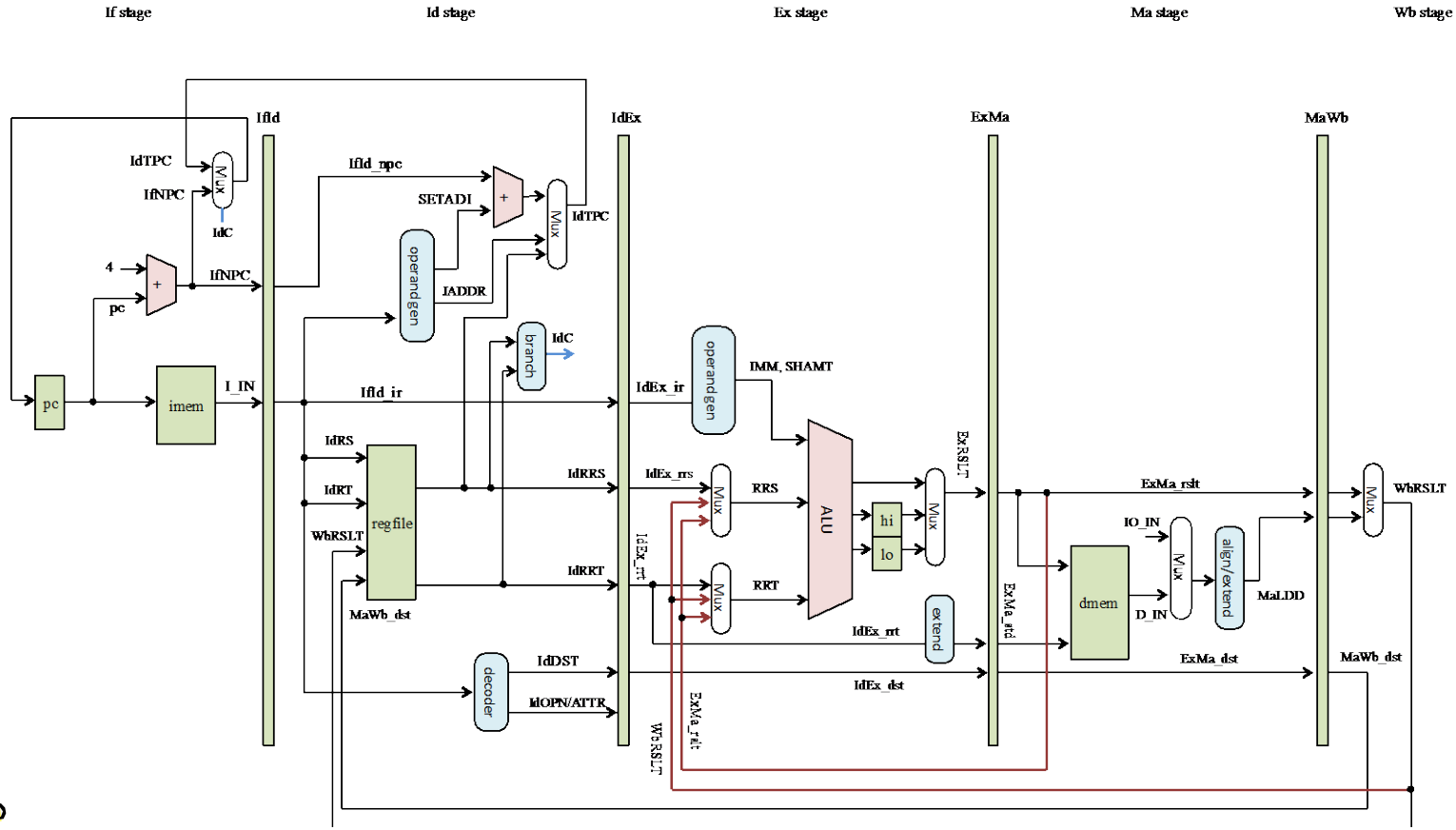
```
{
    volatile int *uart_txd = (int*)0;
    *uart_txd = 'A';
}
```

  - ただし，シリアル通信のモジュールは送信バッファを持たないため，複数の文字を送信する場合には，ソフトウェアでウェイトを入れてください．

```
{
    volatile int *uart_txd = (int*)0;
    *uart_txd = 'A';
    mylib_wait();  /* user defined function  */
    *uart_txd = 'B;
}
```

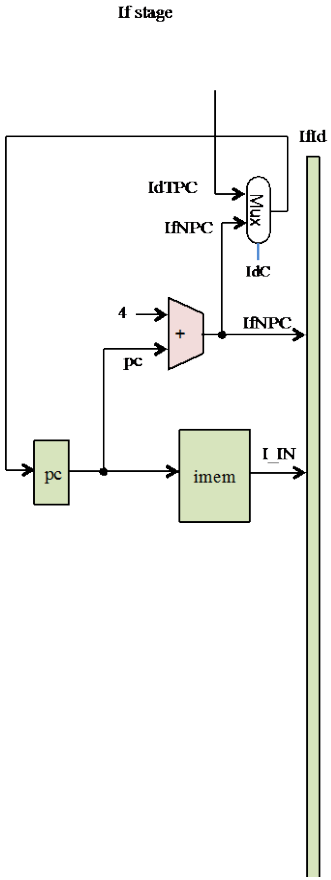
リファレンスデザインに含まれるプロセッサ

- リファレンスデザインには，5段パイプライン処理の典型的なMIPSプロセッサが採用されています．そのデータパスを下に示します．
- このプロセッサは，ロード・ストア命令としてワード単位の命令のみをサポートします．アプリケーションでは char, short, float, double といったデータ型は利用しないでください．



データパス - Ifステージ - (1/5)

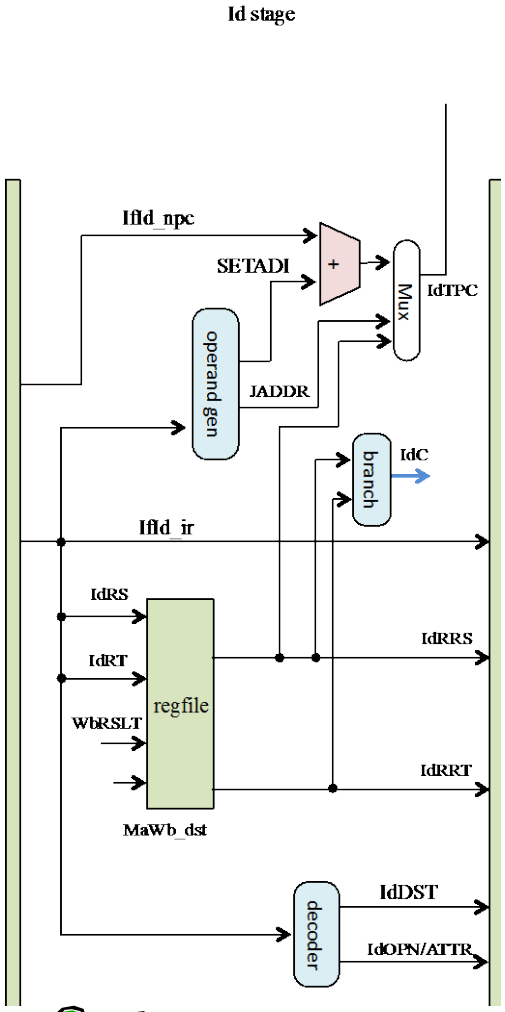
- 命令メモリから命令をフェッチし，プログラムカウンタ(PC)を設定します




```
129     /*****************************************************************************************************/
130     /* Stage 1 : IF, instruction fetch                                                                   */
131     /*****************************************************************************************************/
132     wire [‘ADDR] IfNPC = pc + 4;
133
134     always @( posedge CLK or negedge RST_X ) begin ///// update program counter
135         if(!RST_X)                      pc <= 0;
136         else if(!PSTALL && !bstall) pc <= (IdC) ? IdTPC : IfNPC; // If branch taken, uses taken PC
137     end
138
139     always @( posedge CLK or negedge RST_X ) begin  ///// update pipeline registers
140         if(!RST_X) {IfId_npc, IfId_ir} <= 0;
141         else if(!PSTALL && !bstall) begin
142             IfId_npc <= IfNPC;
143             IfId_ir  <= I_IN;  // if branch taken then NOP else instruction from imem.
144         end
145     end
```

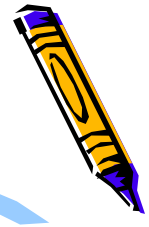
データパス - Idステージ - (2/5)

- フェッチした命令をデコードしながら，レジスタを呼び出します




```
147     /*****************************************************************************************************/
148     /* Stage 2 : ID, instruction decode & operand fetch                                                  */
149     /*****************************************************************************************************/
150     wire [5:0] IdOP    = IfId_ir[31:26]; // opcode field of instruction
151     wire [4:0] IdRS    = IfId_ir[25:21]; // rs field of instruction
152     wire [4:0] IdRT    = IfId_ir[20:16]; // rt field of instruction
153     wire [4:0] IdRD    = IfId_ir[15:11]; // rd field of instruction
154     wire [5:0] IdFCT   = IfId_ir[5:0];   // funct field of instruction
155     wire [31:0] IdRRS, IdRRT;            // register value of rs and rt
156
157     assign bstall = IdB &&
158                     ((IdRS!=0 && (IdRS==IdEx_dst || IdRS==ExMa_dst)) ||
159                      (IdRT!=0 && (IdRT==IdEx_dst || IdRT==ExMa_dst))); // branch stall
160
161     GPR gpr(.CLK(CLK), .REGNUM0(IdRS), .REGNUM1(IdRT), .DOUT0(IdRRS), .DOUT1(IdRRT), // register file
162             .REGNUM2(ExMa_dst), .DIN0(MaRSLT), .WE0(!PSTALL));
163
164     reg [6:0]   IdOPN;   // instruction operation number (unique operation ID)
165     reg [4:0]   IdDST;   // destination register
166     reg [‘ATTR] IdATTR;  // instruction attribute
167     always @(IfId_ir) begin
168         IdOPN   = ‘ERROR____;
169         IdDST  = 0;
170         IdATTR = 0;
171         case (IdOP) // OP
172             6’h00: case (IdFCT) // FUNCT
173                 6’h00: begin IdOPN= (IdRD) ? ‘SLL_____ : ‘NOP_____; IdDST=IdRD; end
174                 6’h02: begin IdOPN=‘SRL_____; IdDST=IdRD;                      end
175                 6’h03: begin IdOPN=‘SRA_____; IdDST=IdRD;                      end
176                 6’h04: begin IdOPN=‘SLLV____; IdDST=IdRD;                      end
177                 6’h06: begin IdOPN=‘SRLV____; IdDST=IdRD;                      end
178                 6’h07: begin IdOPN=‘SRAV____; IdDST=IdRD;                      end
179                 6’h08: begin IdOPN=‘JR______; IdDST=0;                         end
```

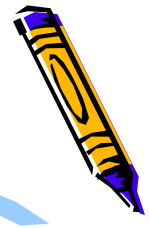
データパス - Idステージ - (2/5)



- フェッチした命令をデコードします．

```
180                 6'h09: begin IdOPN='JALR_____; IdDST=31;                          end
181                 6'h10: begin IdOPN='MFHI_____; IdDST=IdRD;                        end
182                 6'h12: begin IdOPN='MFLO_____; IdDST=IdRD;                        end
183                 6'h18: begin IdOPN='MULT_____; IdDST=0;                           end
184                 6'h19: begin IdOPN='MULTU____; IdDST=0;                           end
185                 6'h20: begin IdOPN='ADD______; IdDST=IdRD;                        end
186                 6'h21: begin IdOPN='ADDU_____; IdDST=IdRD;                        end
187                 6'h22: begin IdOPN='SUB______; IdDST=IdRD;                        end
188                 6'h23: begin IdOPN='SUBU_____; IdDST=IdRD;                        end
189                 6'h24: begin IdOPN='AND______; IdDST=IdRD;                        end
190                 6'h25: begin IdOPN='OR_______; IdDST=IdRD;                        end
191                 6'h26: begin IdOPN='XOR______; IdDST=IdRD;                        end
192                 6'h27: begin IdOPN='NOR______; IdDST=IdRD;                        end
193                 6'h2a: begin IdOPN='SLT______; IdDST=IdRD;                        end
194                 6'h2b: begin IdOPN='SLTU_____; IdDST=IdRD;                        end
195                 6'h1a: begin IdOPN='DIV______; IdDST=IdRD;                        end
196                 6'h1b: begin IdOPN='DIVU_____; IdDST=IdRD;                        end
197             endcase
198             6'h01: case (IdRT)
199                 5'h00: begin IdOPN='BLTZ_____; IdDST=0;                           end
200                 5'h01: begin IdOPN='BGEZ_____; IdDST=0;                           end
201             endcase
202             6'h02:     begin IdOPN='J________; IdDST=0;                           end
203             6'h03:     begin IdOPN='JAL______; IdDST=31;                          end
204             6'h04:     begin IdOPN='BEQ______; IdDST=0;                           end
205             6'h05:     begin IdOPN='BNE______; IdDST=0;                           end
206             6'h06:     begin IdOPN='BLEZ_____; IdDST=0;                           end
207             6'h07:     begin IdOPN='BGTZ_____; IdDST=0;                           end
208             6'h08:     begin IdOPN='ADDI_____; IdDST=IdRT;                        end
209             6'h09:     begin IdOPN='ADDIU____; IdDST=IdRT;                        end
210             6'h0a:     begin IdOPN='SLTI_____; IdDST=IdRT;                        end
211             6'h0b:     begin IdOPN='SLTIU____; IdDST=IdRT;                        end
212             6'h0c:     begin IdOPN='ANDI_____; IdDST=IdRT;                        end
213             6'h0d:     begin IdOPN='ORI______; IdDST=IdRT;                        end
214             6'h0e:     begin IdOPN='XORI_____; IdDST=IdRT;                        end
215             6'h0f:     begin IdOPN='LUI______; IdDST=IdRT;                        end
216             6'h20:     begin IdOPN='LB_______; IdDST=IdRT; IdATTR='LD_1B;         end
217             6'h21:     begin IdOPN='LH_______; IdDST=IdRT; IdATTR='LD_2B;         end
218             6'h23:     begin IdOPN='LW_______; IdDST=IdRT; IdATTR='LD_4B;         end
219             6'h24:     begin IdOPN='LBU______; IdDST=IdRT; IdATTR='LD_1B;         end
```

データパス - Idステージ - (2/5)

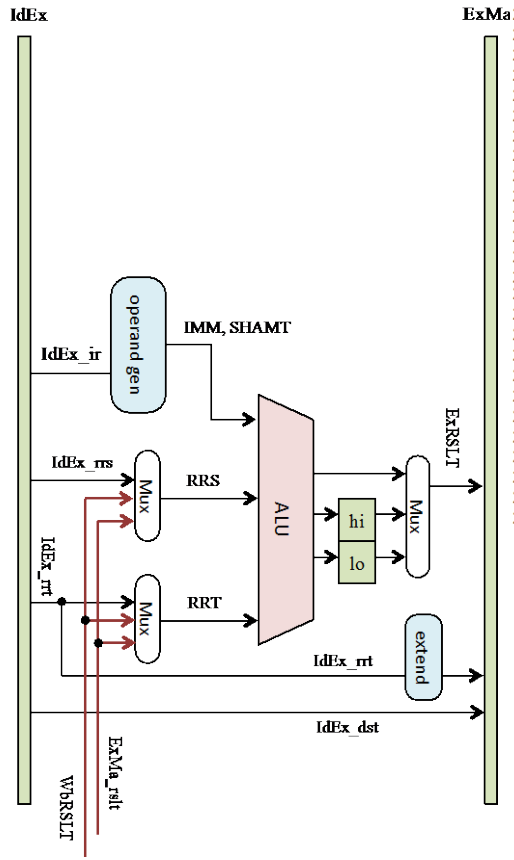


- フェッチした命令をデコードします.

```
220             6'h25:     begin IdOPN='LHU_____; IdDST=IdRT; IdATTR='LD_2B;         end
221             6'h28:     begin IdOPN='SB______; IdDST=0;    IdATTR='ST_1B;         end
222             6'h29:     begin IdOPN='SH______; IdDST=0;    IdATTR='ST_2B;         end
223             6'h2b:     begin IdOPN='SW______; IdDST=0;    IdATTR='ST_4B;         end
224         endcase
225     end
226
227     wire ['ADDR] IdBPC = IfId_npc + ({{16{IfId_ir[15]}}, IfId_ir[15:0]} << 2);
228     always @(*) begin  ////// branch & jump resolution unit
229         {IdTPC, IdC, IdB} = 0;
230         case (IdOP)
231             6'h04: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS == IdRRT);          end        // BEQ
232             6'h05: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS != IdRRT);          end        // BNE
233             6'h06: begin IdB=1; IdTPC = IdBPC; IdC = ( IdRRS[31]||(IdRRS==0)); end        // BLEZ
234             6'h07: begin IdB=1; IdTPC = IdBPC; IdC = (~IdRRS[31]&&(IdRRS!=0)); end        // BGTZ
235             6'h01: begin IdB=1; IdTPC = IdBPC; IdC = (IdRT) ? ~IdRRS[31] : IdRRS[31]; end // BGEZ,BLTZ
236             6'h02: begin IdB=1; IdTPC = IfId_ir['ADDR]<<2; IdC = 1; end                   // J
237             6'h03: begin IdB=1; IdTPC = IfId_ir['ADDR]<<2; IdC = 1; end                   // JAL
238             6'h00: if      (IdFCT==6'h08) begin IdB=1; IdTPC = IdRRS; IdC = 1; end        // JR
239                    else if (IdFCT==6'h09) begin IdB=1; IdTPC = IdRRS; IdC = 1; end        // JALR
240         endcase
241     end
242
243     always @(posedge CLK or negedge RST_X) begin ///// update pipeline registers
244         if(!RST_X) {IdEx_npc, IdEx_rrs, IdEx_rrt, IdEx_dst, IdEx_ir, IdEx_opn, IdEx_attr} <= 0;
245         else if(!PSTALL) begin
246             IdEx_npc   <= (bstall) ? 0 : IfId_npc;
247             IdEx_rrs   <= (bstall) ? 0 : IdRRS;    // data from general-purpose register file
248             IdEx_rrt   <= (bstall) ? 0 : IdRRT;    // data from general-purpose register file
249             IdEx_dst   <= (bstall) ? 0 : IdDST;
250             IdEx_ir    <= (bstall) ? 0 : IfId_ir;
251             IdEx_opn   <= (bstall) ? 0 : IdOPN;
252             IdEx_attr  <= (bstall) ? 0 : IdATTR;
253         end
254     end
255
```

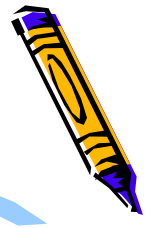
データパス - Exステージ - (3/5)

- 命令操作の実行またはアドレス生成を行います




```
256  /*****************************************************************************************************/
257  /* Stage 3 : EX, execute & address generation for LD/ST                                              */
258  /*****************************************************************************************************/
259  wire        [31:0] RRS_U, RRT_U;                          // output of data forwarding unit
260  wire signed [31:0] RRS_S = RRS_U;                         // signed wire
261  wire signed [31:0] RRT_S = RRT_U;                         // signed wire
262  wire         [4:0] SHAMT  = IdEx_ir[10:6];                // Shift amount
263  wire        [15:0] IMM    = IdEx_ir[15:0];                // Immediate
264  wire        [31:0] SET32I = {{16{IMM[15]}}, IMM};         // Sign Extended Imm.
265  wire       ['ADDR] SETADI = SET32I['ADDR] << 2;           // shifted immediate of address bit width
266  wire       ['ADDR] JADDR  = IdEx_ir['ADDR] << 2;          // Juma address
267  wire       ['ADDR] ExA = RRS_U['ADDR] + SET32I['ADDR];    // mem address of LD/ST
268  reg         [31:0] ExRSLT;                                // execution result
269  reg          [3:0] ExWE;                                  // memory write enable vector
270
271  wire DIVINIT  = IdOPN=='DIV_____  || IdOPN=='DIVU_____;
272  wire EXSIGNED = IdEx_opn=='DIV_____;
273  wire [63:0] DURSLT;
274
275  DIVUNIT divu(.CLK(CLK), .RST_X(RST_X), .INIT(DIVINIT), .SIGNED(EXSIGNED),
276               .A(RRS_U), .B(RRT_U), .RSLT(DURSLT), .BUSY(DIVBUSY));
277
278  FORWARDING frs(.SRC(IdEx_ir[25:21]), .DIN0(IdEx_rrs), .DOUT(RRS_U),
279                 .DST1(ExMa_dst), .DIN1(ExMa_rslt), .DST2(MaWb_dst), .DIN2(MaWb_rslt));
280  FORWARDING frt(.SRC(IdEx_ir[20:16]), .DIN0(IdEx_rrt), .DOUT(RRT_U),
281                 .DST1(ExMa_dst), .DIN1(ExMa_rslt), .DST2(MaWb_dst), .DIN2(MaWb_rslt));
```

データパス - Exステージ - (3/5)



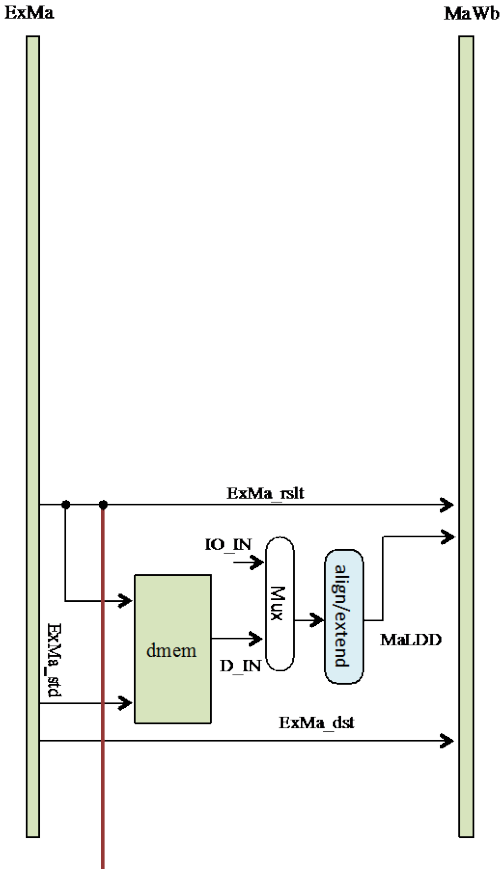
- 命令操作の実行またはアドレス生成を行います.

```
282
283     always @(*) begin
284         {ExRSLT, ExWE} = 0;
285         case ( IdEx_opn )
286           'ADD_____ : begin ExRSLT = RRS_U + RRT_U;                                                    end
287           'ADDI____ : begin ExRSLT = RRS_U + SET32I;                                                   end
288           'ADDIU___ : begin ExRSLT = RRS_U + SET32I;                                                   end
289           'ADDU____ : begin ExRSLT = RRS_U + RRT_U;                                                    end
290           'SUB_____ : begin ExRSLT = RRS_U - RRT_U;                                                    end
291           'SUBU____ : begin ExRSLT = RRS_U - RRT_U;                                                    end
292           'AND_____ : begin ExRSLT = RRS_U & RRT_U;                                                    end
293           'ANDI____ : begin ExRSLT = RRS_U & {16'h0, IMM};                                             end
294           'NOR_____ : begin ExRSLT = ~(RRS_U | RRT_U);                                                 end
295           'OR______ : begin ExRSLT = RRS_U | RRT_U;                                                    end
296           'ORI_____ : begin ExRSLT = RRS_U | {16'h0, IMM};                                             end
297           'XOR_____ : begin ExRSLT = RRS_U ^ RRT_U;                                                    end
298           'XORI____ : begin ExRSLT = RRS_U ^ {16'h0, IMM};                                             end
299           'SLL_____ : begin ExRSLT = RRT_U <<  SHAMT;                                                  end
300           'SRL_____ : begin ExRSLT = RRT_U >>  SHAMT;                                                  end
301           'SRA_____ : begin ExRSLT = RRT_S >>> SHAMT;                                                  end
302           'SLLV____ : begin ExRSLT = RRT_U <<  RRS_U[4:0];                                             end
303           'SRLV____ : begin ExRSLT = RRT_U >>  RRS_U[4:0];                                             end
304           'SRAV____ : begin ExRSLT = RRT_S >>> RRS_U[4:0];                                             end
305           'SLT_____ : begin ExRSLT = (RRS_U[31] ^ RRT_U[31]) ? RRS_U[31] : (RRS_U < RRT_U);            end
306           'SLTI____ : begin ExRSLT = (RRS_U[31] ^   IMM[15]) ? RRS_U[31] : (RRS_U < SET32I);           end
307           'SLTIU___ : begin ExRSLT = (RRS_U < SET32I);                                                 end
308           'SLTU____ : begin ExRSLT = (RRS_U < RRT_U);                                                  end
309           'JAL_____ : begin ExRSLT = IdEx_npc + 4;                                                     end
310           'JALR____ : begin ExRSLT = IdEx_npc + 4;                                                     end
311           'LUI_____ : begin ExRSLT = {IMM, 16'h0};                                                     end
312           'LB______ : begin ExRSLT = ExA;                                                              end
313           'LBU_____ : begin ExRSLT = ExA;                                                              end
314           'LH______ : begin ExRSLT = {ExA[31:1], 1'b0 };                                               end
315           'LHU_____ : begin ExRSLT = {ExA[31:1], 1'b0 };                                               end
316           'LW______ : begin ExRSLT = {ExA[31:2], 2'b00};                                               end
317           'SB______ : begin ExRSLT = ExA;                    ExWE = {4'b0001<<ExA[1:0]};               end
318           'SH______ : begin ExRSLT = {ExA[31:1], 1'b0 }; ExWE = ExA[1] ? 4'b1100 : 4'b0011;            end
319           'SW______ : begin ExRSLT = {ExA[31:2], 2'b00}; ExWE = 4'b1111;                               end
320           'MFHI____ : begin ExRSLT = hi;                                                               end
321           'MFLO____ : begin ExRSLT = lo;                                                               end
322         endcase
323     end
324
325     always @( posedge CLK or negedge RST_X ) begin  ///// update hi and lo register
326         if(!RST_X) {hi, lo} <= 0;
327         else if(!PSTALL) begin
328             if(IdEx_opn == 'MULT_____) {hi, lo} <= RRS_S * RRT_S;
329             if(IdEx_opn == 'MULTU____) {hi, lo} <= RRS_U * RRT_U;
330             if(IdEx_opn == 'DIV______) {hi, lo} <= (RRT_U) ? DURSLT : 0;
331             if(IdEx_opn == 'DIVU_____) {hi, lo} <= (RRT_U) ? DURSLT : 0;
332         end
333     end
334
335     always @( posedge CLK or negedge RST_X ) begin ///// update pipeline registers
336         if(!RST_X) {ExMa_rslt, ExMa_dst, ExMa_mwe, ExMa_oe, ExMa_lds, ExMa_std} <= 0;
337         else if(!PSTALL) begin
338             ExMa_rslt <= ExRSLT;
339             ExMa_dst  <= IdEx_dst;
340             ExMa_std  <= (IdEx_opn=='SB_______) ? {4{RRT_U[7:0]}} :
341                          (IdEx_opn=='SH_______) ? {2{RRT_U[15:0]}} : RRT_U;
342             ExMa_mwe  <= ExWE;
343             ExMa_oe   <= (IdEx_attr & 'LDST_ANY) ? 1 : 0;
344             ExMa_lds  <= (IdEx_opn=='LH_______) ?  1 : (IdEx_opn=='LHU______) ? 2 : // Load selector
345                          (IdEx_opn=='LB_______) ?  4 : (IdEx_opn=='LBU______) ? 8 :
346                          (IdEx_opn=='LW_______) ? 16 : 0;
347         end
348     end
349
```

データパス - Maステージ - (4/5)

- データ・メモリ中のオペランドにアクセスします.

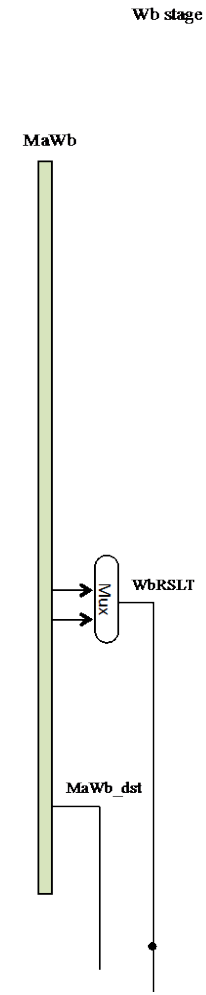
Ma stage




```
350     /*****************************************************************************************************/
351     /* Stage 4 : MA, memory access for LD/ST                                                             */
352     /*****************************************************************************************************/
353     assign D_ADDR = (ExMa_oe)            ? ExMa_rslt : 0;
354     assign D_OUT  = (ExMa_oe)            ? ExMa_std  : 0;
355     assign D_WE   = (ExMa_oe && !PSTALL) ? ExMa_mwe  : 0;
356     assign D_OE   =  ExMa_oe;
357
358     // wire [31:0] MaLD_ = (ExMa_rslt[23]) ? IO_IN : D_IN; // use I/O input if the high address
359     wire [31:0] MaLD_ = D_IN; // This edition does not use I/O.
360     wire [31:0] MaLDD = MaLD_ >> (8*ExMa_rslt[1:0]);    // loaded data, align data here
361
362     assign MaRSLT = (ExMa_lds[0]) ? {{16{MaLDD[15]}}, MaLDD[15:0]} : // opn==‘LH_____
363                     (ExMa_lds[1]) ? { 16'd0         , MaLDD[15:0]} : // opn==‘LHU____
364                     (ExMa_lds[2]) ? {{24{MaLDD[ 7]}}, MaLDD[ 7:0]} : // opn==‘LB_____
365                     (ExMa_lds[3]) ? { 24'd0         , MaLDD[ 7:0]} : // opn==‘LBU____
366                     (ExMa_lds[4]) ?                   MaLDD        : ExMa_rslt;
367
```

データパス - Wbステージ - (5/5)

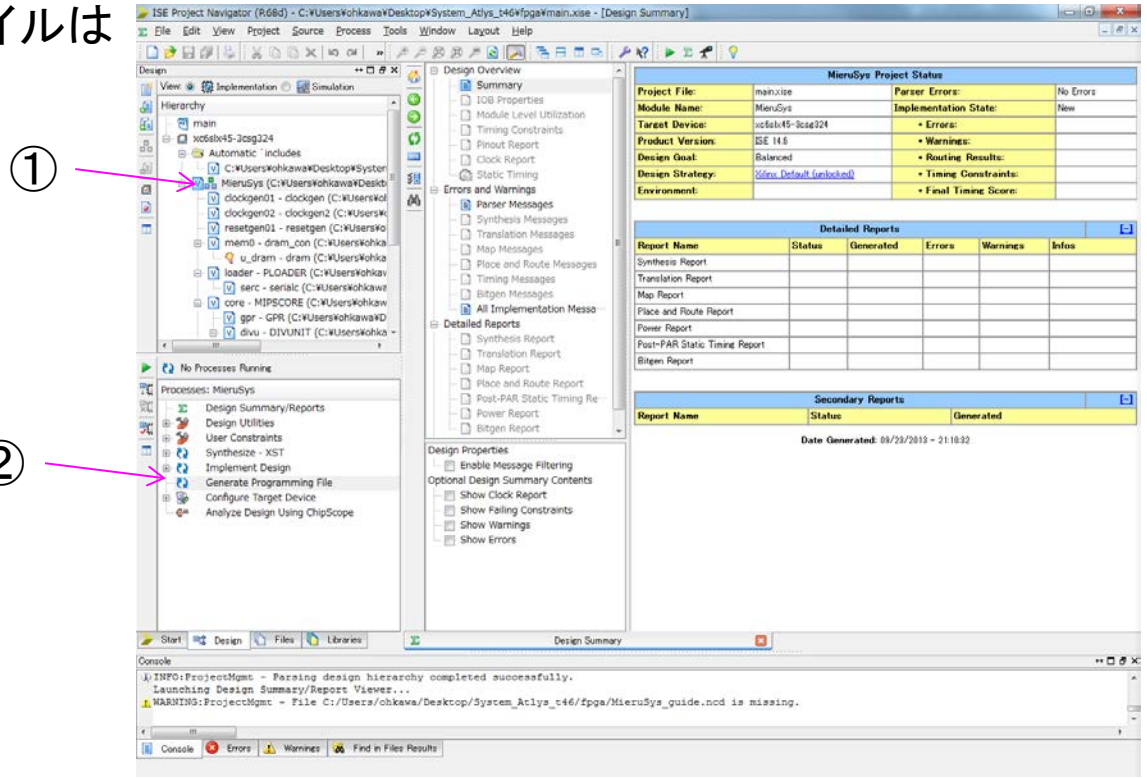
- 結果をレジスタに書き込みます

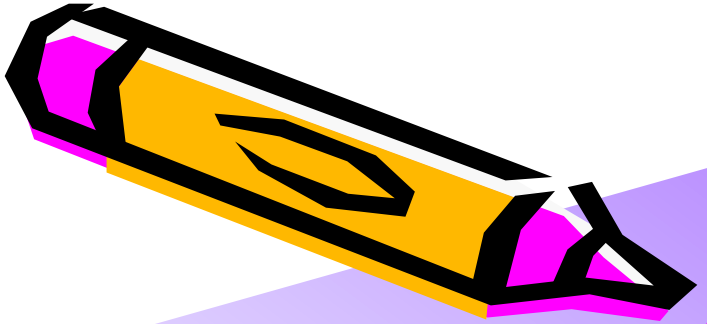



```
368       always @( posedge CLK or negedge RST_X ) begin ///// update pipeline registers
369           if(!RST_X) {MaWb_rslt, MaWb_dst} <= 0;
370           else if(!PSTALL) begin
371               MaWb_rslt <= MaRSLT;
372               MaWb_dst  <= ExMa_dst;
373           end
374       end
375
```

Atlysボード
FPGA回路データ(bitファイル)の作成方法

- ホームページからSystem_Atlys_t63.zipをダウンロードし，展開します．
- fpgaディレクトリ中にある main.xise をISEで開きます．
- ①トップモジュール(MieruSys)をクリックで選択し、
  ②Generate Programming Fileをダブルクリックすると，数分で完了します．
  「Process "Generate Programming File" completed successfully」
- 作成した回路データファイルは
  fpga/mierusys.bit
  です．
- README.txt
  も参考にしてください．



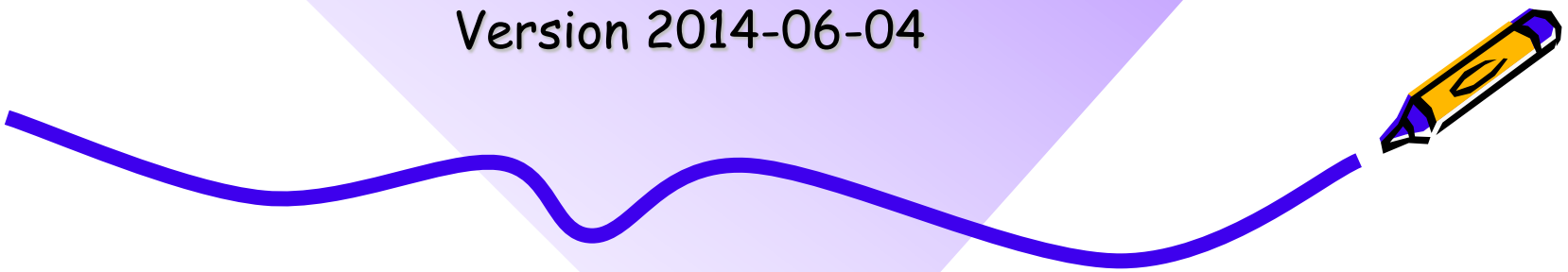


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

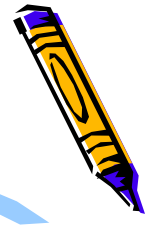
DRAM Memory Interface (ATLYS board)

コンテスト実行委員会コアチーム

Version 2014-06-04



このドキュメント

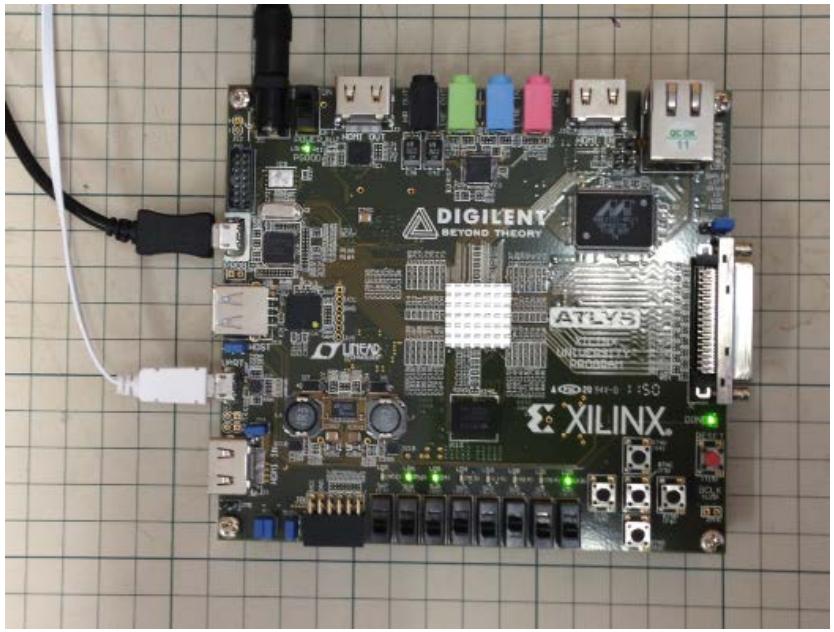


- このドキュメントでは，Digilent Atlysに搭載されているFPGAとDRAMを対象とし，Xilinx Memory Interface Generator を用いたDRAMモジュールの生成方法を説明します．
- ここでは，リファレンスデザインに含まれるシンプルなDRAMコントローラの生成方法を説明しています．高性能化のための様々な設定がありますので，性能向上のために試してみてください．
- ただし，このDRAMモジュールの変更は難しいので，FPGA開発に慣れてきた段階で挑戦してください．

- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/
- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

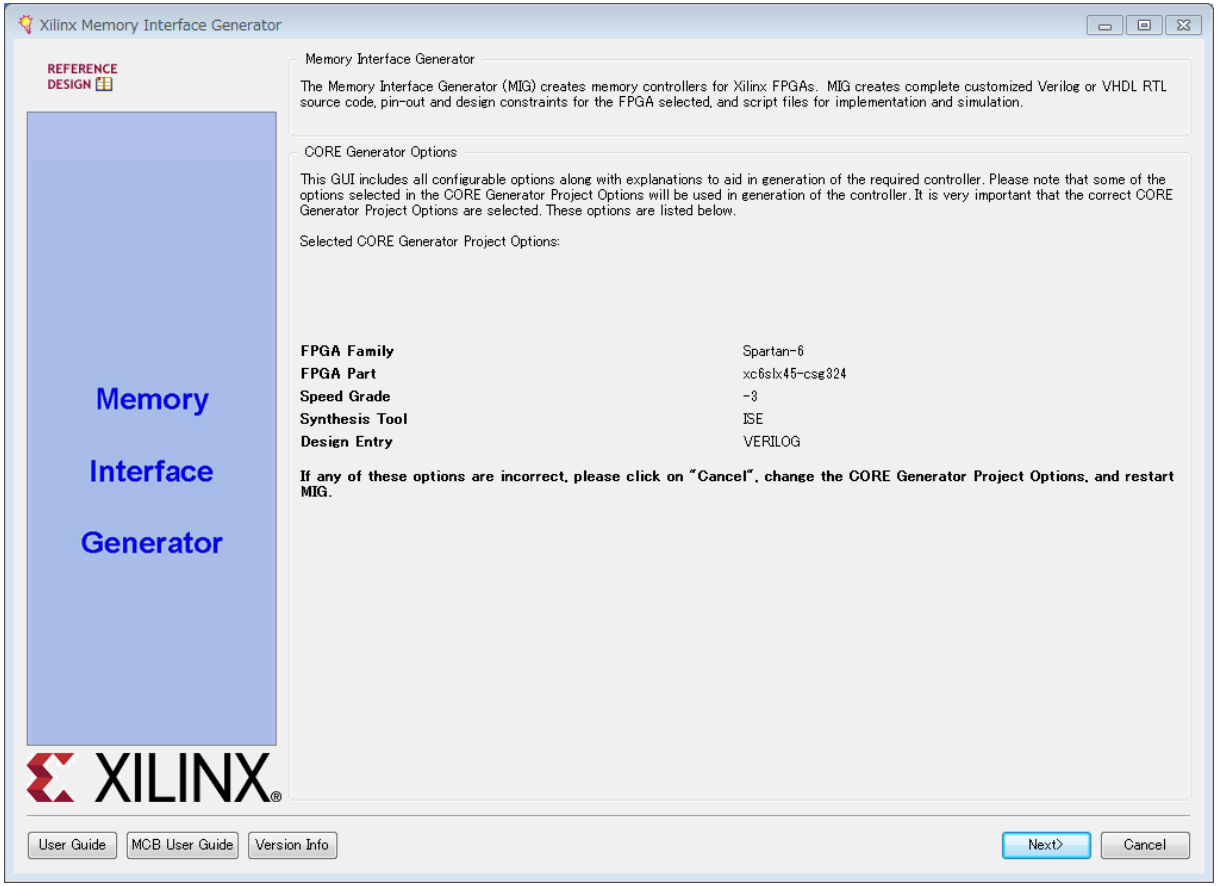
DRAM on Atlys Board

- Atlysに搭載されているDRAM
- DDR2 128MByte
  - MIRA P3R1GE3EGF G8E DDR2
  - 16-bit data bus, 64M locations
- Xilinx Memory Interface Generator で用いる主なパラメータ
  - selecting the "EDE1116AXXX-8E" device
  - RZQ pin location : L6
  - ZIO pin location : C2



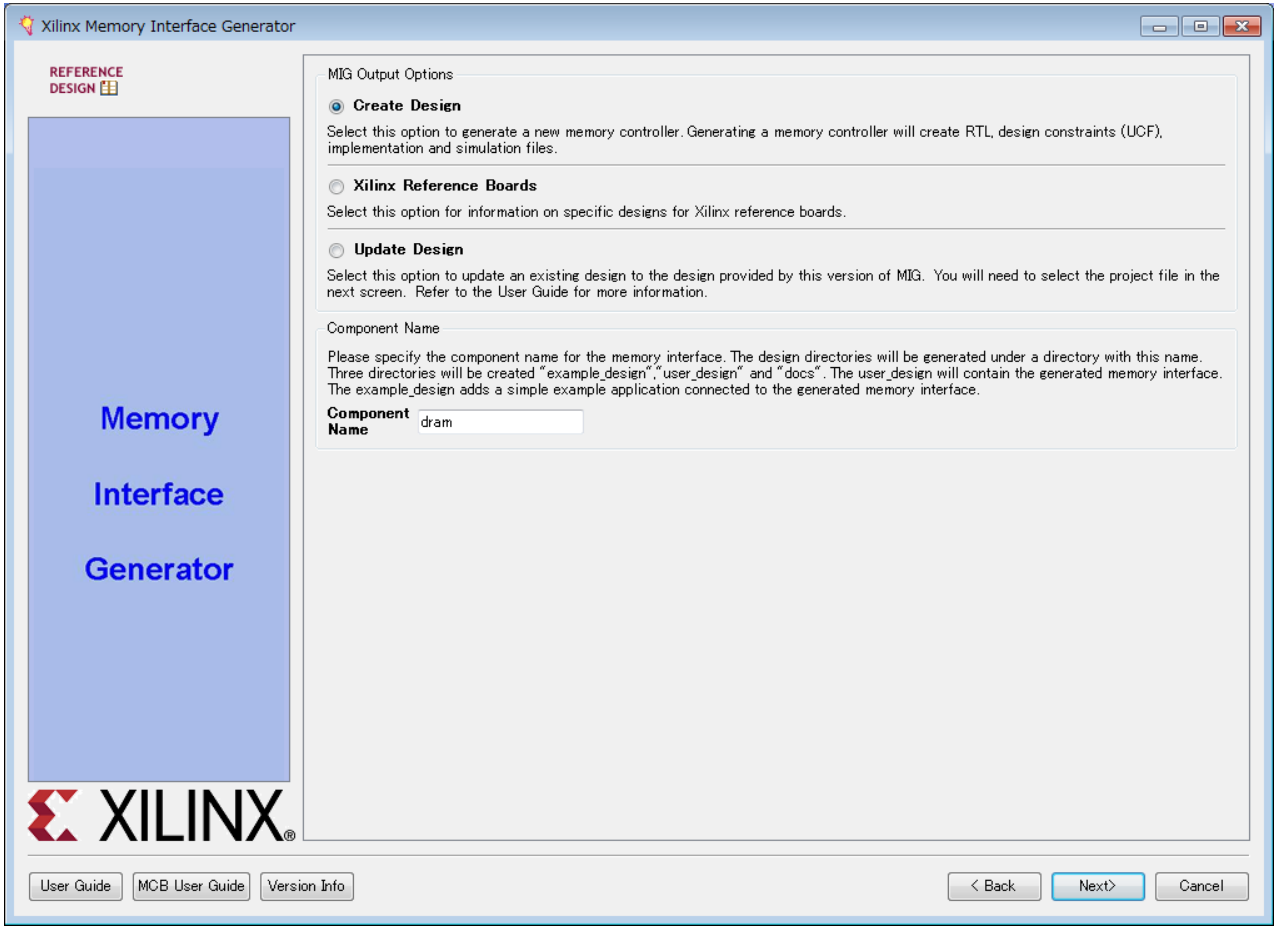
Xilinx Memory Interface Generator (1)

- ISEから Project -> New Source -> IP
  - Memories & Storage Elements -> Memory Interface Generator
  - MIG を 選択して MIG を起動
- Next を選択



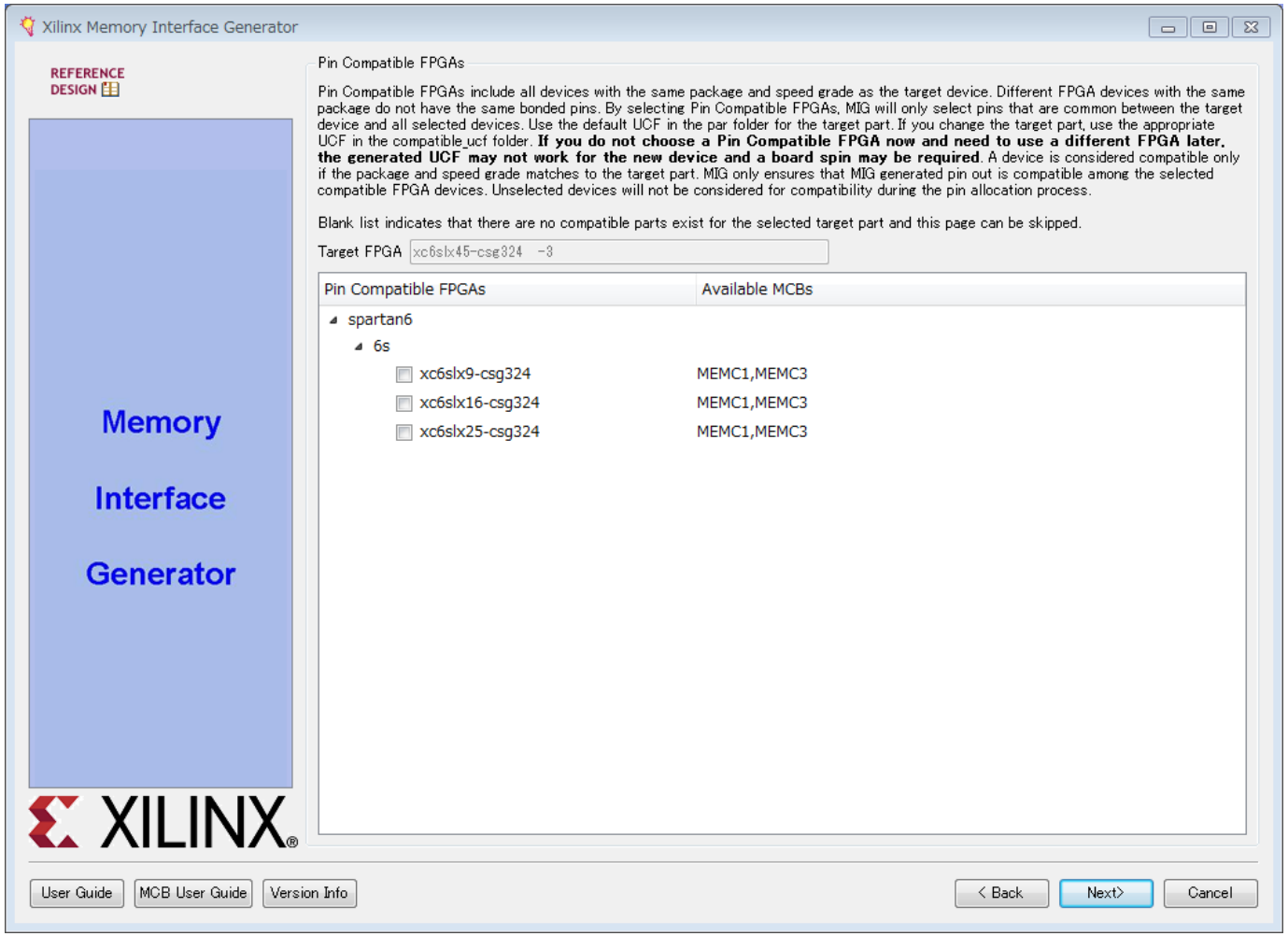
Xilinx Memory Interface Generator (2)

- Create Design を選択
- Component Name : dram に設定



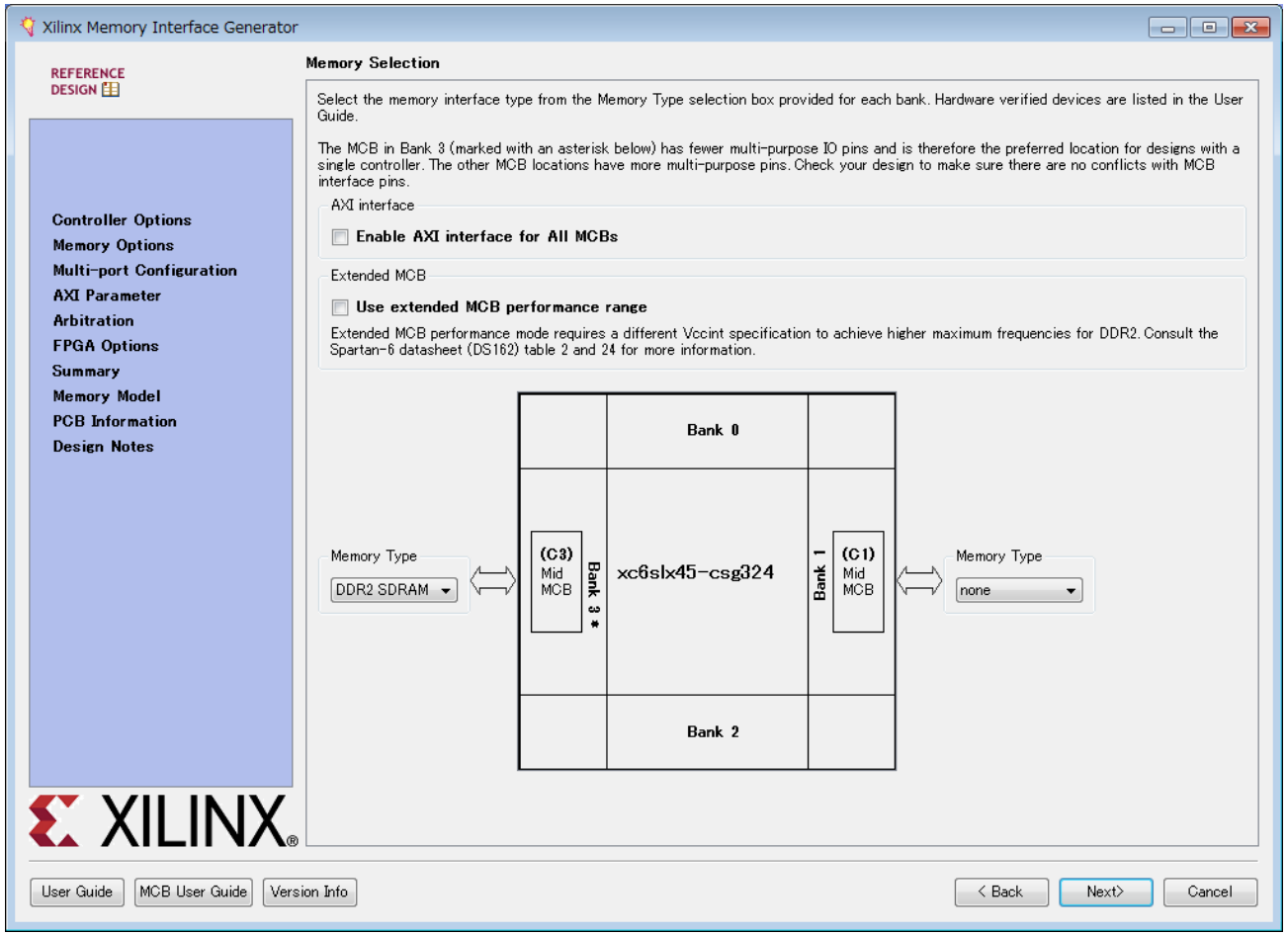
Xilinx Memory Interface Generator (3)

- Pin Compatible FPGAs では，チェックをいれない



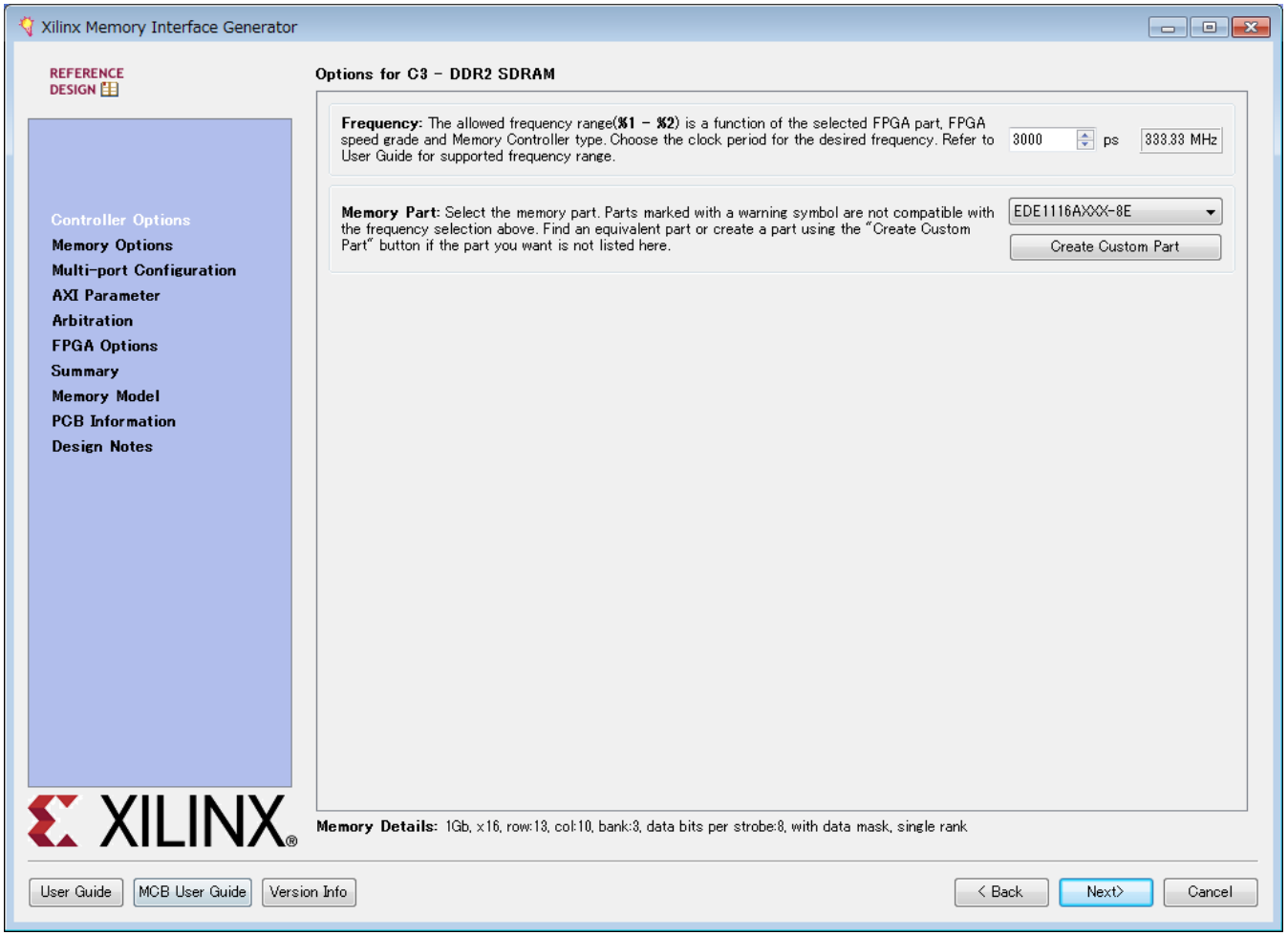
Xilinx Memory Interface Generator (4)

- チェックボックスはチェックしない
- Bank 3 を DDR2 SDRAM に設定



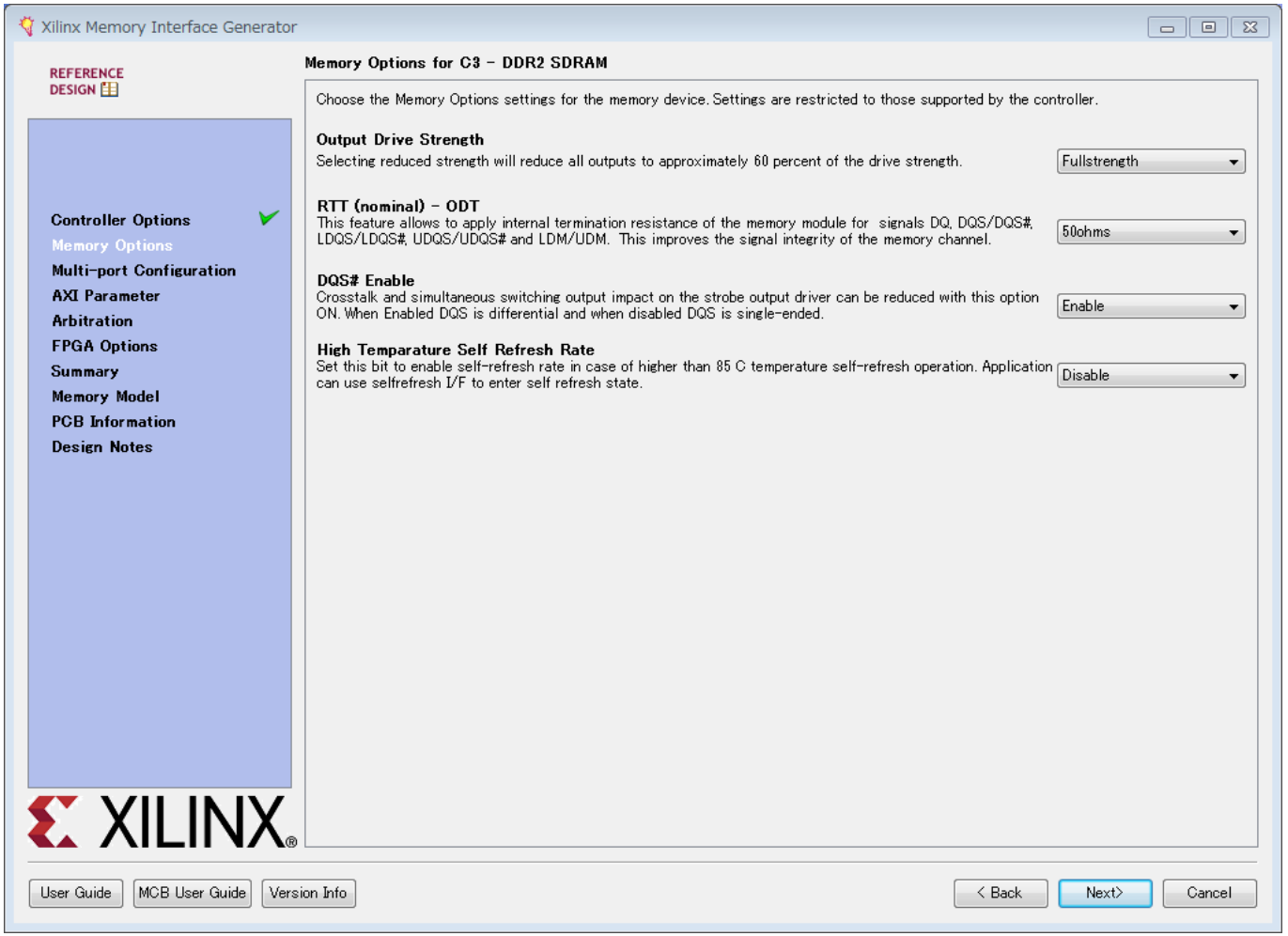
Xilinx Memory Interface Generator (5)

- 3000 ps, EDE1116AXXX-8E を選択



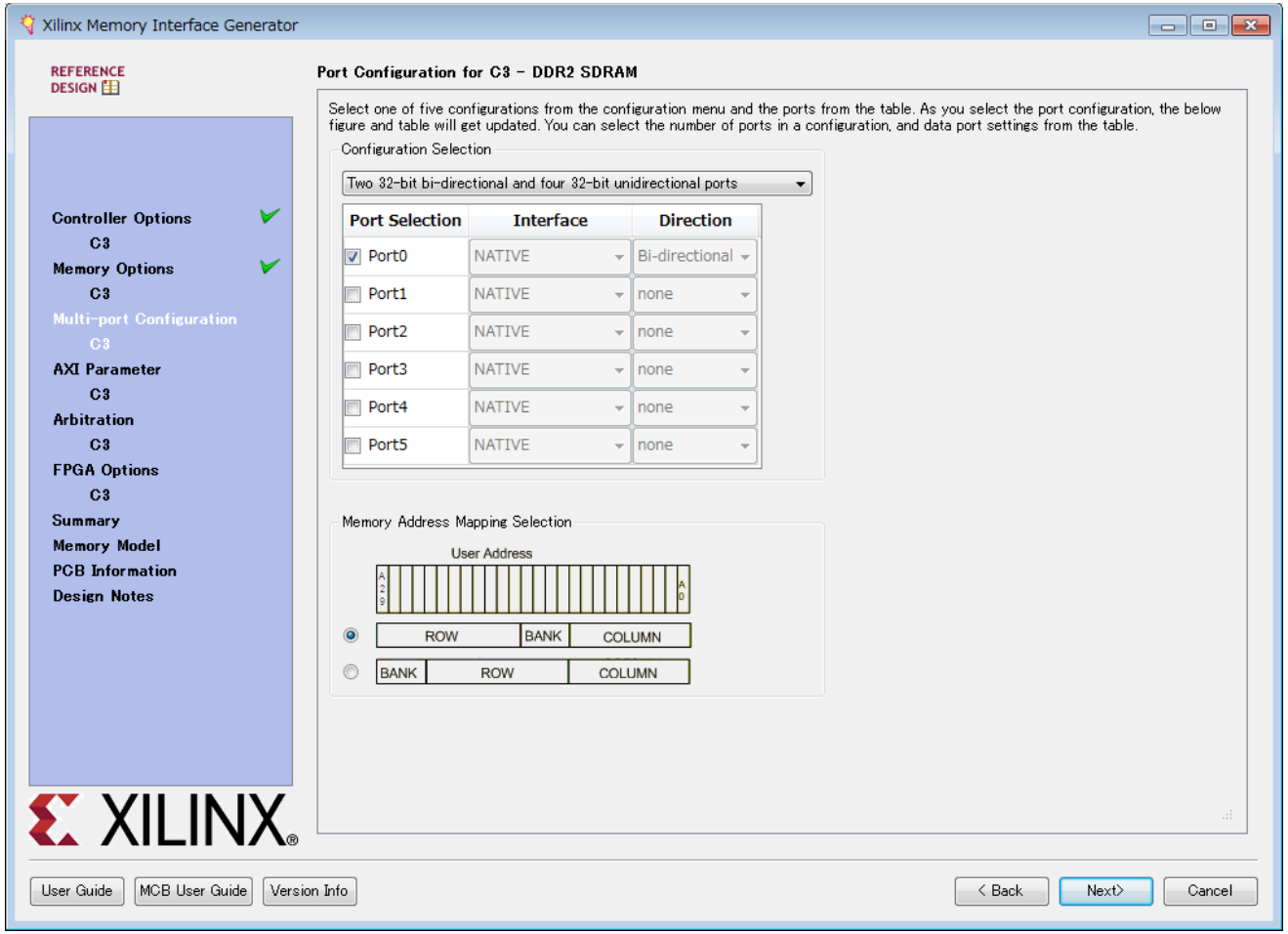
Xilinx Memory Interface Generator (6)

- Fullstrength, 50ohms, Enable, Disable を選択



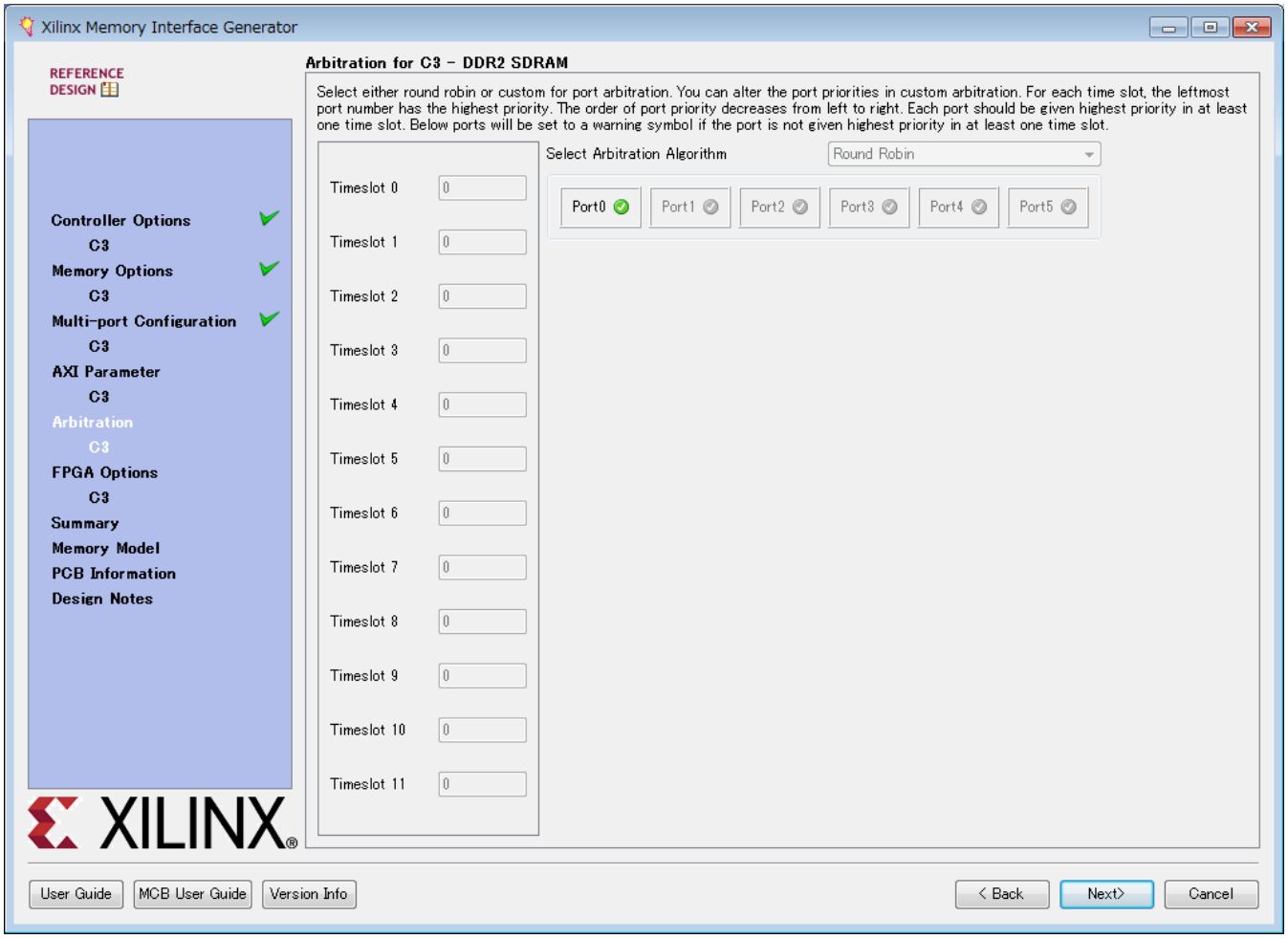
Xilinx Memory Interface Generator (7)

- Two 32-bit bi-directional and four 32-bit unidirectional ports を選択
- Port0 をチェック，[ROW, BANK, COLUMN] をチェック



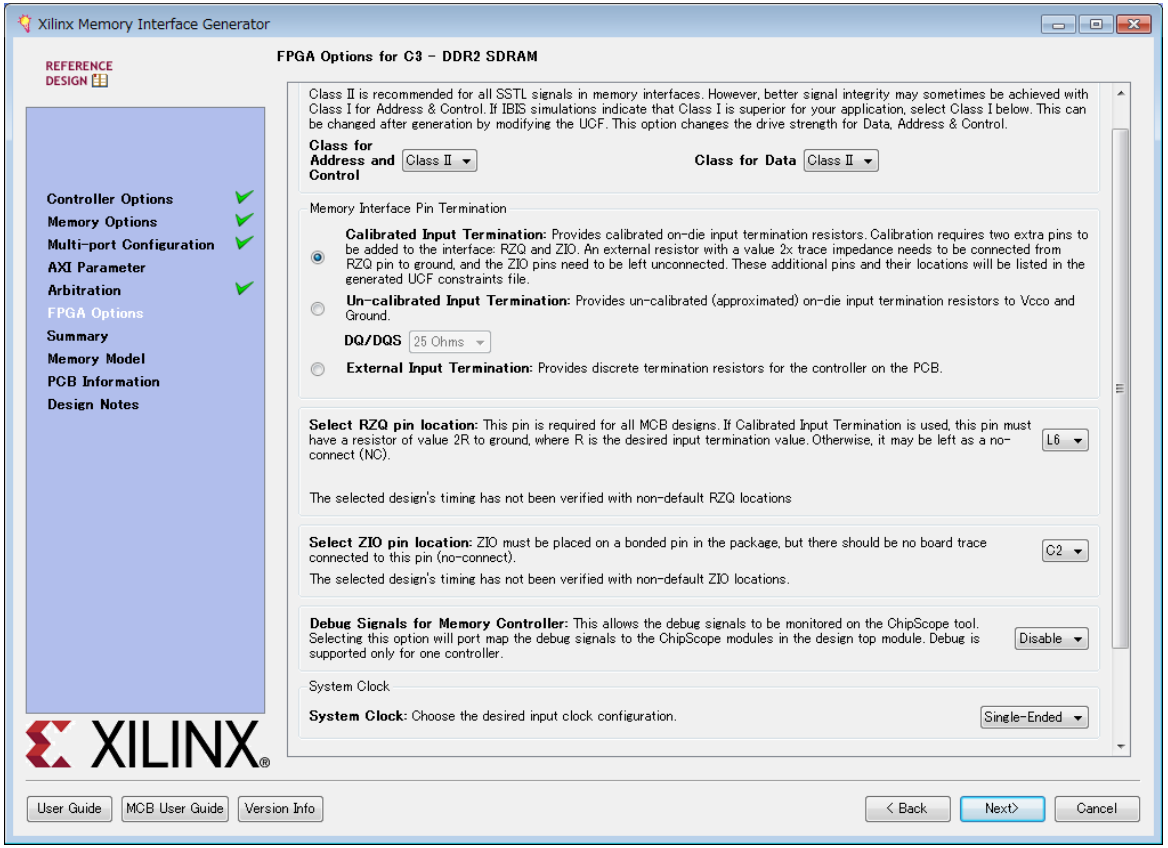
Xilinx Memory Interface Generator (8)

- Next で次に進む



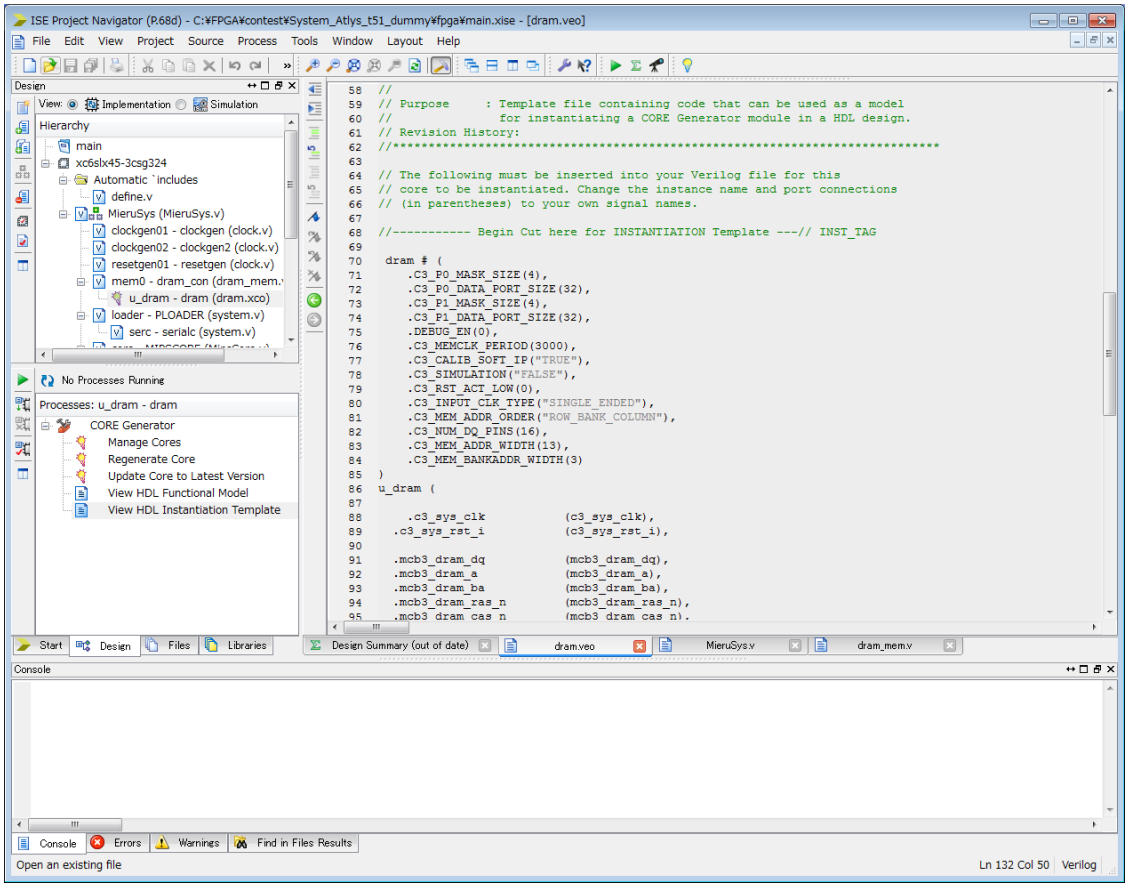
Xilinx Memory Interface Generator (9)

- STTL output Drive Strength : Class II, Class II を選択
- Calibrated Input Termination を選択
- RZQ pin location : L6
- ZIO pin location : C2
- Debug : Disable
- Clock : Single-Ended

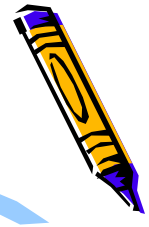


Xilinx Memory Interface Generator (10)

- ISE CORE Generator
  - View HDL Instantiation Template のコードを dram_mem.v にコピーしてワイヤを修正

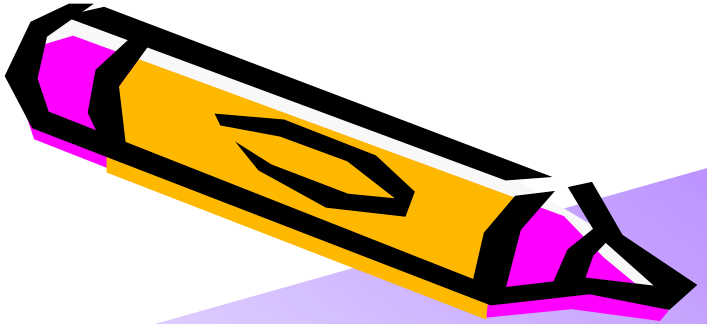


Xilinx Memory Interface Generator (11)



- 論理合成でエラーになるので，生成されたVerilogコードを修正
  - fpga/ipcore_dir/dram/user_design/rtl/infrastructure.v の 151行目付近を編集
  - IBUFG を用いていたものを，利用しないように変更（上位のモジュールにて，IBUFGを利用しているため）

```
152         //***********************************************
153         // SINGLE_ENDED input clock input buffers
154         //***********************************************
155  /*
156         IBUFG  u_ibufg_sys_clk
157             (
158              .I  (sys_clk),
159              .O  (sys_clk_ibufg)
160              );
161
162  */
163     assign sys_clk_ibufg = sys_clk;
164     end
165
166
```

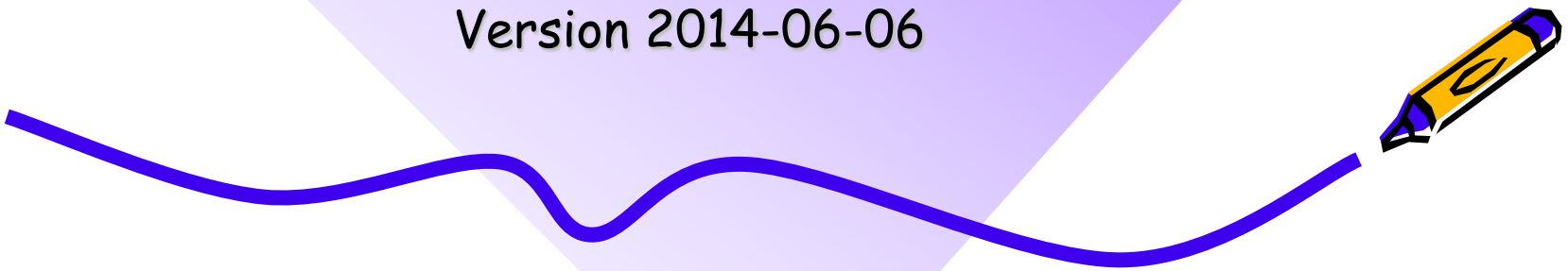


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

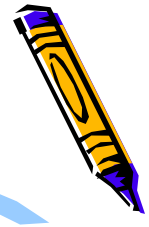
MIPS Cross Compiler Setup Manual

コンテスト実行委員会コアチーム

Version 2014-06-06



このドキュメント

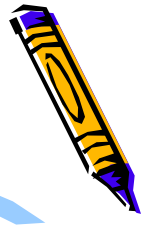


- リファレンスデザインに含まれるプロセッサは命令セットアーキテクチャとして，MIPSを採用しています．
- リファレンスデザインで実行するアプリケーションを生成するために，MIPSクロス開発環境（クロスコンパイラなどを含む環境）が必要となります．
- このドキュメントでは，MIPSクロス開発環境の構築方法を説明します．

- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

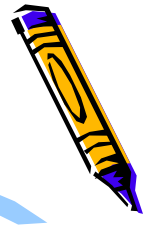
コンパイル済みのMIPSクロス開発環境の設定方法



- 幾つかのLinuxのためのMIPSクロスコンパイラのバイナリを準備しています．
  - このバイナリでは，buildroot-2009.08.tar.gz を利用しています．
  - RedHat5 x86版（64ビット），CentOS6.4 x86版（32ビット），Ubuntu12.04 x86版（64ビット）のいずれかがインストールされたLinuxマシンの利用を推奨します．

- お手軽に環境を構築するためには，使っているLinuxに最も近いバイナリをダウンロード，展開します．
- /home/share/cad/ というディレクトリを作成し，そこで，バイナリを展開し，mipsel-contest というシンボリックリンクを作成します．
- 具体的なコマンド例は次のスライドを参照してください．（コマンドの先頭には $ を追加しています．）

コンパイル済みのMIPSクロス開発環境の設定方法

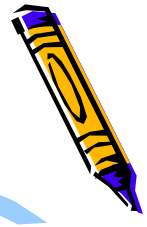


- RedHat Linux Version 5 (aquabase)
  - mips_procdesign_redhat5.tgz
    を次のURLからダウンロード http://www.arch.cs.titech.ac.jp/contest/
  - $ cd /home/share/cad/
  - $ tar xvfz mips_procdesign_redhat5.tgz
  - $ ln -s mips_proc_redhat5 mipsel-contest

- CentOS release 6 (plumbase)
  - mips_procdesign_cent63.tgz
    を次のURLからダウンロード http://www.arch.cs.titech.ac.jp/contest/
  - $ cd /home/share/cad/
  - $ tar xvfz mips_procdesign_cent63.tgz
  - $ ln -s mips_proc_cent63 mipsel-contest

- Ubuntu 12.04 LTS (gn001)
  - mips_procdesign_ubuntu1204.tgz
    を次のURLからダウンロード http://www.arch.cs.titech.ac.jp/contest/
  - $ cd /home/share/cad/
  - $ tar xvfz mips_procdesign_ubuntu1204.tgz
  - $ ln -s mips_proc_ubuntu1204 mipsel-contest

コンパイル済みのMIPSクロス開発環境の動作確認

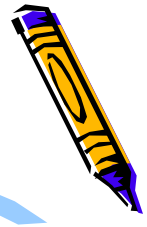


- $ /home/share/cad/mipsel-contest/usr/bin/mipsel-linux-gcc -v
- コンパイラのバージョンなどが表示されることを確認してください.

- 簡単なCのプログラム main.c を作成して次のコマンドを実行してください.
- $ /home/share/cad/mipsel-contest/usr/bin/mipsel-linux-gcc -S main.c
- コンパイルされて main.s というファイルが生成されます.

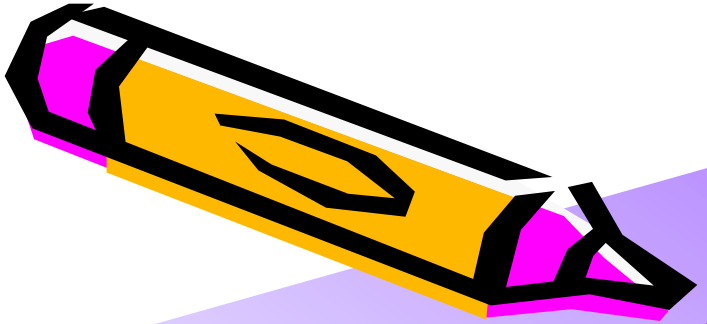
- 補足
  - 提供しているバイナリには シミュレータ SimMips と, メモリイメージ作成のための memgen がインストールされています.
  - 以下のコマンドで正しく動作することを確認してください.
  - $ /home/share/cad/mipsel-contest/usr/bin/SimMips
  - $ /home/share/cad/mipsel-contest/usr/bin/memgen

MIPSクロス開発環境の構築方法



- 先に説明したバイナリを用いることでお手軽にMIPSクロスコンパイラが利用できるようになります．

- 自分でMIPSクロスコンパイラを構築したい場合には次のページを参考にしてください．
  - http://www.arch.cs.titech.ac.jp/mcore/buildroot.html
  - ただし，
    fakeroot_1.9.5.tar.gz が無いと言われてエラーになることがあるので，ダウンロードして展開したディレクトリ下の dl というディレクトリに
    fakeroot_1.9.5.tar.gz をコピーして，再度 make してください．

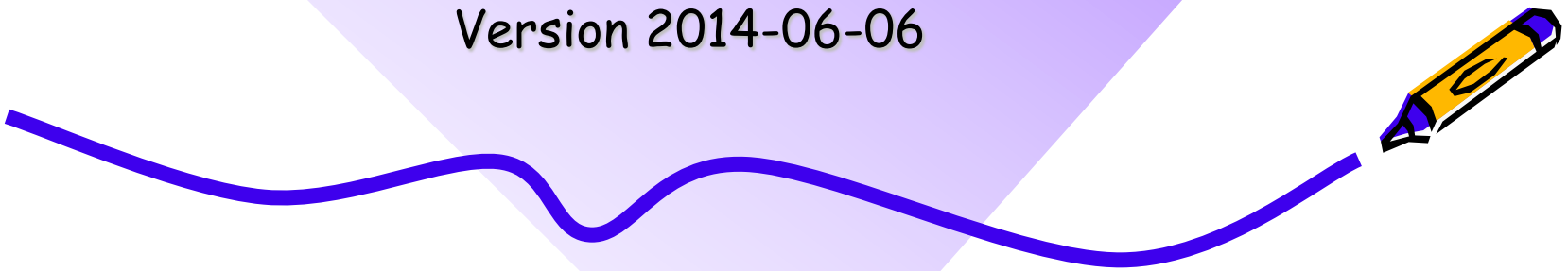


The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

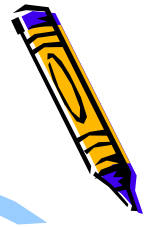
SDK Setup Manual

コンテスト実行委員会コアチーム

Version 2014-06-06



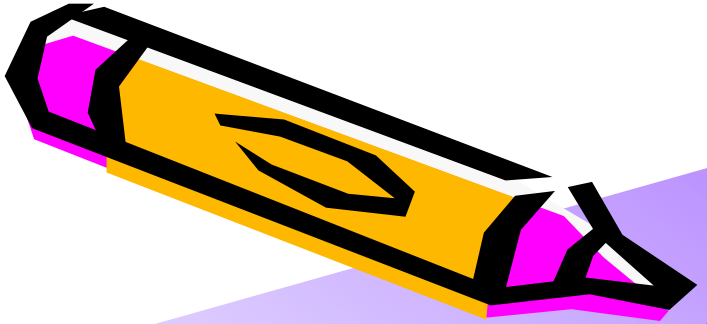
リファレンスデザインのためのアプリケーション開発



- リファレンスデザインのためのアプリケーションをコンパイルするためにSDK(ソフトウェア開発キット)を提供します.
- ホームページから DesignCon_SDK.1.0.2.tgz というファイルをダウンロードし展開してください(バージョンアップによりファイル名が異なることがあります).
- 展開したディレクトリの README.txt に使い方の説明があります.

- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください.
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc



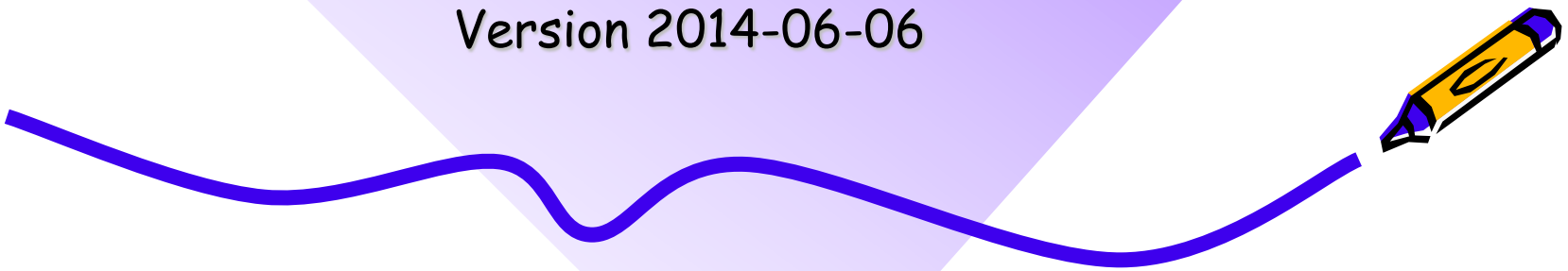
プロセッサ設計部門の
SDKに含まれるソースコードは
今後改善する予定です

The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

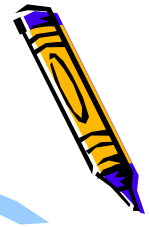
Application Specification of Processor Design Category

コンテスト実行委員会コアチーム

Version 2014-06-06



このドキュメント

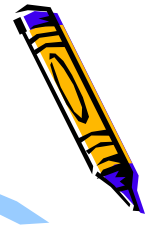


- このドキュメントでは，4種類のアプリケーションプログラムの仕様を説明します

- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

310_sort

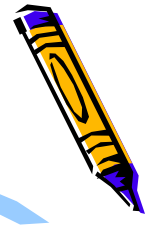


- 256KBのデータファイル 310sort.dat には，次の構造体で定義される，ソートすべきデータを初期化するためのランダムデータと，ソートの要素数 n が格納されます．
- このファイルの生成方法は，data.c と Makefile を参考にしてください．
- iを自然数として，ソートの要素数 n = i * 1024 により指定されます．

```
struct data_t {
    unsigned int buf[SIZE-1];  // 256KB -4Byte buffer
    int n;                     // the number of elements
};
```

- サンプルアプリケーションは main.c に記述されています．
  - まず，データファイルの値を用いて，配列 data を初期化します．
  - 確認のため，先頭の100要素の値を表示します．
  - 次に，qsort によりソーティングをおこないます．main.c ではクイックソートが実装されていますが，任意のソーティングアルゴリズムを用いて修正しても構いません．
  - ソーティング後の値を適切にサンプリングして表示します．

320_mm

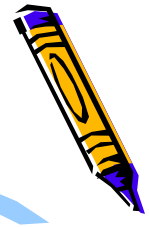


- 256KBのデータファイル 320mm.dat には，次の構造体で定義される，行列を初期化するためのランダムデータと，行列サイズ n が格納されます．
- このファイルの生成方法は，data.c と Makefile を参考にしてください．
- iを自然数として，ソートの要素数 n = i * 16 により指定されます．

```
struct data_t {
    unsigned int buf[SIZE-1];  // 256KB -4Byte buffer
    int n;                     // matrix size
};
```

- サンプルアプリケーションは main.c に記述されています．
  - まず，データファイルの値を用いて，正方行列 a, b, c を初期化します．
  - 行列積 c = a x b を計算します．
  - 確認のため，c の一部の要素を表示します．
  - 確認のため，c の全ての要素の加算（オーバフローは無視）の結果を表示します．

330_stencil

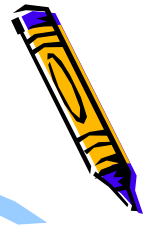


- 256KBのデータファイル 330stencil.dat には，次の構造体で定義される，配列の初期化のためのランダムデータと，配列サイズn ， イタレーション回数 iter が格納されます．
- このファイルの生成方法は， data.c と Makefile を参考にしてください．
- iを自然数として，ソートの要素数 n = i * 32，iter = i *2 により指定されます．

```
struct data_t {
    int buf[SIZE-2];  // 256KB -4Byte buffer
    int n;
    int iter;
};
```

- サンプルアプリケーションは main.c に記述されています．
  - まず，データファイルの値を用いて，配列 buf1, buf2 を初期化します．
  - それぞれの要素の自分自身を含む9近傍の平均により，次のイタレーションの値を計算します．配列間のコピーを排除するため， buf1, buf2 を相互に更新する手法を採用しています．ある要素は常に 0x9999999 の値を保持するものとしています．
  - 計算終了後，確認のため，一部の要素を表示します．
  - 確認のため，全ての要素の加算（オーバフローは無視）の結果を表示します．

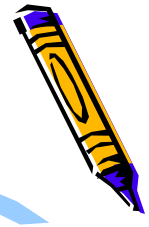
340_spath：最短路問題の概要



- 概要
  - 与えられたグラフ上の，指定された2点間の最短距離を求める問題です．
  - グラフ・問題定義は，ファイル（.gr, .p2p）にて定義されています．
    - 問題のグラフは，最短路がただ一つ存在することが保証されています．
      【第1回からの追加点】

- グラフ・問題定義の出典に関する情報
  - グラフ・問題定義は，9th DIMACS Implementation Challenge - Shortest Pathsにて用いられた，ランダムグラフ生成ツールを用いて作ります．
    - 生成ツール類は，パブリックドメインのソフトウェアとして配布されています．
      - http://www.dis.uniroma1.it/challenge9/download.shtml
  - グラフ・問題定義のファイル形式の情報は以下のURLからご確認ください．
    - http://www.dis.uniroma1.it/challenge9/format.shtml

340_spath：入力データと実装の概要

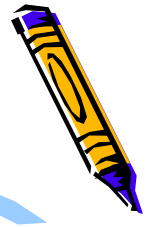


- 256KBのデータファイル 340spath.dat には，次の構造体で定義される，対象グラフのノード間接続（エッジ）を表すデータ(buf)と，ノード数 n，エッジ数m，始点ノード番号start，終点ノード番号 goalが格納されます．
- このファイルの生成方法は，generator.rbと Makefile を参考にしてください．

```
struct data_t {
    unsigned int buf[SIZE-4];  // 256KB -16(4x4)Byte buffer
    int n;                     // the number of nodes
    int m;                     // the number of edges
    int start;                 // ID of the start node
    int goal;                  // ID of the goal node
};
```

- 入力ファイルのエッジのデータは，1つのエッジあたり3ワードで構成されます．
  詳しくは，「340_spath：データ形式について」のスライドを見てください．
- アルゴリズムは main.cに記述されています．
  - メインの関数では，データの初期化後にfindShortestPath関数を呼びます．
  - findShortestPath関数は，Dijkstraのアルゴリズムにより最短路を求めます．
  - 最後に最短ルートのノード番号を順番に表示し，コスト（距離）の合計を表示します．

340_spath：データ形式について



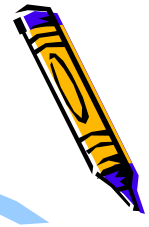
- エッジのデータ形式（入力データ）
  - エッジのデータは次の構造体で表されます．1つのエッジあたり3ワード（12バイト）で構成されます．
  - 入力データのbufはedgeの配列になります．
  - fromとtoはそれぞれ始点・終点のノード番号です．
  - costはノード間の距離（整数値）です．

```
typedef struct {
  int from;
  int to;
  int cost;
} edge;
```

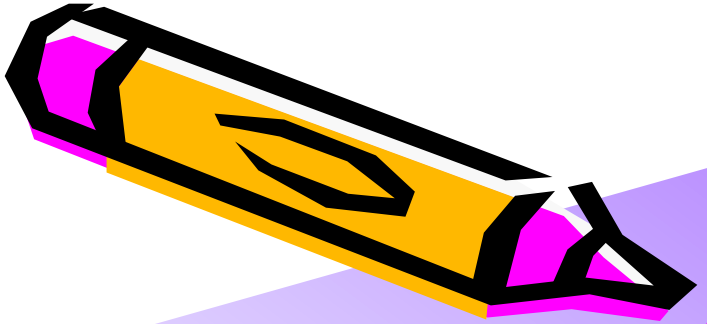
- ノードのデータ形式（内部データ）
  - ノードのデータは次の構造体で表されます．1つのノードあたり3ワード（12バイト）で構成されます．
  - Dijkstraのアルゴリズムを想定しています．
  - currentCostはノードの現在のコスト（合計値）です．
  - isMinimumCostは，TRUE(1)かFALSE(0)で最小コストであることが確定していることを示します．
  - fromNodeForMinimumCostは、最小コスト（最短路）となる場合の，直前のノード番号を保持します．

```
typedef struct {
  int currentCost;
  int isMinimumCost;
  int fromNodeForMinimumCost;
} node;
```

340_spath：ファイルの説明（生成ファイルを含む）



- バイナリ
  - 340spath　　ーシミュレータ用実行バイナリ（elf）
  - 340spath512.bin　ーFPGA用バイナリ(512KB)
  - 340spath.bin　　ーFPGA用バイナリ　コード部分のみ(256KB)
  - 340spath.dat　　ー入力データ(256KB), data.binも同じもの
- スクリプト
  - Makefile　　ーデフォルトでFPGA用バイナリ(512KB)を生成
  - generator.rb　ーグラフ・問題定義ファイル（.gr, .p2p）から、データファイルを生成
  - sim.m　　ーシミュレーションでデータを読み込むための定義
- ソースファイル
  - main.c　　ーメイン関数と最短路問題に関する実装
  - startup.S　　ーブートアップコード
- データファイル
  - n2048.gr　　ーグラフ定義ファイル　（2048ノード）
  - n2048.p2p　　ー問題定義ファイル（始点・終点の指定）
  - n6.gr　　ー【参考】単純なグラフ定義ファイル　（6ノード）
  - n6.p2p　　ー【参考】問題定義ファイル（始点・終点の指定）



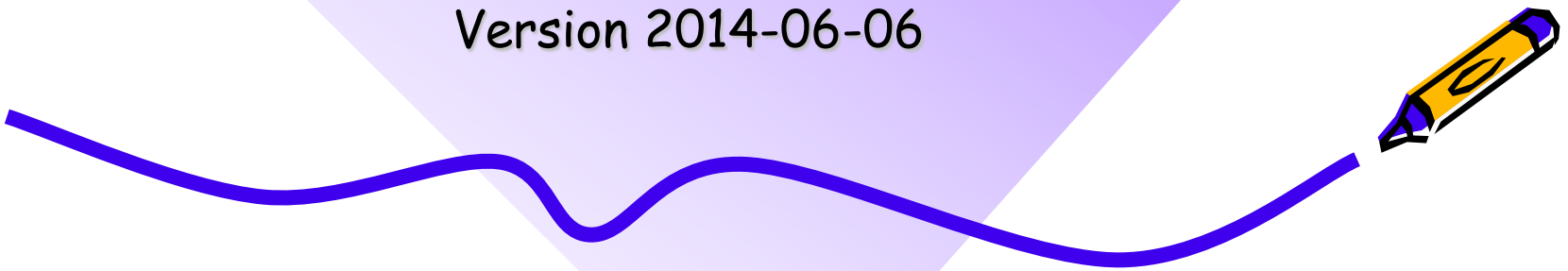
コンピュータシステム設計部門
のアプリ仕様は今後変更される
可能性があります

The 2nd ARC/CPSY/RECONF
High-Performance Computer System Design Contest

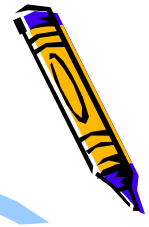
Application Specification of Computer System Design Category

コンテスト実行委員会コアチーム

Version 2014-06-06



このドキュメント



- このドキュメントでは，コンピュータシステム部門のアプリケーションプログラムの仕様を説明します

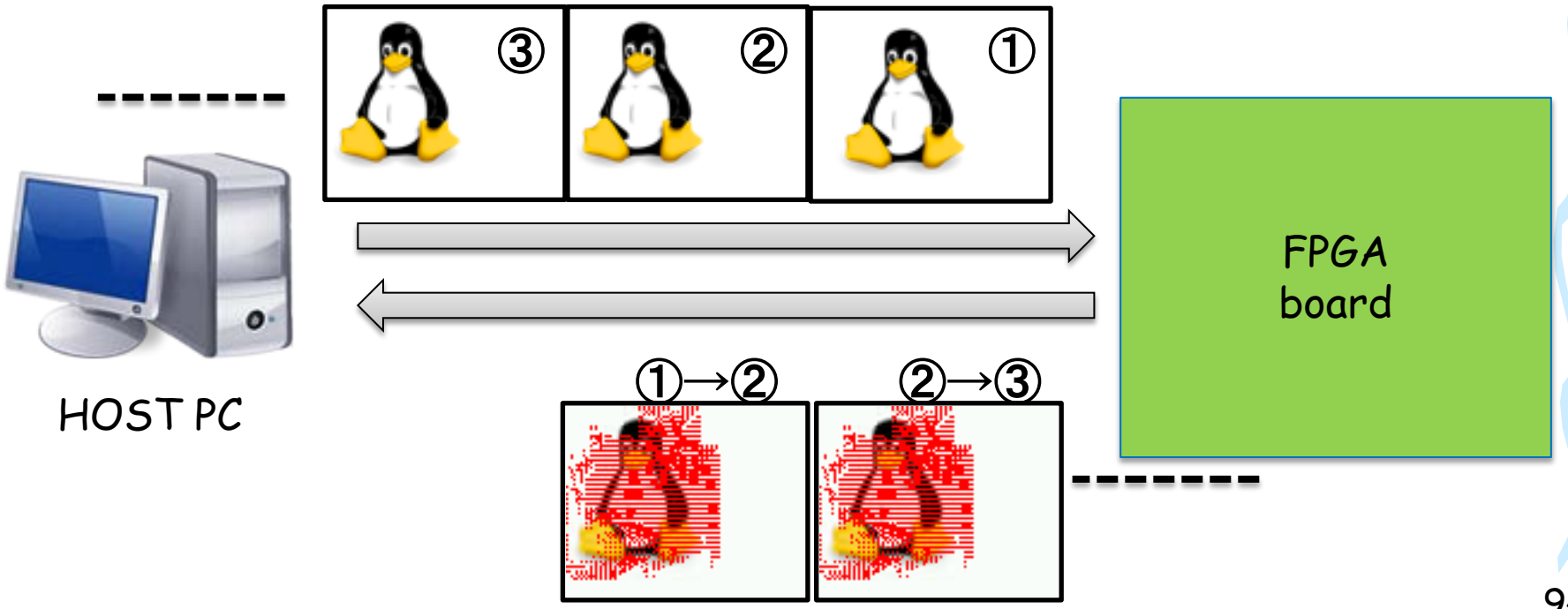
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

コンピュータシステム設計部門のターゲット「ネットワーク画像オプティカルフロー処理」

- 目的
  - ホストPCから送られる複数の画像フレーム間の動きベクトルを、オプティカルフロー処理により計算し、画像に書き入れてホストPCに送り返す。

入力画像
（JPEGに似た形式で圧縮されている）



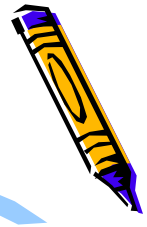
出力画像 （同じ形式で圧縮されている）

概要

- コンピュータシステム部門では，以下の処理を繰り返します
  - ホストから2つの入力画像を受け取る
    - 画像は独自のJPEG圧縮されたフォーマットで渡されます．
    - オプティカルフロー以外の画像関係の処理に関しては，参考文献[1]に記載されている処理をベースに作成しています．

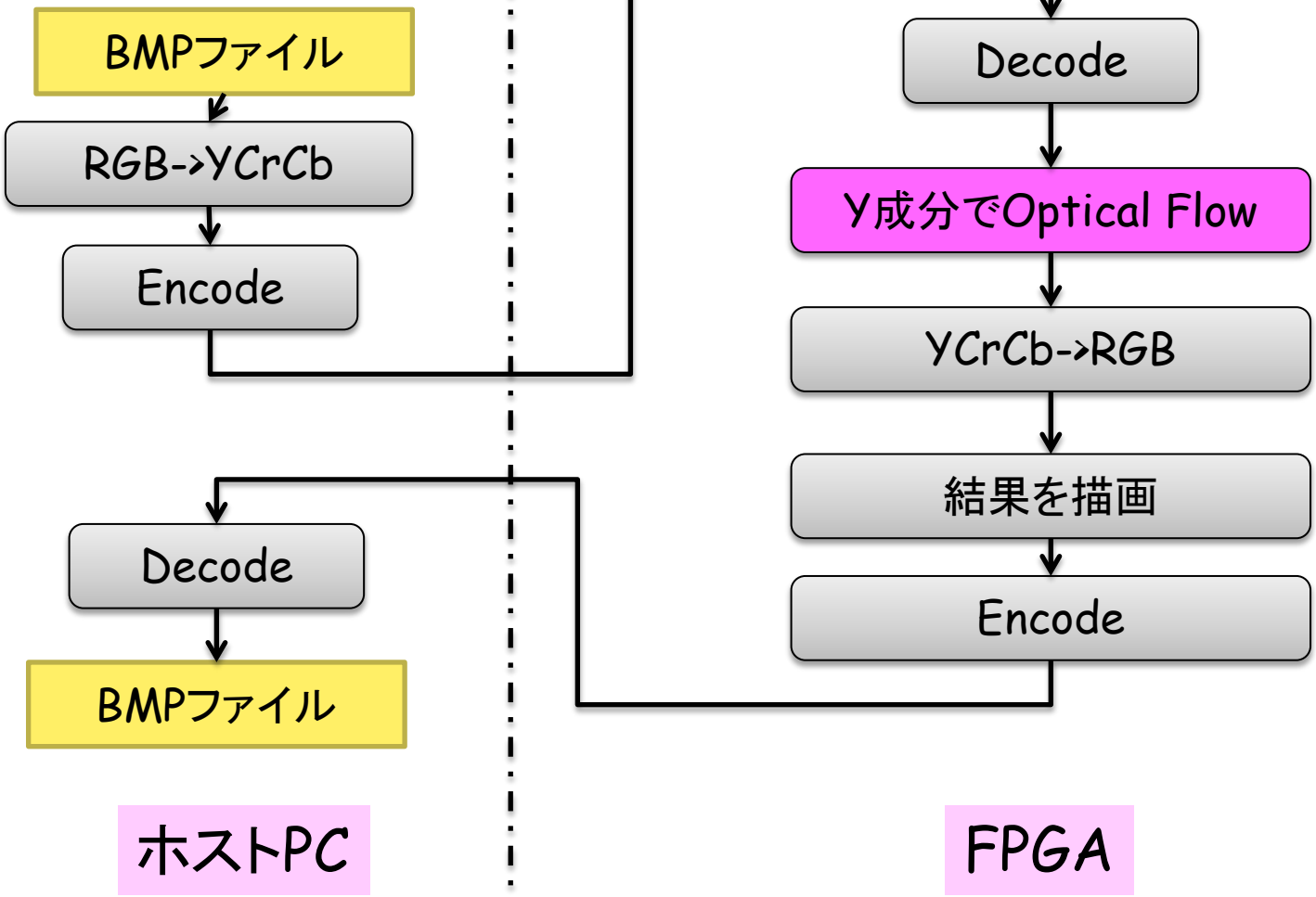
- 参考文献[1]
  - 昌達 慶仁 著，「詳解 画像処理プログラミング」，ソフトバンククリエイティブ株式会社，2008．

SDK内のファイルについて



- Makefile
  - コンテストのホストで動作する処理，および，FPGAボードで実装する処理をPC上で実行できるファイルを生成します．(Unix系のOSを想定しています)
- run_contest.sh
  - コンピュータシステム部門のプログラムとして，下記の一連の処理をPC上で仮想的に実現します．
    - ホストでの前処理(FPGA入力データの生成)
    - FPGAでの処理(入力画像の展開，オプティカルフローの計算，オプティカルフローの描画，出力画像の圧縮)
    - ホストでの後処理(FPGAから受け取った画像の展開)
- 以下を実行する事で，どのような処理が実行されるのか分かると思います．
  - make
  - ./run_contest.sh
- 以降のページで，run_contest.shが行っている処理について説明します

全体の処理フロー


BMPファイル
RGB->YCrCb
Encode
Decode
BMPファイル
Decode
Y成分でOptical Flow
YCrCb->RGB
結果を描画
Encode
ホストPC
FPGA

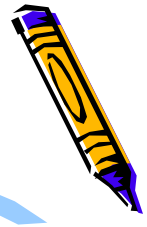
ホストでの前処理

- 2つの入力画像(tux00.bmp, tux01.bmp)をそれぞれYCrCb変換し，その変換後の画像をJPEG圧縮します．
  - 読み込み可能なBMP画像フォーマットについては，参考文献[1]を参考にして下さい
  - YCrCb変換に関しては参考文献[1]にある手法をベースに整数化したバージョンを使用しています．(参考文献[1]では浮動小数点を用いています)
  - YCrCb変換後の画像フォーマットとしては，BMP画像フォーマットのR部分にY，G部分にCr，B部分にCbを格納して保存しています
  - JPEG圧縮に関しては，参考文献[1]では1要素(RGBならRのみ，今回の場合Yのみ)に対しての圧縮処理でしたが，YCrCbそれぞれに対して圧縮処理をするように改良しています．
  - JPEG圧縮では，参考文献[1]では浮動小数点のDCTが使われていましたが，整数化したDCTに変更しました．

FPGAでの処理

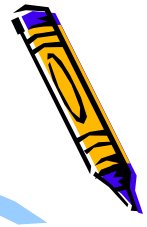
- FPGAでは以下の処理をします．
  - 2つのJPEG圧縮画像（image0.encoded, image1.encoded）をデコード
  - デコードされた画像のそれぞれY成分を用いて，オプティカルフローを求めます．
  - 最初の入力画像（image0）に対してYCrCb->RGB変換（関数名はconvert2bmp）をし，RGB画像を得て，そのRGB画像に対してオプティカルフローの線を描画します
  - オプティカルフローが描画された画像をJPEG圧縮します

ホストでの後処理



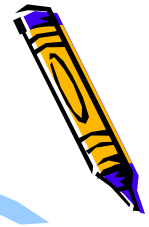
- FPGAからJPEG圧縮された画像を受け取り，展開します．
- 展開後のファイルを，output.bmpとして保存します．

謝辞



- 参考文献[1]の著者である昌達慶仁氏，ならびに，ソフトバンククリエイティブ社には，書籍のプログラムを本コンテストで使用させて頂く事をご快諾して頂きました．おかげさまで，コンピュータシステム部門の競技としてJPEG圧縮，展開を使用した競技にすることができました．この場をお借りしまいて，厚く御礼申し上げます．

改訂履歴



- Ver.2014-06-06 初版（第1回コンテストの内容に追加変更をした）