Document P30

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 2014-07-03

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
  - P61 User Manual of Optical Flow System Reference Design (Atlys board)
  - P62 User Manual of Optical Flow System Reference Design (DE2-115 board)
  - 【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　1/6

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。
   - 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　2/6

3. プロセッサ設計部門の競技内容
   - 以下の4種類のいずれかのFPGAボードが参加可能です。
     - Digilent社 Atlys Spartan-6 FPGA Development Board (Xilinx)
     - Digilent社 Nexys4 Artix-7 FPGA Board (Xilinx)
     - Terasic社 Altera DE2-115 Development and Education Board
     - Terasic社 Cyclone V GX Starter Kit
   - 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
   - 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.
   - 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)

コンテストルール　3/6

4. コンピュータシステム設計部門の競技内容
   - 以下の4種類のいずれかのFPGAボードが参加可能です。
     - Digilent社 Atlys Spartan-6 FPGA Development Board (Xilinx)
     - Digilent社 Nexys4 Artix-7 FPGA Board (Xilinx)
     - Terasic社 Altera DE2-115 Development and Education Board
     - Terasic社 Cyclone V GX Starter Kit
   - 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
   - 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、64KB x 8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）
     - プログラムの起動方法に関するドキュメントと、必要なデータ・プログラム(elf)等 (2014/7/3訂正)

コンテストルール　4/6

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

   基本的には予選に準じます。【後程公開】

コンテストルール　5/6

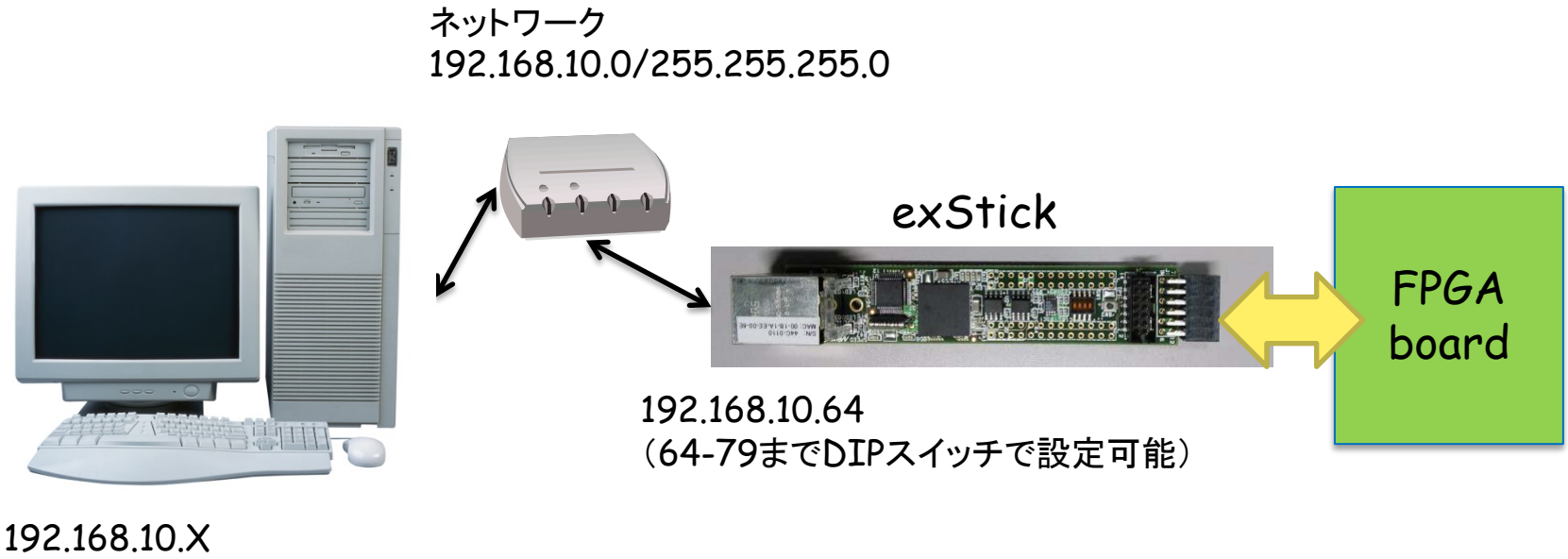
7. 実行時間の計測方法
   - ホストからのUDP/IP(Ethernet)通信にて512KBのデータ（256KBのプログラムと256KBのアプリケーションデータ）の送信開始時刻T1から，計算結果を受け取り，最後にENDという文字列を受け取るまでの時刻T2を実行時間(execution time) として計測します．すなわち，execution time = T2 – T1 です．
   - UDP/IPの通信は100Mbpsで行う事とします．
8. 実行結果の検証
   - 設計した回路は，実行委員会が提供するツールキットと全く同じ結果を出力（通信にてホスト計算機に送信）する必要があります．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データ(2014/7/3訂正)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．

コンテストルール　6/6

9. ホストとの通信方法(2014/7/3追記)
   - プロセッサ設計部門
     - ホストとの通信には，実行委員から配布するイーサネット(100Mbps)・シリアル(1Mbps)変換の小型ボードを用います．
   - コンピュータシステム設計部門
     - ホストとの通信には，実行委員から配布するイーサネット(100Mbps)・シリアル(1Mbps)変換の小型ボード，もしくはFPGAボード上に搭載されているイーサネットコネクタ(RJ-45)を用います．
   - 両部門共通
     - FPGAボードと小型ボードの間の接続に用いるピンは，リファレンスデザインのピン配置に準じます．すなわち，1Mbpsのシリアル通信と，リセット信号がリファレンスデザインと同様に動作する必要があります．
     - 各ボード毎に以下のコネクタを使用してください．
       - Atlys: PMODコネクタ
       - Nexys4: PMODコネクタ（JA）
       - DE2-115：2x20 GPIOコネクタJP5
       - Cyclone V GX Starter Kit：2x20 GPIOコネクタJP9

ハードウェアセットアップ

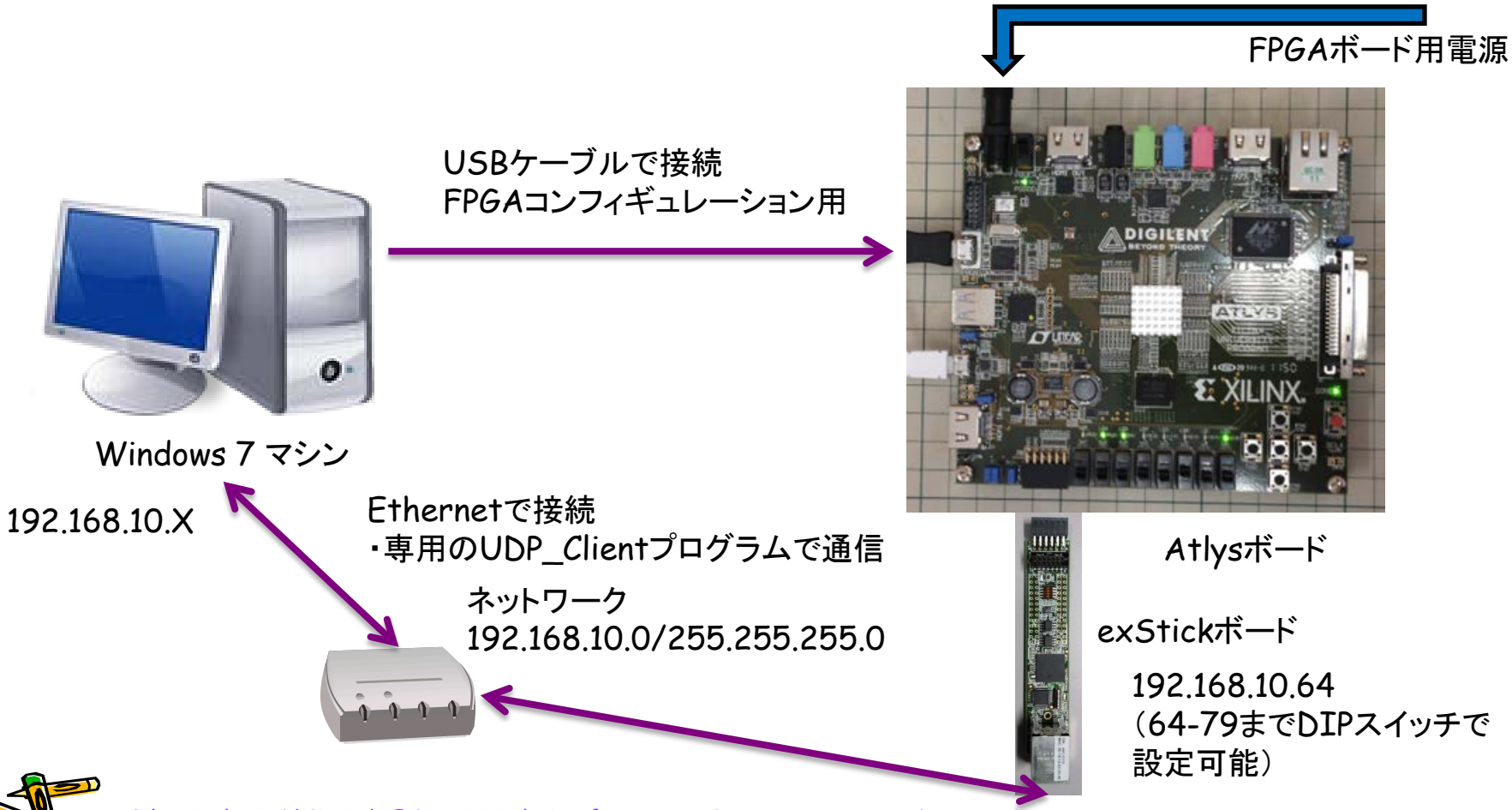
- PCとFPGAボードを、小型ボード(exStick)を介して接続します
  - イーサネット(100Mbps)・シリアル(1Mbps)変換



【注意】
プロセッサ設計部門においては、かならずexStickを介してホストPCと通信します
コンピュータシステム設計部門においては、ボード上のEthernetを用いてもOKです

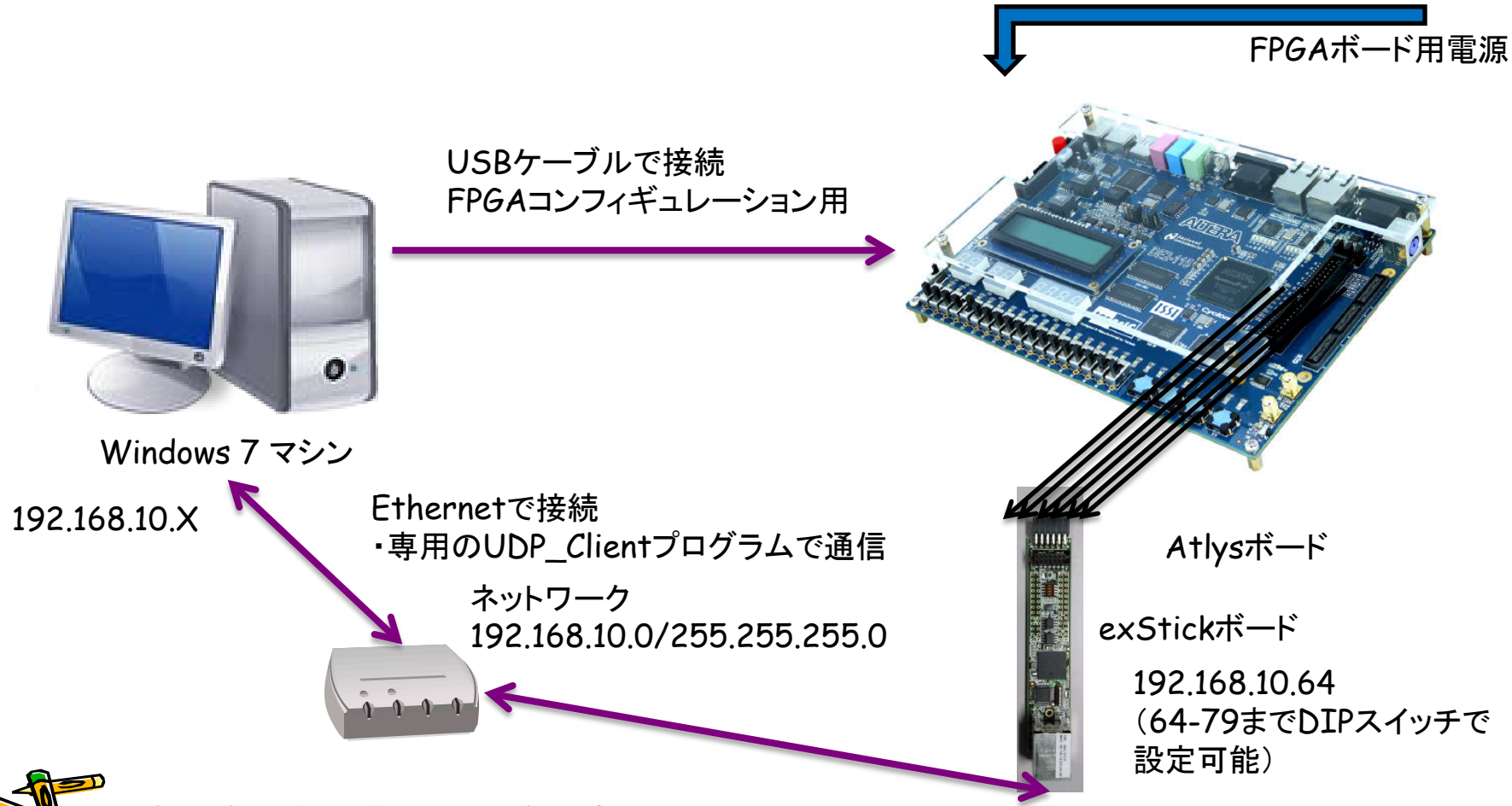
参考）ATLYSボードの開発環境セットアップ

- 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

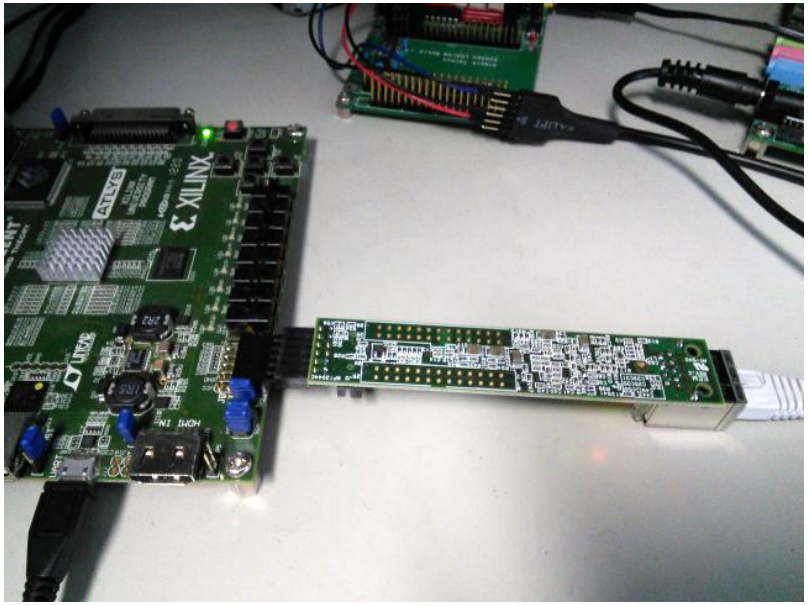


参考）DE2-115ボードの開発環境セットアップ

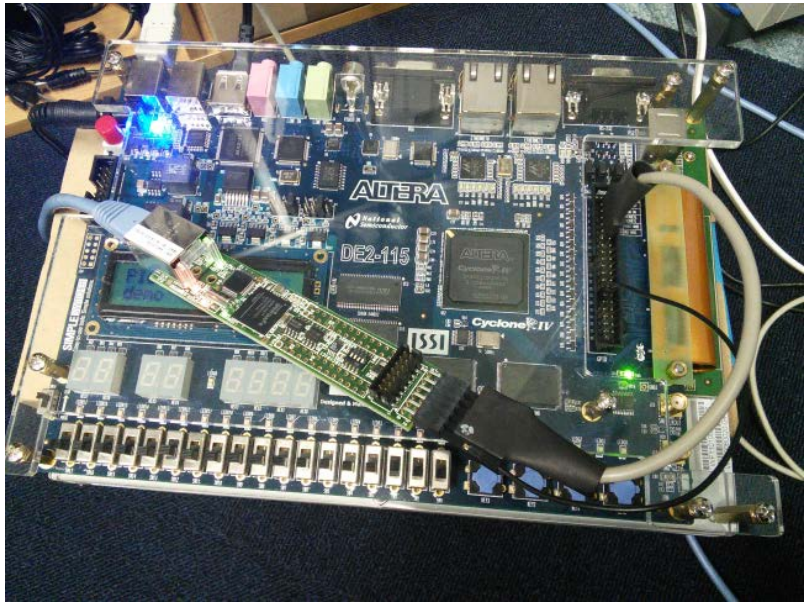
- Windows 7 マシン と DE2-115ボード を USBケーブル で接続
- FPGAボード用電源 を使ってDE2-115ボードに電源供給



exStickボードの接続例



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

標準ピンヘッダ6ピンのケーブル（1ピンずらす）＋電源用ジャンパワイヤ、もしくはジャンパワイヤで1本ずつ接続します

開発環境

- ホストPC
  - OS： Windows7
- 動作確認済みのホストPC上のソフトウェア環境
  - java version "1.7.0_60"
- 既知の問題： CentOS6.5をホストPCとして用いた際に，UDP_Clientが正常に通信できない問題が報告されています。

- exStickBridgeでの動作確認済みスイッチ一覧
  - BUFFALO LSW3-TX-5EPL
  - BUFFALO BSL-WS-G2116M
  - （今後追加します）

設計を始めましょう

- FPGA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・Nexys4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

- FPGA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.

Document P32

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

Reference Design Test Manual (ATLYS board)

コンテスト実行委員会コアチーム

Version 2014-07-03

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．

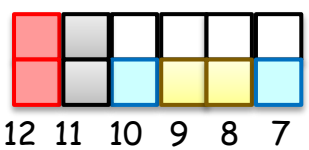
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 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

exStickの接続方法

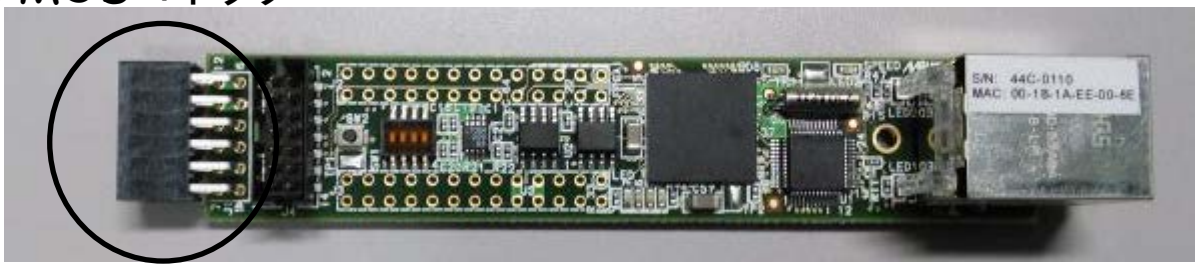
- 入出力ピン
  - exStickのPMODコネクタ(6x2)
    - P7: UART-TX (out)
    - P8: UART-RX (in)
    - P9: RST_X (in)
    - P10: INIT_FPGA_X (out)

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Atlysボードの接続方法

- 両方が長いヘッダピン(6x1)を用いて、AtlysボードのPMODコネクタに、逆さにしたexStickを接続して下さい
- ピンの接続については以下の表の様になります

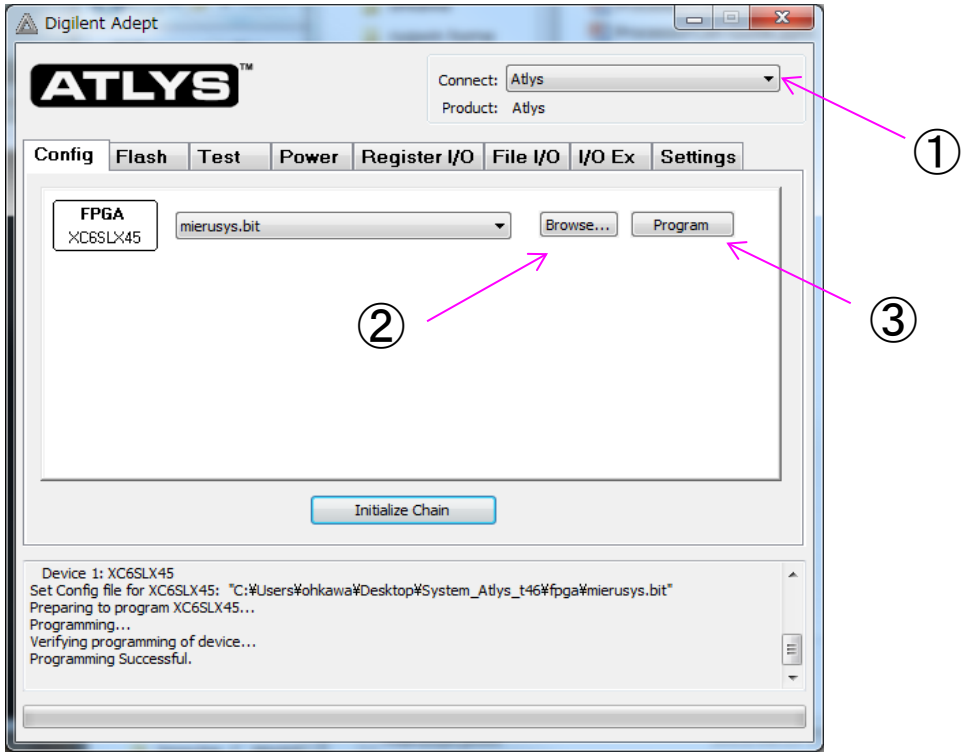
exStickBridge	方向	Atlys
VDD(Pmod12番)	-	PMOD[6]: 3.3V
GND(Pmod11番)	-	PMOD[5]: GND
INIT_FPGA_X(Pmod10番)	→	PMOD[4]
RST_X(Pmod9番)	←	PMOD[3]
UART-RX (Pmod8番)	←	PMOD[2]
UART-TX (Pmod7番)	→	PMOD[1]

PMODコネクタによる接続

・両方が長いヘッダピン(6x1)は、exStick貸出し時に付属します。

Atlysボード
FPGAのコンフィギュレーション

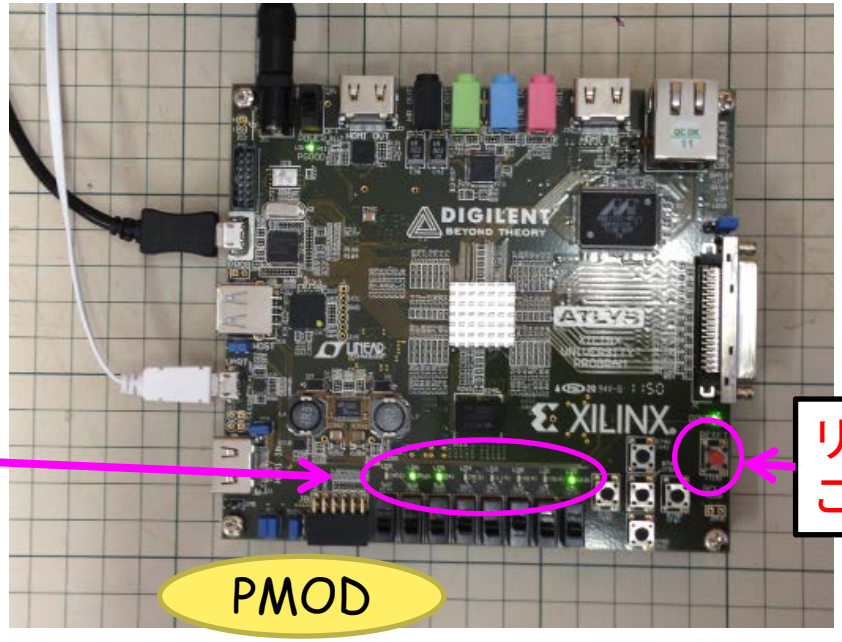
- コンテストホームページから，プロセッサを含むリファレンスデザインの回路データ System_Atlys_t63 をダウンロードしてください．
- System_Atlys_t63 をAdeptで書き込みます
  - ①Atlysボードが認識されていることを確認し、
    ②Browseで先程のbitファイルを選択して、
    ③Programで書き込みます



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）

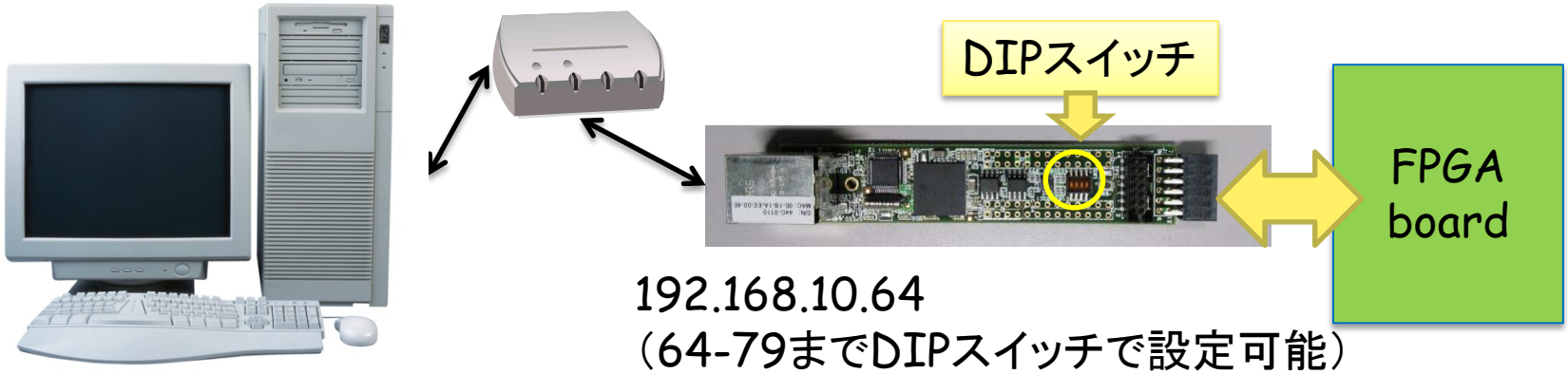
- 以下のファイルを用いて、exStickBridgeの動作検証を行います。
  - exStickBridge_v051.bit.zip
  - System_Atlys_t63.bit.zip
  - UDP_Client_v06.jar
  - SDK.1.1.1.tgz

- 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
  - System_Atlys_t63.zip (Xilinx ISE14.7プロジェクト)
  - UDP_Client_v06.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



192.168.10.64
（64-79までDIPスイッチで設定可能）

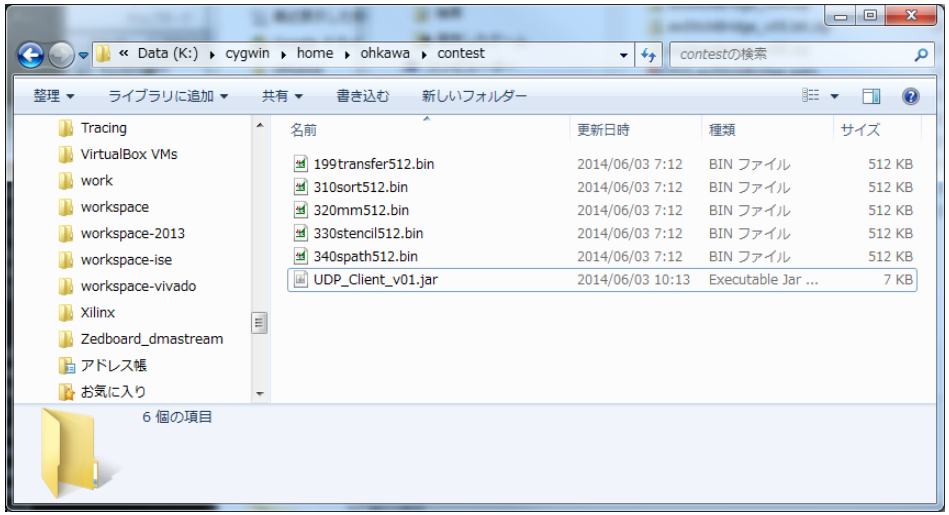
192.168.10.X

MACアドレスもDIPスイッチで変化します

exStickBridgeを用いた リファレンスデザインの動作検証 （3）

- SDK-1.1.1.tgzを展開します
- bin/xilinxディレクトリにある以下のファイルを、UDP_Client_v06.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。

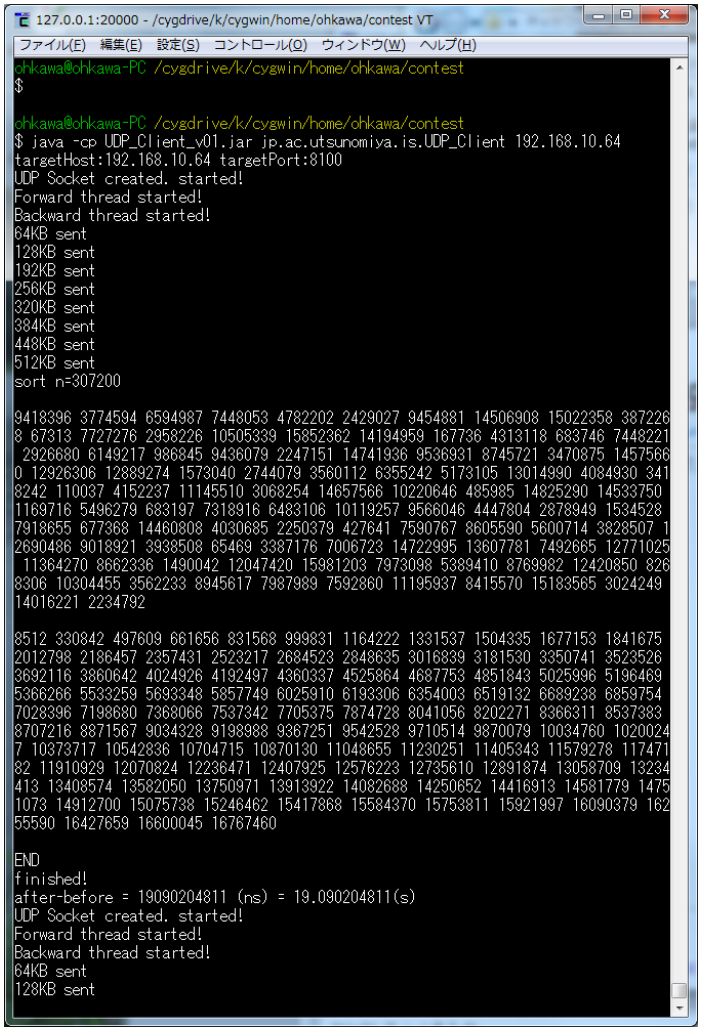
```
% java -cp UDP_Client_v06.jar jp.ac.utsunomiya.is.UDP_Client 192.168.10.64
```



exStickBridgeを用いた リファレンスデザインの動作検証 （4）

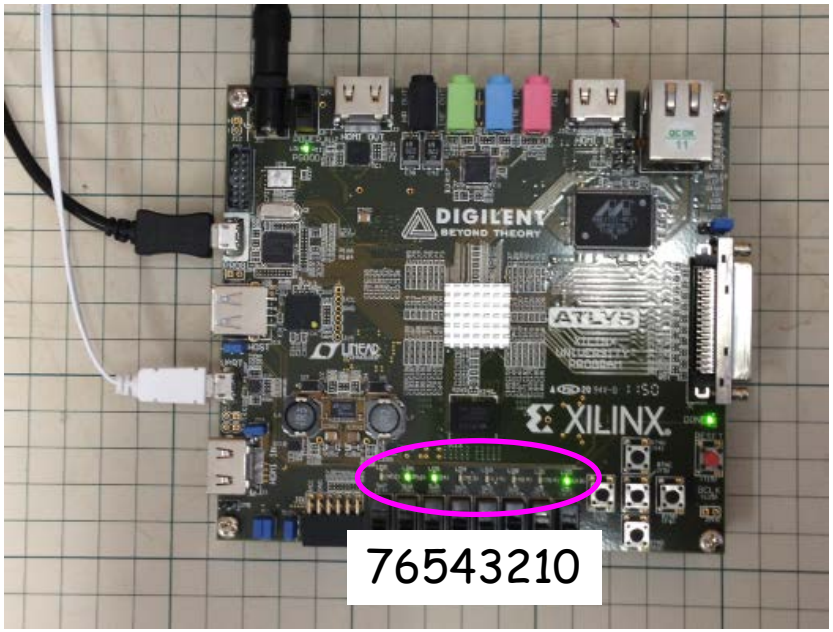
- 実行
  - データ転送は8秒程度
  - 4つのアプリが順次実行される
  - アプリ開始時にFPGAボードが毎回リセットされる（リセットのために16バイトのUDPパケットを送信）
- 以下のログファイルが出力
  - ToUDP: 送信したデータ
  - FromUDP: 受信したデータ
- UDP通信のエラーが無いか確認するには、199transfer512.binを使用可能
  - FPGA上のプログラムが、データ領域のデータ全てを、PCに送ります
  - つまり199transfer512.binの後半256KB(199transfer.dat)とFromUDP.binが一致する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

```
% java -cp UDP_Client_v06.jar
jp.ac.utsunomiya.is.UDP_Client 192.168.10.64
199transfer512.bin
```



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
```

Document P40

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 2014-07-03

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

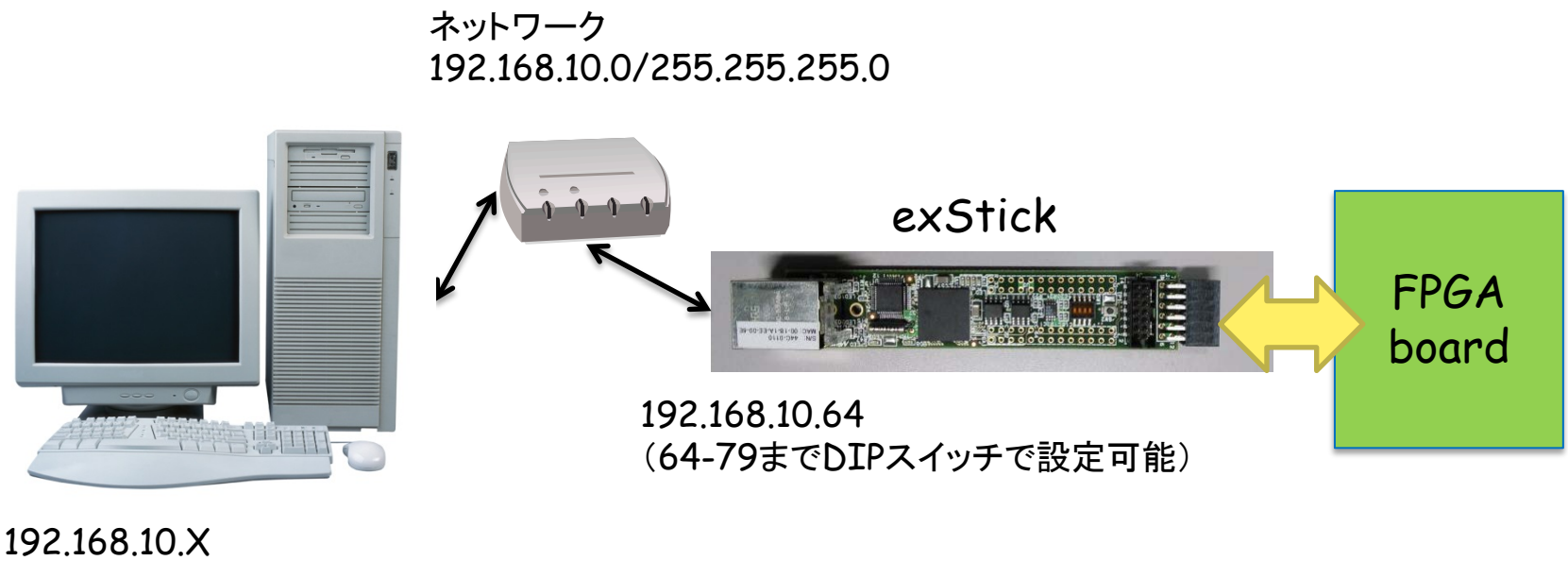
- 本ドキュメントを読む前に，本コンテストの「General Guide and Contest Rule」を熟読されている事を前提とします．

- 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）

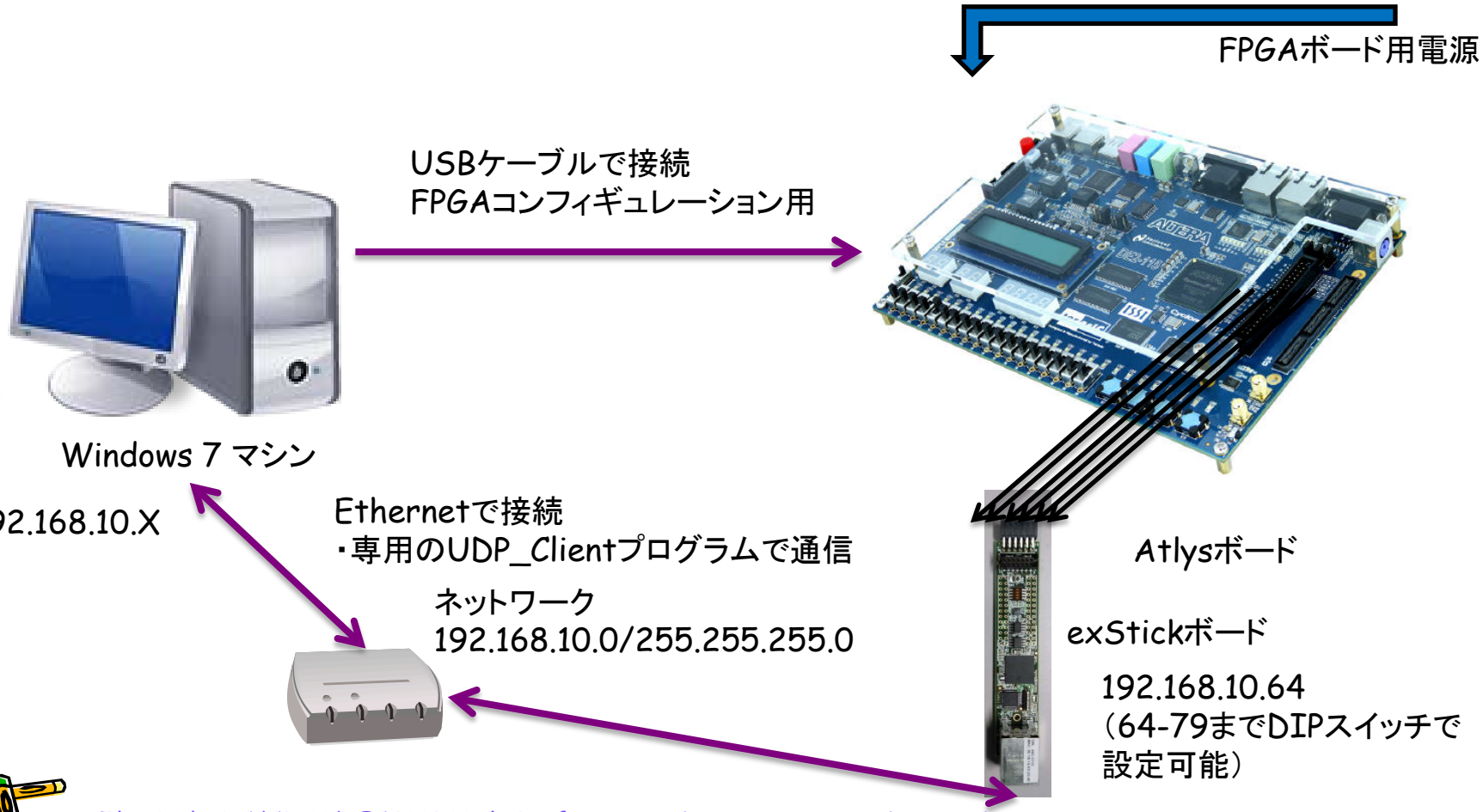
ハードウェアセットアップ

- 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 2014-07-03

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．

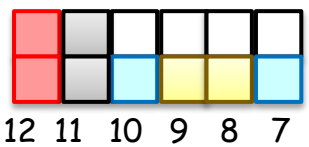
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 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の接続方法

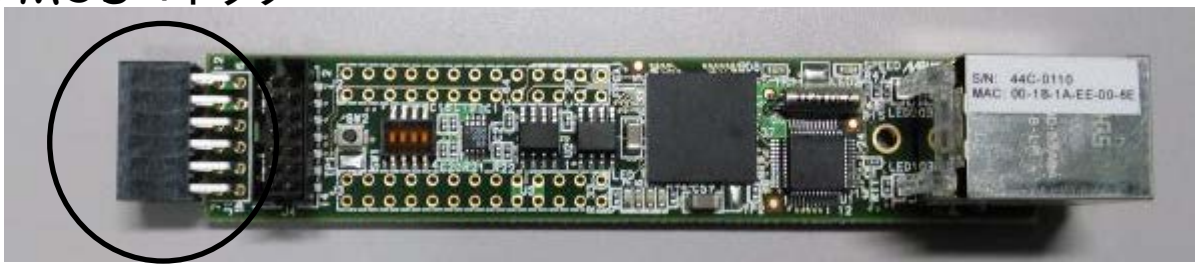
- 入出力ピン
  - exStickのPMODコネクタ(6x2)
    - P7: UART-TX (out)
    - P8: UART-RX (in)
    - P9: RST_X (in)
    - P10: INIT_FPGA_X (out)

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)

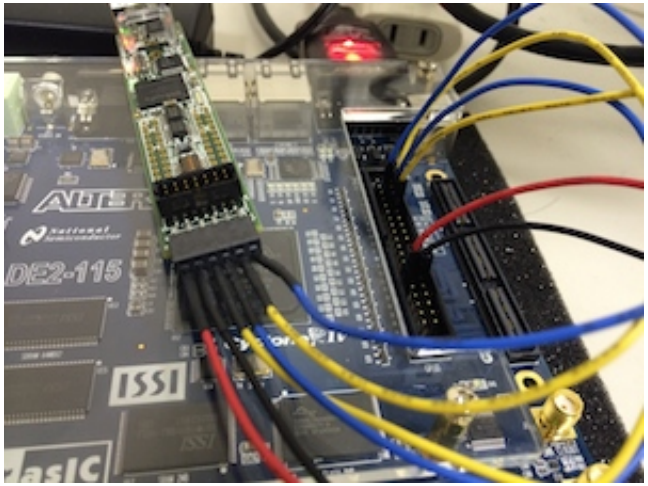
PMODコネクタ



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以下の表の様にします

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

・標準ピンヘッダ間を繋ぐケーブル(5x1や6x1)があれば使用可能です。
・両方が長いヘッダピン(6x1)は、exStick貸出し時に付属します。

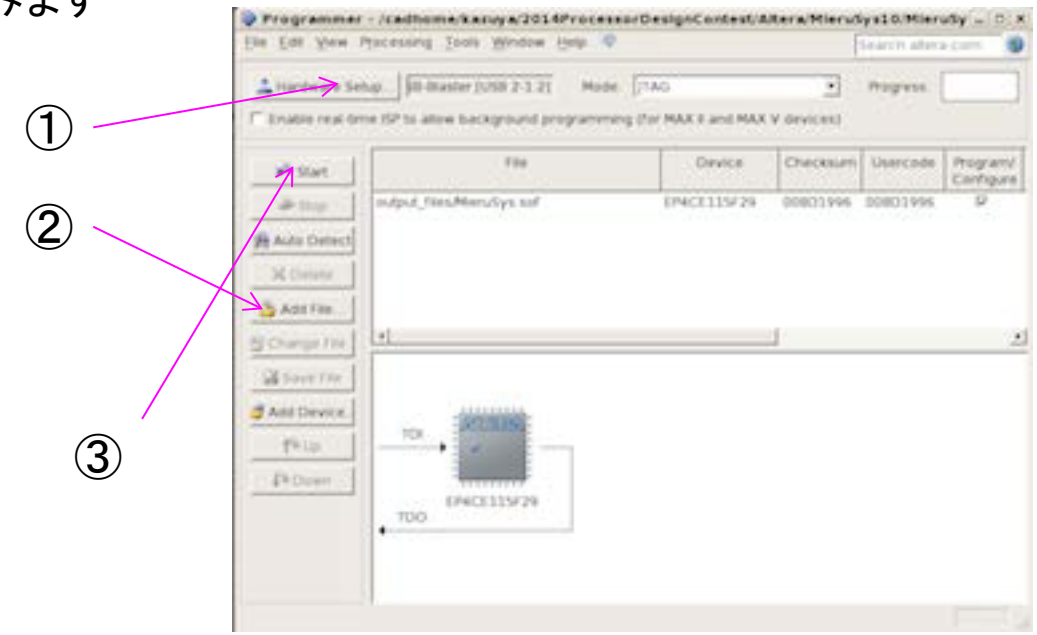
exStickBridgeを用いた
リファレンスデザインの動作検証に必要なファイル

- 以下のファイルを用いて、exStickBridgeの動作検証を行います。
  - exStickBridge_v051.bit.zip
  - MieruSys10.sof.zip
  - UDP_Client_v06.jar
  - 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
  - MieruSys10.zip (Aletra Quartus 13.1プロジェクト)
  - UDP_Client_v06.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

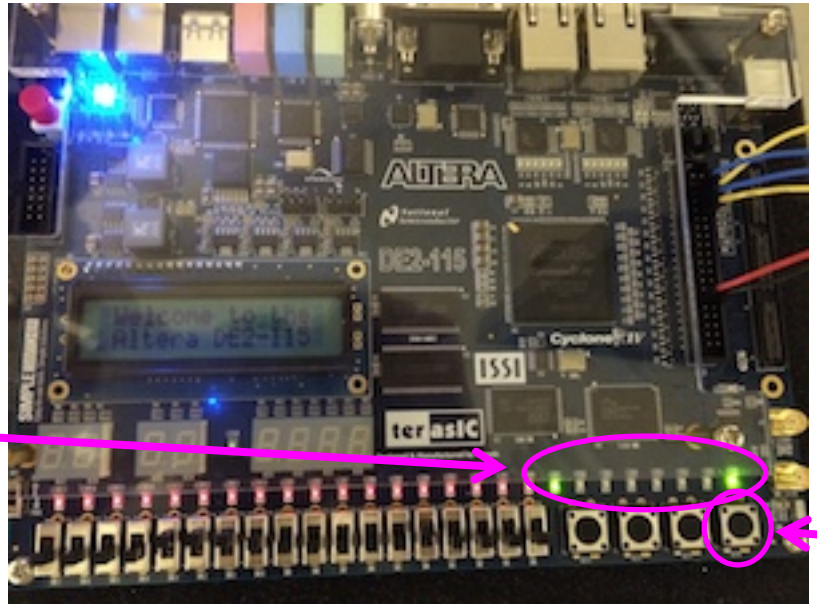
- コンテストホームページから，プロセッサを含むリファレンスデザインの回路データMieruSys10.sof.zipをダウンロードしてください．
（解凍するとMieruSys.sofというファイルになります）
- De2-115ボードの電源をいれます．
- MieruSys.sofをQuartusから起動できるProgrammerで書き込みます
  - ①Hardware SetupでUSB Blasterが選択されている事を確認
  - ②Add fileでMieruSys10.sofを選択して追加します
  - ③Startで書き込みます



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．

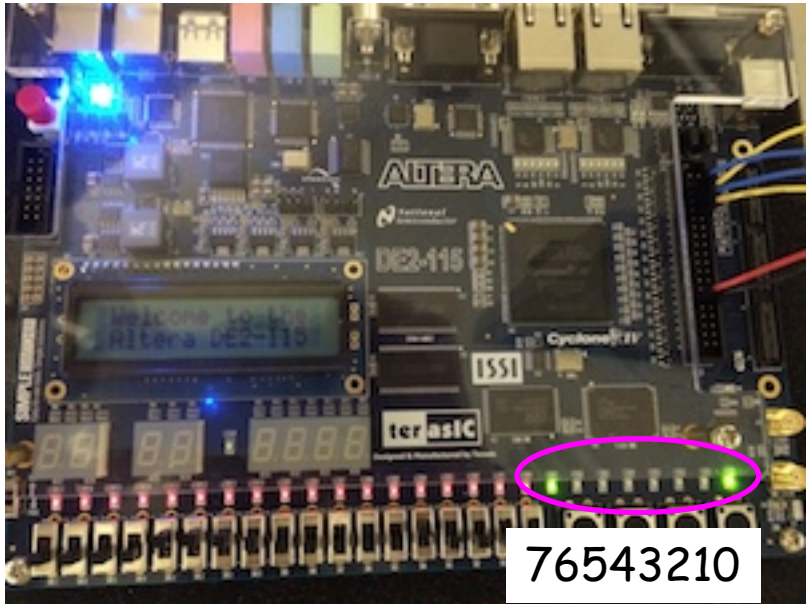
LEDはこちら
LD0：点灯
LD7：点滅
で正常動作です



リセットボタンは
こちら

参考:LEDの説明

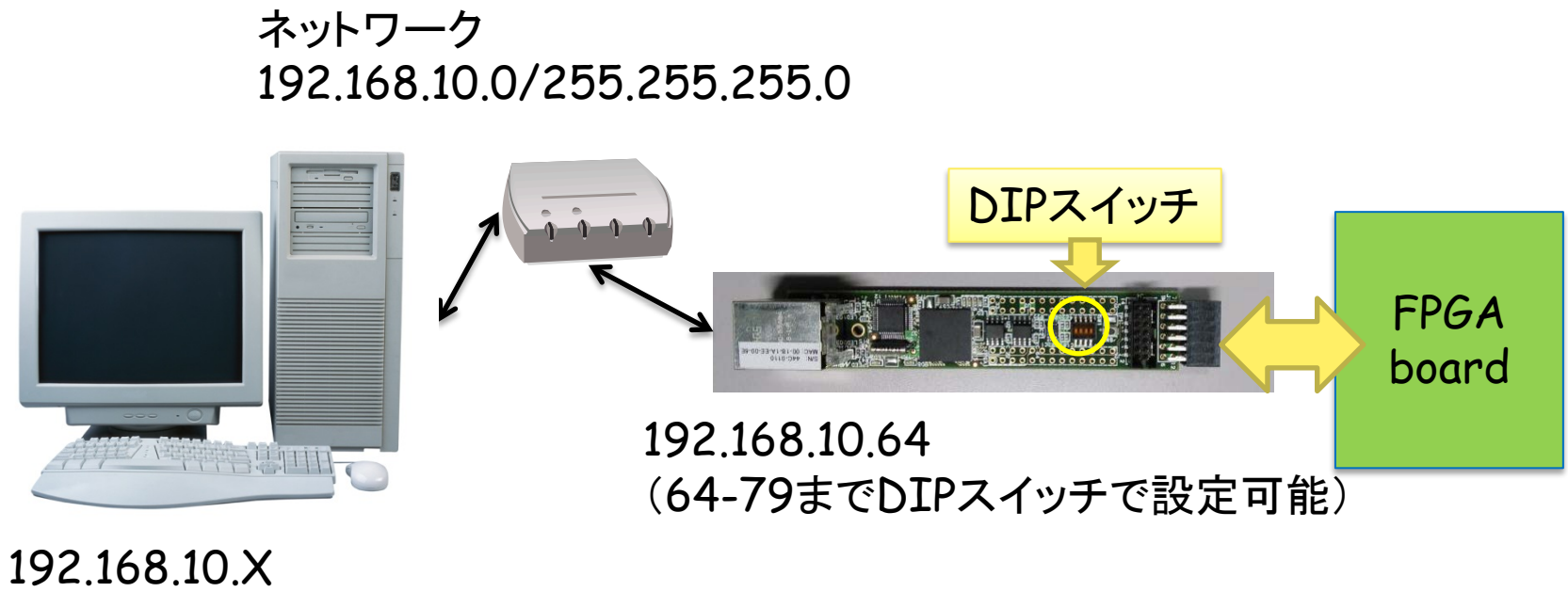
- リファレンスデザインにおける下図の赤枠で囲んだ各LEDは以下の状態を示します.



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）

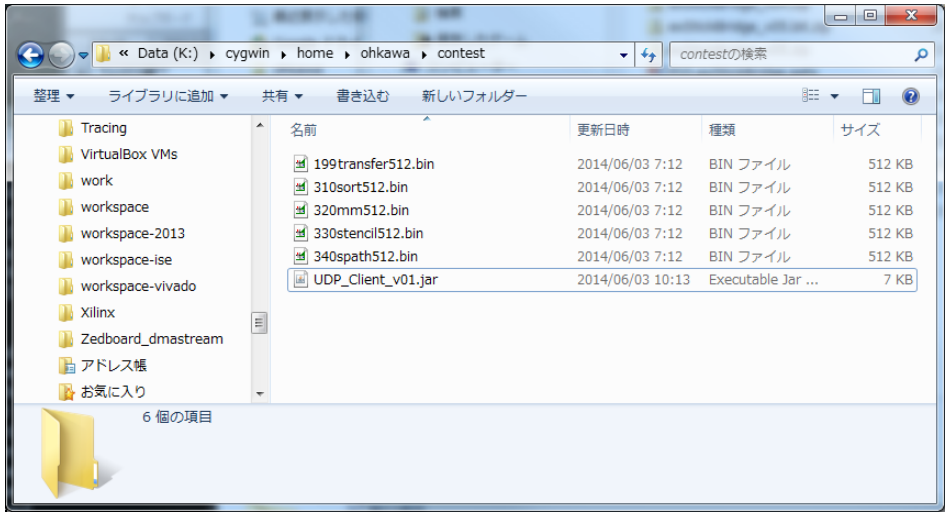
- 動作検証のためのネットワーク環境
  - Pingコマンドで応答を確認可能
  - 動作確認済みのスイッチについては別表を参照



exStickBridgeを用いた リファレンスデザインの動作検証 （2）

- SDK-1.1.1.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_v06.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。

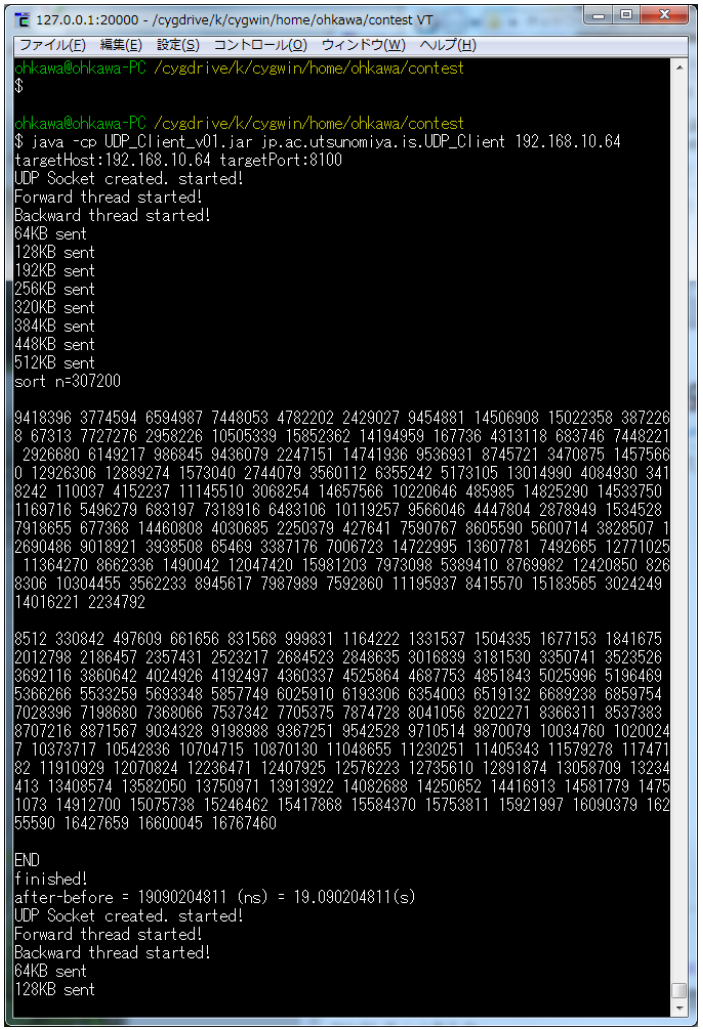
```
% java -cp UDP_Client_v06.jar jp.ac.utsunomiya.is.UDP_Client 192.168.10.64
```



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_v06.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 (ALTERA DE-2 board)

コンテスト実行委員会コアチーム

Version 2014-07-03

このドキュメント

- このドキュメントでは，Aletara DE2-115ボード用のリファレンスデザインに含まれるシステム構成について説明します．
- また，Altera Quartusを用いて，リファレンスデザインの回路ファイル（sofファイル）を生成する方法を示します．

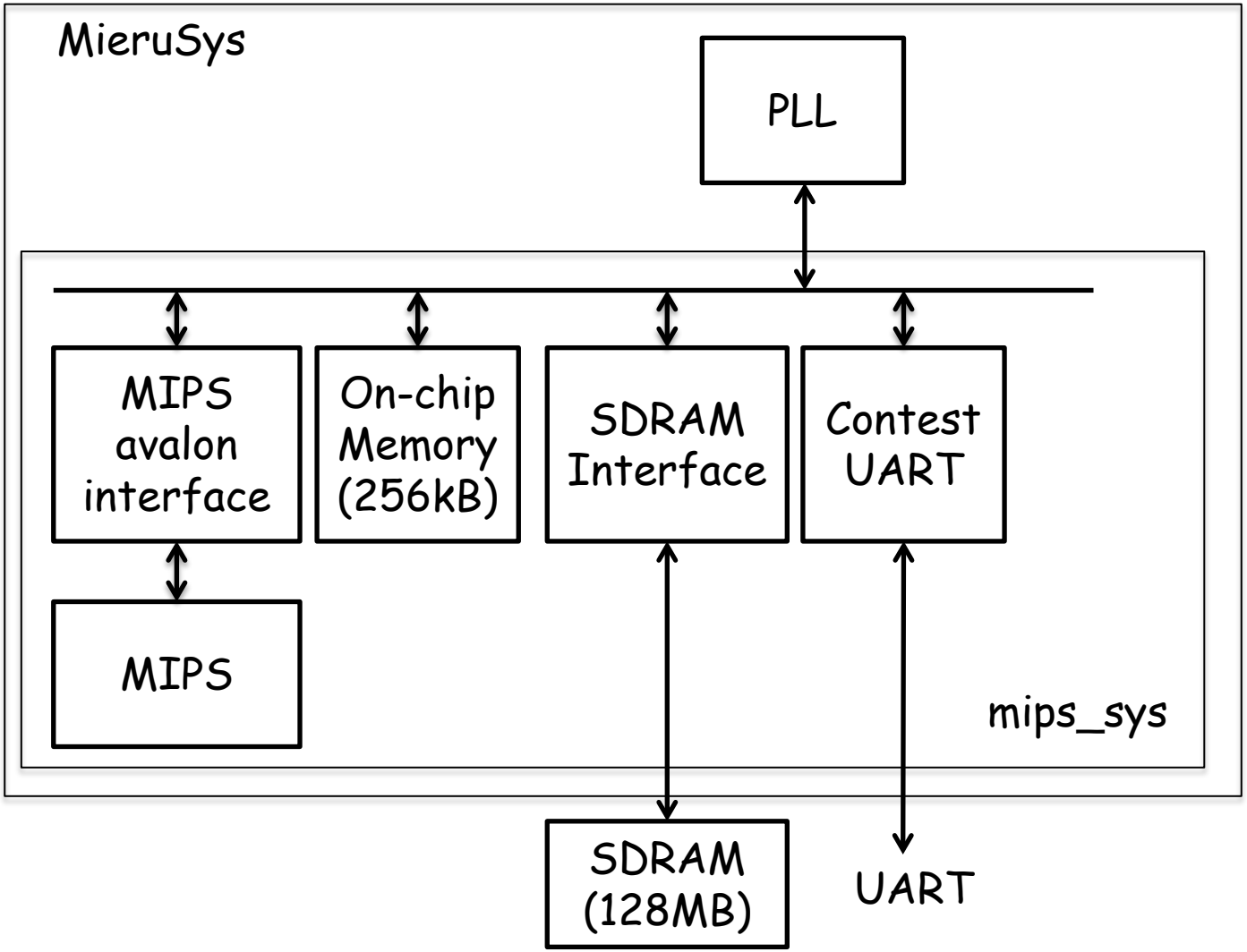
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 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

最初に

- Altera版MieruSysはXilinx版MieruSysと機能的には同じですが，異なる所が多々ありますので，注意して下さい．

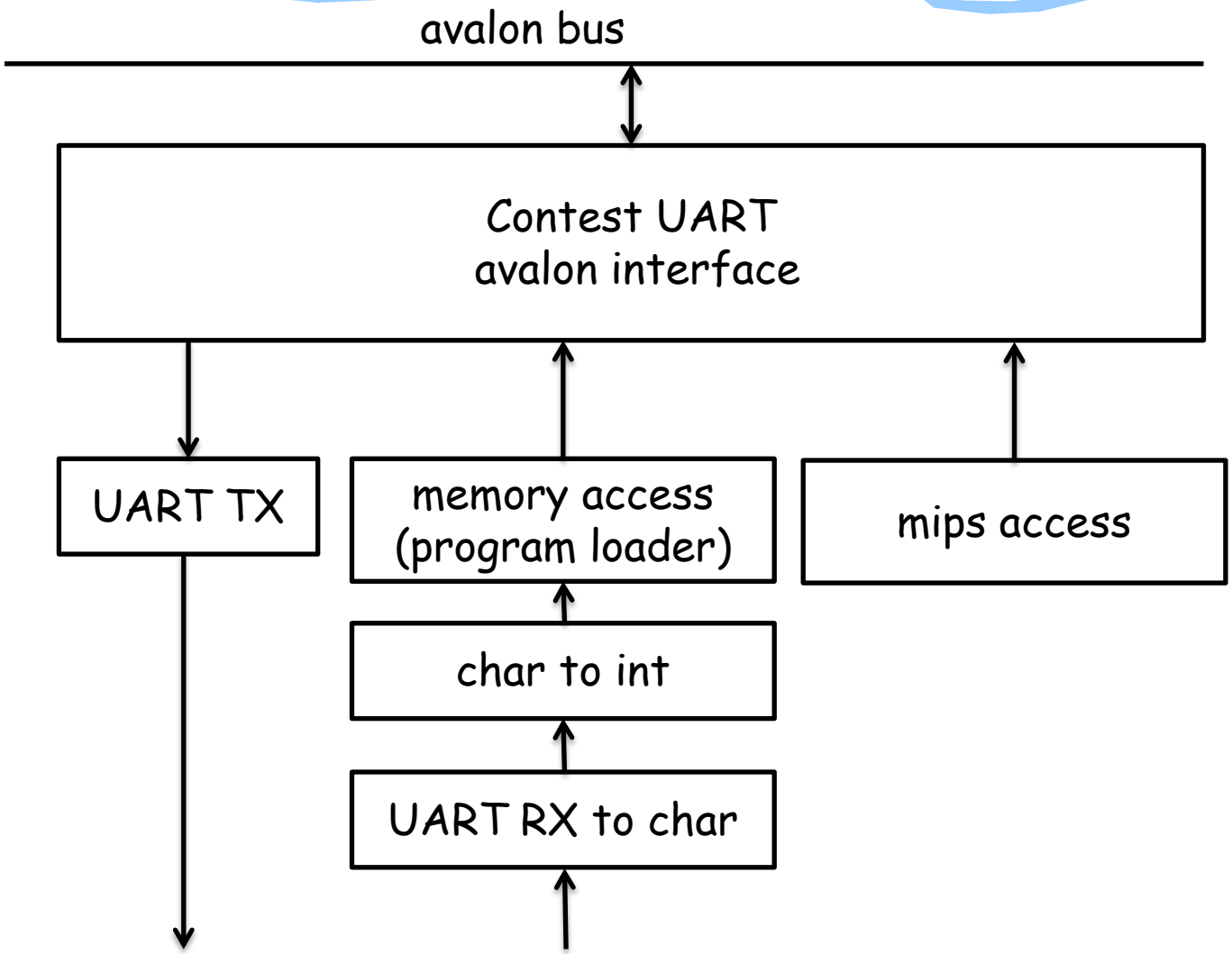
MieruSysのブロック図



各機能の役割

- PLL
  - 40MHzのクロックを供給
- On-chip Memory（256kB）
  - MIPSの命令メモリとして使用
- SDRAM
  - MIPSのデータメモリとして使用.
- Contest UART
  - データ転送用のインターフェース.
  - UARTから受信したデータを命令メモリ，データメモリに保存する
  - MIPSからはUARTの送信ポート(TX)しかアクセスできない.
- MIPS
  - MIPS本体の記述はAtlysボード用のMIPSと同じ.

Contest UARTの詳細



Contest UARTの詳細

- contest UART avalon interface
  - プログラムローダーやUART TXに対してavalonバスとのinterfaceになる
- memory access
  - モジュール名はmemory aceessとなっているがプログラムローダーとして働く．つまり受信データの先頭256kBを命令メモリに書き込み，残り256kBをデータメモリに書く
  - char to int
    - UART RXを8ビットのデータにしたものを32ビットのデータにしてmemory accessモジュールに渡す
  - UART RX to char
    - シリアルで入力されるUART RXのデータを8ビットのデータに変換する
- mips access
  - データ受信前にMIPSをリセット状態でkeepさせ，データ受信後にそれを解除する

メモリマップ

メモリアドレス		
開始アドレス	終了アドレス	
0x0000_0000	0x0003_FFFF	on-chip memory
0x0004_1000	0x0004_1000	UART_TX
0x0004_1020	0x0004_102F	MIPS
0x0800_0000	0x0FFF_FFFF	SDRAM

補足

- SDRAMの領域は0x0800_0000番地から使用出来るのですが，
  リファレンスデザインでは0x0c00_0000番地から使用しています．
  （SDRAM領域の0x0800_0000番地からの領域をデバッグに用いていたため）

リファレンスデザインの作成について

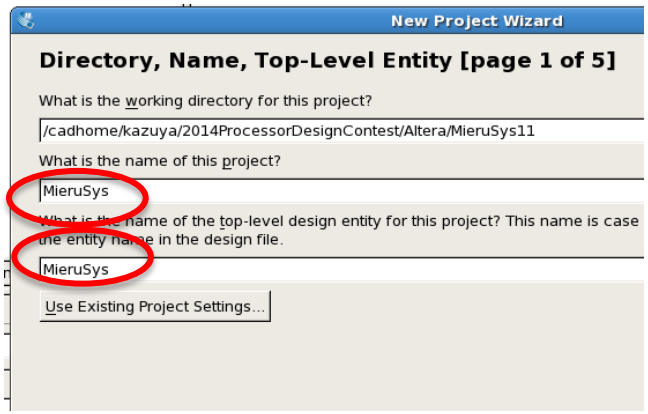
- 次ページ以降で作成するリファレンスデザインの作成方法では，Verilog-HDLの記述は既に完成しているものを使用するとします．
- 本ドキュメントで参照するファイルは以下のものです．
  - MieruSys10.tar.gz
    - DE2-115ボード用プロセッサ設計部門リファレンスデザインのプロジェクトファイル
  - DE2-115.qsf
    - DE2-115ボード用ピン設定ファイル．下記のURLから取得できます http://www.altera.com/education/univ/materials/boards/de2-115/unv-de2-115-board.html

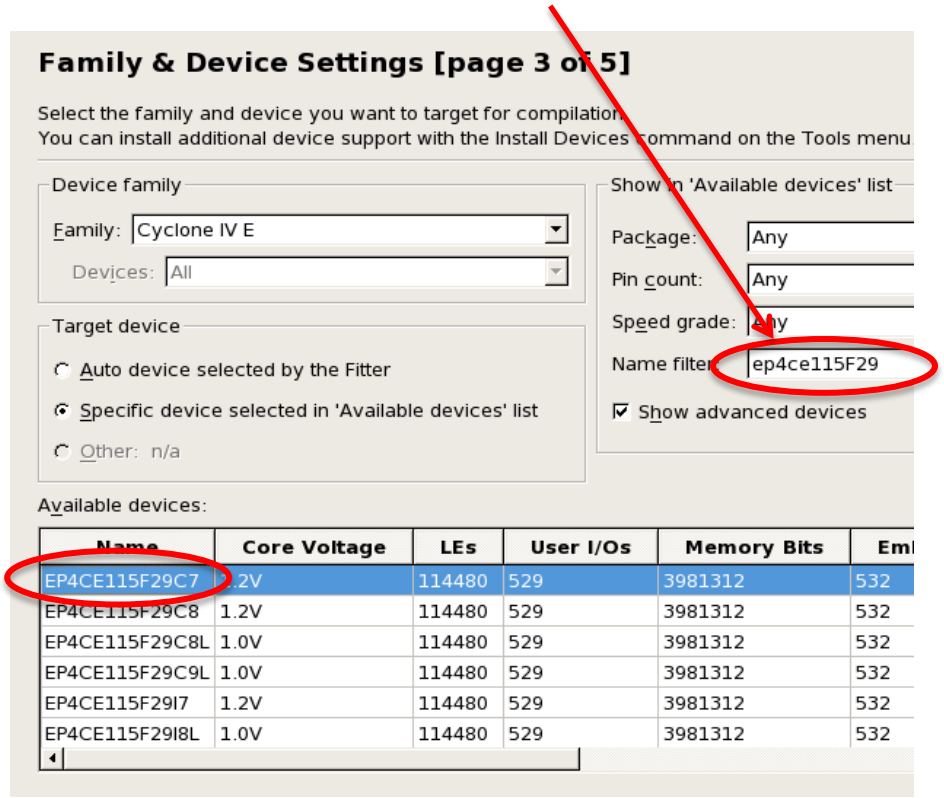
新規プロジェクトの作成

1. プロジェクトを置くディレクトリは新規に空のディレクトリを作成する．ここではMieruSys11とします
2. 新規に作成したディレクトリでQuartus を起動する
3. File -> New Project Wizardを選択
   1. プロジェクト名をトップモジュール名（MieruSys）にする（次ページ左写真）．その後Next
   2. Add fileでは何も追加しない
   3. デバイス名はCyclone IV E EP4CE115F29C7を選ぶ（次ページ右写真）
   4. EDA toolの設定でSimulationツールとしてModelsimを選んでいる場合はFormatをVerilog HDLにする
   5. その他はデフォルトでNextを押し，finishまでいく

新規プロジェクトの作成

Name filterでデバイス名の候補をfilteringできる

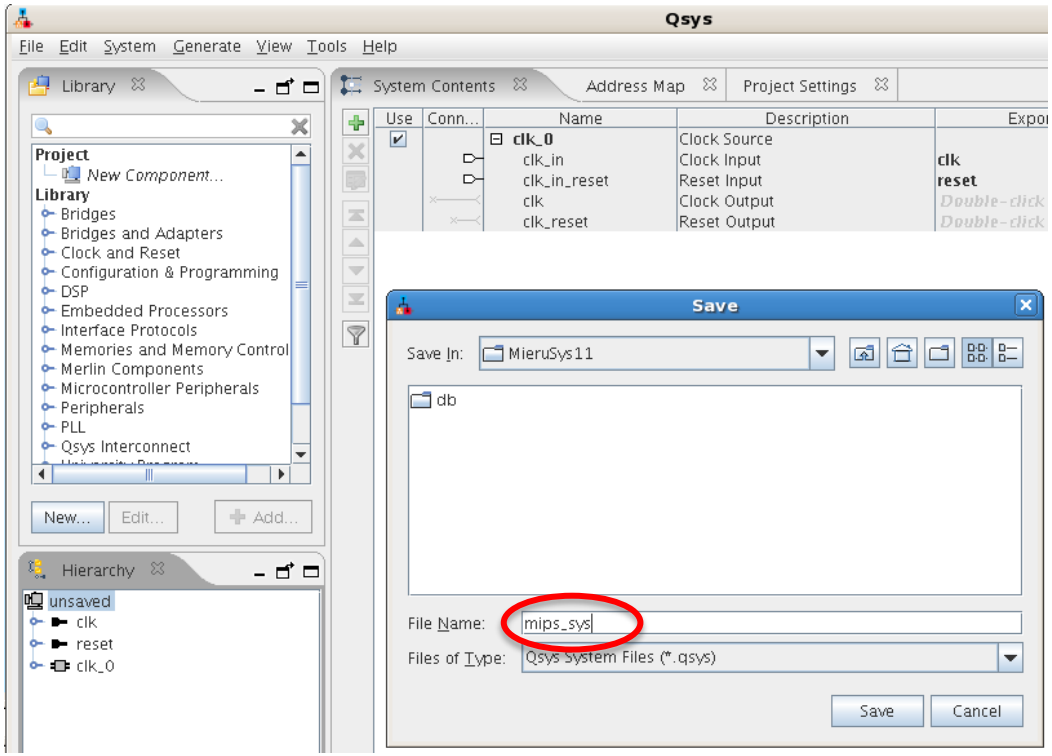




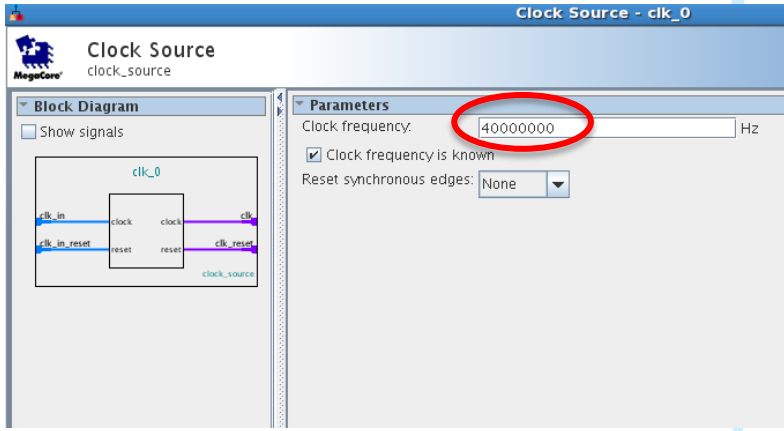
Qsysの起動と外部クロックの設定

- ここではQsysのシステムとしてmips sysを作成します.
- Tools -> QsysでQsysを起動する
  （以下Qsysでの操作です．）
- File -> Save でmips_sysという名前をつけて保存
- clk_0を右クリックして，Edit
  - そこで40MHzと設定

Qsysの起動と外部クロックの設定



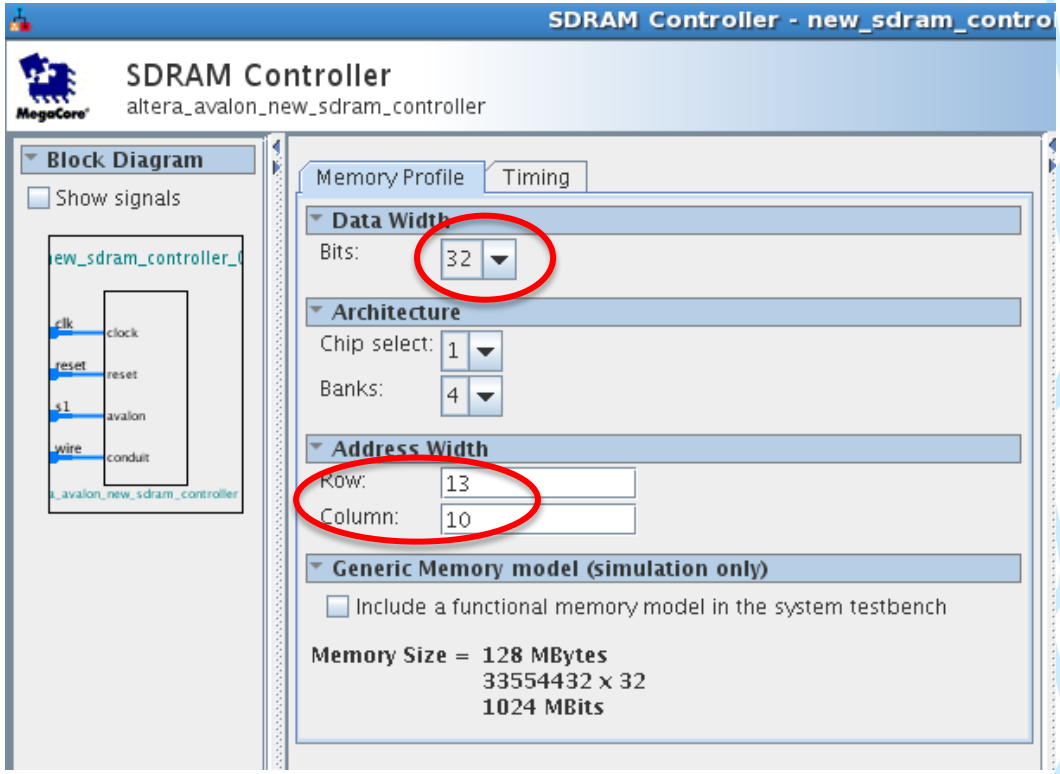
ファイルの保存



クロックの設定

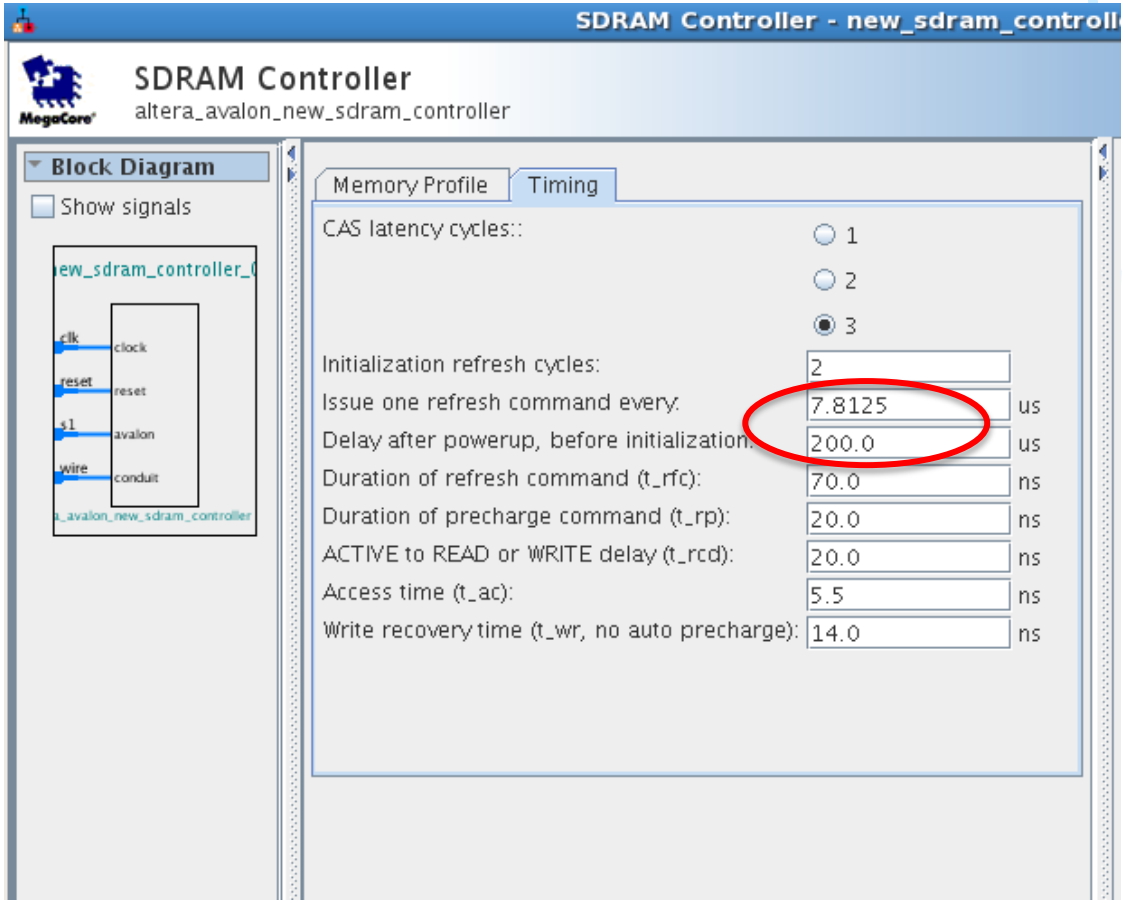
SDRAM Controllerの追加

- Library から Memories and Memory Controllers->External Memory Interfaces-> SDRAM Interfaces -> SDRAM Controllerを選択し，Add
- 現れたWindowで以下を設定する
  - Data Width: 32
  - Address Width
    - Row: 13，Colum: 10



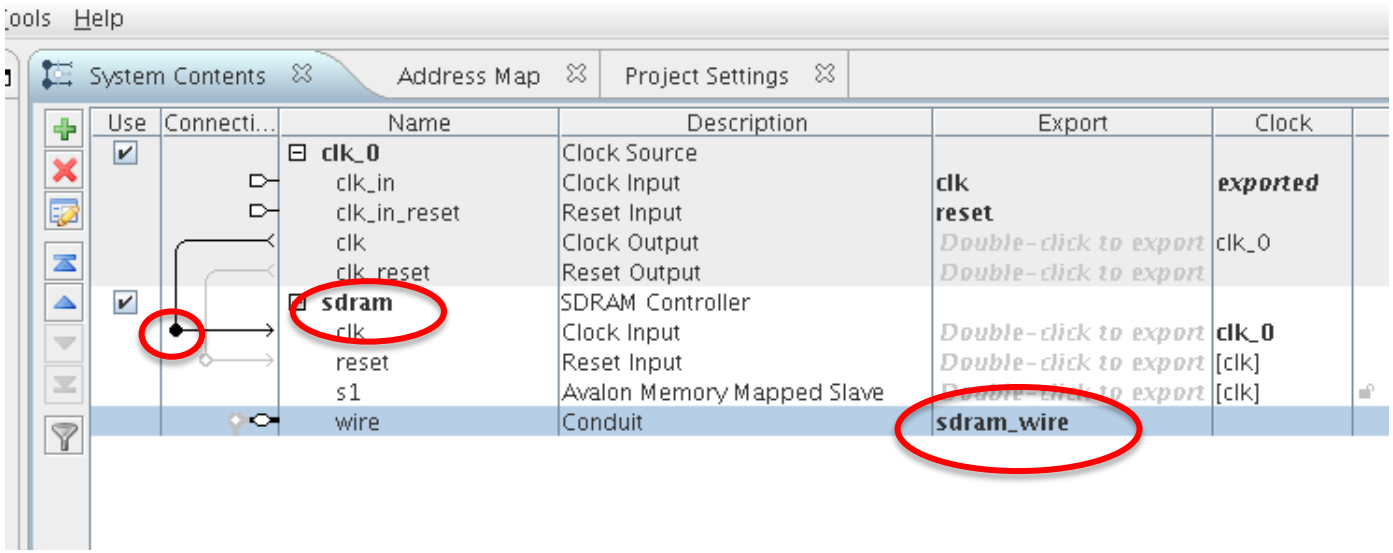
SDRAM ControllerのTiming設定

- Timingタブをクリック
  - Issue one refresh command every を7.8125us
  - Delay after powerupを200us
- Finishボタンをクリック



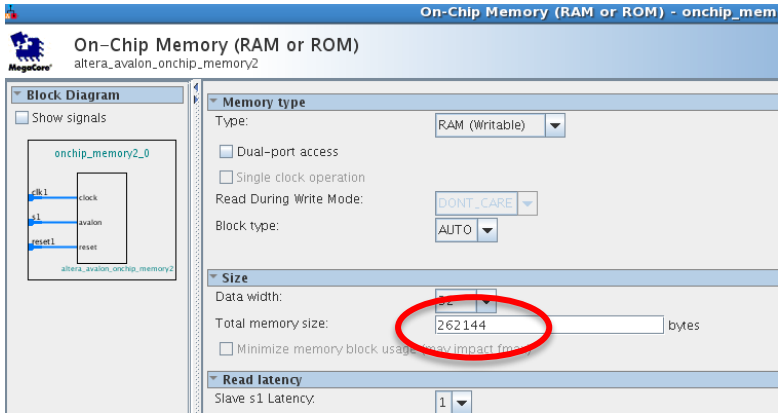
SDRAM Controllerのclk配線など

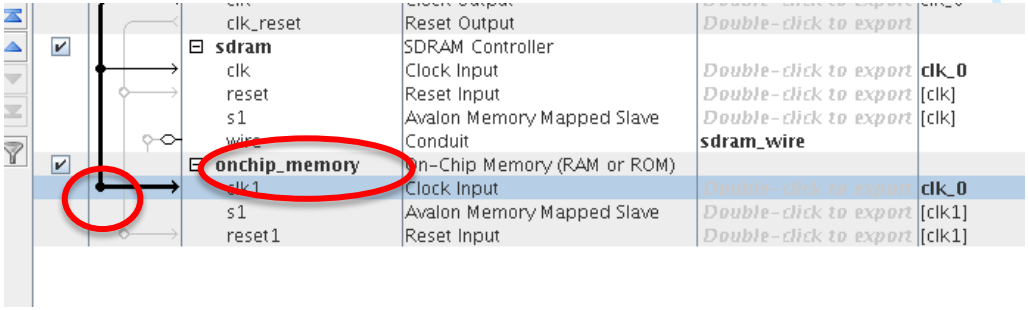
- 追加したsdram controllerの名前をsdramに変更（new_sdram_controller_0という名前の上で右クリックを押して現れるメニューからRenameを選択）
- クロックモジュールからclkをsdram controllerのclkに配線
- sdramのwireをExportするようにExport欄をダブルクリック.（Export名をsdram_wireとする）



On-Chip memoryの追加

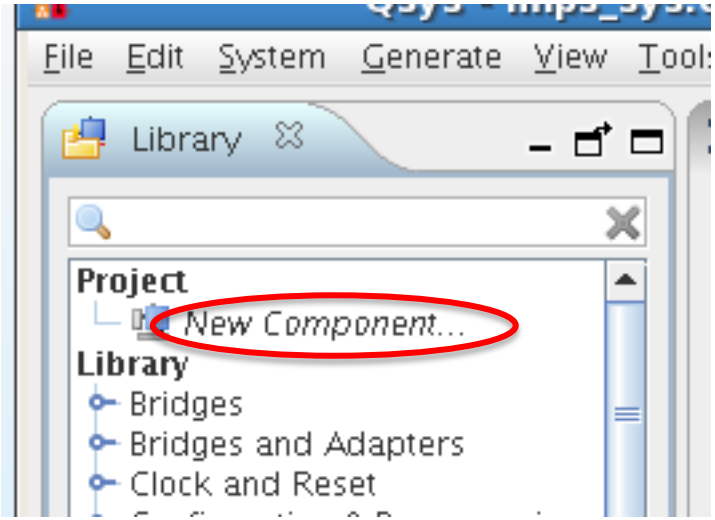
- Library から Memories and Memory Controllers-> On-Chip -> On-Chip Memory(RAM or ROM)を選択し，Add
  - Total Memory Sizeを262144とする(256k)
  - Finishをクリックする
- 名前をonchip_memoryに変更
- onchip_memoryのclk1をclk_0のclkと接続する

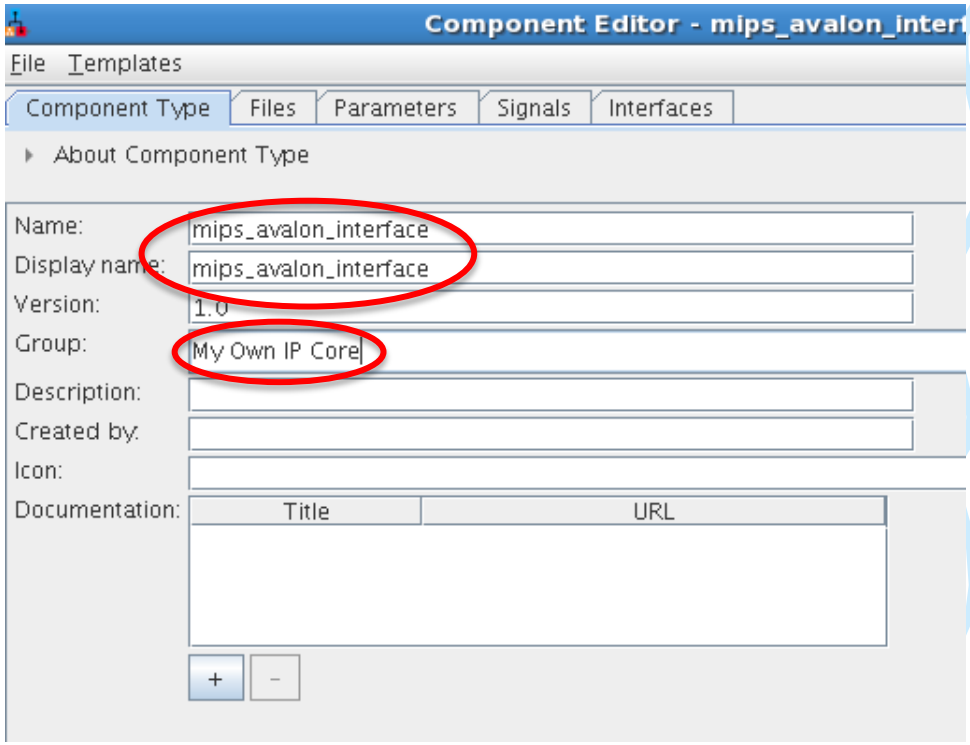




MIPS avalon interfaceをComponentとして登録

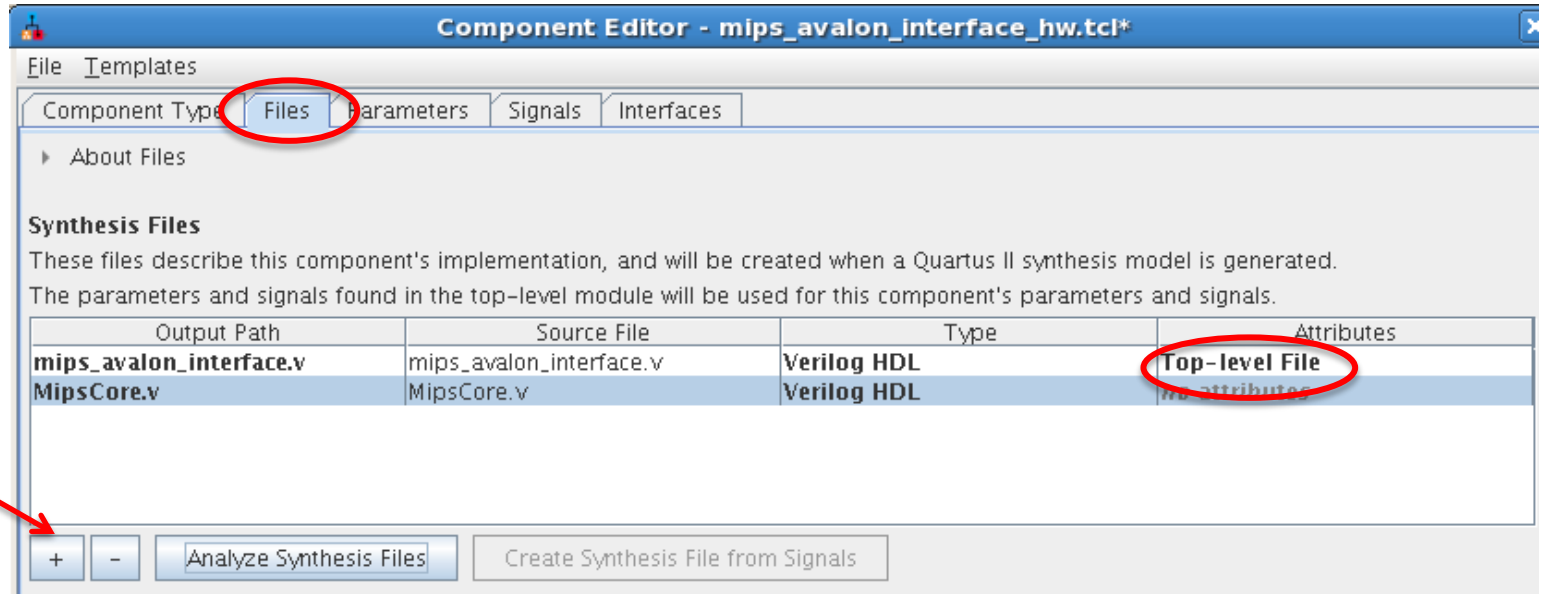
- ProjectのNew Componentを選択して，Add
- Component Typeタブ内
  - Name, Display Nameを“mips_avalon_interface”とする
  - Groupを“My Own IP Core”とする





MIPS avalon interfaceをComponentとして登録

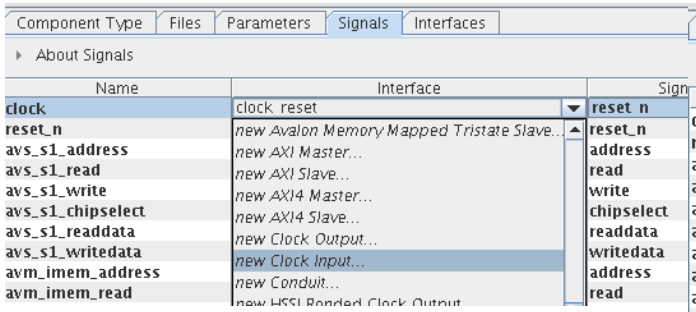
- Filesタブ内
  - （プロジェクトディレクトリ内にリファレンスデザインのプロジェクトディレクトリにあるMipsCore.vとmips_avalon_interface.vをコピーしておく）
  - Synthesis Filesとして，mips_avalon_interface.vとMipsCore.vを追加する（mips_avalon_interface.vがTop-level Fileとなっているのを確認）
  - Analyze Synthesis Filesボタンをクリック



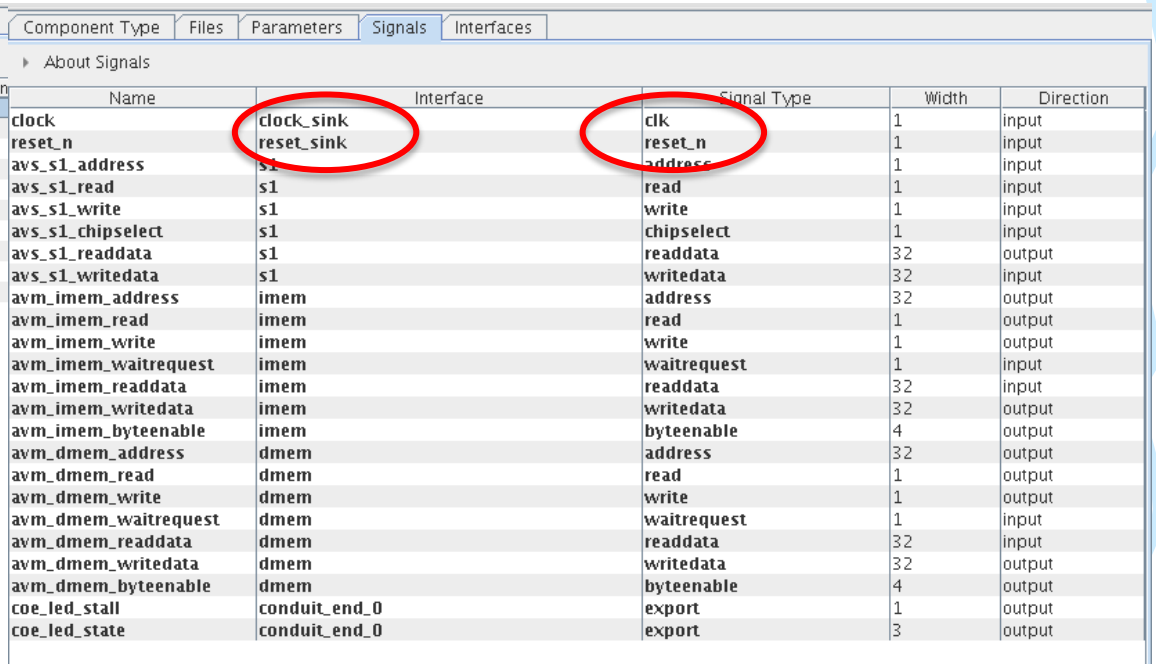
このボタンを押す事でファイルの追加

MIPS avalon interfaceをComponentとして登録

- Signalsタブ内
  - clockのinterfaceの所を選択して，new Clock Inputを選択し，interface欄がclock_sinkとなるようにし，Signal Typeをclkとする
  - resetのinterfaceの所を選択して，new Reset Inputを選択し，interface欄がreset_sinkとなるようにする．



clock_sinkを選択している所

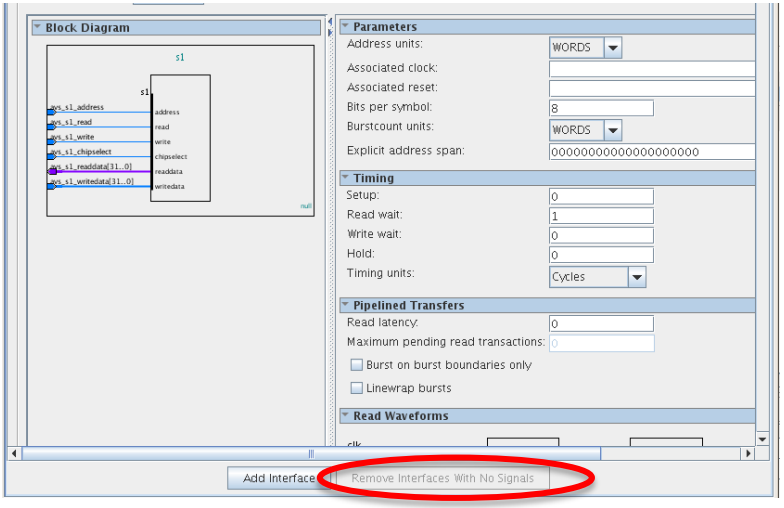


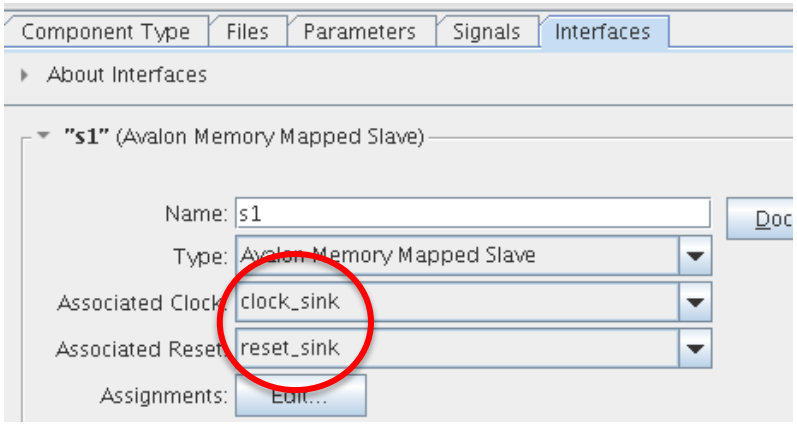
Name	Interface	Signal Type	Width	Direction
clock	clock_sink	clk	1	input
reset_n	reset_sink	reset_n	1	input
avs_s1_address	s1	address	1	input
avs_s1_read	s1	read	1	input
avs_s1_write	s1	write	1	input
avs_s1_chipselect	s1	chipselect	1	input
avs_s1_readdata	s1	readdata	32	output
avs_s1_writedata	s1	writedata	32	input
avm_imem_address	imem	address	32	output
avm_imem_read	imem	read	1	output
avm_imem_write	imem	write	1	output
avm_imem_waitrequest	imem	waitrequest	1	input
avm_imem_readdata	imem	readdata	32	input
avm_imem_writedata	imem	writedata	32	output
avm_imem_byteenable	imem	byteenable	4	output
avm_dmem_address	dmem	address	32	output
avm_dmem_read	dmem	read	1	output
avm_dmem_write	dmem	write	1	output
avm_dmem_waitrequest	dmem	waitrequest	1	input
avm_dmem_readdata	dmem	readdata	32	input
avm_dmem_writedata	dmem	writedata	32	output
avm_dmem_byteenable	dmem	byteenable	4	output
coe_led_stall	conduit_end_0	export	1	output
coe_led_state	conduit_end_0	export	3	output

MIPS avalon interfaceをComponentとして登録

- Interfaceタブ内
  - Remove Interfaces With No Signalsボタンを押す
  - s1のAssociated Clockをclock_sinkに，Associated Resetをreset_sinkにする
  - imem, dmem, conduit_end_0のAssociated Clock, Associated Resetも同様にする
  - reset_sinkのAssociated Clockをclock_sinkにする
- Finishボタンを押し，保存するかどうかを訪ねるWindowが出たら，Saveを選ぶ.

MIPS avalon interfaceをComponentとして登録





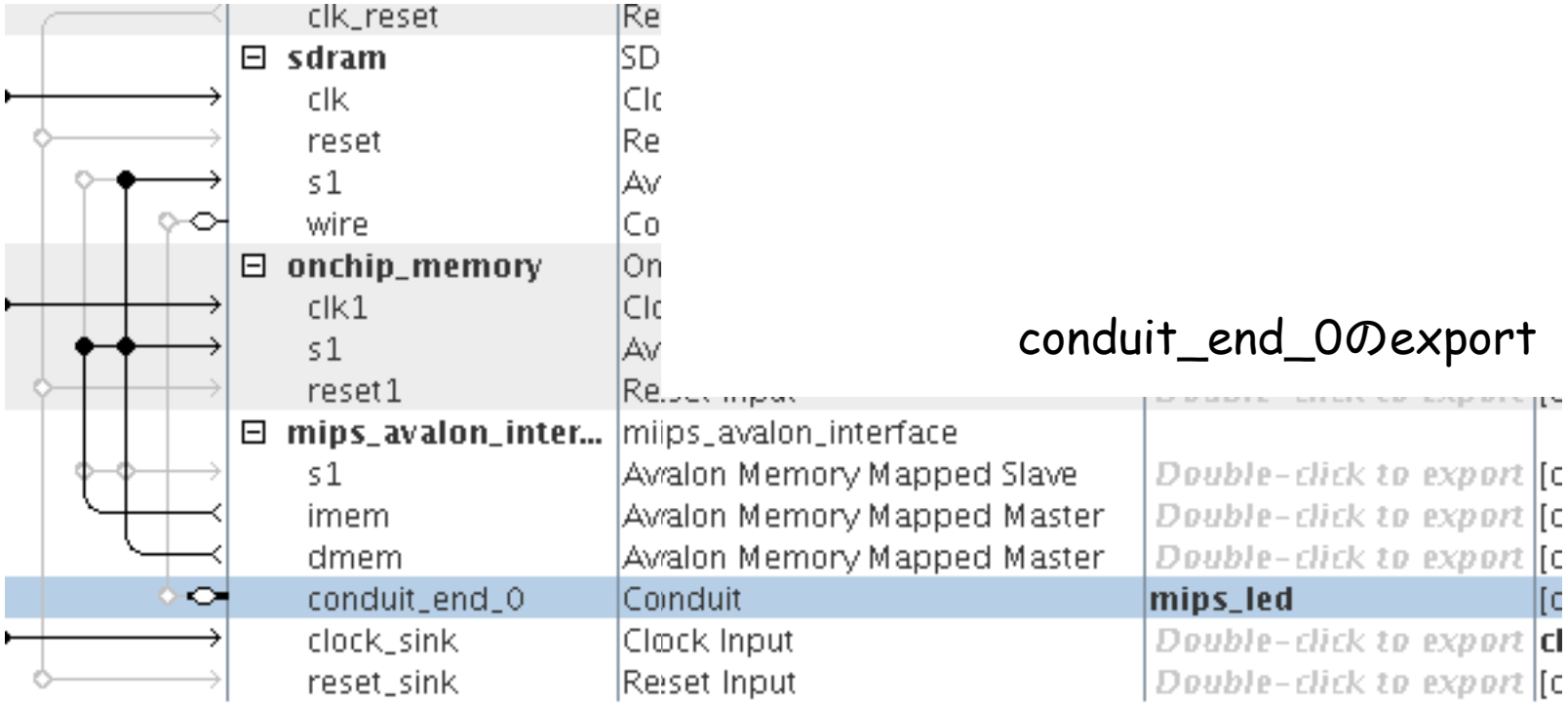
Remove Interfaces With No SignalsボタンはWindowの下の方にあり，クリックすると図の様にクリックできない状態になる

MIPS avalon interfaceを追加

- ProjectのMy Own IP Core内のmips_avalon_interfaceを選択してAdd
  - Finishボタンをクリック
- mips_avalon_interfaceのclock_sinkをclk_0のclkと接続
- mips_avalon_interfaceのimemをonchip_memoryのs1と接続
- mips_avalon_interfaceのdmemをsdramのs1，onchip_memoryのs1と接続
- mips_avalon_interfaceのconduit_end_0のexport欄をダブルクリックして，export名をmips_ledとする

MIPS avalon interfaceを追加

配線の様子



conduit_end_0のexport

contest uart avalon interfaceをComponentとして登録

- ProjectのNew Componentを選択して，Add
- Component Typeタブ内
  - Name, Display Nameを“contest_uart_avalon_interface”とする
  - Groupを“My Own IP Core”とする
- Filesタブ内
  - （プロジェクトディレクトリ内にリファレンスデザインのプロジェクトディレクトリにある以下のファイルをコピーしておく
    - contest_uart_avalon_interface.v
    - memory_access_module.v
    - mips_access_module.v
    - system.v
    - define.v
  - Synthesis Filesとして，define.v以外の上記4つのVerilog HDLファイルを追加する（contest_uart_avalon_interface.vがTop-level Fileとなっているのを確認）
  - Analyze Synthesis Filesボタンをクリック

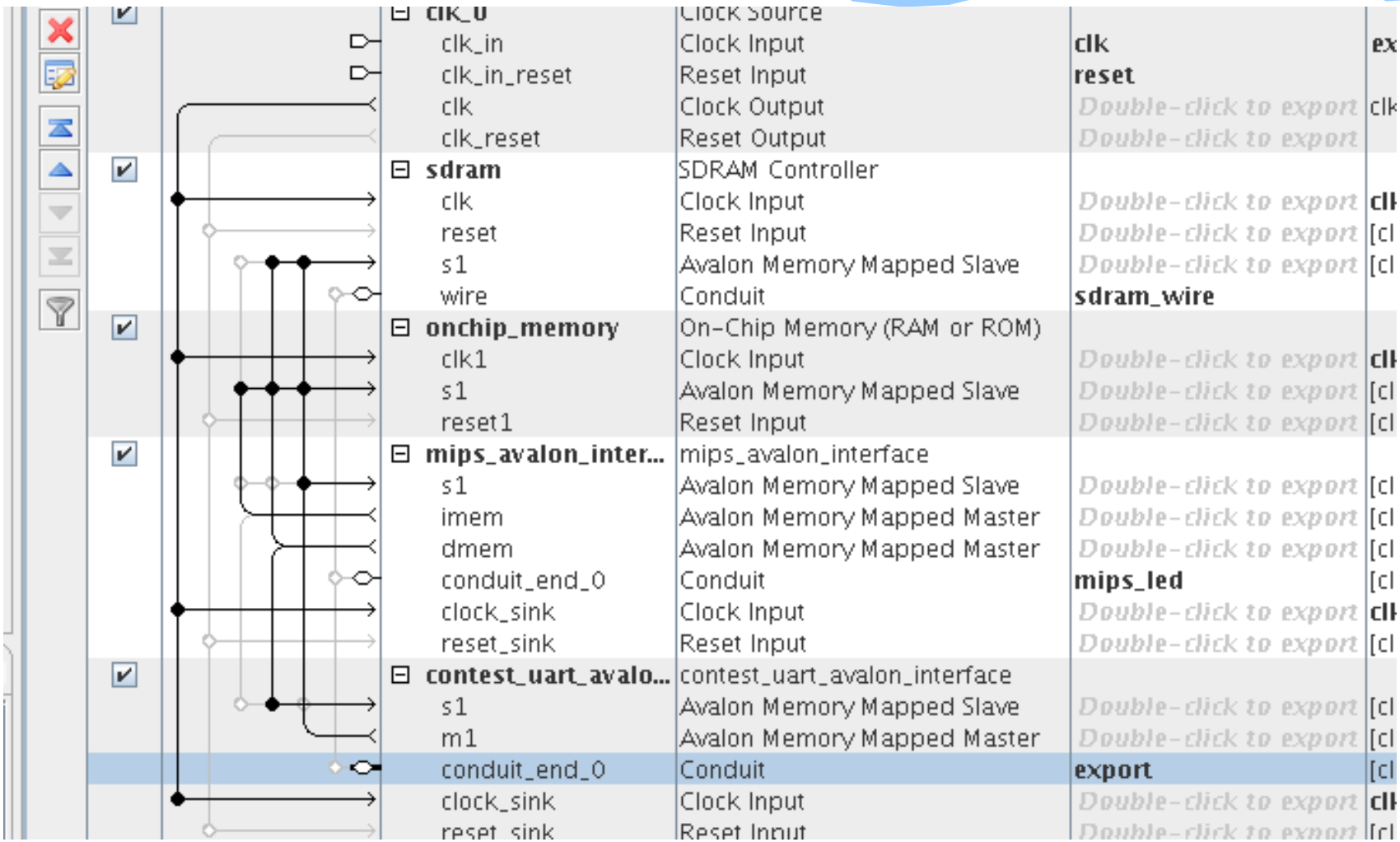
contest uart avalon interfaceをComponentとして登録

- Signalsタブ内
  - clockのinterfaceの所を選択して，new Clock Inputを選択し，interface欄がclock_sinkとなるようにし，Signal Typeをclkとする
  - resetのinterfaceの所を選択して，new Reset Inputを選択し，interface欄がreset_sinkとなるようにする.
- Interfaceタブ内
  - Remove Interfaces With No Signalsボタンを押す
  - s1のAssociated Clockをclock_sinkに，Associated Resetをreset_sinkにする
  - m1, conduit_end_0のAssociated Clock, Associated Resetも同様にする
  - reset_sinkのAssociated Clockをclock_sinkにする
- Finishボタンを押し，保存するかどうかを訪ねるWindowが出たら，Saveを選ぶ.

contest uart avalon interfaceを追加

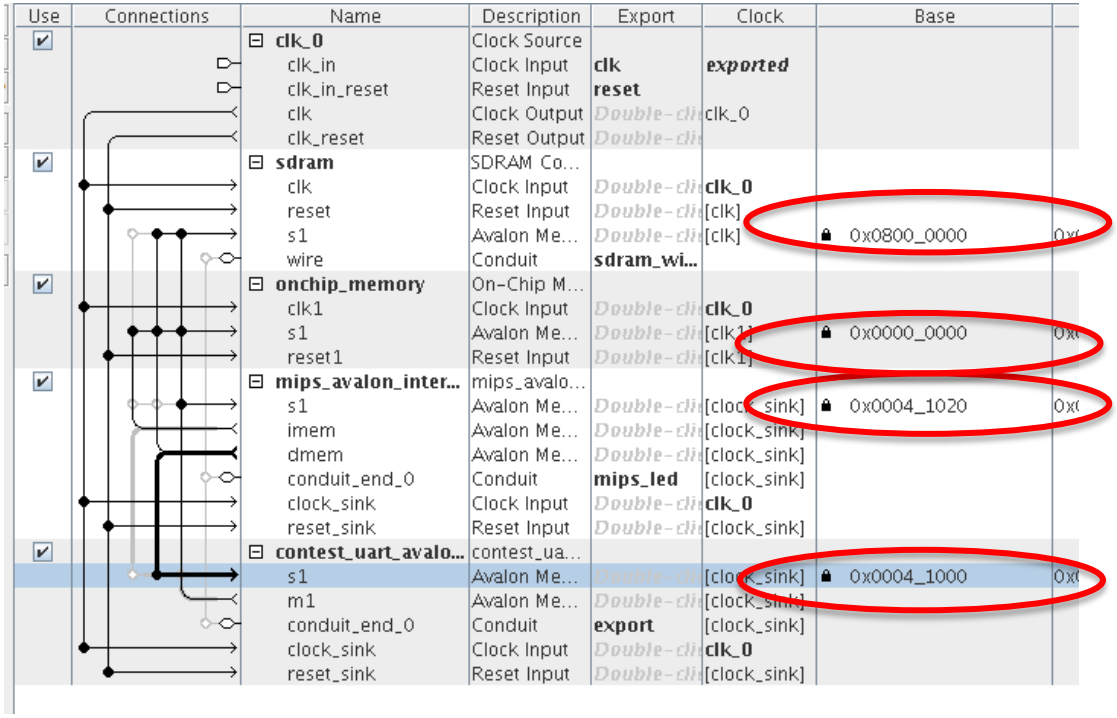
- ProjectのMy Own IP Core内のcontest_uart_avalon_interfaceを選択してAdd
  - Finishボタンをクリック
- contest_uart_avalon_interfaceのclock_sinkをclk_0のclkと接続
- contest_uart_avalon_interfaceのm1をmips_avalon_interfaceのs1, onchip_memoryのs1, sdramのs1と接続
- contest_uart_avalon_interfaceのs1をmips_avalon_interfaceのdmemと接続
- contest_uart_avalon_interfaceのconduit_end_0のexport欄をダブルクリックして, export名をexportとする

contest uart avalon interfaceを追加



メモリマップの作成

- メモリマップとしてbase addressを以下のように指定して，ロック
  - sdramのmemory: 0x0800_0000
  - onchip_memory: 0x0000_0000
  - mips_avalon_interface_0: 0x0004_1020
  - contest_uart_avalon_interface_0: 0x0004_1000

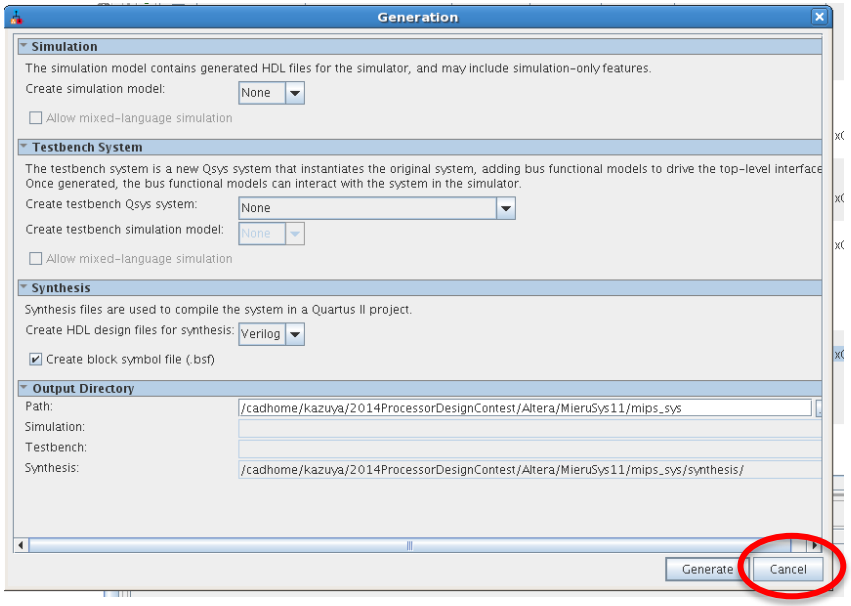


Use	Connections	Name	Description	Export	Clock	Base
✔		⊟ **clk_0**	Clock Source			
		clk_in	Clock Input	**clk**	*exported*	
		clk_in_reset	Reset Input	**reset**		
		clk	Clock Output	Double-cli	clk_0	
		clk_reset	Reset Output	Double-cli		
✔		⊟ **sdram**	SDRAM Co...			
		clk	Clock Input	Double-cli	**clk_0**	
		reset	Reset Input	Double-cli	[clk]	
		s1	Avalon Me...	Double-cli	[clk]	🔒 0x0800_0000
		wire	Conduit	**sdram_wi...**		
✔		⊟ **onchip_memory**	On-Chip M...			
		clk1	Clock Input	Double-cli	**clk_0**	
		s1	Avalon Me...	Double-cli	[clk1]	🔒 0x0000_0000
		reset1	Reset Input	Double-cli	[clk1]	
✔		⊟ **mips_avalon_inter...**	mips_avalo...			
		s1	Avalon Me...	Double-cli	[clock_sink]	🔒 0x0004_1020
		imem	Avalon Me...	Double-cli	[clock_sink]	
		dmem	Avalon Me...	Double-cli	[clock_sink]	
		conduit_end_0	Conduit	**mips_led**	[clock_sink]	
		clock_sink	Clock Input	Double-cli	**clk_0**	
		reset_sink	Reset Input	Double-cli	[clock_sink]	
✔		⊟ **contest_uart_avalo...**	contest_ua...			
		s1	Avalon Me...	Double-cli	[clock_sink]	🔒 0x0004_1000
		m1	Avalon Me...	Double-cli	[clock_sink]	
		conduit_end_0	Conduit	**export**	[clock_sink]	
		clock_sink	Clock Input	Double-cli	**clk_0**	
		reset_sink	Reset Input	Double-cli	[clock_sink]	

ロックを外した状態で
アドレスを入力し，
その後ロックをかける

mips_sysの生成

- メニューSysmem -> Create Global Reset Networkをクリック
- この時点で，Qsys下部のMessageからエラー表示がなくなっているはず
- mips_sysを保存
- メニューGenerate -> Generateをクリック
  - Generateボタンをクリック
- ちなみに，Mips_Core.vやmips_avalon_interface.vを変更する度に，QsysでGenerateする必要があります
- 以上で，Qsysは終了してOK

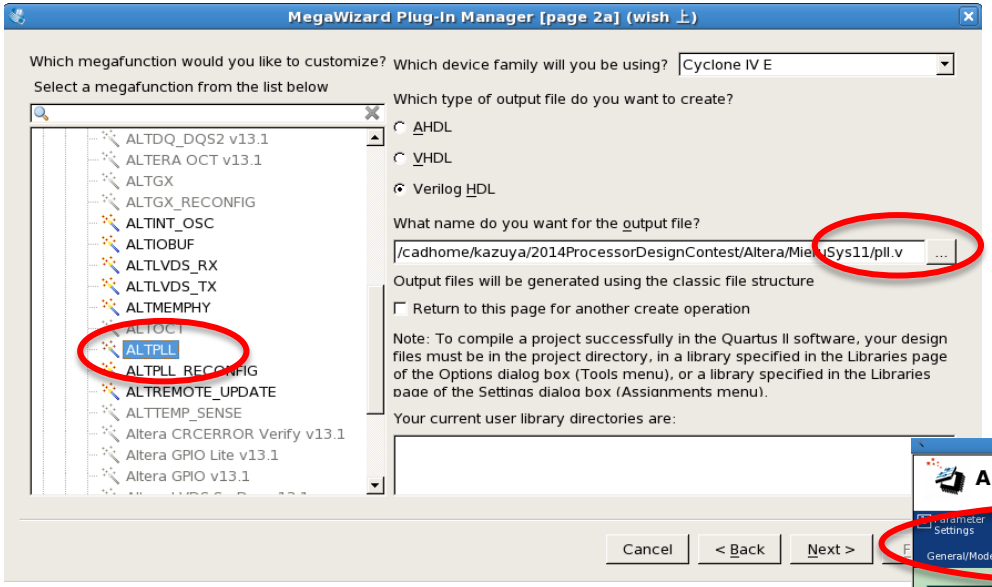


Generateでは
右下のGenerateボタンを
押すだけでよい

PLLの追加

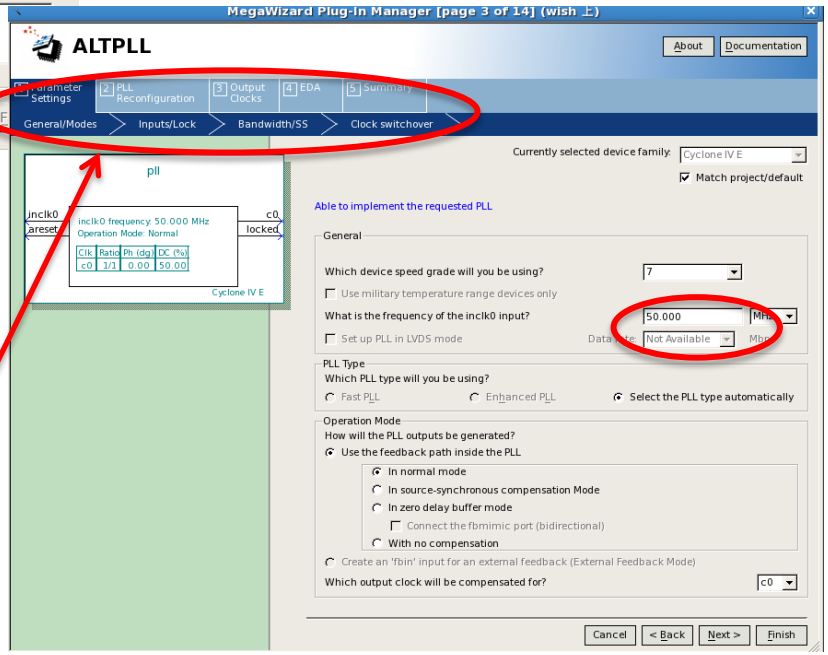
- QuartusのメニューTools->MegaWizard Plug-In Manageをクリック
  - Nextをクリック
  - I/OのALTPLLを選択し，出力ファイル名をpll.vとし，Nextをクリック（次ページの画面1）
  - PLLの設定
    - Parameter Setting -> General/Mode内の入力クロックを50MHzに設定（次ページの画面2）
    - Output Clock -> clk c0のEnter output clock frequencyのラジオボタンを選択し，40MHzと入力（2ページ先の画面3）
    - Summaryでは，選択可能なチェックボックスを全て外し，Finishボタンを押す
    - IPをprojectに登録するかどうか聞かれるので，登録する

PLLの追加



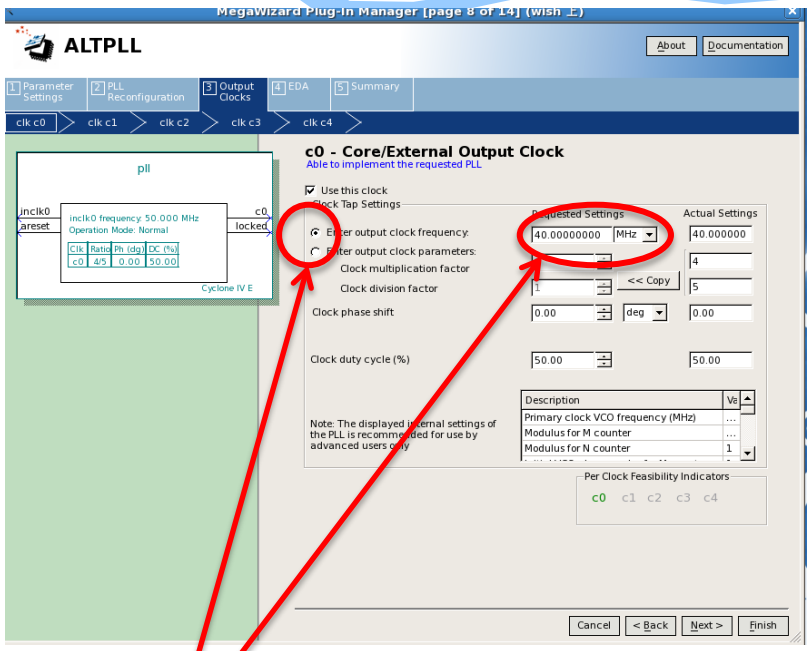
画面1

画面2



ここの欄が現在設定している
項目を示すタブになっている

PLLの追加

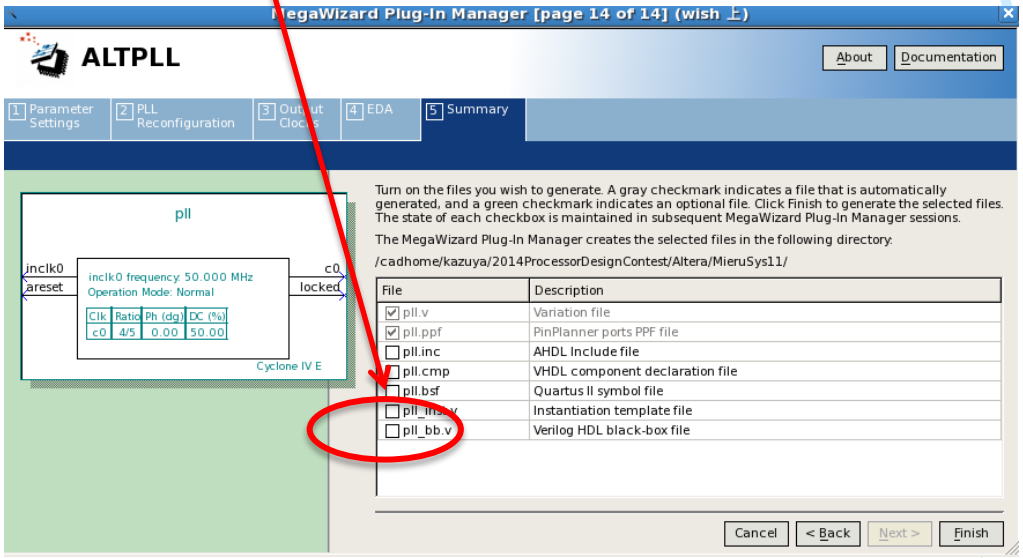


画面3

最初にラジオボタンを選択してから，
周波数の数値を入力する

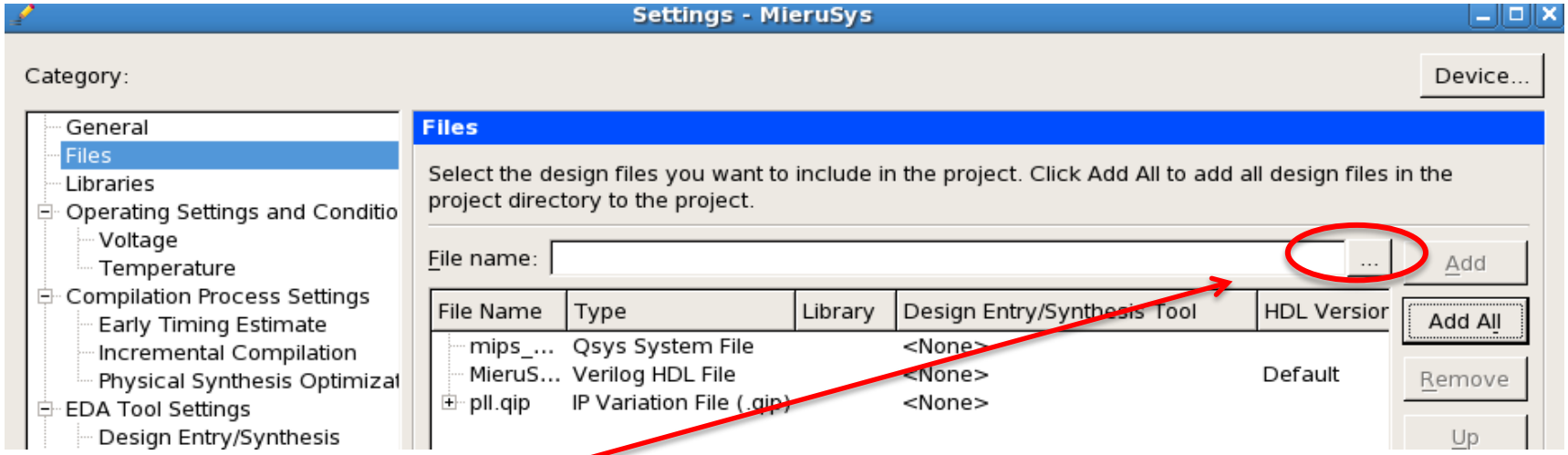
チェックを外す

画面4



プロジェクトへのファイルの追加

- あらかじめ，MieruSys.vをプロジェクトディレクトリにコピーしておく
- Porject -> Add/Remove files in Projectで以下のファイルを追加する
  - MieruSys.v
  - mips_sys.qsys
- 追加したらOKボタンを押してウィンドウを閉じる



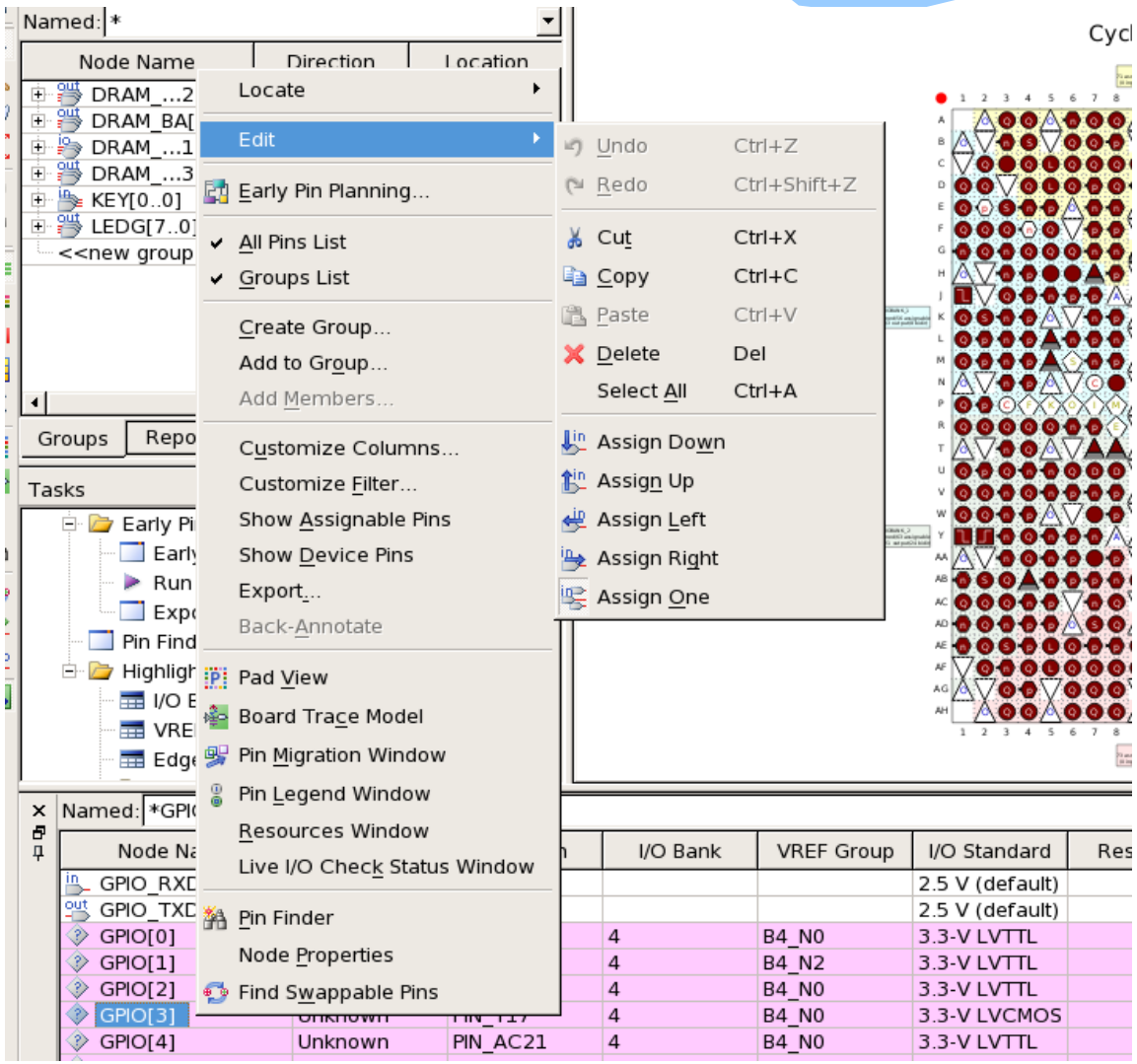
ここをクリックして追加するファイルを選択し，その後addボタンを押す

ピン配置情報のimportと割当

- Assignments -> Import Assignmentsをクリック
  - DE2_115.qsfを選択し，importする
- Assignments -> Pin Plannerをクリック
  - 現れるウィンドウの下の方にある各ピンの設定で，GPIO[3]を右クリックし現れるメニューの中から，Edit->Deleteを選択して削除する（次ページの画面1）
  - 同様に GPIO[5], GPIO[7], GPIO[9]のピン設定を削除する
  - 以下の表の様に未割当の入出力信号にピンを割り当てる
  - メニューFile -> closeをクリックして，ウィンドウを閉じる

Node Name	Location	I/O Standard	補足
GPIO_RXD	PIN_Y17	3.3V LVCMOS	GPIO[3]
GPIO_TXD	PIN_Y16	3.3V LVCMOS	GPIO[5]
GPIO_FLUSH_X	PIN_AE16	3.3V LVCMOS	GPIO[7]
GPIO_INIT_X	PIN_AE15	3.3V LVCMOS	GPIO[9]

ピン配置情報のimportと割当



ピン割当結果

Named: *GPIO* Edit:

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Diff
GPIO_FLUSH_X	Output	PIN_AE16	4	B4_N2	3.3-V LVCMOS		2mA (default)	2 (default)	
GPIO_INIT_X	Input	PIN_AE15	4	B4_N2	3.3-V LVCMOS		2mA (default)		
GPIO_RXD	Input	PIN_Y17	4	B4_N0	3.3-V LVCMOS		2mA (default)		
GPIO_TXD	Output	PIN_Y16	4	B4_N0	3.3-V LVCMOS		2mA (default)	2 (default)	
GPIO[0]	Unknown	PIN_AB22	4	B4_N0	3.3-V LVTTL		8mA (default)		
GPIO[1]	Unknown	PIN_AC15	4	B4_N2	3.3-V LVTTL		8mA (default)		

構成情報の生成

- Processing -> Start Compilationをクリックして，論理合成＆配置配線
- DE2-115ボードの電源をいれる.
- Taskウィンドウ内のProgram Device(Open Programmer)をダブルクリック
- Programmer内にて，Hardware Setupボタンをクリック.
  - No Hardwareとなっている所をUSB-Blaster USBを選択して，closeボタンをクリック
- Start ボタンを押して，Progress が100%(Successful)になればOK
- Programmerを閉じる

- プロセッサ設計部門のプログラムの転送方法は，「P42Reference Design Test Manual」を参照して下さい.

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 2014-07-03

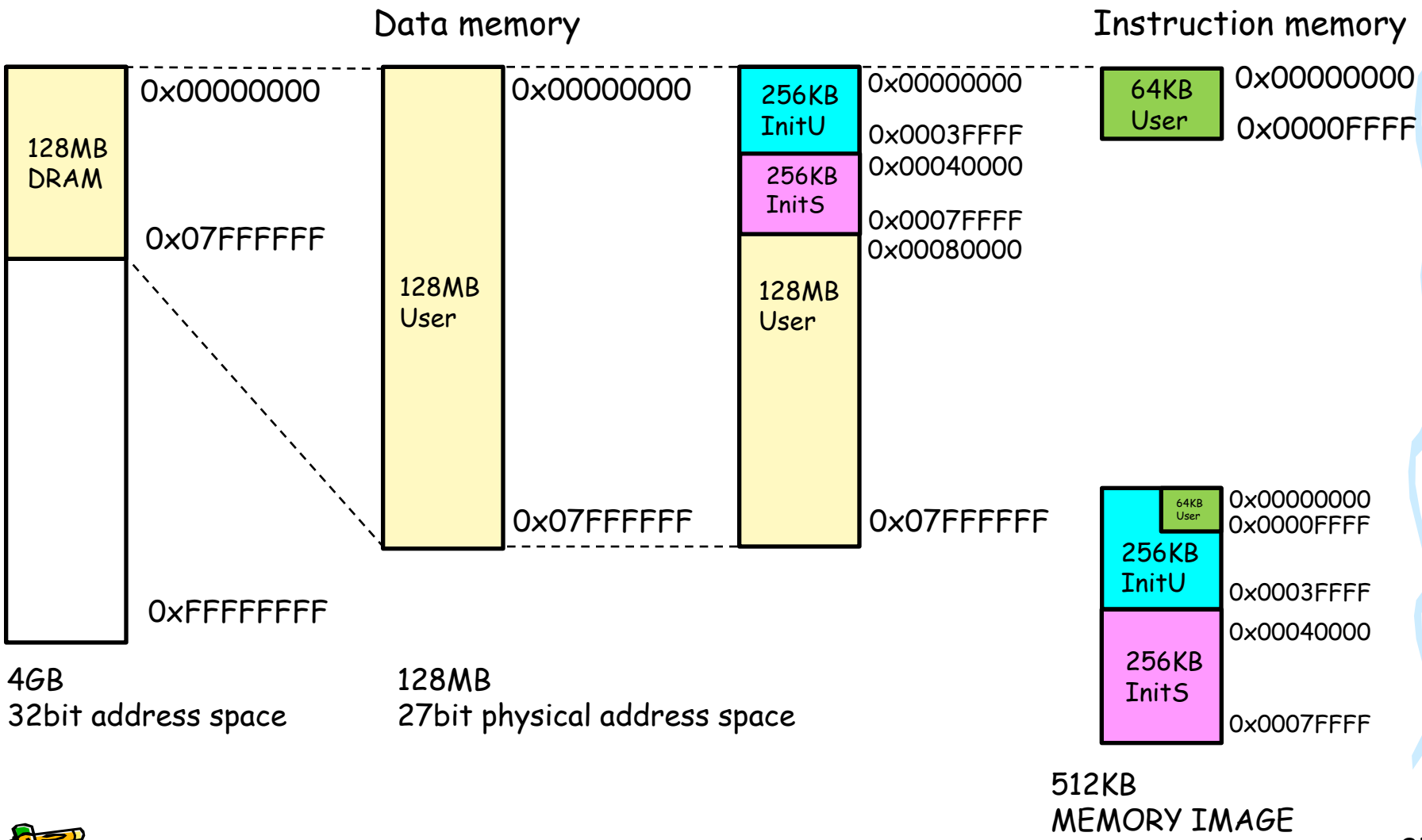
このドキュメント

- このドキュメントでは，Atlysボード向け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．
- 但し，UDPパケットの送信は1024バイトのデータがたまってから行われます．
  - 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．

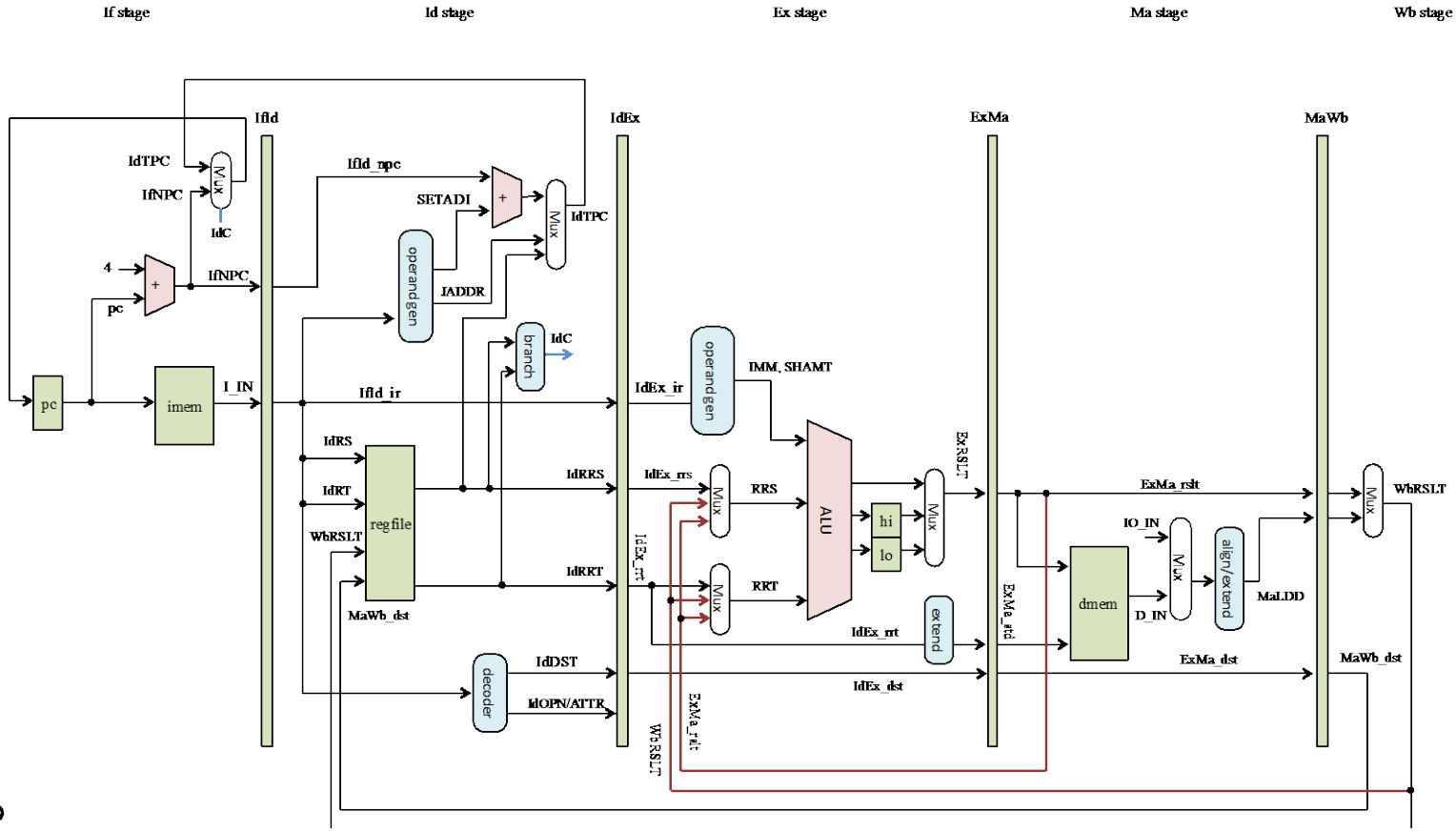
```
{
    volatile int *uart_txd = (int*)0;
    *uart_txd = 'A';
}
```

  - 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;
}
```

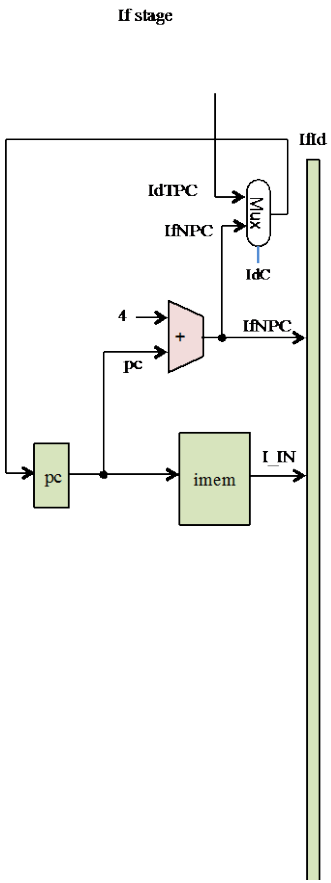
リファレンスデザインに含まれるプロセッサ

- 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



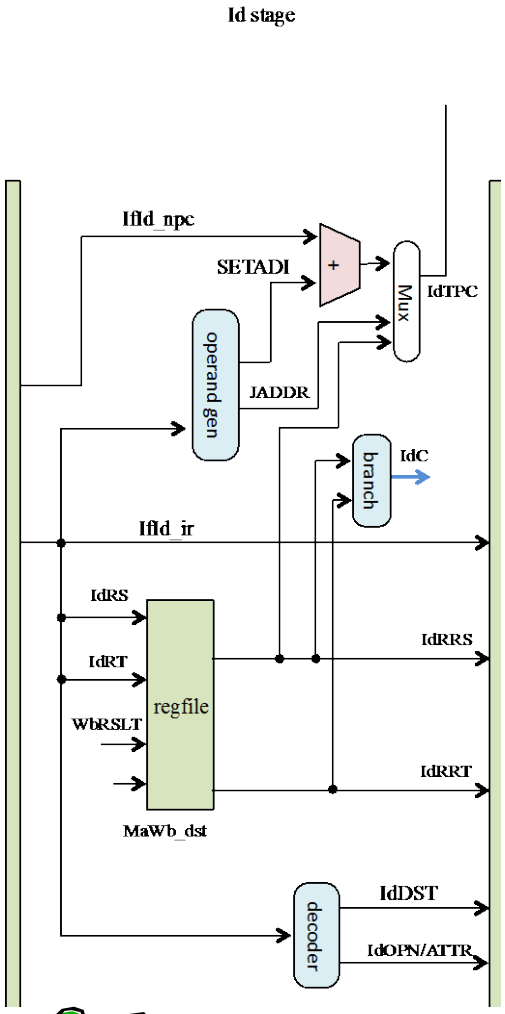

```
/*****************************************************************************************/
/* Stage 1 : IF, instruction fetch                                                       */
/*****************************************************************************************/
wire [‘ADDR] IfNPC = pc + 4;

always @( posedge CLK or negedge RST_X ) begin ///// update program counter
    if(!RST_X)                    pc <= 0;
    else if(!PSTALL && !bstall) pc <= (IdC) ? IdTPC : IfNPC; // If branch taken, uses taken PC
end

always @( posedge CLK or negedge RST_X ) begin  ///// update pipeline registers
    if(!RST_X) {IfId_npc, IfId_ir} <= 0;
    else if(!PSTALL && !bstall) begin
        IfId_npc <= IfNPC;
        IfId_ir  <= I_IN;  // if branch taken then NOP else instruction from imem.
    end
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.

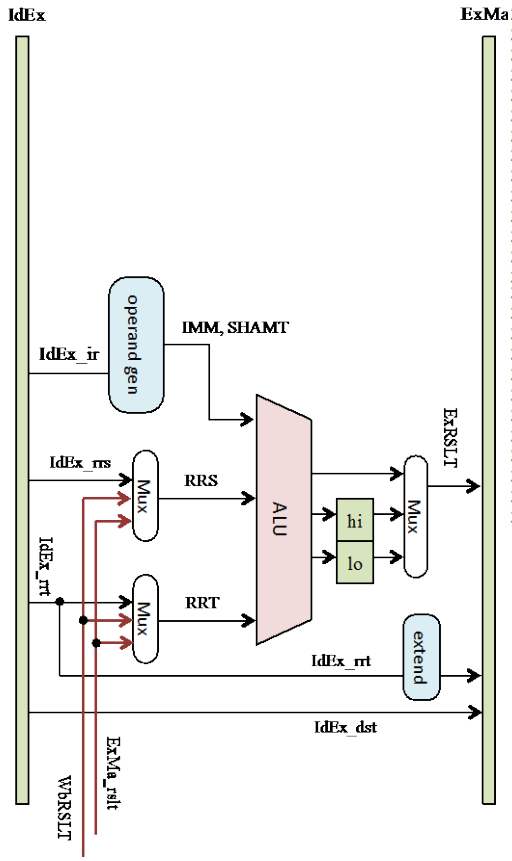
```
            6'h25:     begin IdOPN='LHU_____; IdDST=IdRT; IdATTR='LD_2B;         end
            6'h28:     begin IdOPN='SB______; IdDST=0;    IdATTR='ST_1B;         end
            6'h29:     begin IdOPN='SH______; IdDST=0;    IdATTR='ST_2B;         end
            6'h2b:     begin IdOPN='SW______; IdDST=0;    IdATTR='ST_4B;         end
        endcase
    end

    wire ['ADDR] IdBPC = IfId_npc + ({{16{IfId_ir[15]}}, IfId_ir[15:0]} << 2);
    always @(*) begin  ////// branch & jump resolution unit
        {IdTPC, IdC, IdB} = 0;
        case (IdOP)
            6'h04: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS == IdRRT);         end        // BEQ
            6'h05: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS != IdRRT);         end        // BNE
            6'h06: begin IdB=1; IdTPC = IdBPC; IdC = ( IdRRS[31]||(IdRRS==0)); end        // BLEZ
            6'h07: begin IdB=1; IdTPC = IdBPC; IdC = (~IdRRS[31]&&(IdRRS!=0)); end        // BGTZ
            6'h01: begin IdB=1; IdTPC = IdBPC; IdC = (IdRT) ? ~IdRRS[31] : IdRRS[31]; end // BGEZ,BLTZ
            6'h02: begin IdB=1; IdTPC = IfId_ir['ADDR]<<2; IdC = 1; end                   // J
            6'h03: begin IdB=1; IdTPC = IfId_ir['ADDR]<<2; IdC = 1; end                   // JAL
            6'h00: if      (IdFCT==6'h08) begin IdB=1; IdTPC = IdRRS; IdC = 1; end        // JR
                   else if (IdFCT==6'h09) begin IdB=1; IdTPC = IdRRS; IdC = 1; end        // JALR
        endcase
    end

    always @(posedge CLK or negedge RST_X) begin ///// update pipeline registers
        if(!RST_X) {IdEx_npc, IdEx_rrs, IdEx_rrt, IdEx_dst, IdEx_ir, IdEx_opn, IdEx_attr} <= 0;
        else if(!PSTALL) begin
            IdEx_npc   <= (bstall) ? 0 : IfId_npc;
            IdEx_rrs   <= (bstall) ? 0 : IdRRS;    // data from general-purpose register file
            IdEx_rrt   <= (bstall) ? 0 : IdRRT;    // data from general-purpose register file
            IdEx_dst   <= (bstall) ? 0 : IdDST;
            IdEx_ir    <= (bstall) ? 0 : IfId_ir;
            IdEx_opn   <= (bstall) ? 0 : IdOPN;
            IdEx_attr  <= (bstall) ? 0 : IdATTR;
        end
    end
```

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




```
/*****************************************************************************************/
/* Stage 3 : EX, execute & address generation for LD/ST                                  */
/*****************************************************************************************/
wire         [31:0] RRS_U, RRT_U;                        // output of data forwarding unit
wire signed  [31:0] RRS_S = RRS_U;                       // signed wire
wire signed  [31:0] RRT_S = RRT_U;                       // signed wire
wire          [4:0] SHAMT  = IdEx_ir[10:6];              // Shift amount
wire         [15:0] IMM    = IdEx_ir[15:0];              // Immediate
wire         [31:0] SET32I = {{16{IMM[15]}}, IMM};       // Sign Extended Imm.
wire       [‘ADDR] SETADI = SET32I[‘ADDR] << 2;          // shifted immediate of address bit width
wire       [‘ADDR] JADDR  = IdEx_ir[‘ADDR] << 2;         // Juma address
wire       [‘ADDR] ExA = RRS_U[‘ADDR] + SET32I[‘ADDR];   // mem address of LD/ST
reg          [31:0] ExRSLT;                              // execution result
reg           [3:0] ExWE;                                // memory write enable vector

wire DIVINIT  = IdOPN==‘DIV_____   || IdOPN==‘DIVU_____;
wire EXSIGNED = IdEx_opn==‘DIV_____;
wire [63:0] DURSLT;

DIVUNIT divu(.CLK(CLK), .RST_X(RST_X), .INIT(DIVINIT), .SIGNED(EXSIGNED),
             .A(RRS_U), .B(RRT_U), .RSLT(DURSLT), .BUSY(DIVBUSY));

FORWARDING frs(.SRC(IdEx_ir[25:21]), .DIN0(IdEx_rrs), .DOUT(RRS_U),
               .DST1(ExMa_dst), .DIN1(ExMa_rslt), .DST2(MaWb_dst), .DIN2(MaWb_rslt));
FORWARDING frt(.SRC(IdEx_ir[20:16]), .DIN0(IdEx_rrt), .DOUT(RRT_U),
               .DST1(ExMa_dst), .DIN1(ExMa_rslt), .DST2(MaWb_dst), .DIN2(MaWb_rslt));
```

データパス - Exステージ - (3/5)

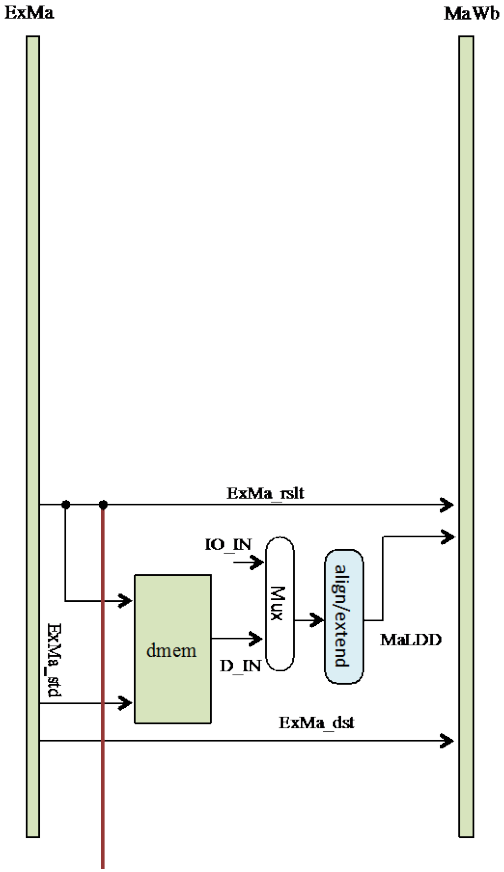
- 命令操作の実行またはアドレス生成を行います.

```
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



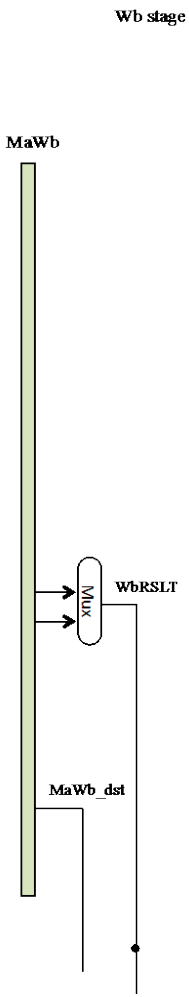

```
/*****************************************************************************************/
/* Stage 4 : MA, memory access for LD/ST                                                 */
/*****************************************************************************************/
assign D_ADDR = (ExMa_oe)            ? ExMa_rslt : 0;
assign D_OUT  = (ExMa_oe)            ? ExMa_std  : 0;
assign D_WE   = (ExMa_oe && !PSTALL) ? ExMa_mwe  : 0;
assign D_OE   =  ExMa_oe;

// wire [31:0] MaLD_ = (ExMa_rslt[23]) ? IO_IN : D_IN; // use I/O input if the high address
wire [31:0] MaLD_ = D_IN; // This edition does not use I/O.
wire [31:0] MaLDD = MaLD_ >> (8*ExMa_rslt[1:0]);    // loaded data, align data here

assign MaRSLT = (ExMa_lds[0]) ? {{16{MaLDD[15]}}, MaLDD[15:0]} : // opn=='LH_______
                (ExMa_lds[1]) ? { 16'd0         , MaLDD[15:0]} : // opn=='LHU______
                (ExMa_lds[2]) ? {{24{MaLDD[ 7]}}, MaLDD[ 7:0]} : // opn=='LB_______
                (ExMa_lds[3]) ? { 24'd0         , MaLDD[ 7:0]} : // opn=='LBU______
                (ExMa_lds[4]) ?                  MaLDD         : ExMa_rslt;
```

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



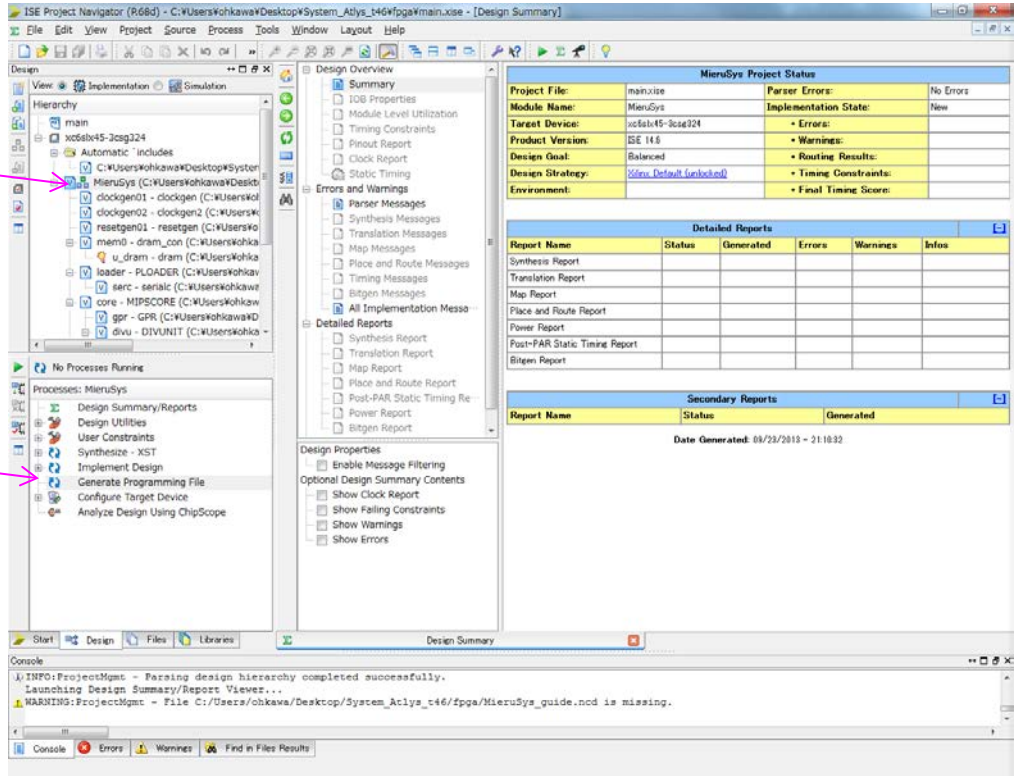

```
368     always @( posedge CLK or negedge RST_X ) begin ///// update pipeline registers
369         if(!RST_X) {MaWb_rslt, MaWb_dst} <= 0;
370         else if(!PSTALL) begin
371             MaWb_rslt <= MaRSLT;
372             MaWb_dst  <= ExMa_dst;
373         end
374     end
375
```

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．

①

②



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 2014-07-03

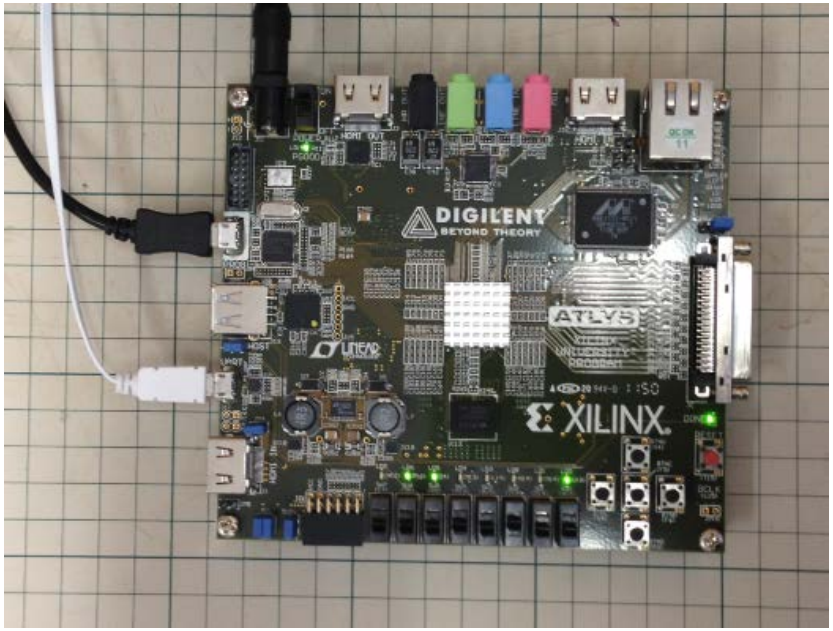
このドキュメント

- 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

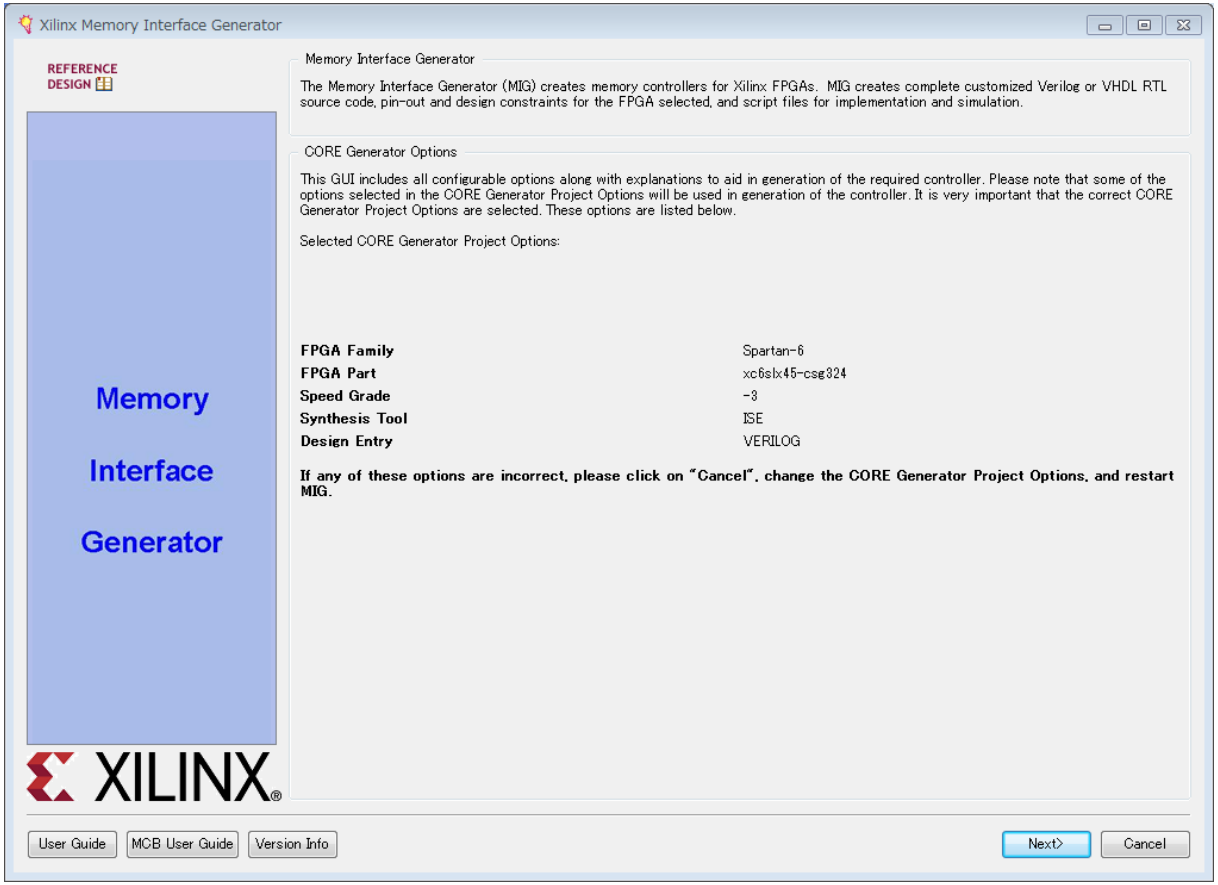
DRAM on Atlys Board

- 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



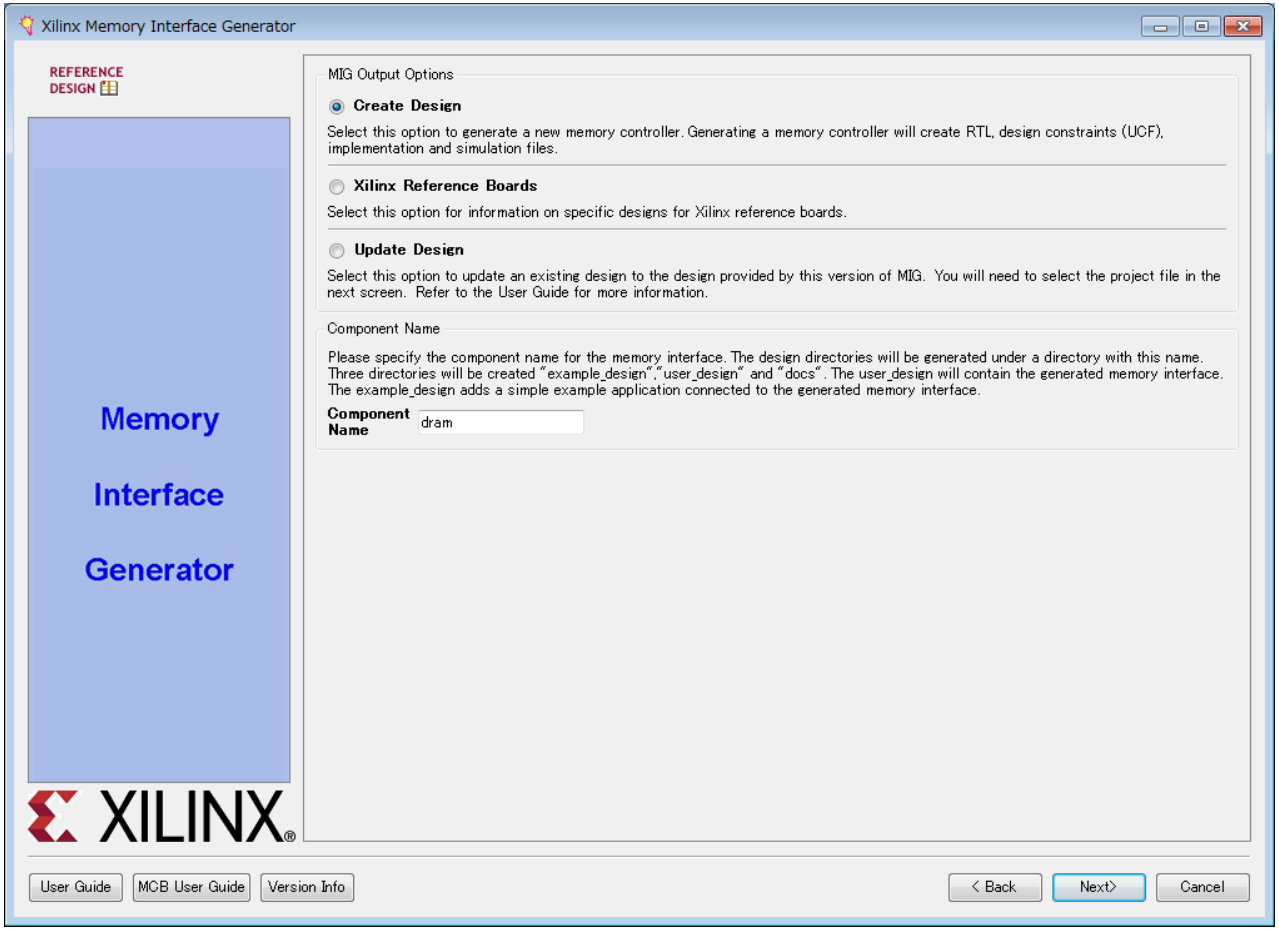
Xilinx Memory Interface Generator (1)

- 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



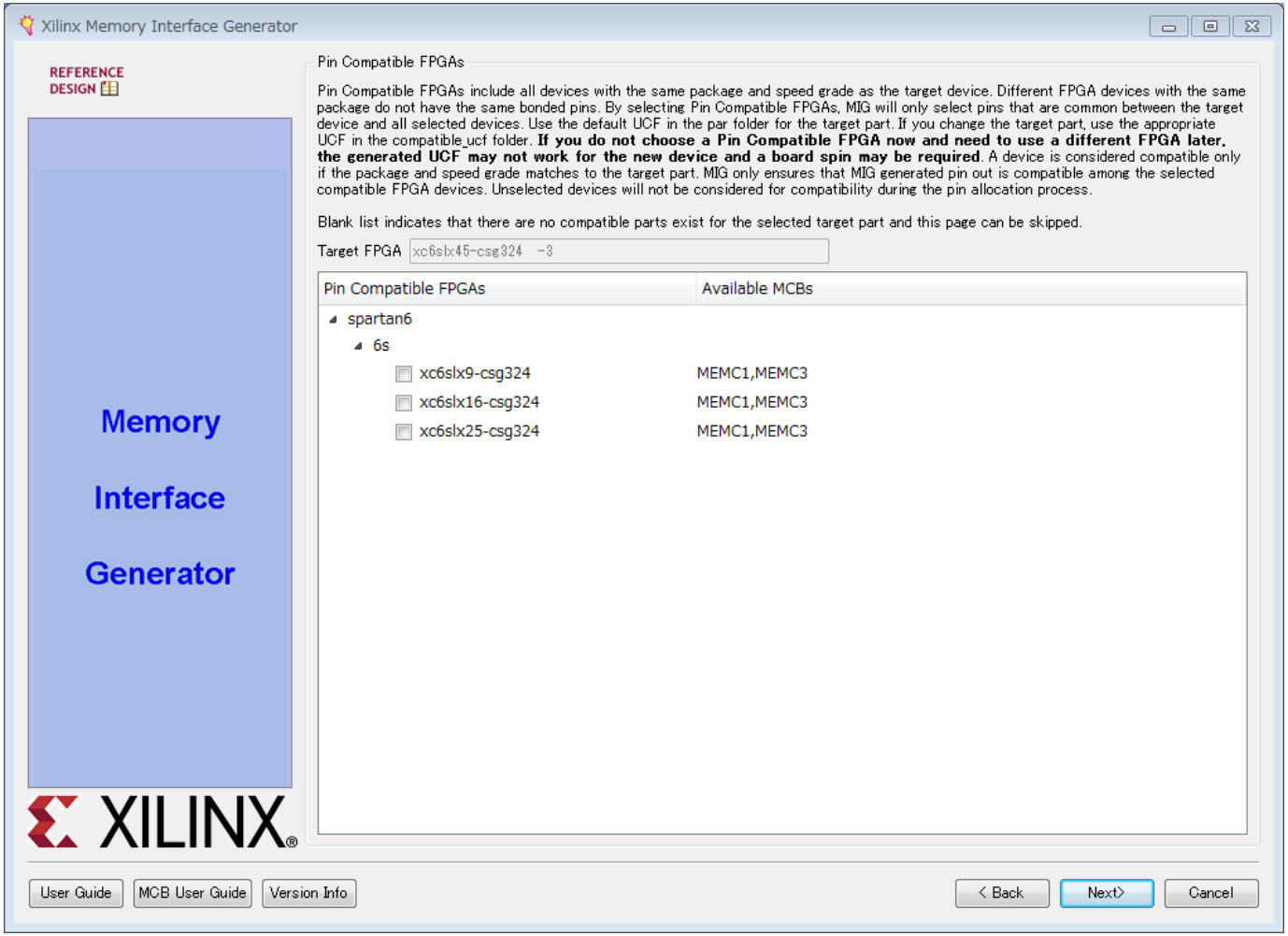
Xilinx Memory Interface Generator (2)

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



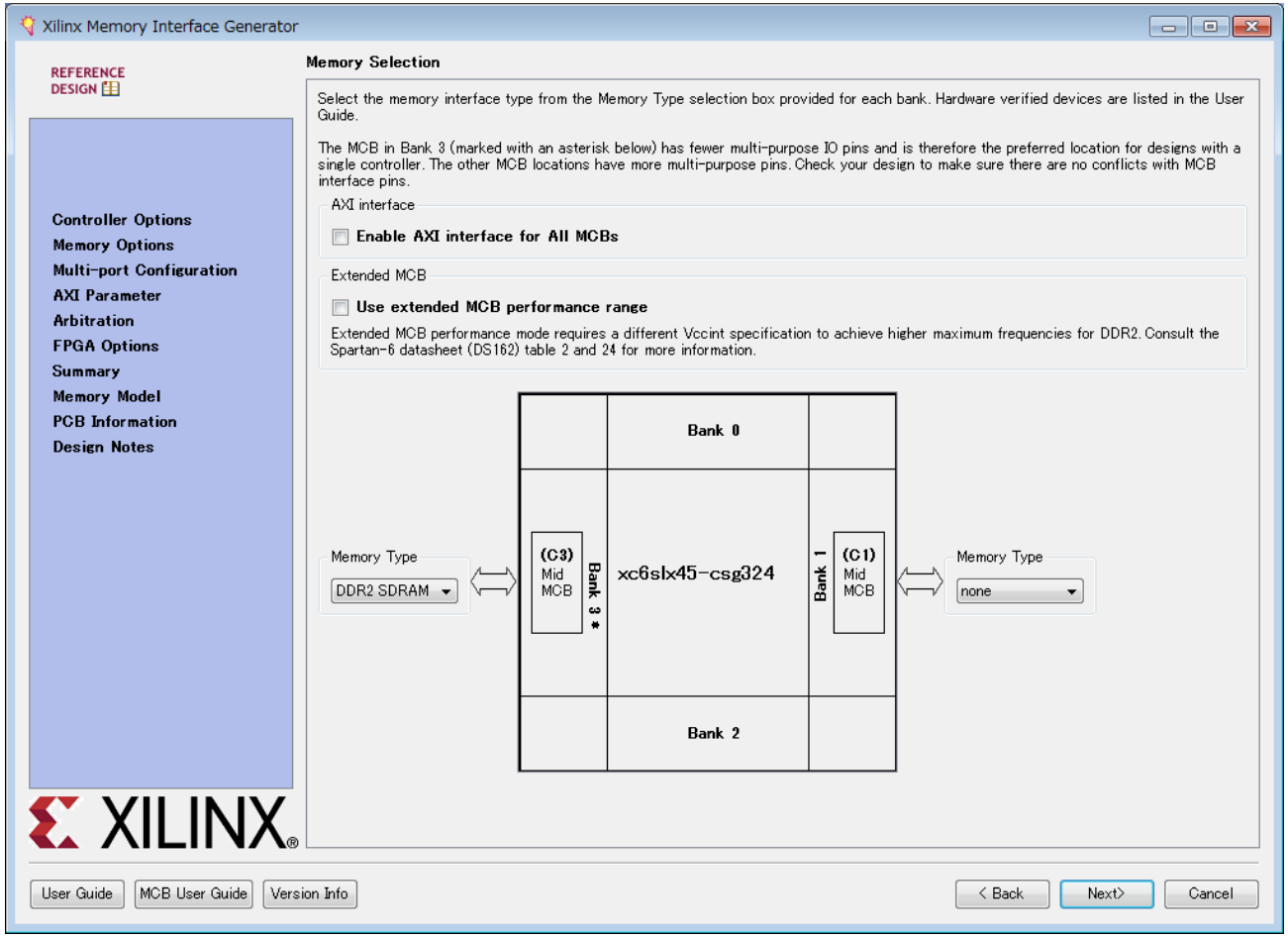
Xilinx Memory Interface Generator (3)

- Pin Compatible FPGAs では，チェックをいれない



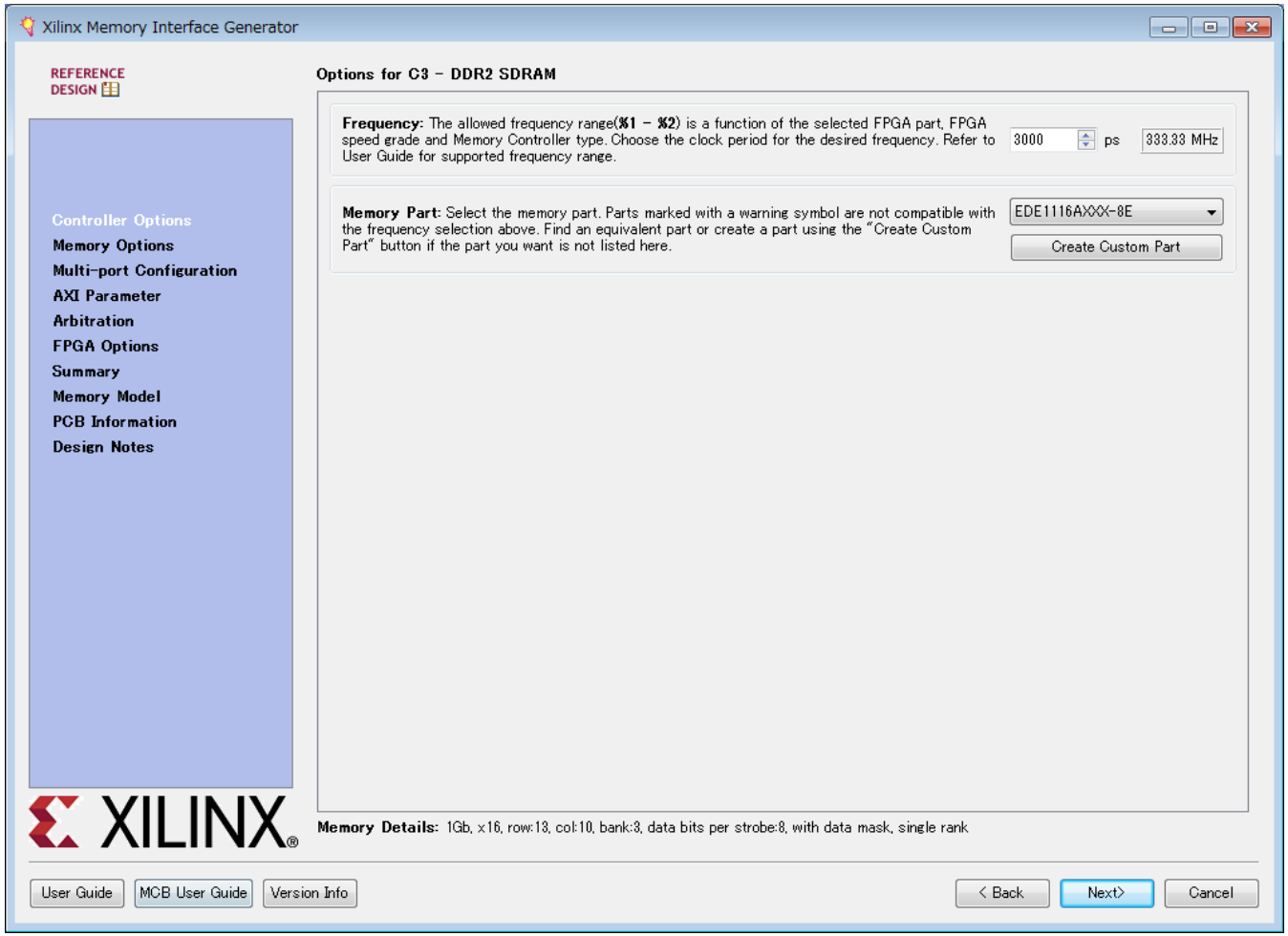
Xilinx Memory Interface Generator (4)

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



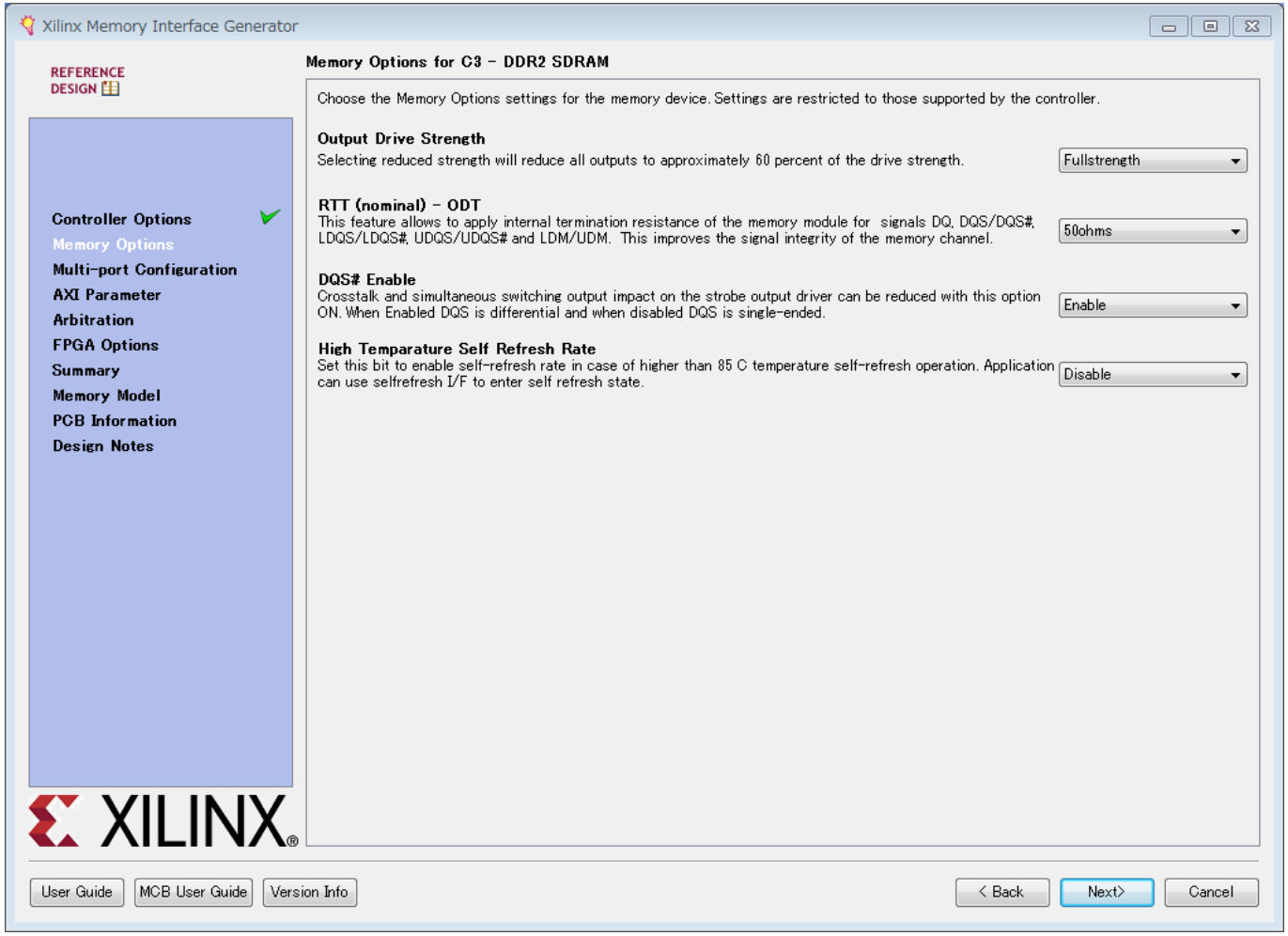
Xilinx Memory Interface Generator (5)

- 3000 ps, EDE1116AXXX-8E を選択



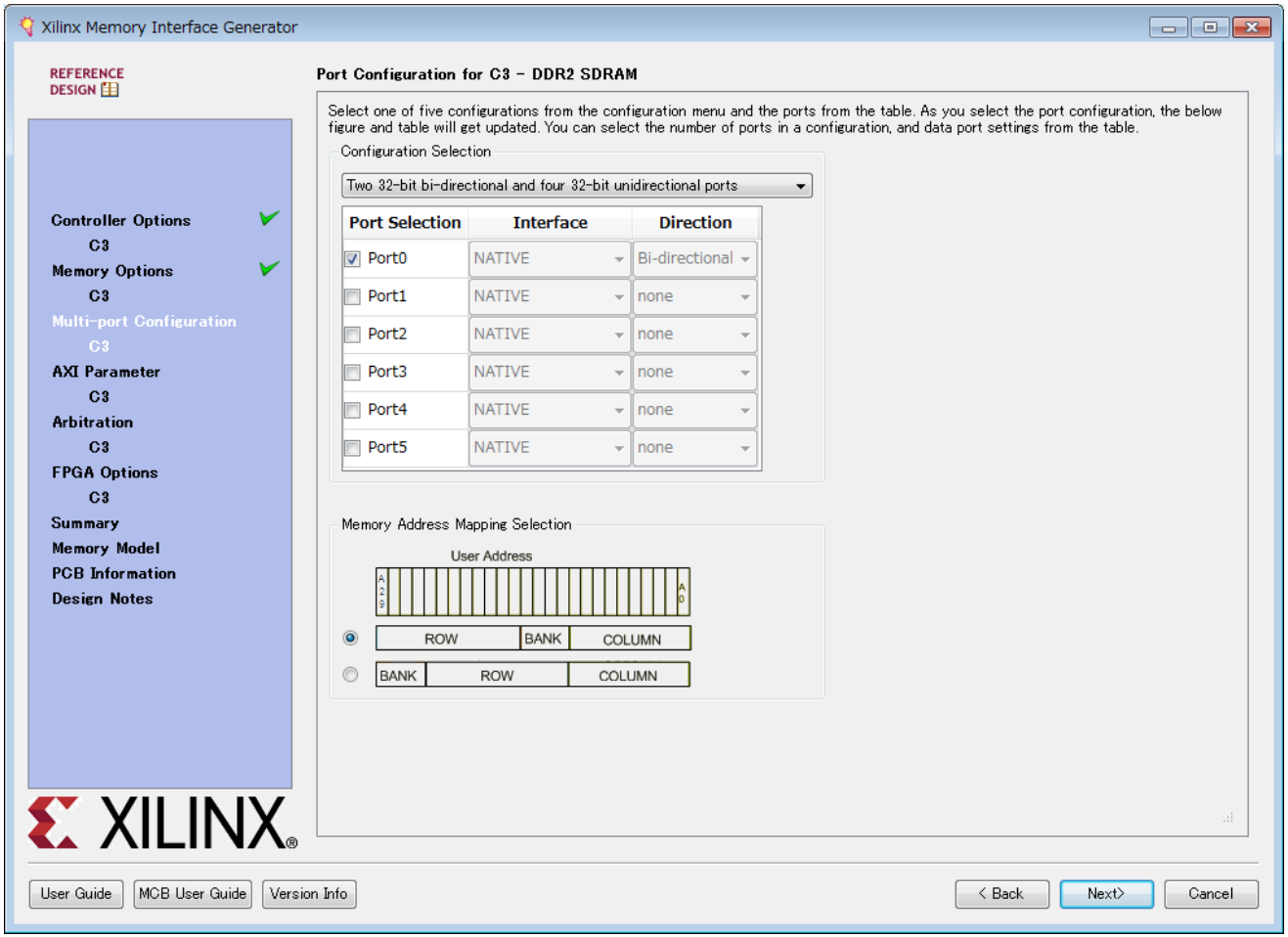
Xilinx Memory Interface Generator (6)

- Fullstrength, 50ohms, Enable, Disable を選択



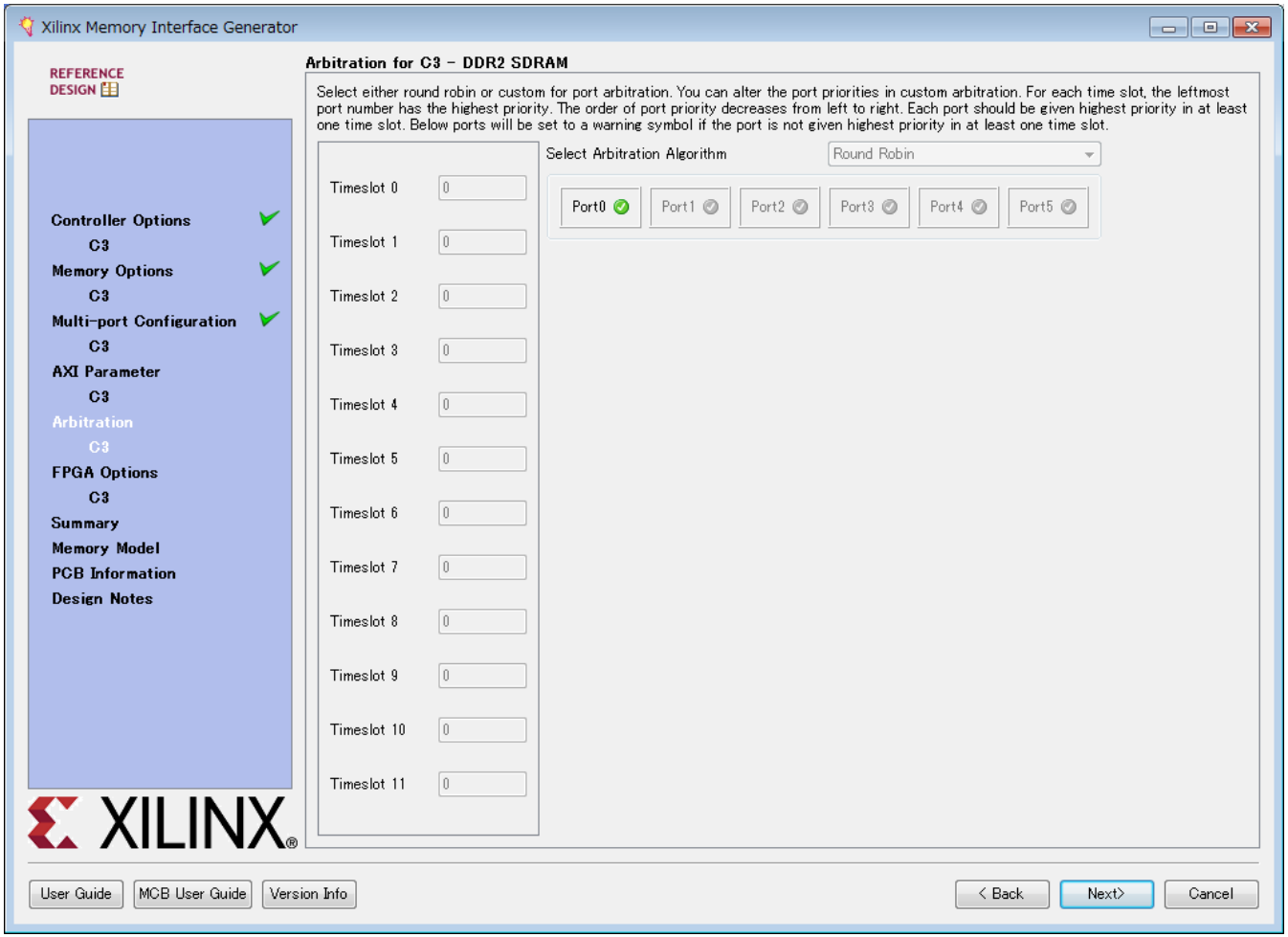
Xilinx Memory Interface Generator (7)

- 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



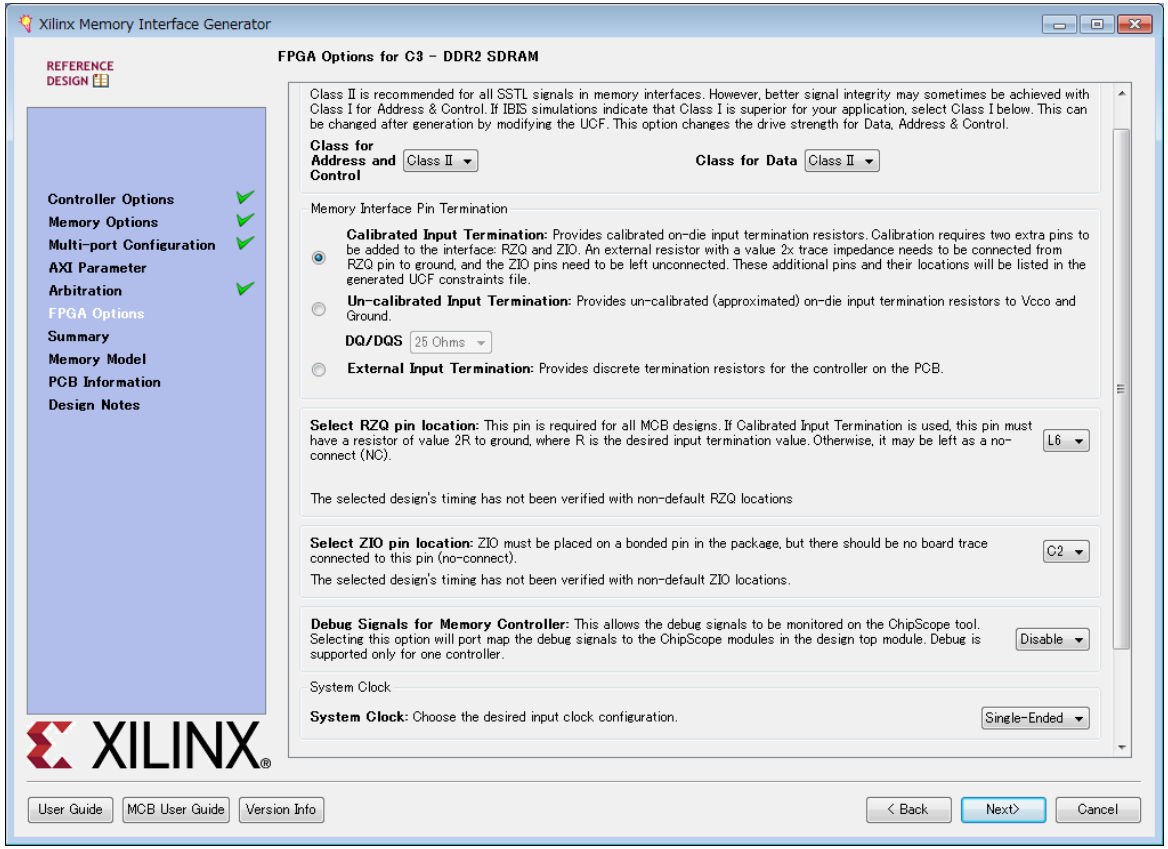
Xilinx Memory Interface Generator (8)

- 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

ISE Project Navigator (P.68d) - C:¥FPGA¥contest¥System_Atlys_t51_dummy¥fpga¥main.xise - [dram.veo]

File Edit View Project Source Process Tools Window Layout Help

Design

View: Implementation Simulation

Hierarchy

- main
- xc6slx45-3csg324
  - Automatic `includes
    - define.v
  - MieruSys (MieruSys.v)
    - clockgen01 - clockgen (clock.v)
    - clockgen02 - clockgen2 (clock.v)
    - resetgen01 - resetgen (clock.v)
    - mem0 - dram_con (dram_mem.
      - u_dram - dram (dram.xco)
    - loader - PLOADER (system.v)
      - serc - serialc (system.v)

No Processes Running

Processes: u_dram - dram

- CORE Generator
  - Manage Cores
  - Regenerate Core
  - Update Core to Latest Version
  - View HDL Functional Model
  - View HDL Instantiation Template

```
58  //
59  // Purpose     : Template file containing code that can be used as a model
60  //               for instantiating a CORE Generator module in a HDL design.
61  // Revision History:
62  //*****************************************************************************
63
64  // The following must be inserted into your Verilog file for this
65  // core to be instantiated. Change the instance name and port connections
66  // (in parentheses) to your own signal names.
67
68  //---------- Begin Cut here for INSTANTIATION Template ---// INST_TAG
69
70   dram # (
71      .C3_P0_MASK_SIZE(4),
72      .C3_P0_DATA_PORT_SIZE(32),
73      .C3_P1_MASK_SIZE(4),
74      .C3_P1_DATA_PORT_SIZE(32),
75      .DEBUG_EN(0),
76      .C3_MEMCLK_PERIOD(3000),
77      .C3_CALIB_SOFT_IP("TRUE"),
78      .C3_SIMULATION("FALSE"),
79      .C3_RST_ACT_LOW(0),
80      .C3_INPUT_CLK_TYPE("SINGLE_ENDED"),
81      .C3_MEM_ADDR_ORDER("ROW_BANK_COLUMN"),
82      .C3_NUM_DQ_PINS(16),
83      .C3_MEM_ADDR_WIDTH(13),
84      .C3_MEM_BANKADDR_WIDTH(3)
85  )
86  u_dram (
87
88      .c3_sys_clk           (c3_sys_clk),
89    .c3_sys_rst_i           (c3_sys_rst_i),
90
91    .mcb3_dram_dq           (mcb3_dram_dq),
92    .mcb3_dram_a            (mcb3_dram_a),
93    .mcb3_dram_ba           (mcb3_dram_ba),
94    .mcb3_dram_ras_n        (mcb3_dram_ras_n),
95    .mcb3_dram_cas_n        (mcb3_dram_cas_n),
```

Start Design Files Libraries | Design Summary (out of date) | dram.veo | MieruSys.v | dram_mem.v

Console

Console Errors Warnings Find in Files Results

Open an existing file | Ln 132 Col 50 | Verilog

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
```

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 2014-07-03

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://aquila.is.utsunomiya-u.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://aquila.is.utsunomiya-u.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://aquila.is.utsunomiya-u.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 2014-07-03

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 SDK-1.1.1.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
近日中に改善する予定です

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 2014-07-03

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）

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 2014-07-03

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
- また、SDKの使用方法について解説します．

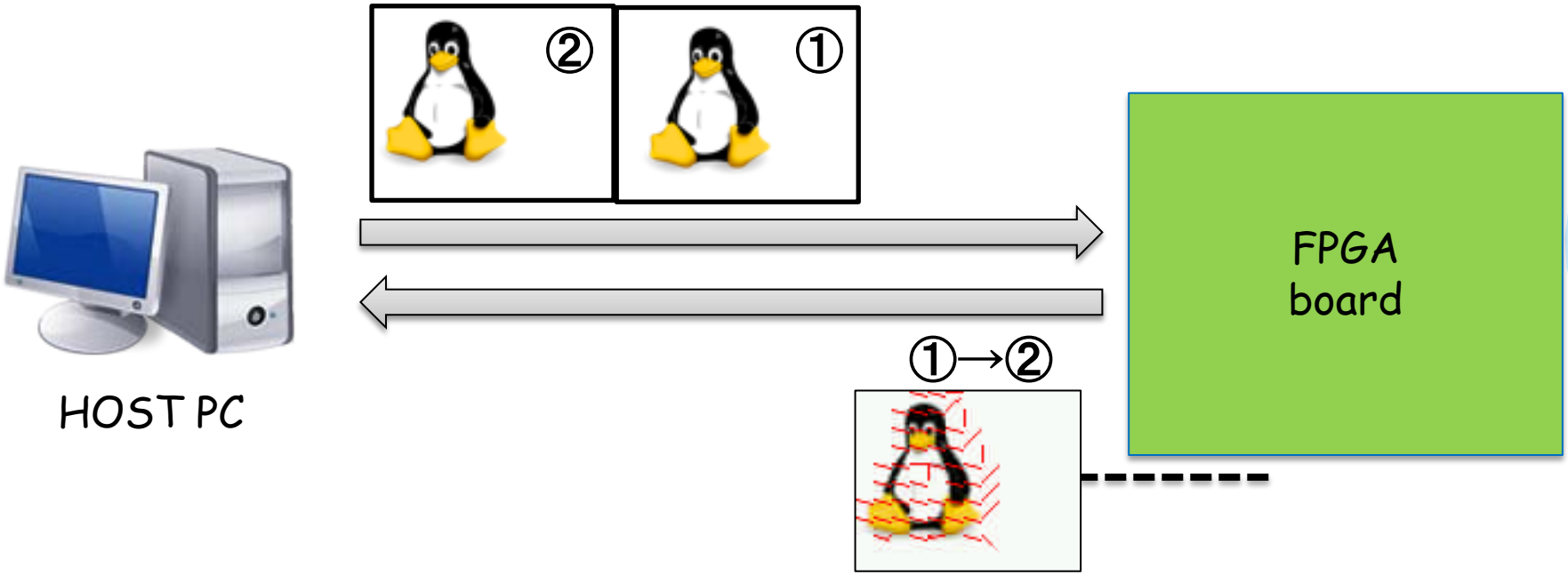
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 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2つ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
（YCrCb成分をJPEGに似た形式で圧縮）



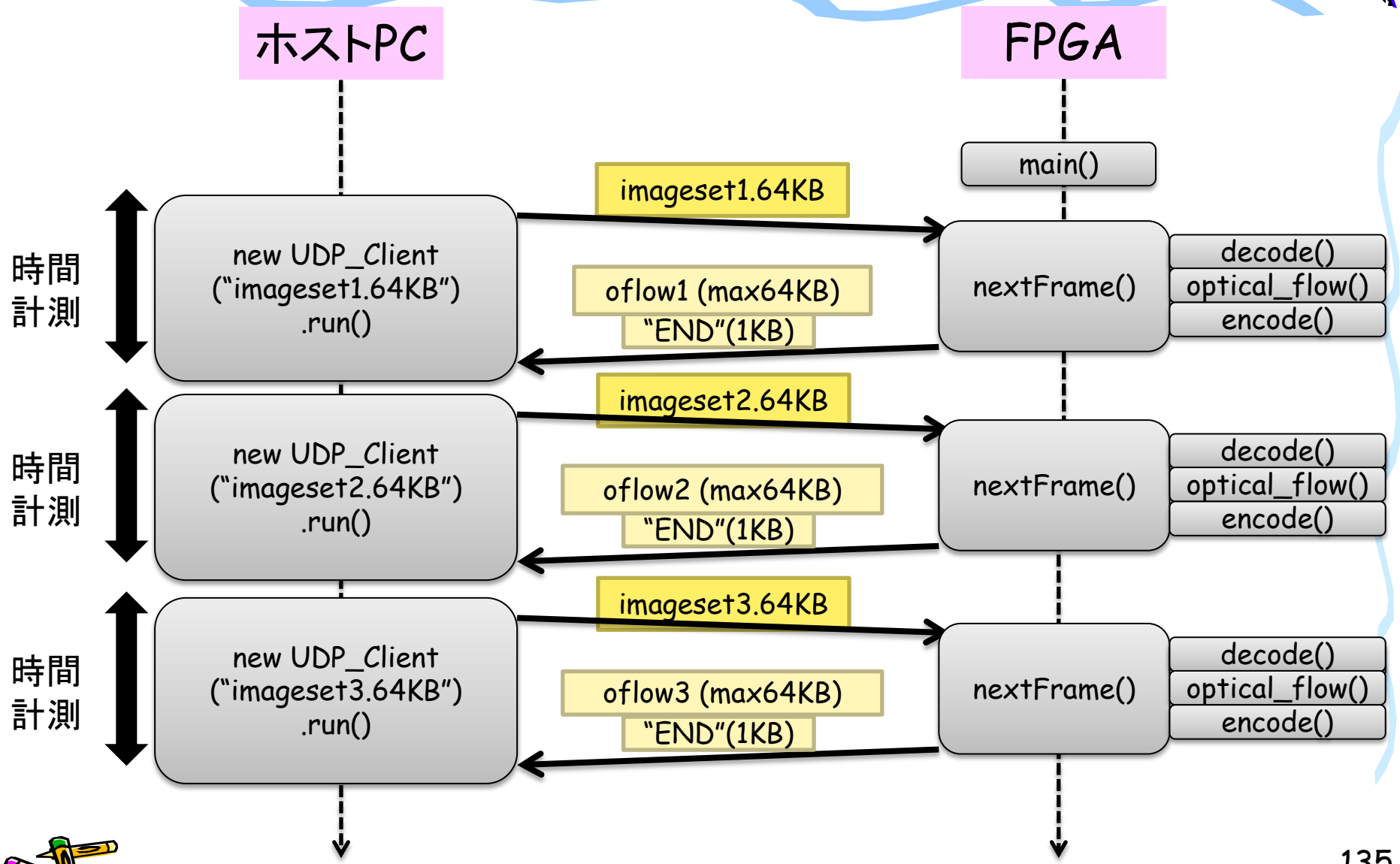
出力画像 （RGB成分を同様の形式で圧縮）

概要

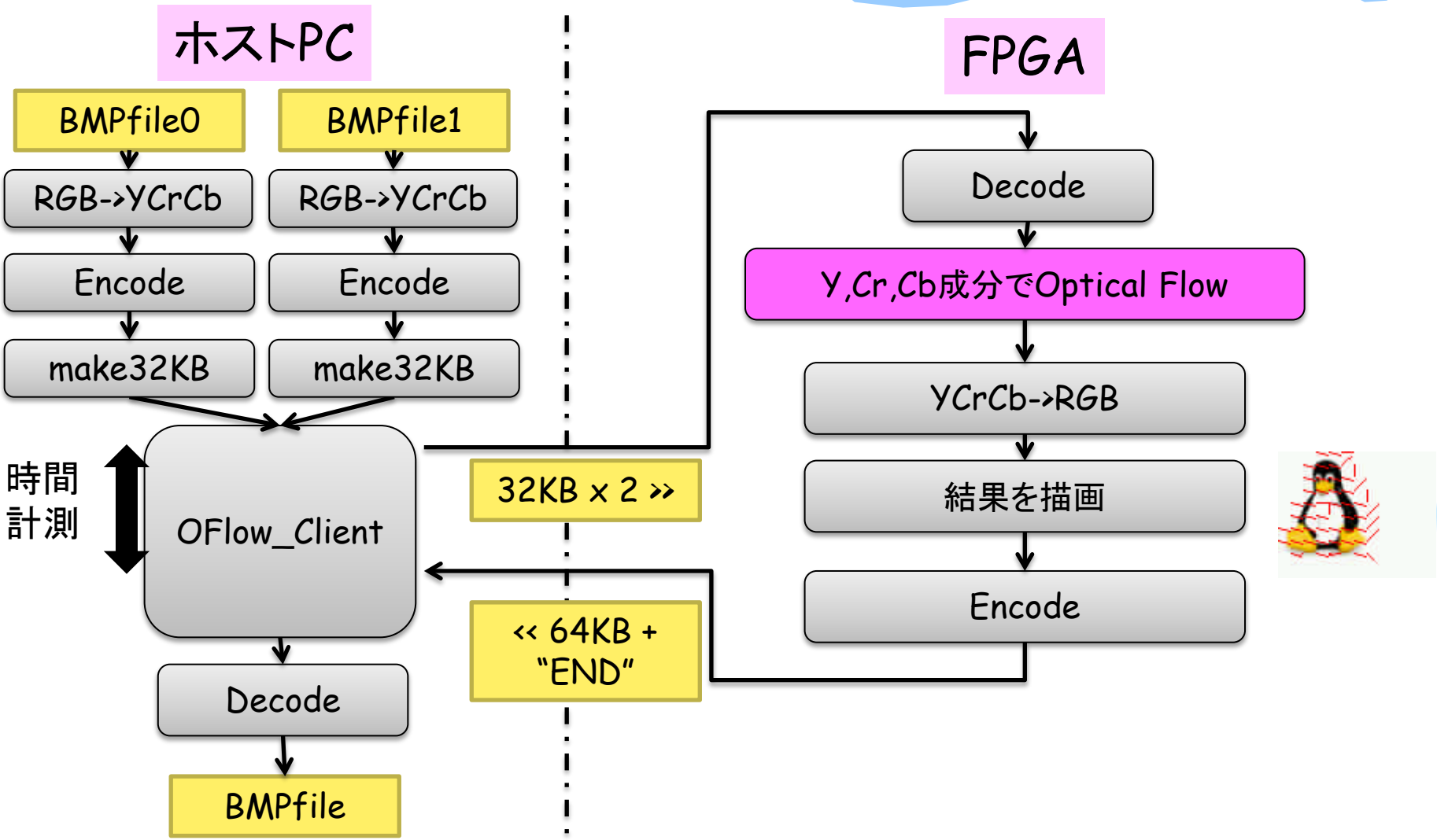
- コンピュータシステム部門では，以下の処理をFPGAボードで行います
  - ホストから2つの入力画像フレーム（32KB x 2）を受け取り、デコードする
    - 画像は独自のJPEGに似た形式で圧縮されたフォーマットで渡されます．
  - 2つのフレーム間のオプティカルフローを計算する
    - オプティカルフローを画像に書き入れる（赤線）
    - 結果画像をエンコードして結果画像フレームにする
  - 結果画像フレーム（max64KB）をホストに送信し、更に“END”（1KB）を送信
    - 注意）UDP・シリアル変換のexStickBridgeは1KB単位でのみパケットを送信

- 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.

FPGA・ホストPC間の通信シーケンス



FPGA・ホストPCの処理フロー



ホストPCでの前処理内容

- 2つの入力画像（410_tux0.bmp，410_tux1.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．
- 2つの入力画像をそれぞれ32KBのサイズに揃えます（0で埋める）
- 2つをあわせて64KBのデータとし、UDP/IP(8100番ポート)で送信します．
  - 送信したデータは、ToUDP.（画像データ名）.binとして保存されます

FPGAでの処理内容

- FPGAでは以下の処理をします.
  - 32KBの2つのデータを受信
  - 2つのJPEG圧縮画像(img0, img1)をデコード
  - デコードされた画像のそれぞれY, Cr, Cb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(img0)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
  - 最大64KBの結果画像データを送信し, 最後に"END"を送る

ホストでの後処理内容　と　時間計測・結果検証

- FPGAからJPEG圧縮された画像をUDP/IP経由で受け取り，ファイルに保存します．ファイル名：FromUDP.(画像データ名).binとして保存します．
- 上記ファイルをデコードし，RESULT.(画像データ名).bmpとして保存します．
- 正しいデータと一致しているか確認します．

- 競技の時間計測について
  - ホストPCにおいて、以下のT2-T1の経過時間を計測します
    - T1: UDPにて64KBの画像データを送信
    - T2: UDPにて“END”を受信
  - その後、BMP画像データが、リファレンスデザインの結果BMP画像データと完全一致しているかを検証します

コンピュータシステム設計部門 リファレンスデザインSDKの 使用方法

- コンテストWEBサイトから、コンピュータシステム設計部門のリファレンスデザインSDK（400_oflow_v10.tgz）をダウンロードして展開。
- 開発環境： LinuxもしくはCygwin
  - http://aquila.is.utsunomiya-u.ac.jp/contest/toolkit.html

開発環境

- ホストPC
  - OS： Windows7
- 動作確認済みのホストPC上のソフトウェア環境
  - java version "1.7.0_60"
- 既知の問題： CentOS6.5をホストPCとして用いた際に，UDP通信が失敗する事が報告されています。

- exStickBridgeでの動作確認済みスイッチ一覧
  - BUFFALO LSW3-TX-5EPL
  - BUFFALO BSL-WS-G2116M
  - （今後追加します）

SDKの使い方 (1) ディレクトリ構成

- ディレクトリ構成の簡単な説明です.
  - common : 共通ソースファイル
  - xilinx : Xilinx用ソースファイル
  - altera : Altera用ソースファイル
  - linux : Linux用ソースファイル
  - client : ホスト用クライアントプログラムのディレクトリ

主なソースファイルの構成

ディレクトリ	ファイル名	概要
Platform (linux, xilinx, altera)	addressmap.h addressmap.c	アドレスマップ
	communication.h communication.c	ホストとの通信
	platform.h platform.c	プラットフォーム固有の処理
	bmp.h, bmp.c	ビットマップファイル関連(linuxのみ)
Common	codec.h codec.c	Codecその他画像の変換関連のライブラリ
	fileio.h fileio.c	メモリファイル入出力 （stdioのFILE置き換え）
	image.h, image.c	画像関連ライブラリ
	main.c	メイン
	oflow.h, oflow.c	オプティカルフロー関連処理

ユーティリティツール一覧

ディレクトリ	ファイル名	概要
Linux	decoder.c	デコーダ
	encoder.c	エンコーダ
	make32KBdata.c	32KBデータ作成
	rgb2ycbcr.c	RGB→YCrCb色変換

SDKの使い方 (2) Linuxプログラムの動作確認-1

- FPGAボードが無くても，リファレンスデザインの機能を確認することが出来ます
- トップディレクトリでのmake コマンドにより，FPGAボードでの動作を模擬するLinux上で動作するプログラムを作成することが出来ます．

```
$ make
```

- 出来上がる主なファイルは次の通りです．
  - 400_oflow　　：Linux用のオプティカルフロー処理プログラム
  - encoder　　：BMPファイルのJPEG風エンコーダ
  - decoder　　：BMPファイルのJPEG風デコーダ
  - rgb2ycbcr　　：BMPファイルの色変換(RGB -> YCrCb)
- トップディレクトリで，以下のコマンドにより400_oflowプログラムを起動します．

```
$ ./400_oflow
```

- 以下の様に、ホストプログラムからの画像データを待ち受ける状態になります．
  - UDP/8100ポートにて32KBバイトの画像データ2セットを待ち受けます．

```
$ ./400_oflow
Starting 400_oflow (batchID:410) ===
Waiting recv 32KB at 0906c008
```

SDKの使い方 (2) Linuxプログラムの動作確認-2

- 次に，別ウィンドウを開き，以下のコマンドでホストプログラムを起動します．

```
$ cd client
$ ./all.sh
```

- all.shは画像データセットを送るスクリプトを4つ順次起動します．
  - (410_tux.sh, 411_anim.sh, 412_star.sh, 413_rectangle.sh)
  - この際，スクリプトの内部では，javaのUDP送信プログラムを起動します．
  - all.shスクリプトの引数には，UDPパケットの送信先IPアドレスを指定可能

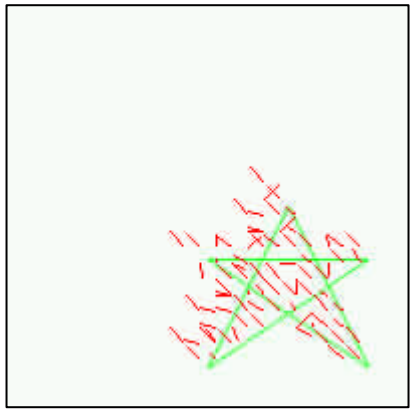
```
$ ./all.sh
targetHost:127.0.0.1 targetPort:8100
UDP Socket created. sending file '410_tux.64KB'
started!
64KB sent

finished!
after-before = 3331318965 (ns) = 3.331318965(s)
（続く）
```

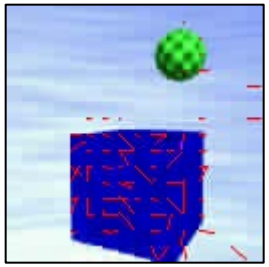
SDKの使い方 (2) Linuxプログラムの動作確認-3

- 正しく動くと、4回のオプティカルフロー計算処理が走ります。
- 結果は、ホストPC側（クライアント側）にRESULT.***.bmpファイルとして保存されます。

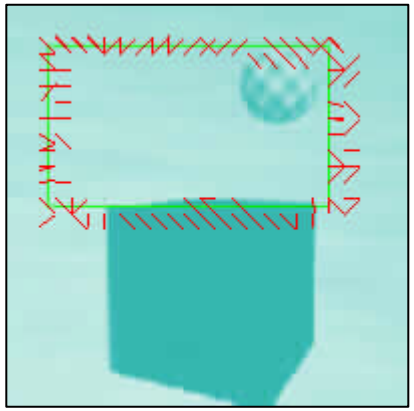
410_tux



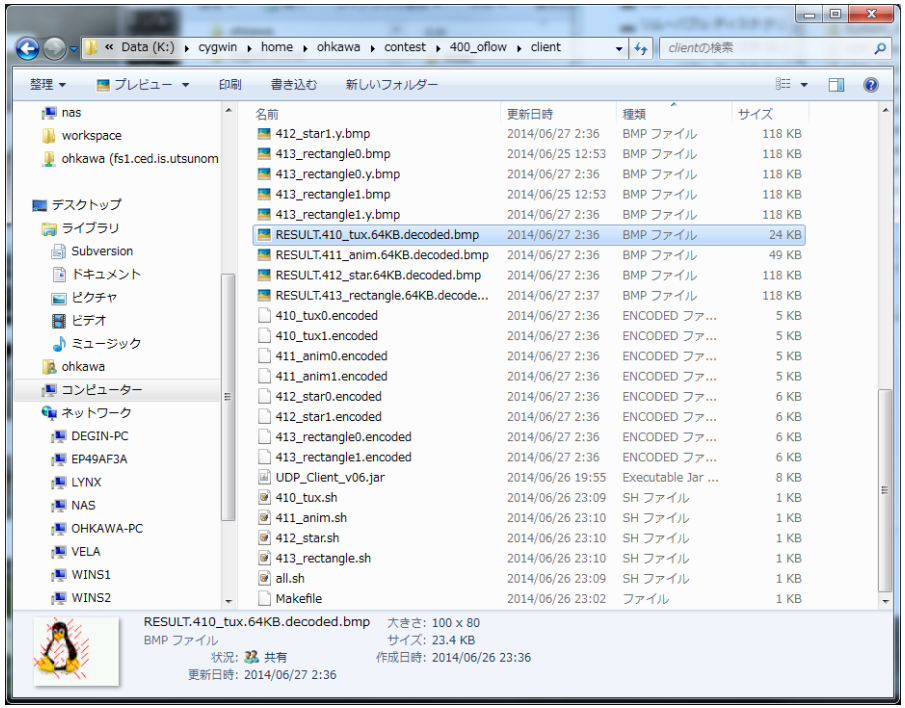
412_star



411_anim



413_rectangle



SDKの使い方 (3) Xilinx/Altera FPGAボード向けソースファイルのexport方法

- FPGAボード上で動作するソフトウェアのソースファイルをexportするには，以下の様にします。

```
$ make export
```

- exportディレクトリ以下に、xilinx/srcディレクトリとaltera/srcディレクトリが出来ますので、その中の.cファイル・.hファイルを，Xilinx MicroBlaze環境もしくはAltera Nios2環境にコピーして使用してください。

- 以降は，別ドキュメントにて，Xilinx/Altera環境での動作方法について説明します.
  - P61　Xilinx環境
  - P62　Altera環境

SDKの使い方 (4) 既知の問題

- 現在のリファレンスデザインには，以下に挙げる問題があることが分かっています．注意してお使いください．
  - 元画像(BMP)をエンコードし，デコードすると元画像に戻らない場合がある．（特に画像サイズが大きい場合）
  - ホストPCとしてCentOS6.5の環境とFPGAボードの間でUDP通信を行った場合に、途中で停止することがある。（Atlysボード・DE2-115ボードいずれの場合においても不具合が発生する様子。・現在調査中）

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．

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

User Manual of Optical Flow System Reference Design (Xilinx ATLYS)

コンテスト実行委員会コアチーム

Version 2014-07-03

このドキュメント

- このドキュメントでは，Atlysボード用のリファレンスデザインに含まれるシステム構成について説明します．
- また，Xilinx Platform Studio (ISE14.7)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）を生成し、プログラムしたFPGA上で400_oflowソフトウェアを動作させる方法を示します．

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

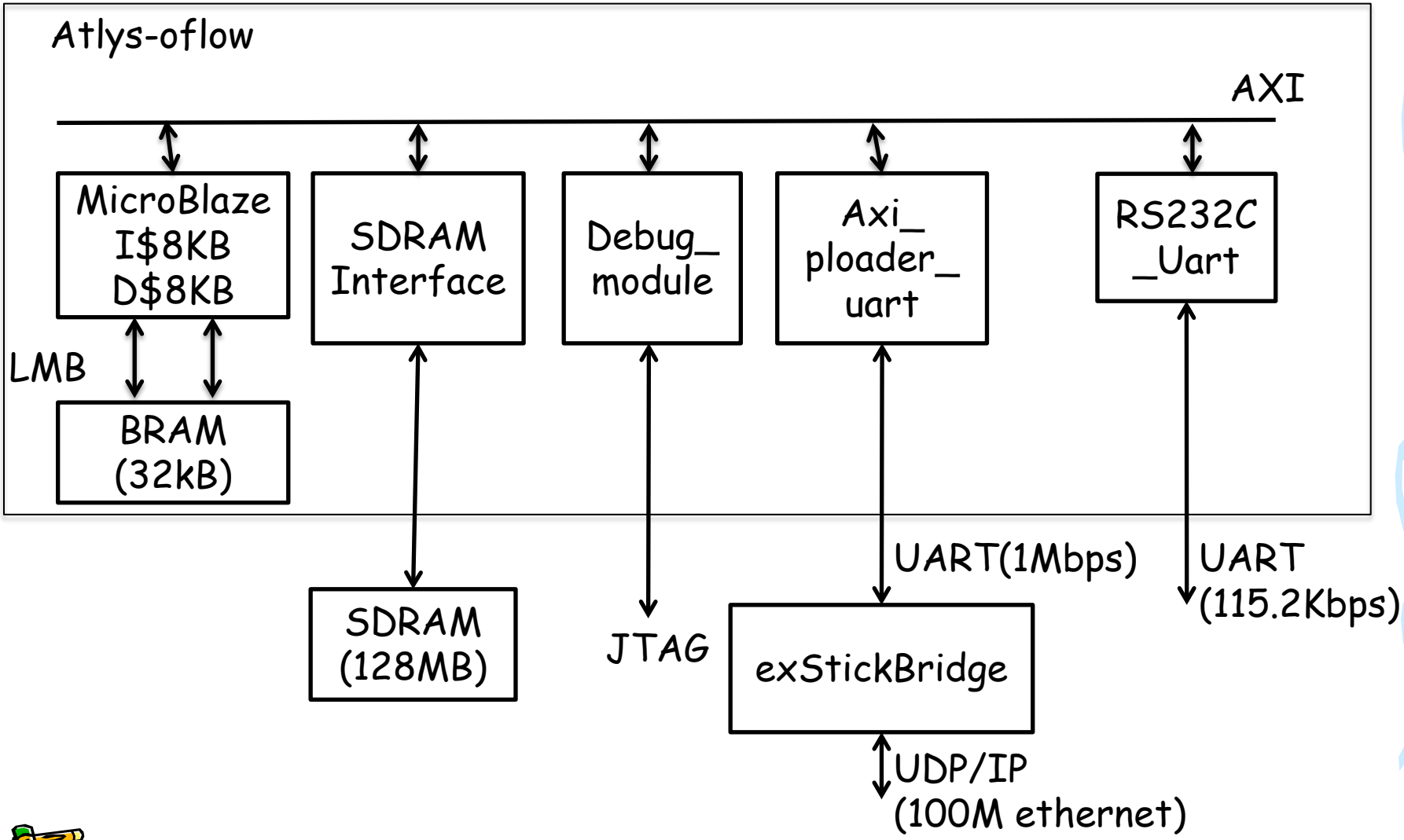
- 不明な点は，以下のいずれかの方法でお問い合わせください．
  - メールアドレス(contest_support@virgo.is.utsunomiya-u.ac.jp)
  - twitter(#arc_procon)
  - 技術情報掲示板
    - Google Group: HpCpsyDC2014
    - https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc

最初に

- 開発環境については、ドキュメントP30を参考にしてください。

- Xilinx版とAltera版は機能的には同じですが，異なる所が多々ありますので，注意して下さい．
- Xilinx版のリファレンスデザインは，ISE14.7のEDK Xilinx Platform Studioで開発できます．ISE WEB Packでは開発できませんのでご注意ください．
- 大学関係者であればXilinx University ProgramのISEライセンス申請が可能です．
  - http://japan.xilinx.com/support/university.html
- またISEの無償評価版ライセンスがXilinxのホームページより申請可能です．
  - http://japan.xilinx.com/ise_eval/index.htm

リファレンスデザインAtlys-oflowのブロック図



システム構成の概要

- 本システムの構成は、Digilent社のWEBサイトにて提供されている，“Atlys board support files for EDK BSB wizard”を用いて作製しました.
  - http://www.digilentinc.com/

- 構成要素
  - MicroBlazeソフトコアプロセッサ
    - 設定）浮動小数点なし，32ビット乗算器搭載，I$8KB，D$8KB
  - Block RAM (BRAM) 32KB
  - SDRAM I/F (128MB)
  - MicroBlaze Debug Module (MDM) – JTAGによるデバッグ接続
  - RS232C_Uart
    - AtlysボードのUART-USB変換用のシリアル通信（115.2Kbps）
  - Axi_ploader_uart
    - 本コンテスト用のシリアル通信(1Mbps)
      → PMODコネクタ経由でexStickBridgeに接続

メモリマップ

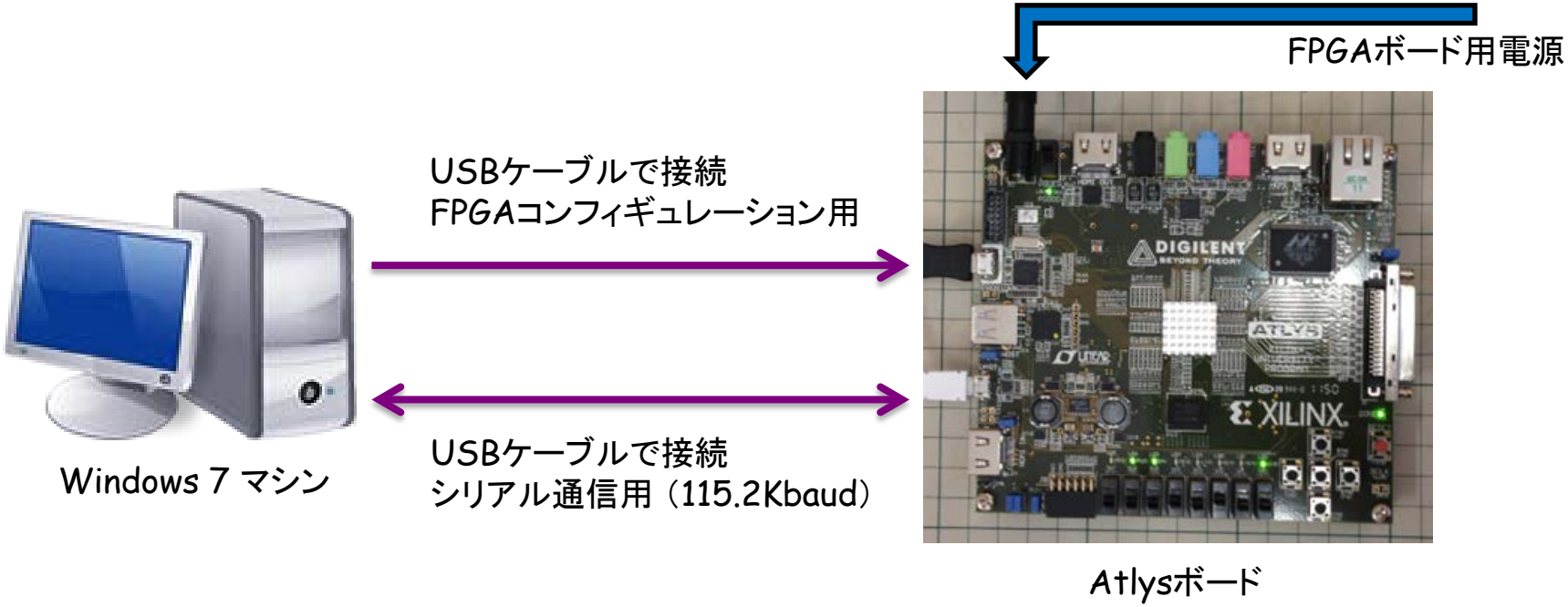
メモリアドレス		
開始アドレス	終了アドレス	
0x0000_0000	0x0000_7FFF	Block RAM (on-chip)
0x4060_0000	0x4060_FFFF	RS232_Uart (115.2kbps USB-UART)
0x77A0_0000	0x77A0_FFFF	Axi_ploader_uart (1Mbps UART via PMOD)
0xA800_0000	0xAFFF_FFFF	SDRAM (128MB)

コンピュータシステム設計部門 ATLYSボード用ハードウェアリファレンスデザインの使用方法

- コンテストWEBサイトから、コンピュータシステム設計部門のXilinx FPGA - Atlysボード用のリファレンスデザインAtlys-oflow_v02.zipをダウンロードして展開します。
  - 展開先例）C:¥workspace-edk147¥Atlys-oflow
  - http://aquila.is.utsunomiya-u.ac.jp/contest/toolkit.html

ハードウェアセットアップ

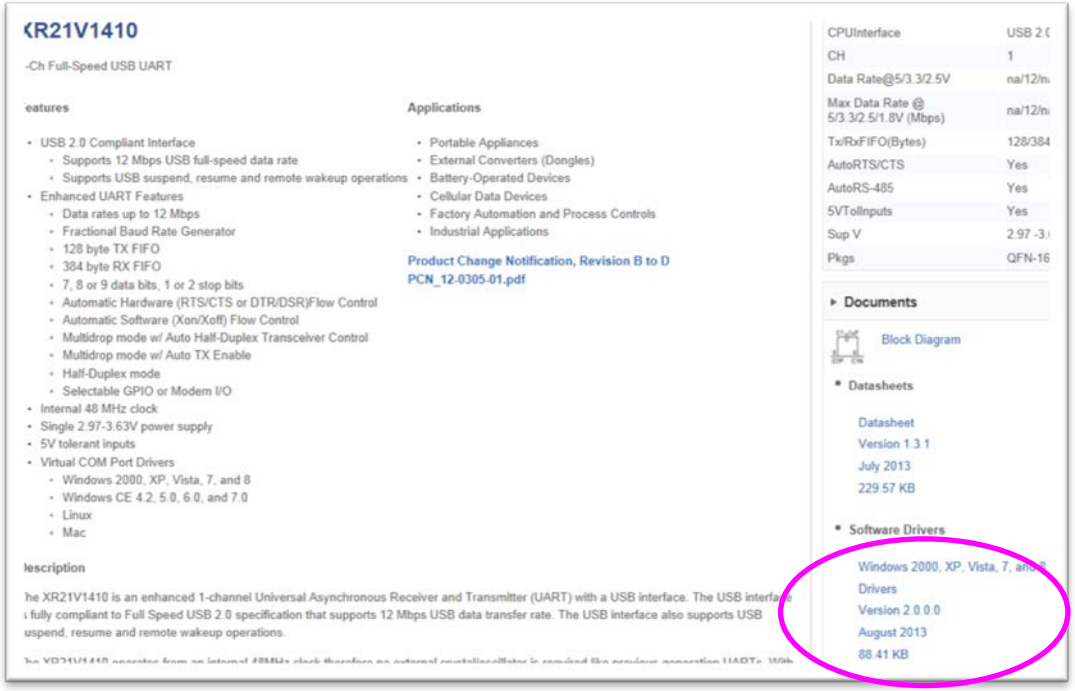
- Windows 7 マシン と Atlysボード を2本の USBケーブル で接続
- FPGAボード用電源 を使ってAtlysボードに電源供給



- 注意点
  - AtlysボードにはUSBケーブルが1本しか添付されていません．別途 1本のUSBケーブル（Aコネクタ-マイクロBコネクタ）を準備してください．

準備 USB-UARTドライバのインストール

- Windows 7 マシンにUSB-UARTドライバをインストール
  - http://www.exar.com/connectivity/uart-and-bridging-solutions/usb-uarts/xr21v1410
    - Software Drivers Windows 2000, XP, Vista, 7, and 8 Drivers Version 2.0.0.0 August 2013

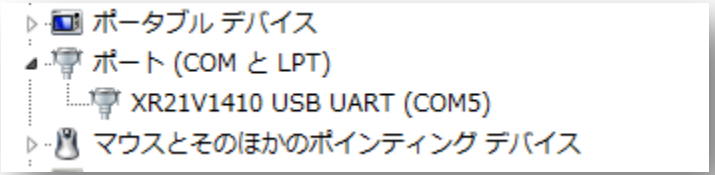


ソフトウェアセットアップ Tera Term のインストール

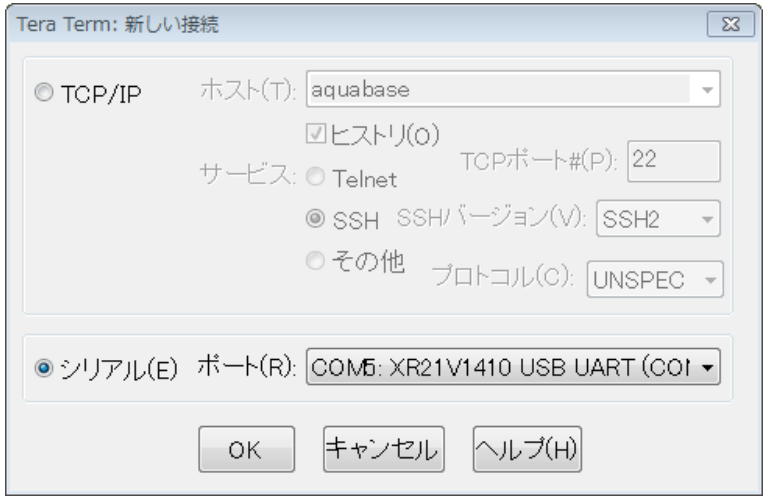
- Windows 7 マシンに Tera Termをインストール
  - http://sourceforge.jp/projects/ttssh2/
  - teraterm-4.78.exe

Tera Term の設定（1）

- ATLYSをコンピュータに接続し，COMポートが見えることを確認
  - Windows の デバイス マネージャー から確認
  - ポート番号(COM5)はコンピュータによって異なります．

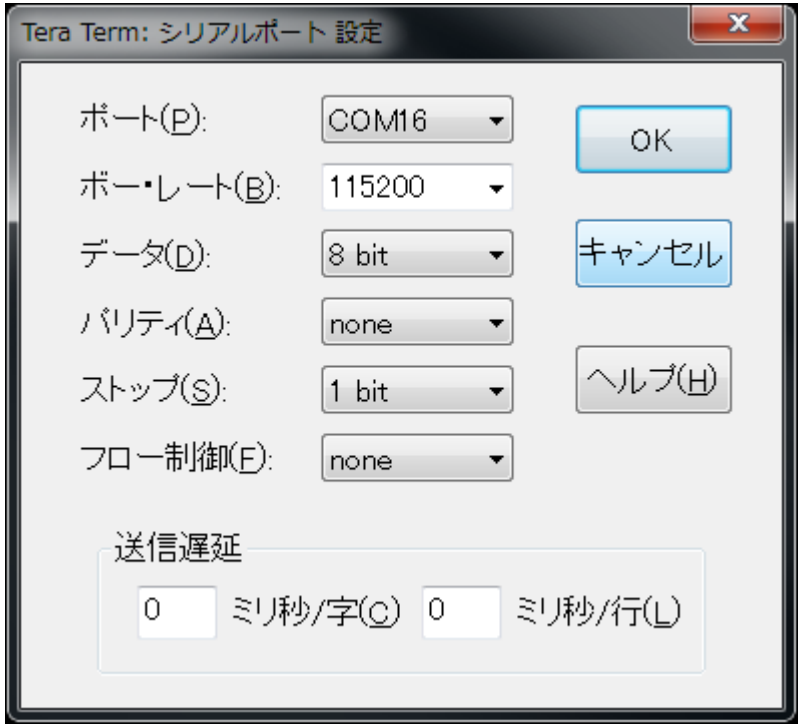


- Tera Term を立ち上げる(シリアル ポートを選択)



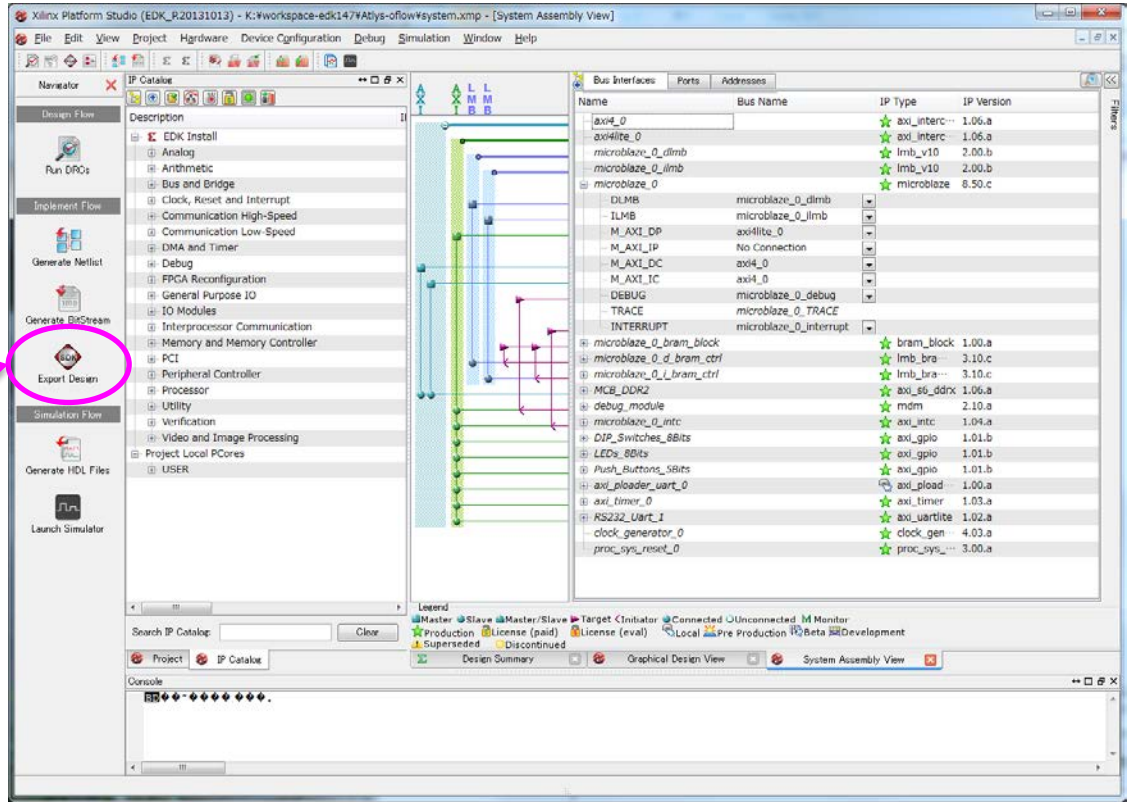
Tera Term の設定（2）

- Tera Termのシリアル設定を 115.2Kbaud に変更します.
  - 設定 -> シリアルポート -> ボー・レート
    115200を選択して，OK をクリック.

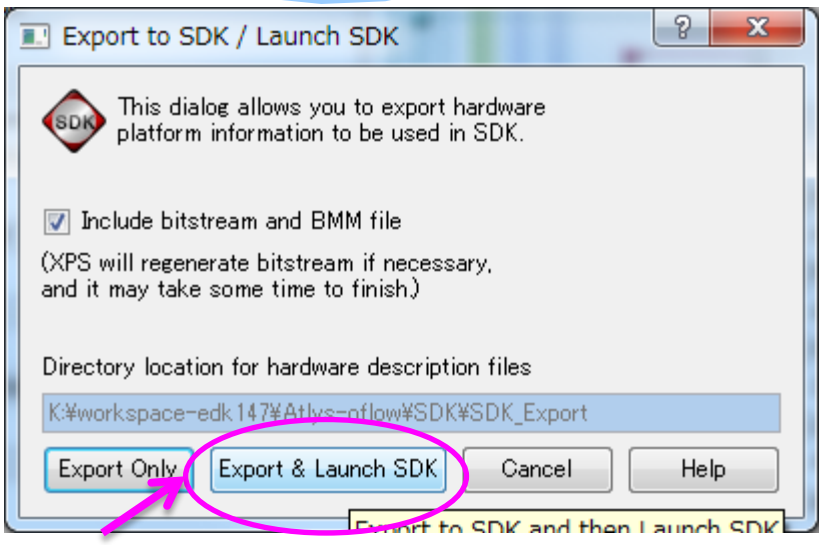


Xilinx Platform Studio (XPS)の起動とExport

- ISE Desugin Suite 14.7 → EDK → Xilinx Platform Studioを起動し、Atlys-oflow/system.xmpを開く
  - 以下、C:¥workspace-edk147¥Atlys-oflowに配置したとする
- 以下の画面になるので、"Export Design"をクリック



Export to SDK, XSDKの起動



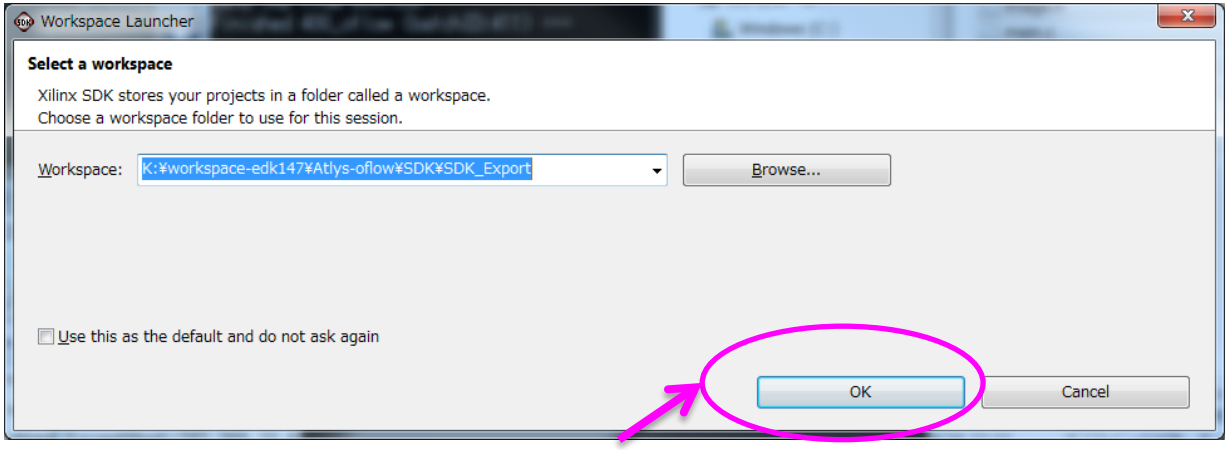
← Export & Launch SDKをクリック

＜HDL生成・合成・配置配線・ビットストリーム生成に約20分程度かかる＞

↓その後、以下の画面が出るので、

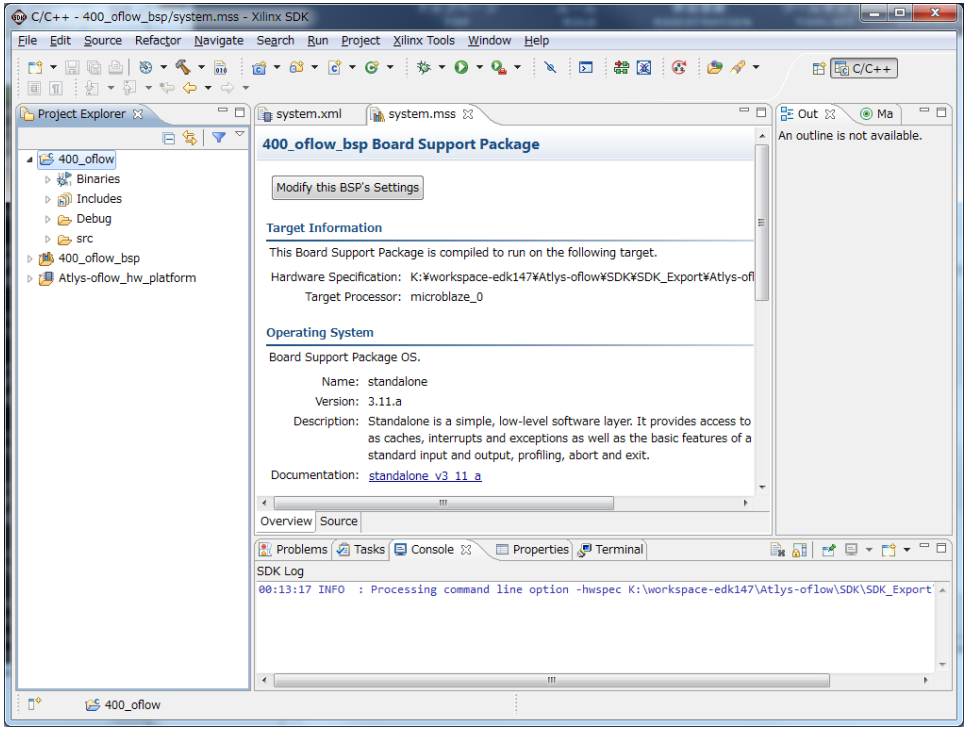
プロジェクトディレクトリの下の、"SDK/SDK_Export"ディレクトリを指定する。

例）C:¥workspace-edk147¥Atlys-oflow¥SDK¥SDK_Export



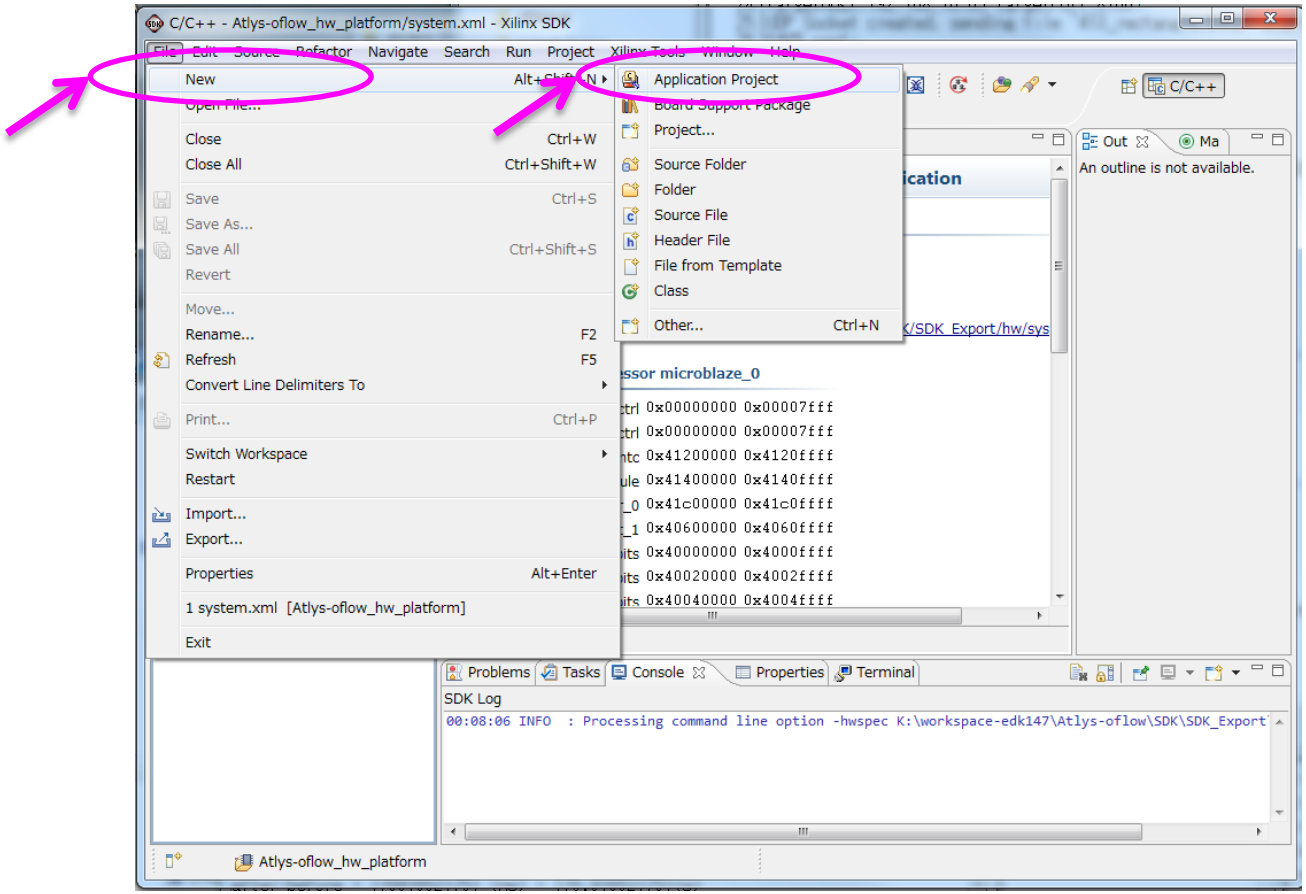
Xilinx SDKの起動

- 以下のような画面が現れるはず
- 次ページ以降は、リファレンスデザインのSDK(400_oflow_v02.tgz)からexportした.c, .hファイルを用いてアプリケーションを作成する方法を説明する。



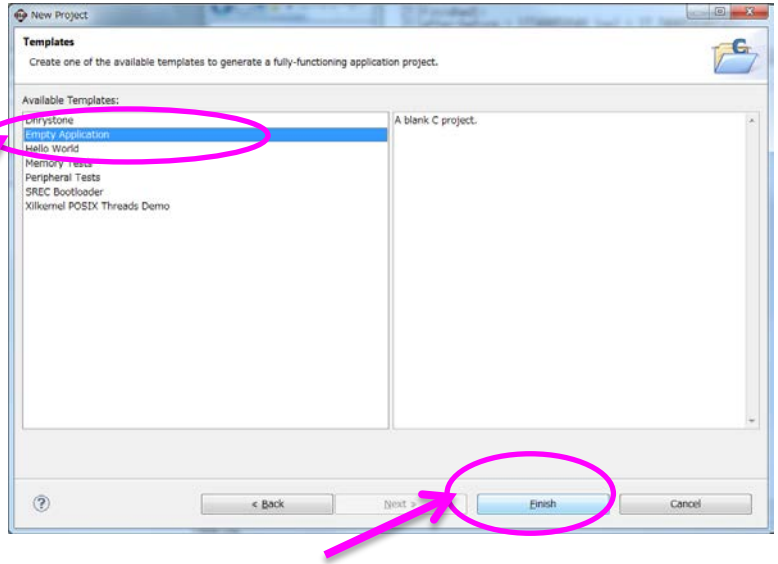
アプリケーションプロジェクトの作成 (1/3)

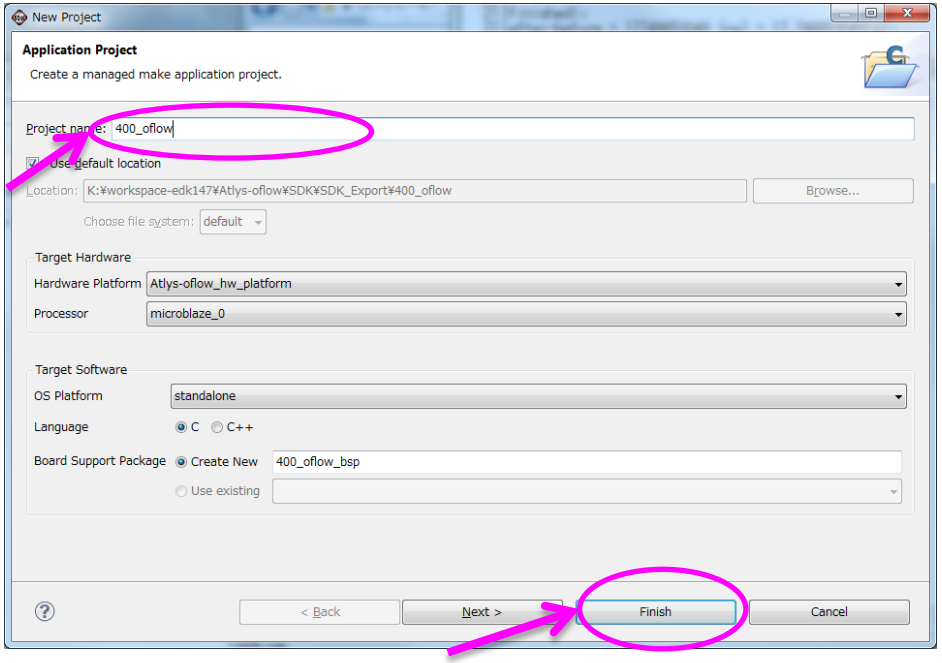
- メニューFile→New..から、Application Projectを選択



アプリケーションプロジェクトの作成 (2/3)

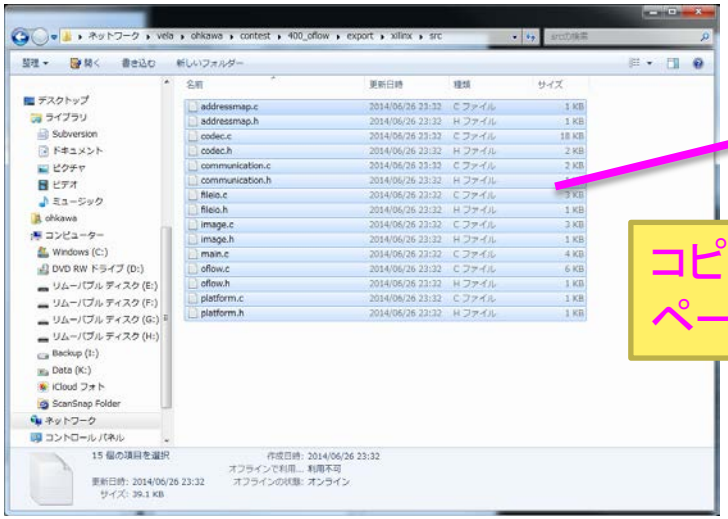
- プロジェクト名を設定（例：400_oflow）
  - Nextをクリック
- 次の画面で、Empty Projectを選択
  - Finishをクリック





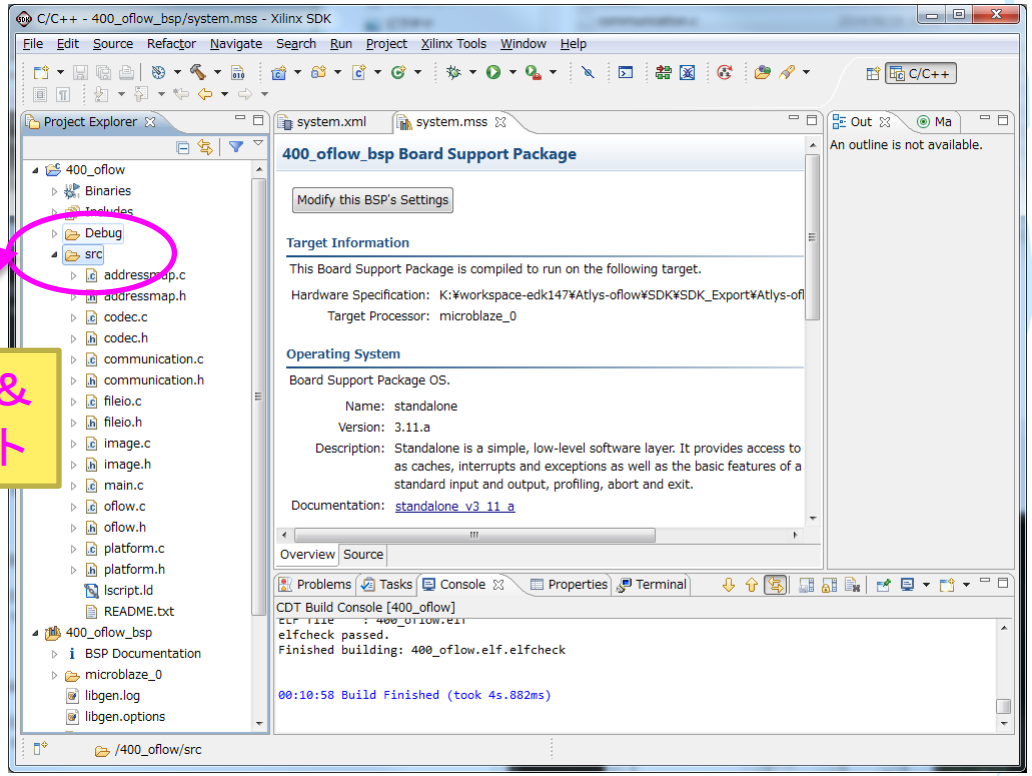
アプリケーションプロジェクトの作成 (3/3)

- SDK(400_oflow_v02.tgz)からexportした.c, .hファイルをコピー(Ctrl+C)
- XSDKの400_oflowプロジェクトのsrcディレクトリをクリックした後に、ペースト(Ctrl+V)
  - Drag&DropでもOK
- 自動的にビルドが始まる



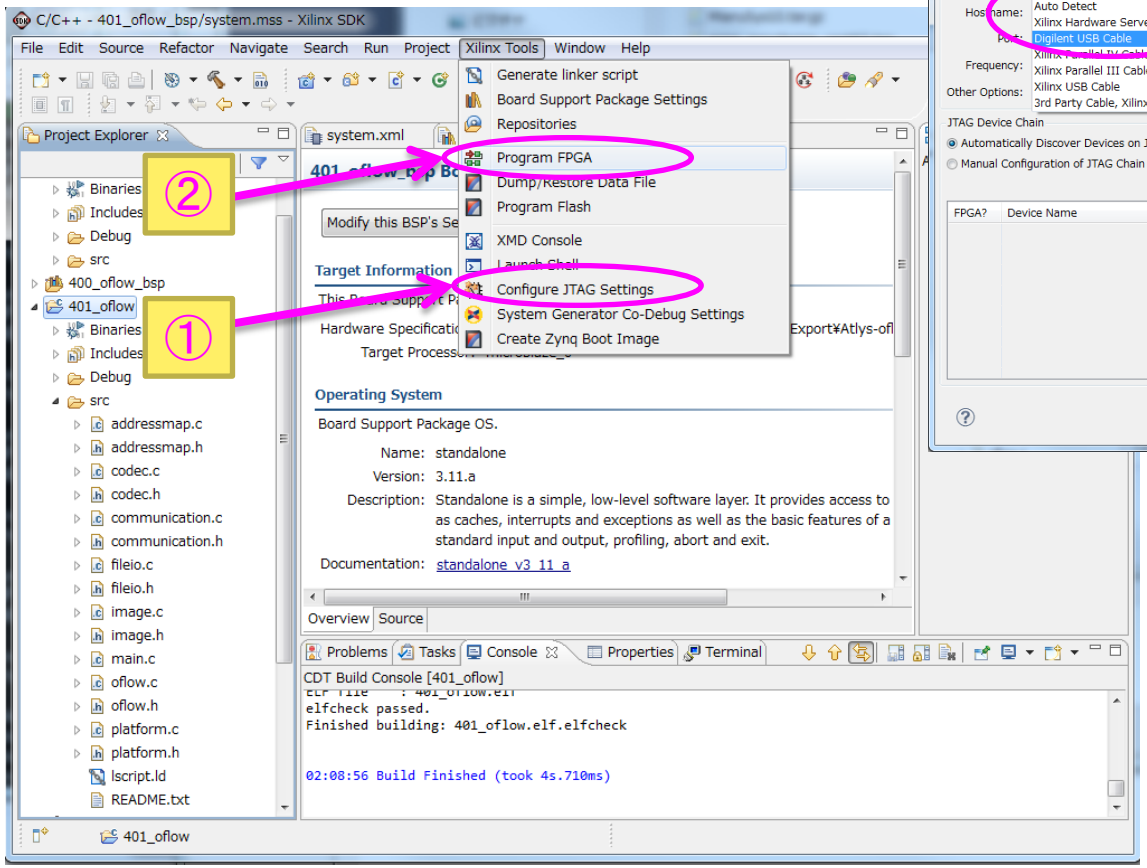
400_oflow¥export¥xilinx¥src

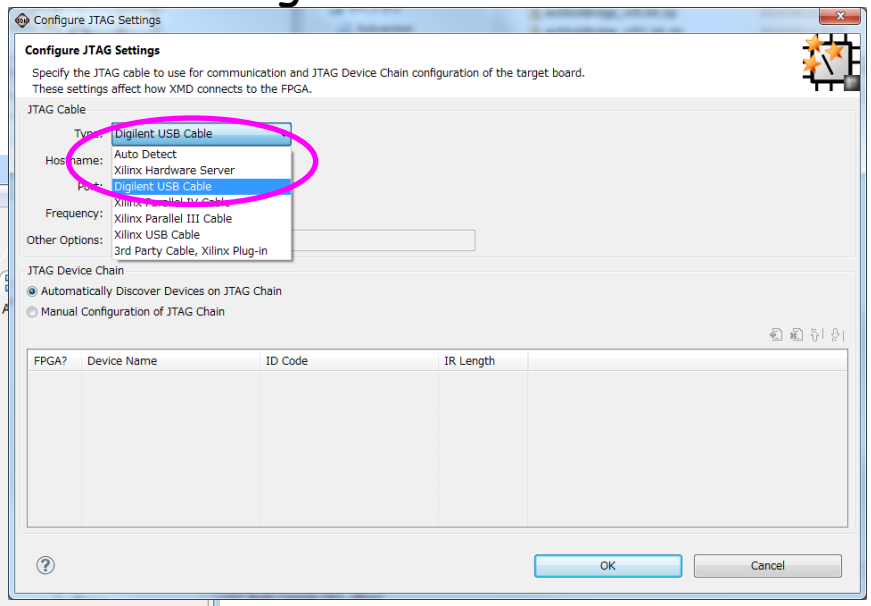
コピー＆ペースト

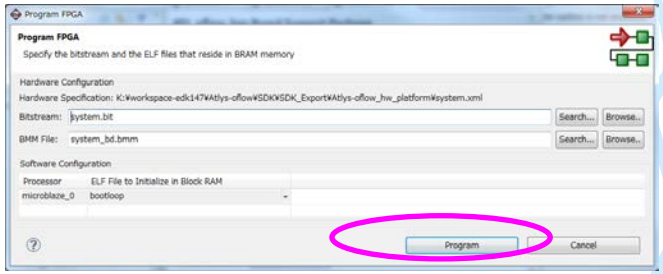


JTAGケーブル設定＆FPGAプログラム

- ①JTAGの設定をします。右のWindowが出るのでDigilentを選択
- ②Program FPGAして下さい。
  - 次のWindowでProgramをクリック

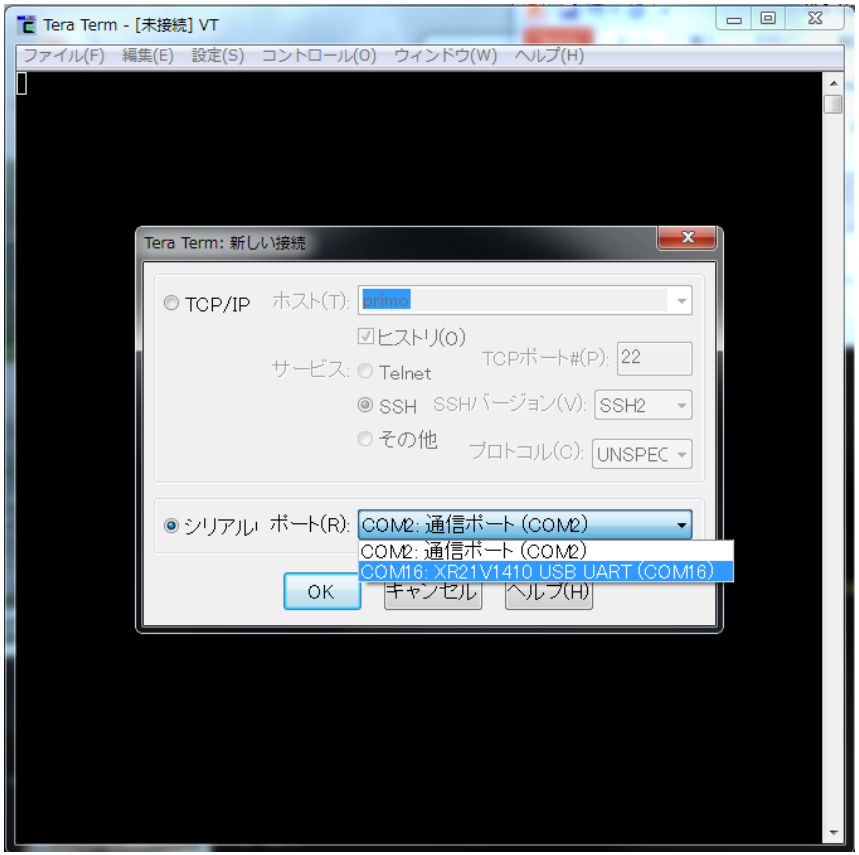






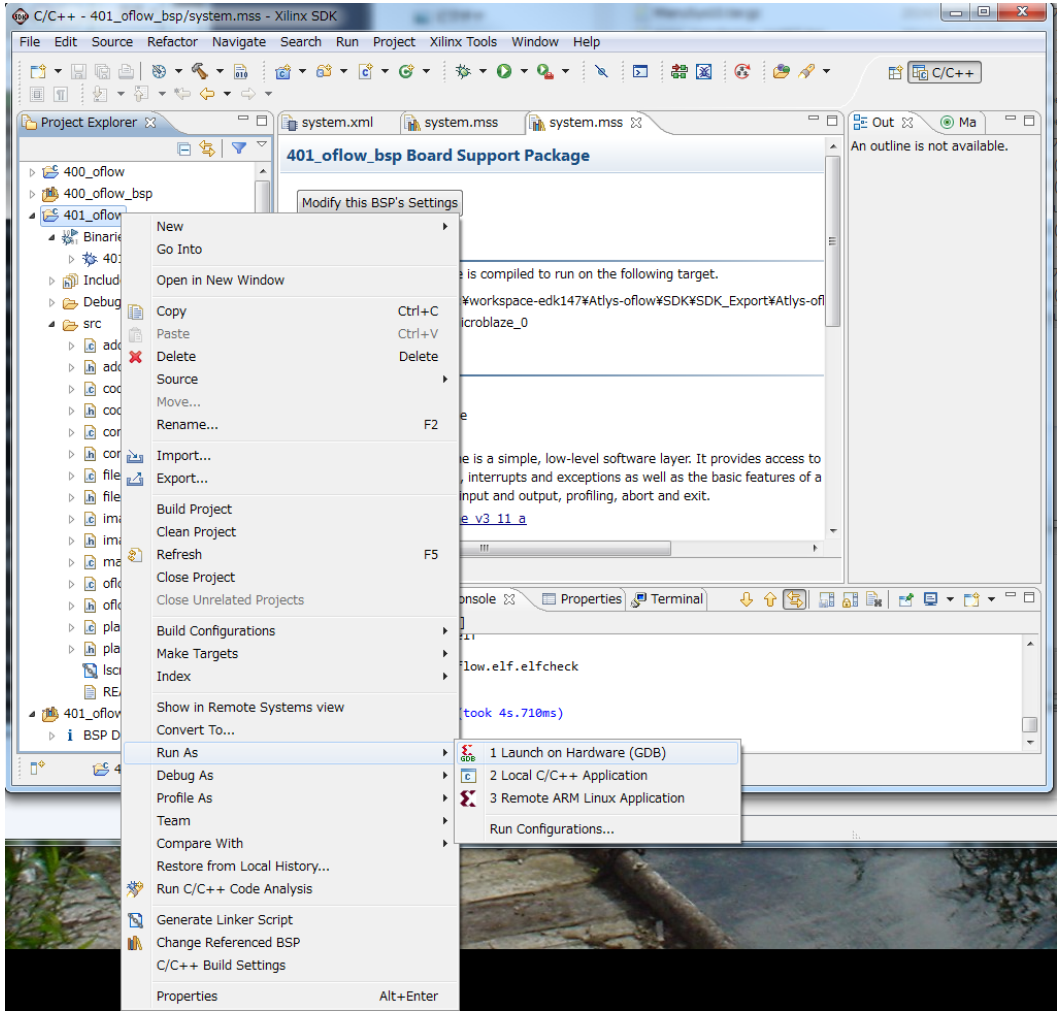
デバッグ用シリアルコンソールの接続(TeraTerm)

- プログラム実行前にTera Termにて、シリアルコンソールを立ち上げておきます。



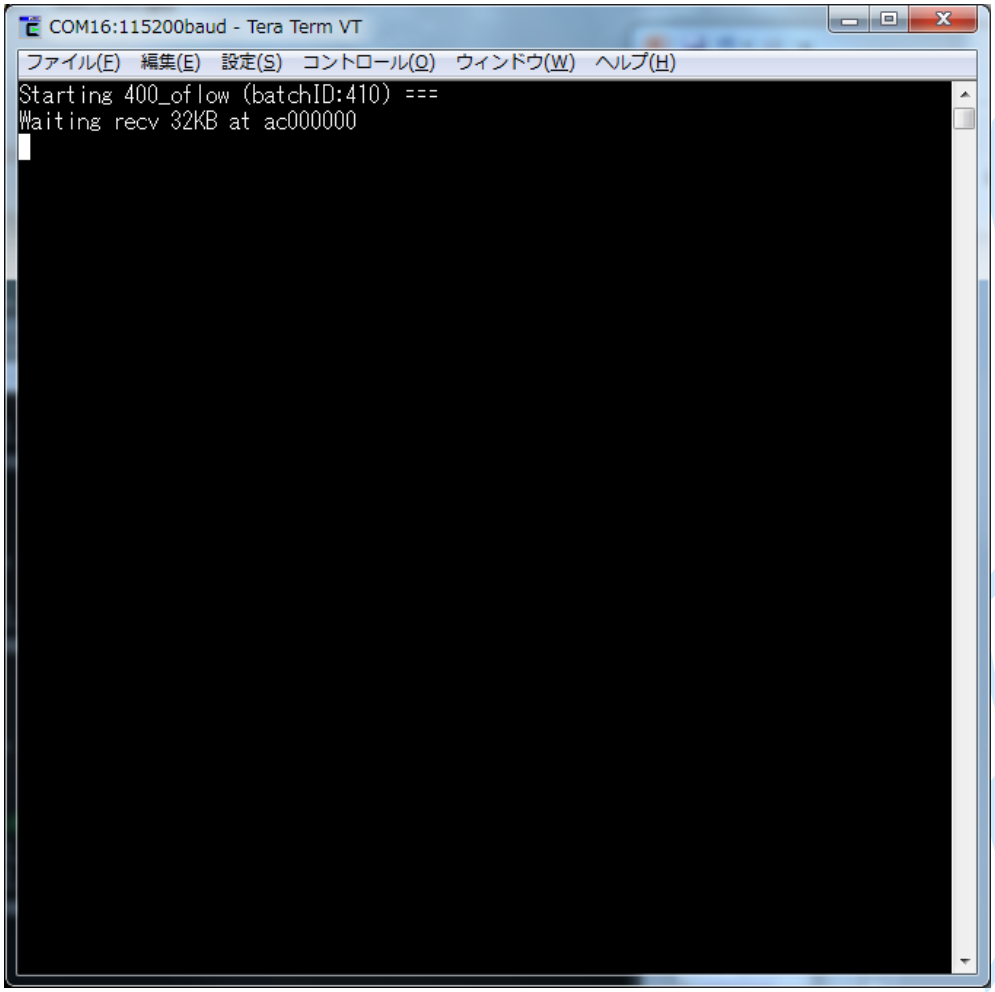
プログラムの実行

- プロジェクトを右クリックし
- Run Asのメニューを選び
- Launch on Hardware (GDB)をクリックしてください



400_oflowの起動

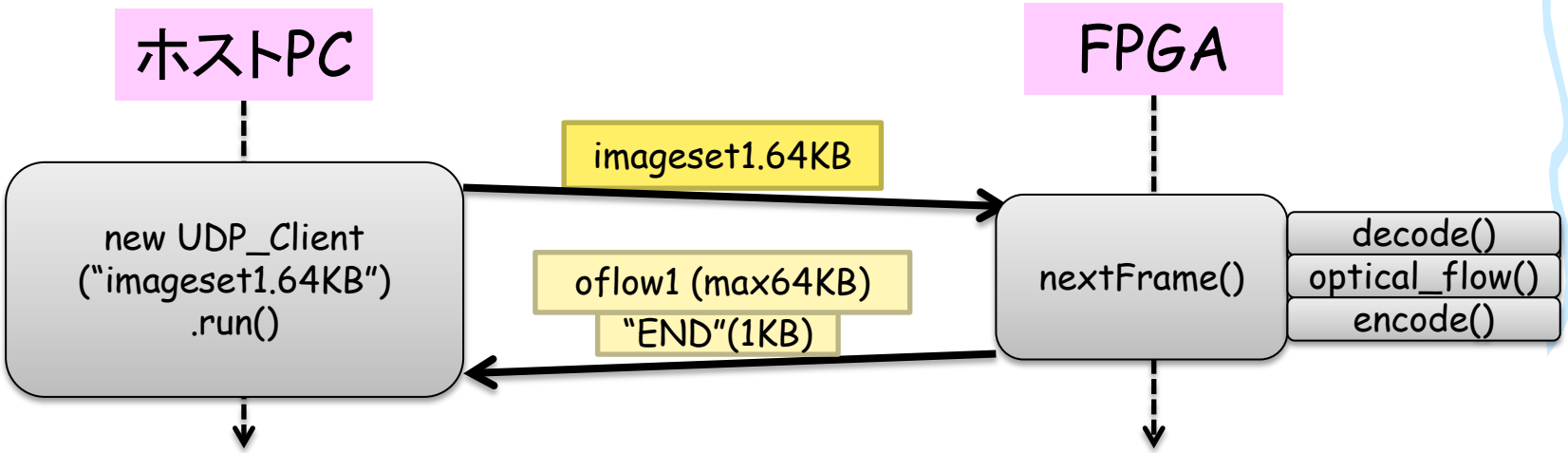
- Starting 400_oflow・・・という起動メッセージが表示されるはず
  - 表示されない場合は
    ここまでの作業を再確認
- 32KB＋32KBの画像データを
  UDP/8100番ポートで
  待ち受けています
  （exStickBridge経由）



ホストPCでのUDP通信プログラムの起動

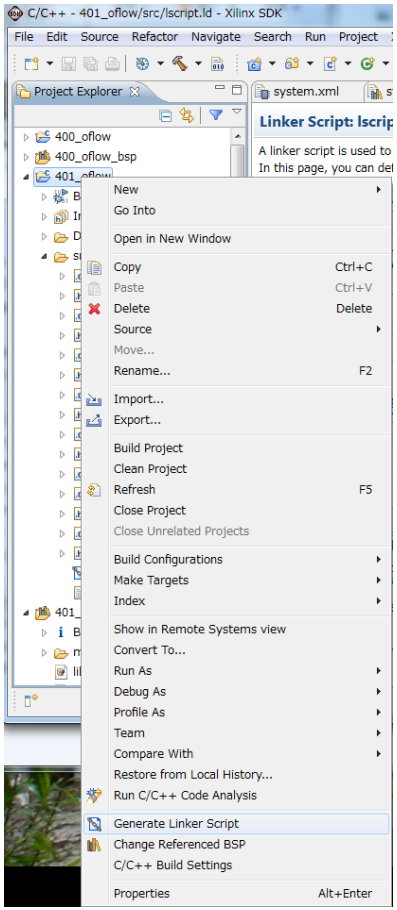
- ホストPCにて、画像データを用意し、UDPによりFPGAに転送する必要があります。
- リファレンスデザインのSDK(400_oflow_v02.tgz)を、ホストPCに展開し、clientディレクトリにて、all.shスクリプトを動かします。（IPアドレスを指定）

```
$ ./all.sh 192.168.10.64
targetHost:192.168.10.64 targetPort:8100
UDP Socket created. sending file '410_tux.64KB'
started!
64KB sent
```



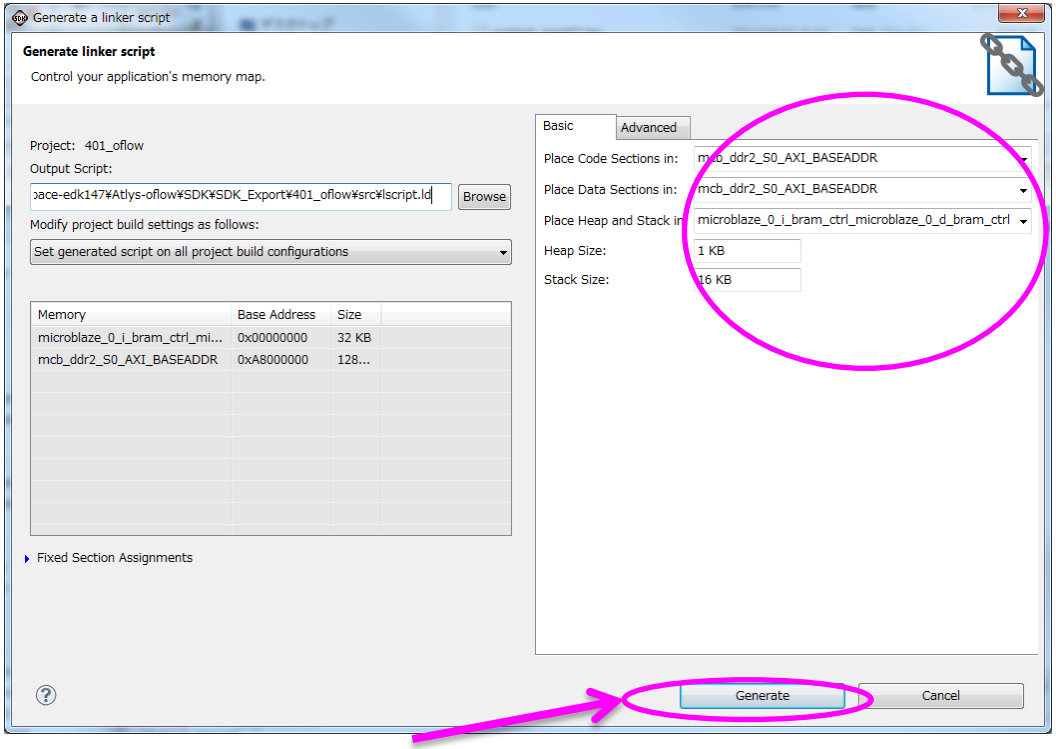
スタック領域の設定 (1/2)

- しかし、最初の画像を読み込んだ後、止まってしまうかもしれません。
  - XSDKのデフォルトでは、スタック領域が1KB(0x400)バイトしかありませんので、上手く動きません。
- スタック領域の大きさを設定するには、プロジェクトを右クリックして、Generate Linker Scriptを選択します



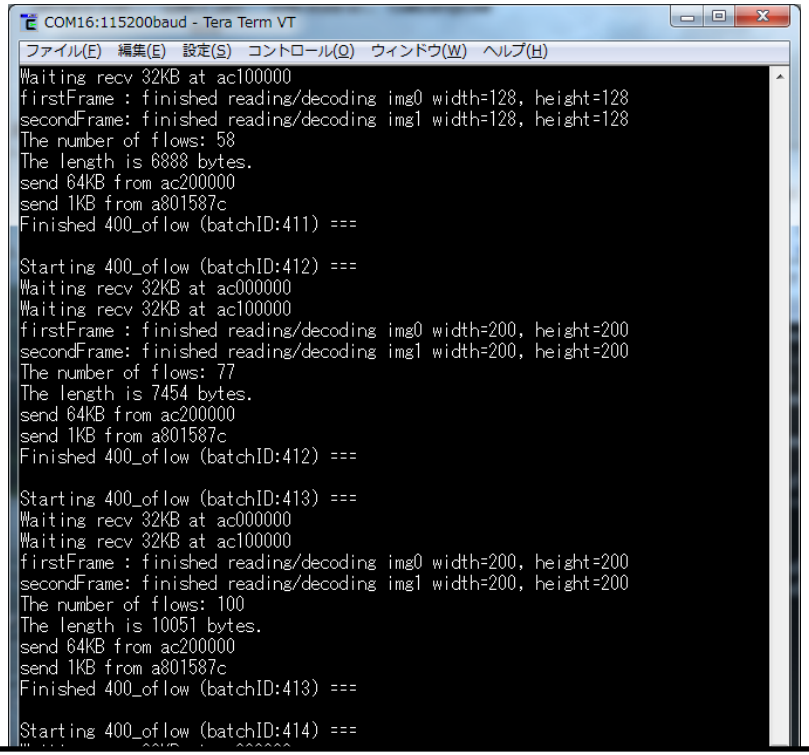
スタック領域の設定 (2/2)

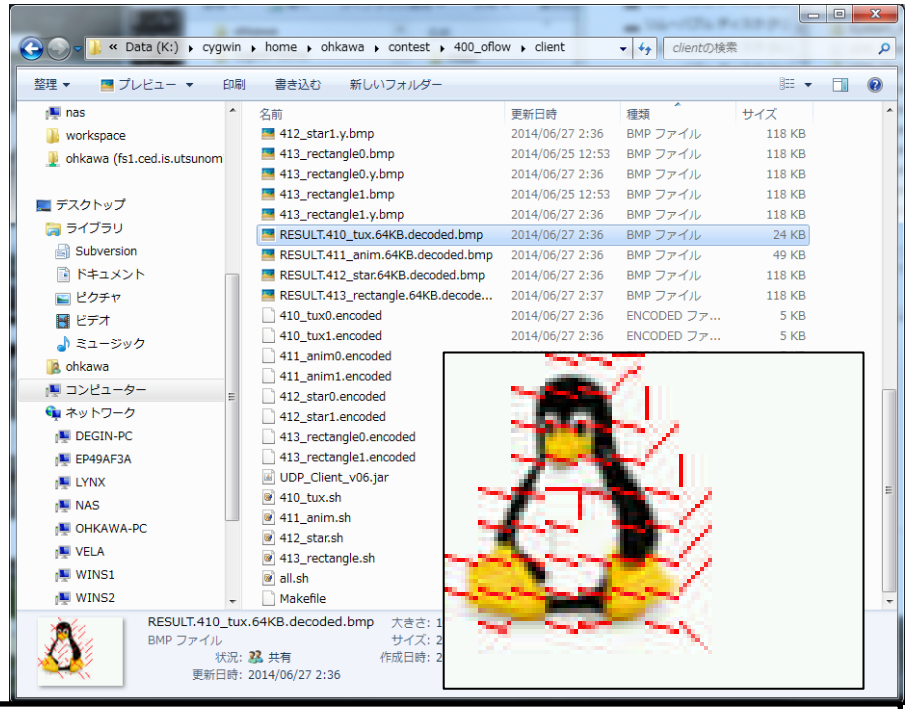
- Generate Linker Scirptダイアログで、例えば以下の設定をします。
  - CodeSectionの配置: DDR2
  - DataSectionの配置：DDR2
  - Heap&Stackの配置：bram
  - HeapSize: 1 KB
  - StackSize: 16 KB
- その後、GenerateするとOverwriteするか聞かれるのでlscript.ldファイルを更新します。



400_oflowの動作確認

- さて、動くでしょうか・・・？400_oflowプロジェクトを右クリックして、Runします
- ホストPC用のクライアントも再度動かしてください。($ ./all.sh 192.168.10.64)
- Linux上で動作させたときと同じ様にRESULTのBMPファイルが出来ればOK！





それではいよいよ高速化に挑戦する開発を始めましょう！！

Document P62

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

User Manual of Optical Flow System Reference Design (Altera DE2-115)

コンテスト実行委員会コアチーム

Version 2014-07-03

このドキュメント

- このドキュメントでは，Altera DE2-115ボード用のリファレンスデザインに含まれるシステム構成について説明します．
- また，Altera Quartusを用いて，リファレンスデザインの回路ファイル（sofファイル）を生成する方法を示します．

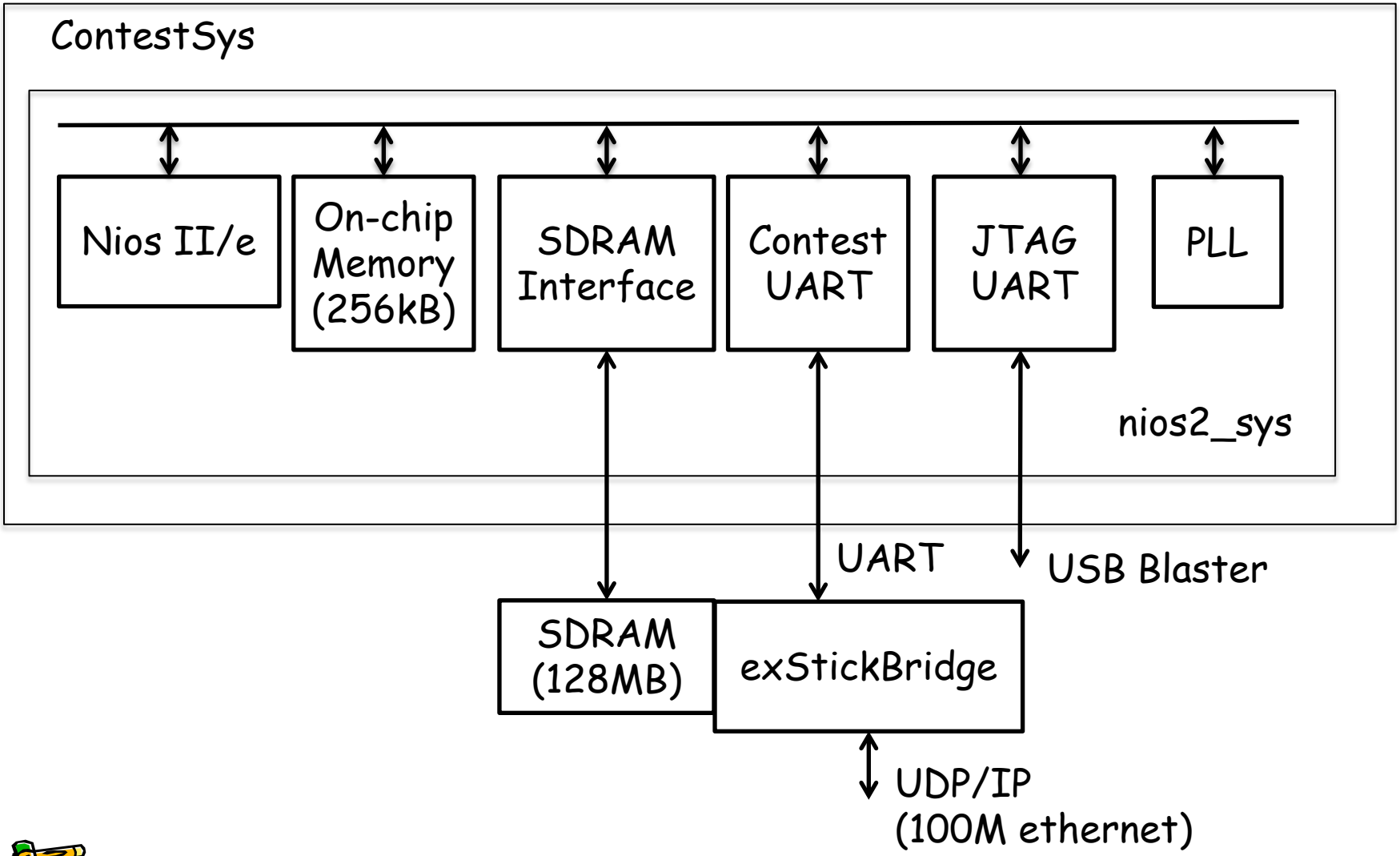
- 設計コンテストのWEBサイト
  - http://aquila.is.utsunomiya-u.ac.jp/contest/

- 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

最初に

- Altera版ContestSysはXilinx版と機能的には同じですが，異なる所が多々ありますので，注意して下さい．
- Altera版のリファレンスデザインは，Quartus Web Editionで開発できるように考慮しております．そのため，Nios2の実装にはNios2/eを使用し，キャッシュ等は搭載しておりません．
- 大学関係者の方はAcademic License（無料）がAlteraのホームページより申請可能ですので，利用される事をお勧め致します．以下，簡単に手順方法を記載しておきます．
  1. AlteraのUniversity Program（http://www.altera.com/education/univ/unv-index.html）のページの左にあるメニューの中の，Member->License Requestをクリックする
  2. myAlteraのアカウント名，パスワードを入力して，申請ページへ行く．

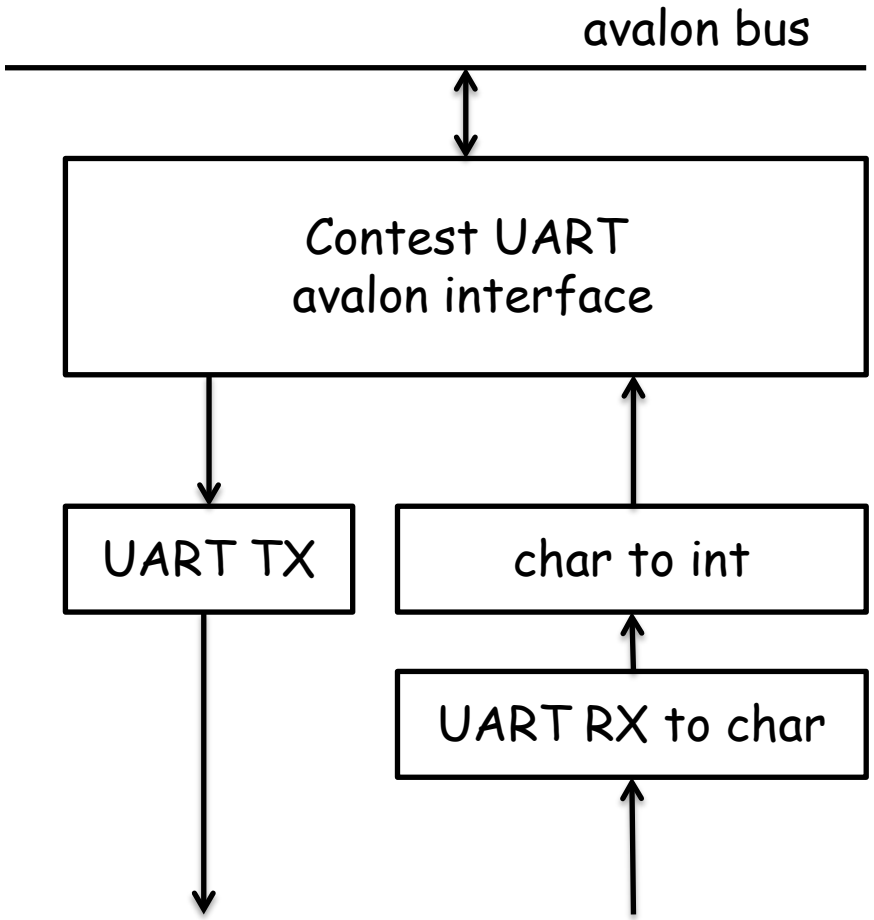
リファレンスデザインContestSysのブロック図



各機能の役割

- PLL
  - 50MHzのクロックを供給
- On-chip Memory（256kB）
  - 命令メモリとして使用
- SDRAM
  - NiosIIの命令メモリ，データメモリとして使用.
- Contest UART
  - データ転送用のインターフェース.
  - UARTから受信したデータを32ビットのデータとして格納する.
  - 8ビットのデータをUART経由で送信する
- JTAG UART
  - NiosIIでの実行経過などをprintfで出力するインターフェース
- NiosII/e
  - オプティカルフローを計算するCPU

Contest UARTの詳細



Contest UARTの詳細

- contest UART avalon interface
  - UART RXやUART TXに対してavalonバスとのinterfaceになる
- char to int
  - UART RXを8ビットのデータにしたものを32ビットのデータに変換する
- UART RX to char
  - シリアルで入力されるUART RXのデータを8ビットのデータに変換する
- UART TX
  - 8ビットのデータをシリアルで送信する

メモリマップ

メモリアドレス		
開始アドレス	終了アドレス	
0x0000_0000	0x0003_FFFF	on-chip memory
0x0004_1020	0x0004_102f	Contest UART
0x0005_0000	0x0005_07FF	Nios II/e
0x0800_0000	0x0FFF_FFFF	SDRAM

補足

- SDRAMの領域は0x0800_0000番地から使用出来るのですが，
  リファレンスデザインでは0x0c00_0000番地から使用しています．
  （SDRAM領域の0x0800_0000番地からの領域をデバッグに用いていたため）

リファレンスデザインの作成について

- 次ページ以降で作成するリファレンスデザインの作成方法では，Verilog-HDLの記述は既に完成しているものを使用するとします．
- 本ドキュメントで参照するファイルは以下のものです．
  - ContestSys05.tar.gz
    - DE2-115ボード用コンピュータシステム設計部門リファレンスデザインのプロジェクトファイル
  - DE2-115.qsf
    - DE2-115ボード用ピン設定ファイル．下記のURLから取得できます http://www.altera.com/education/univ/materials/boards/de2-115/unv-de2-115-board.html

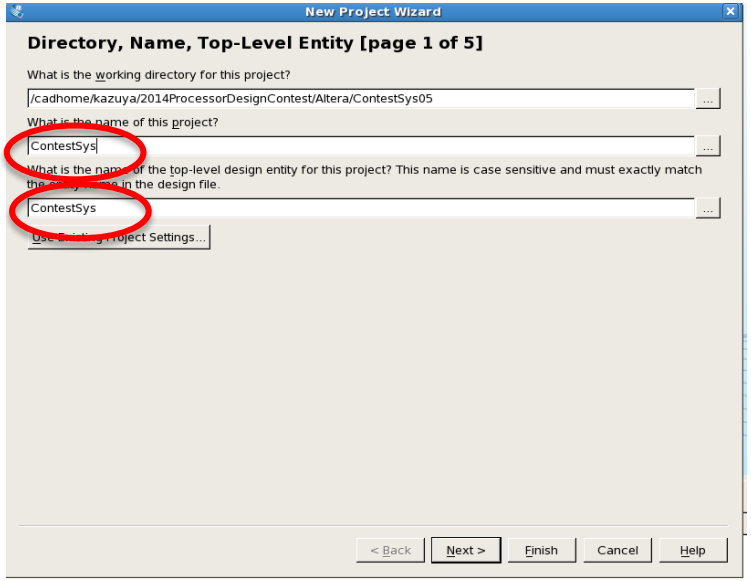
ハードウェアデザイン

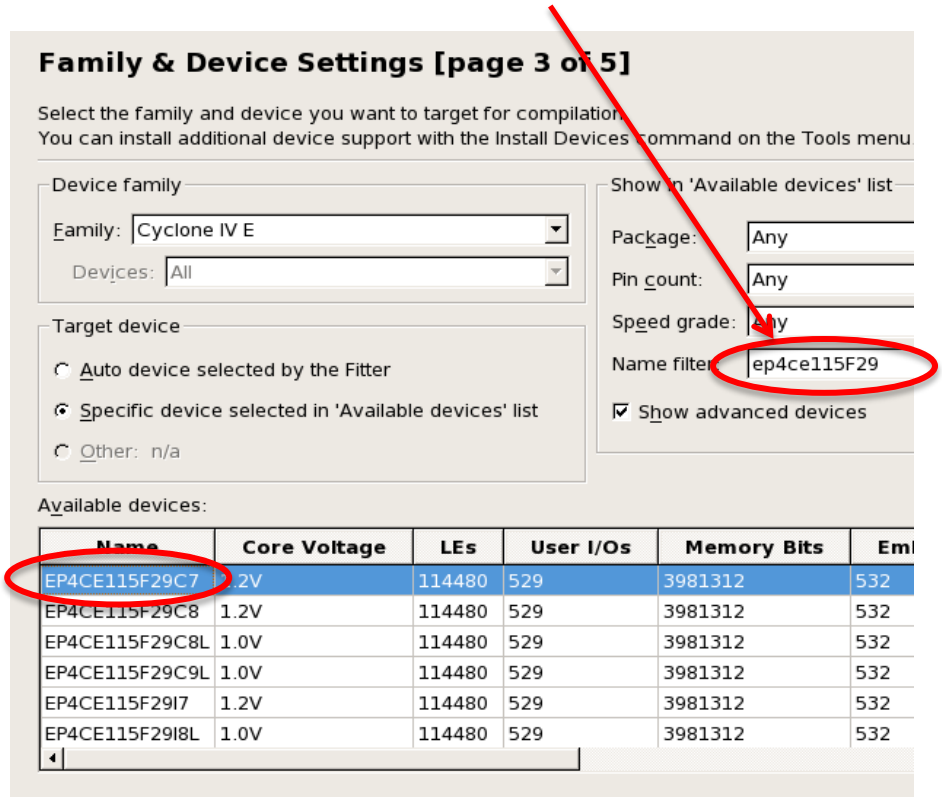
新規プロジェクトの作成

1. プロジェクトを置くディレクトリは新規に空のディレクトリを作成する．ここではContestSys06とします
2. 新規に作成したディレクトリでQuartus を起動する
3. File -> New Project Wizardを選択
   1. プロジェクト名をトップモジュール名（ContestSys）にする（次ページ左写真）．その後Next
   2. Add fileでは何も追加しない
   3. デバイス名はCyclone IV E EP4CE115F29C7を選ぶ（次ページ右写真）
   4. EDA toolの設定でSimulationツールとしてModelsimを選んでいる場合はFormatをVerilog HDLにする
   5. その他はデフォルトでNextを押し，finishまでいく

新規プロジェクトの作成

Name filterでデバイス名の候補をfilteringできる





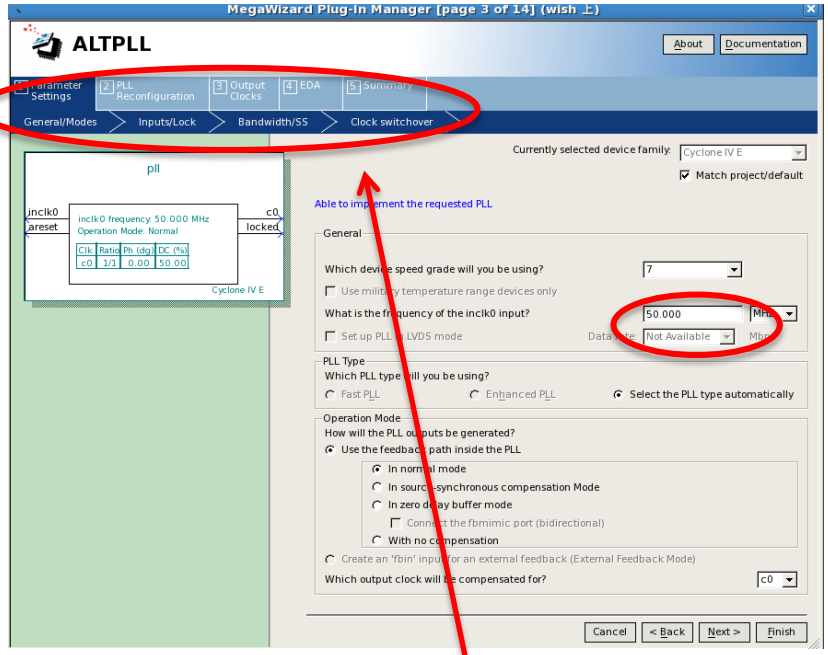
Qsysの起動と外部クロックの設定

- ここではQsysのシステムとしてnios2 sysを作成します.
- Tools -> QsysでQsysを起動する
  （以下Qsysでの操作です．）
- File -> Save でnios2_sysという名前をつけて保存

PLLの追加

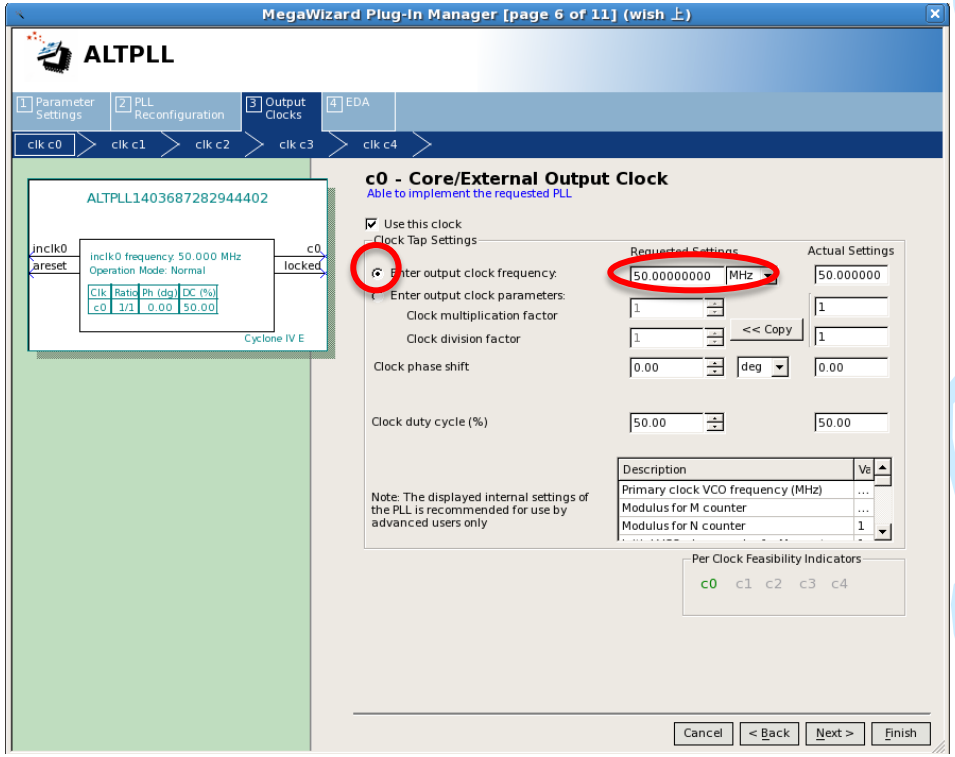
- Library から PLL -> Avalon PLL を選択し，Add
- 現れたWindowで以下を設定
  - Parameter Setting -> General/Mode内の入力クロックを50MHzに設定（次ページの画面1）
  - Output Clock -> clk c0のEnter output clock frequencyのラジオボタンを選択し，50MHzと入力（次ページの画面2）
  - Output Clock -> clk c1のEnter output clock frequencyのラジオボタンを選択し，50MHzと入力し，Clock phase shiftを-3nsとする．（2ページ先の画面3）
  - Finishボタンを押す
- 追加したPLLの名前をpllに変更
- PLLの配線を以下の用にする
  - pllのclk_in_primaryとclk_0のclkを接続
  - pllのclk_in_primary_resetとclk_0のclk_resetを接続
- PLLのc1をexport欄をダブルクリックしてexportし，export名をsdram_clkとする
- PLLのlocked_conduitのexport欄をダブルクリックしてexportし，export名をlockedとする（2ページ先の画面4）

PLLの追加



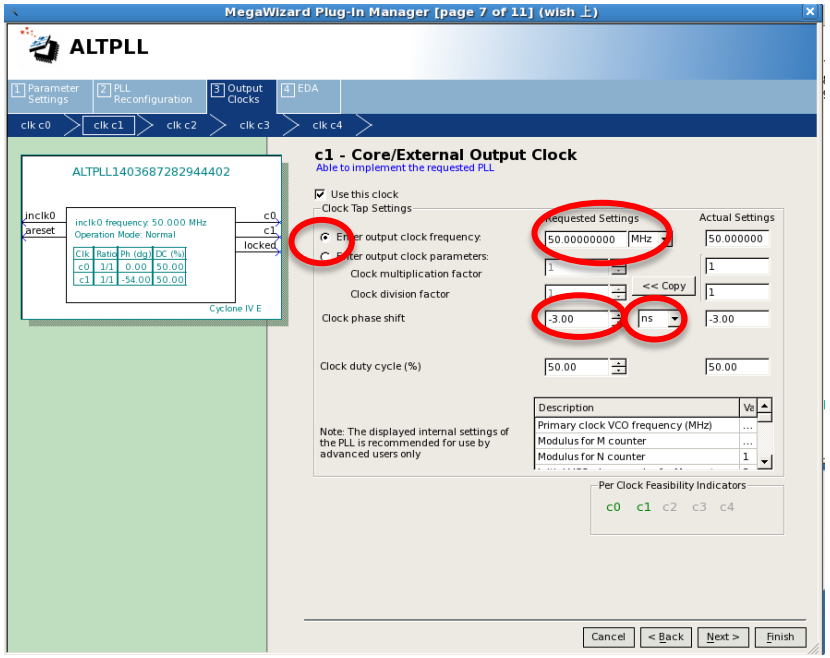
画面1

画面2



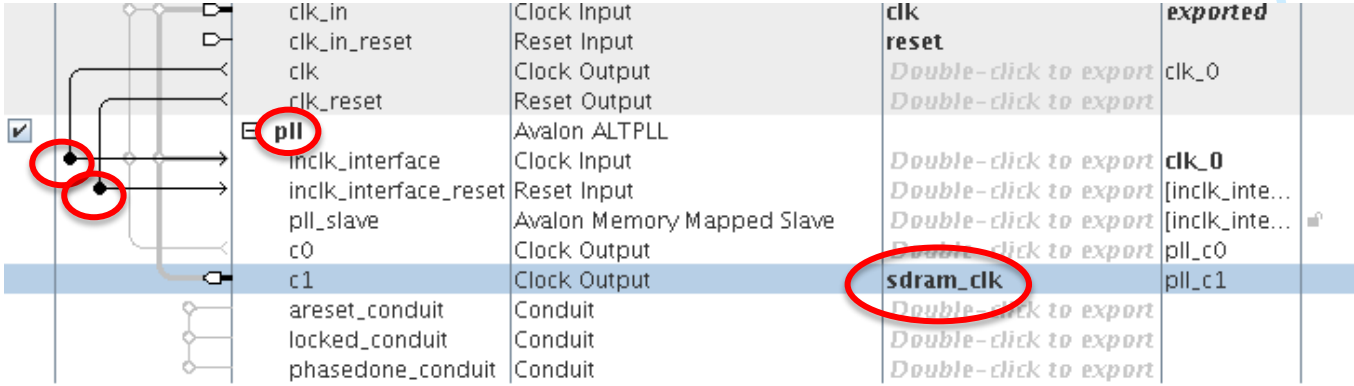
ここの欄が現在設定している
項目を示すタブになっている

PLLの追加



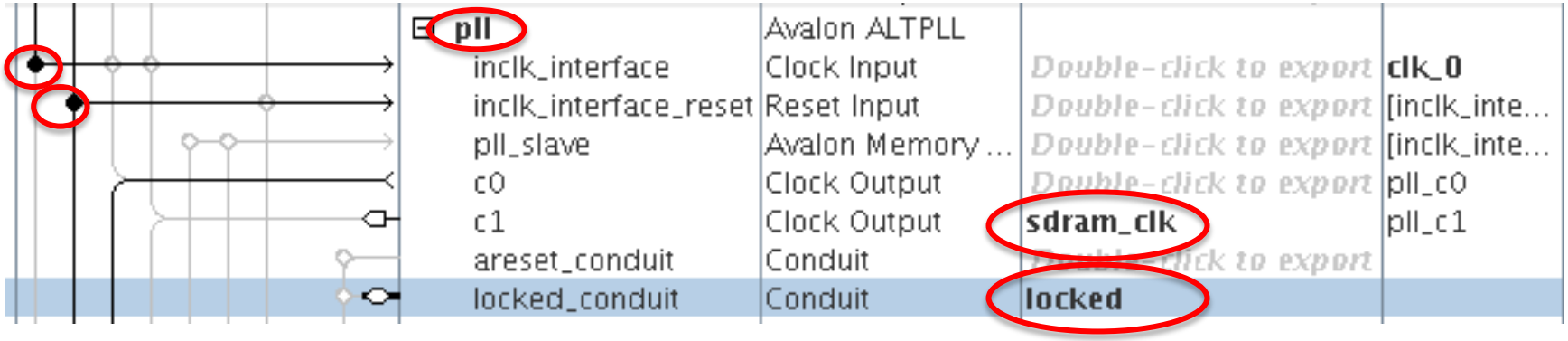
画面3

画面4



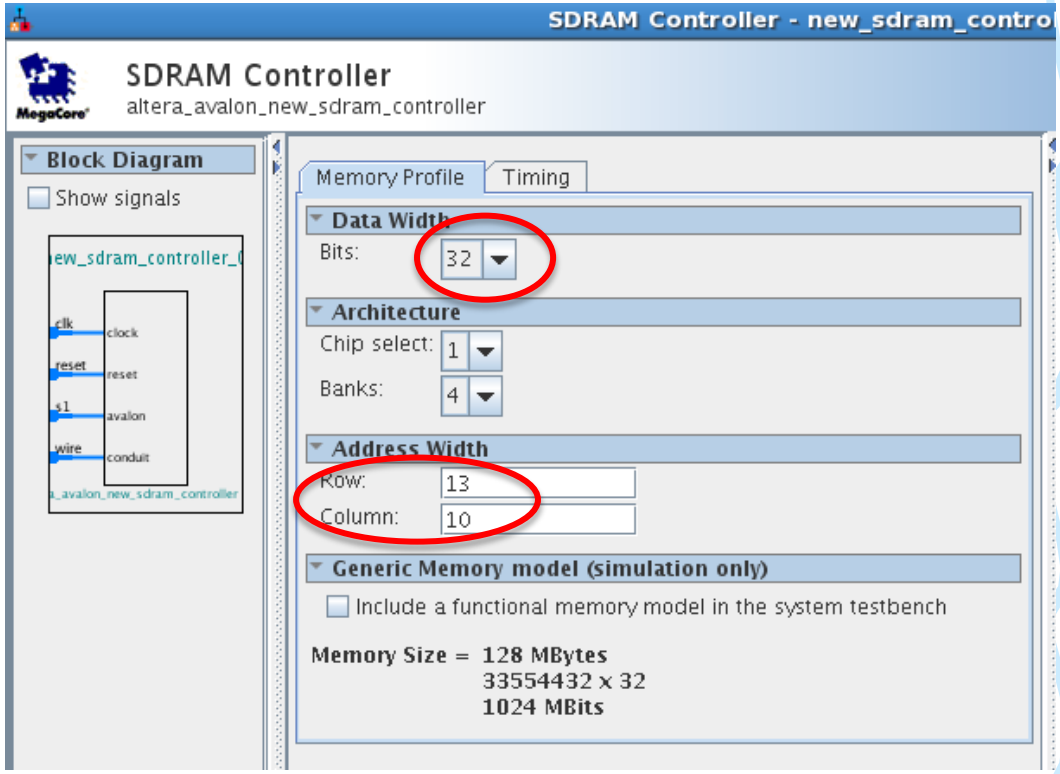
PLLの設定

- 追加したPLLの名前をpllに変更
- PLLの配線を以下の用にする
  - pllのclk_in_primaryとclk_0のclkを接続
  - pllのclk_in_primary_resetとclk_0のclk_resetを接続
- PLLのc1をexport欄をダブルクリックしてexportし，export名をsdram_clkとする
- PLLのlocked_conduitのexport欄をダブルクリックしてexportし，export名をlockedとする



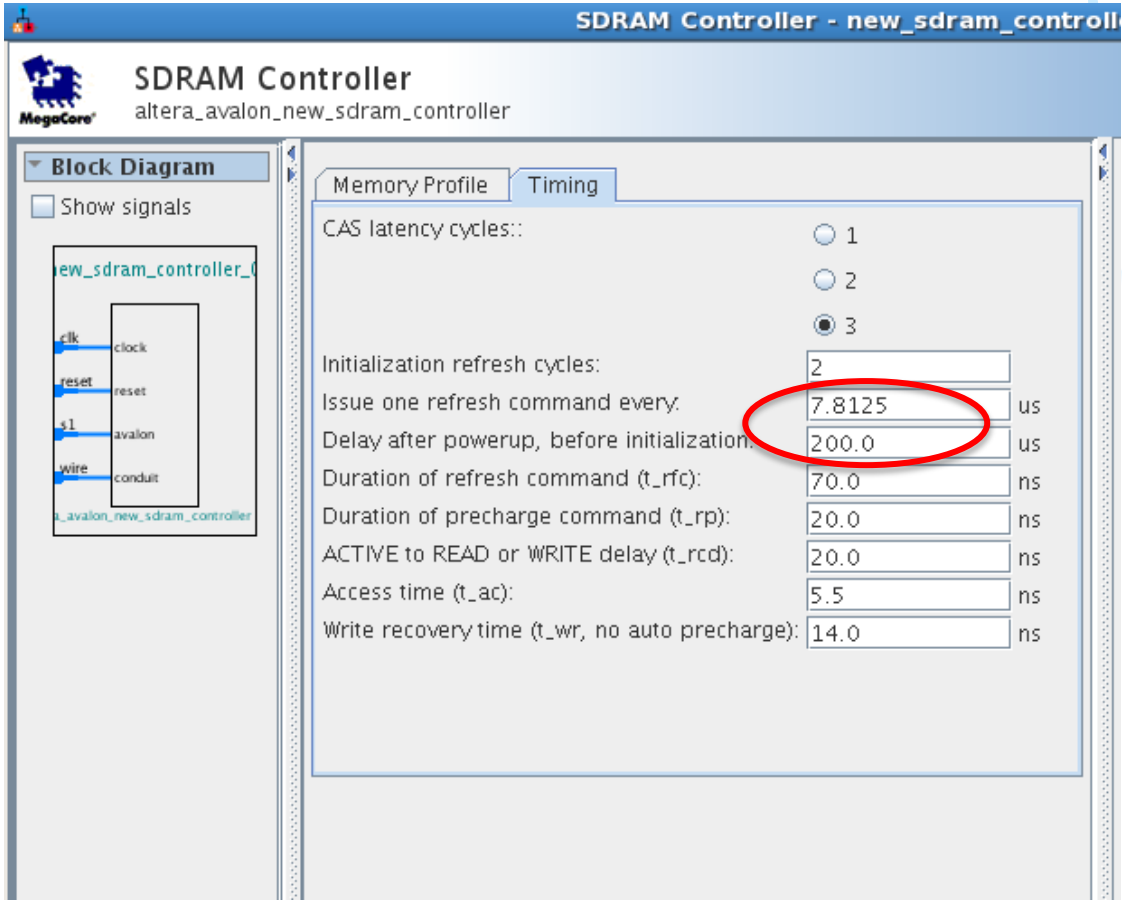
SDRAM Controllerの追加

- Library から Memories and Memory Controllers->External Memory Interfaces-> SDRAM Interfaces -> SDRAM Controllerを選択し，Add
- 現れたWindowで以下を設定する
  - Data Width: 32
  - Address Width
    - Row: 13，Colum: 10



SDRAM ControllerのTiming設定

- Timingタブをクリック
  - Issue one refresh command every を7.8125us
  - Delay after powerupを200us
- Finishボタンをクリック



SDRAM Controllerのclk配線など

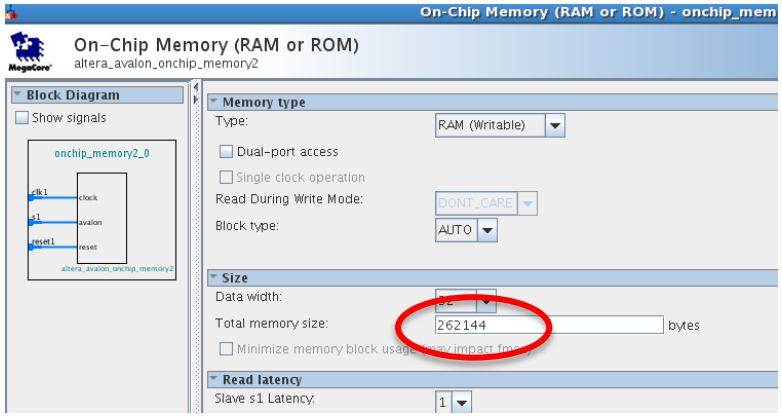
- 追加したsdram controllerの名前をsdramに変更（new_sdram_controller_0という名前の上で右クリックを押して現れるメニューからRenameを選択）
- pllのc0をsdram controllerのclkに配線
- clk_0のclk_resetをsdramのresetに配線
- sdramのwireをExportするようにExport欄をダブルクリック.（Export名をsdram_wireとする）

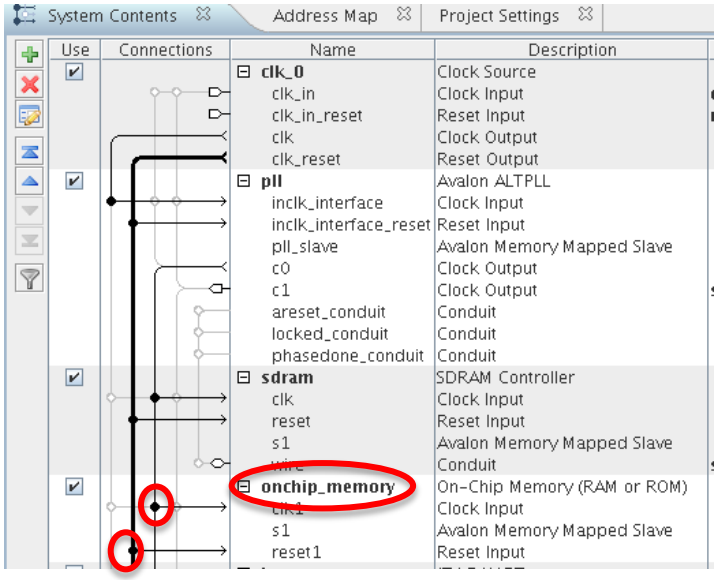
System Contents | Address Map | Project Settings

Use	Connections	Name	Description	Export	Clock
✔		clk_0	Clock Source		
		clk_in	Clock Input	clk	exported
		clk_in_reset	Reset Input	reset	
		clk	Clock Output	Double-click to export	clk_0
		clk_reset	Reset Output	Double-click to export	
✔		pll	Avalon ALTPLL		
		inclk_interface	Clock Input	Double-click to export	clk_0
		inclk_interface_reset	Reset Input	Double-click to export	[inclk_inte...
		pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_inte...
		c0	Clock Output	Double-click to export	pll_c0
		c1	Clock Output	sdram_clk	pll_c1
		areset_conduit	Conduit	Double-click to export	
		locked_conduit	Conduit	Double-click to export	
		phasedone_conduit	Conduit	Double-click to export	
✔		sdram	SDRAM Controller		
		clk	Clock Input	Double-click to export	pll_c0
		reset	Reset Input	Double-click to export	[clk]
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]
		wire	Conduit	sdram_wire	

On-Chip memoryの追加

- Library から Memories and Memory Controllers-> On-Chip -> On-Chip Memory(RAM or ROM)を選択し，Add
  - Total Memory Sizeを262144とする(256k)
  - Finishをクリックする
- 名前をonchip_memoryに変更
- onchip_memoryのclk1をpllのc0と接続する
- onchip_memoryのreset1をclk_0のclk_resetと接続する





JTAG UARTの追加

- Library から Interface Protocols-> Serial -> JTAG UARTを選択し，Add
  - Finishをクリックする
- 名前をjtag_uartに変更
- jtag_uartのclkをpllのc0と接続する
- jtag_uartのresetをclk_0のclk_resetと接続する

System Contents | Address Map | Project Settings

Use	Connections	Name	Description	Export	Clock	Base
✓		⊟ clk_0	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset		
		clk	Clock Output	Double-click to export	clk_0	
		clk_reset	Reset Output	Double-click to export		
✓		⊟ pll	Avalon ALTPLL			
		inclk_interface	Clock Input	Double-click to export	clk_0	
		inclk_interface_reset	Reset Input	Double-click to export	[inclk_inte...	
		pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_inte...	
		c0	Clock Output	Double-click to export	pll_c0	
		c1	Clock Output	sdram_clk	pll_c1	
		areset_conduit	Conduit	Double-click to export		
		locked_conduit	Conduit	Double-click to export		
		phasedone_conduit	Conduit	Double-click to export		
✓		⊟ sdram	SDRAM Controller			
		clk	Clock Input	Double-click to export	pll_c0	
		reset	Reset Input	Double-click to export	[clk]	
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	
		wire	Conduit	sdram_wire		
✓		⊟ onchip_memory	On-Chip Memory (RAM or ROM)			
		clk1	Clock Input	Double-click to export	pll_c0	
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	
		reset1	Reset Input	Double-click to export	[clk1]	
✓		⊟ jtag_uart	JTAG UART			
		clk	Clock Input	Double-click to export	pll_c0	
		reset	Reset Input	Double-click to export	[clk]	
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	

contest uart avalon interfaceをComponentとして登録

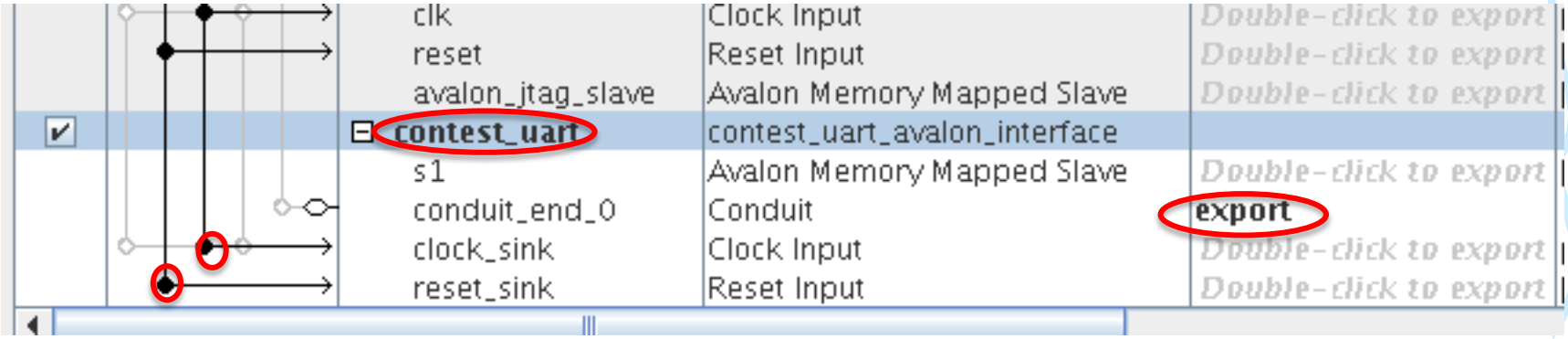
- ProjectのNew Componentを選択して，Add
- Component Typeタブ内
  - Name, Display Nameを“contest_uart_avalon_interface”とする
  - Groupを“My Own IP Core”とする
- Filesタブ内
  - （プロジェクトディレクトリ内にリファレンスデザインのプロジェクトディレクトリにある以下のファイルをコピーしておく
    - contest_uart_avalon_interface.v
    - system.v
    - define.v
  - Synthesis Filesとして，define.v以外の上記2つのVerilog HDLファイルを追加する（contest_uart_avalon_interface.vがTop-level Fileとなっているのを確認）
  - Analyze Synthesis Filesボタンをクリック

contest uart avalon interfaceをComponentとして登録

- Signalsタブ内
  - clockのinterfaceの所を選択して，new Clock Inputを選択し，interface欄がclock_sinkとなるようにし，Signal Typeをclkとする
  - resetのinterfaceの所を選択して，new Reset Inputを選択し，interface欄がreset_sinkとなるようにする.
- Interfaceタブ内
  - Remove Interfaces With No Signalsボタンを押す
  - s1のAssociated Clockをclock_sinkに，Associated Resetをreset_sinkにする
  - conduit_end_0のAssociated Clock, Associated Resetも同様にする
  - reset_sinkのAssociated Clockをclock_sinkにする
- Finishボタンを押し，保存するかどうかを訪ねるWindowが出たら，Saveを選ぶ.

contest uart avalon interfaceを追加

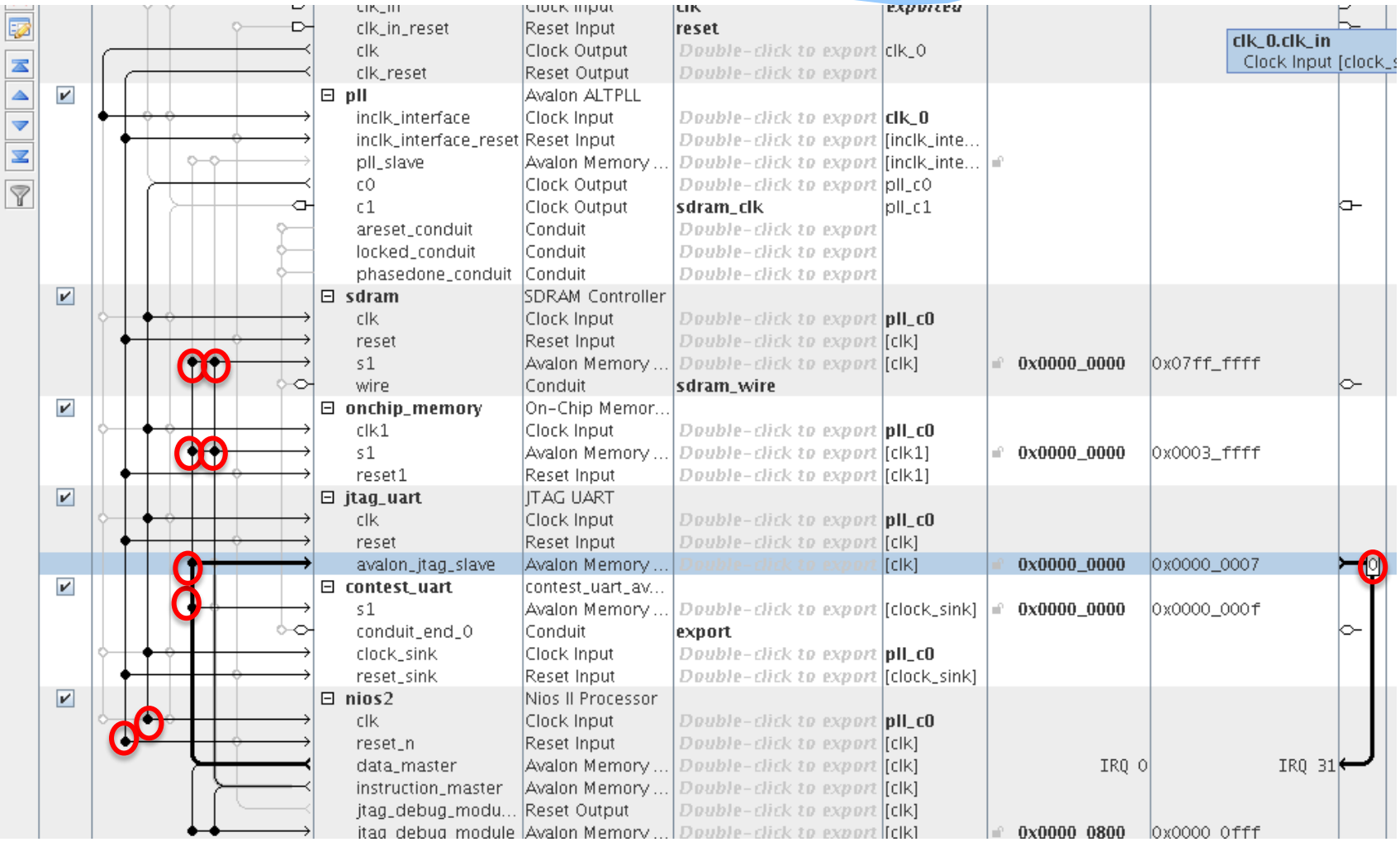
- ProjectのMy Own IP Core内のcontest_uart_avalon_interfaceを選択してAdd
  - Finishボタンをクリック
- 名前をcontest_uartに変更
- contest_uart_avalon_interfaceのclock_sinkをpllのc0と接続
- contest_uart_avalon_interfaceのreset_sinkをclk_0のclk_resetと接続
- contest_uart_avalon_interfaceのconduit_end_0のexport欄をダブルクリックして，export名をexportとする



Nios II/eの追加

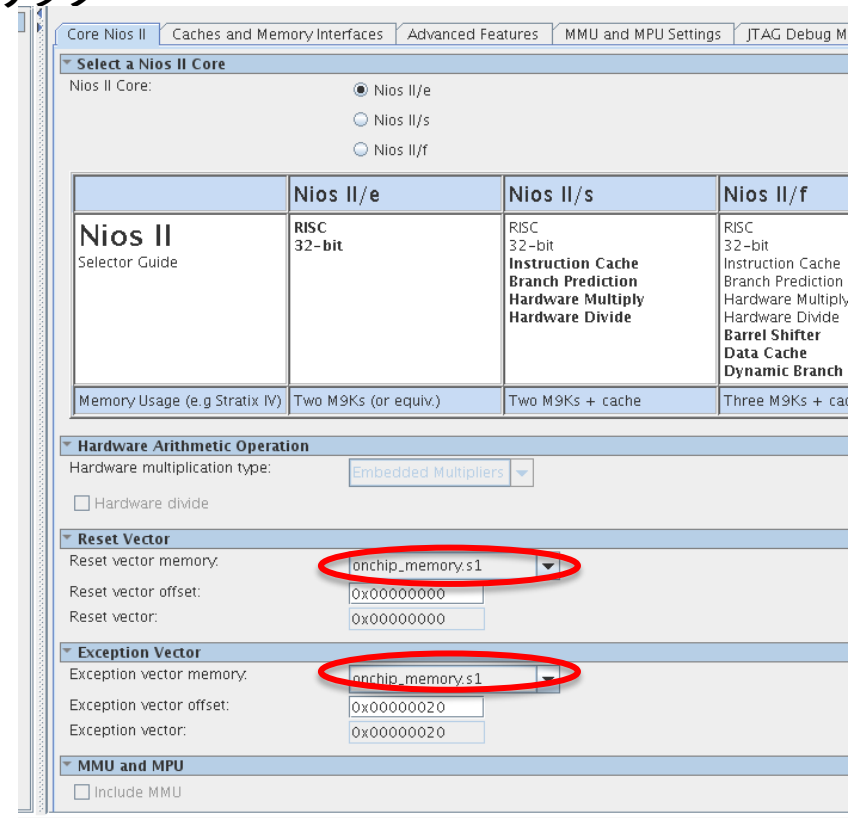
- Library から Embedded Processors-> Nios II Processorを選択し，Add
  - Nios II CoreとしてNios II/eを選択する
  - Finishをクリックする
- 名前をnios2に変更
- nios2のclkをpllのc0と接続する
- nios2のreset_nをclk_0のclk_resetと接続する
- nios2のdata_masterをcontest_uartのs1，jtag_uartのavalon_jtag_slave，onchip_memoryのs1，sdramのs1と接続する
- nios2のinstruction_masterをonchip_memoryのs1，sdramのs1と接続する
- nios2のIRQとjtag_uartのIRQを接続する

NiosII/eの追加


clk_in_reset Reset Input reset
clk Clock Output Double-click to export clk_0
clk_reset Reset Output Double-click to export
clk_0.clk_in
Clock Input [clock_s
pll Avalon ALTPLL
inclk_interface Clock Input Double-click to export clk_0
inclk_interface_reset Reset Input Double-click to export [inclk_inte...
pll_slave Avalon Memory ... Double-click to export [inclk_inte...
c0 Clock Output Double-click to export pll_c0
c1 Clock Output sdram_clk pll_c1
areset_conduit Conduit Double-click to export
locked_conduit Conduit Double-click to export
phasedone_conduit Conduit Double-click to export
sdram SDRAM Controller
clk Clock Input Double-click to export pll_c0
reset Reset Input Double-click to export [clk]
s1 Avalon Memory ... Double-click to export [clk] 0x0000_0000 0x07ff_ffff
wire Conduit sdram_wire
onchip_memory On-Chip Memor...
clk1 Clock Input Double-click to export pll_c0
s1 Avalon Memory ... Double-click to export [clk1] 0x0000_0000 0x0003_ffff
reset1 Reset Input Double-click to export [clk1]
jtag_uart JTAG UART
clk Clock Input Double-click to export pll_c0
reset Reset Input Double-click to export [clk]
avalon_jtag_slave Avalon Memory ... Double-click to export [clk] 0x0000_0000 0x0000_0007
contest_uart contest_uart_av...
s1 Avalon Memory ... Double-click to export [clock_sink] 0x0000_0000 0x0000_000f
conduit_end_0 Conduit export
clock_sink Clock Input Double-click to export pll_c0
reset_sink Reset Input Double-click to export [clock_sink]
nios2 Nios II Processor
clk Clock Input Double-click to export pll_c0
reset_n Reset Input Double-click to export [clk]
data_master Avalon Memory ... Double-click to export [clk] IRQ 0 IRQ 31
instruction_master Avalon Memory ... Double-click to export [clk]
jtag_debug_modu... Reset Output Double-click to export [clk]
jtag_debug_module Avalon Memory ... Double-click to export [clk] 0x0000_0800 0x0000_0fff

nios2の設定

- nios2を選択して，右クリックし，Editを選択する
- Reset vector memoryとException vector memoryとして，onchip_memory.s1を選択する
- Finishボタンをクリック



メモリマップの作成

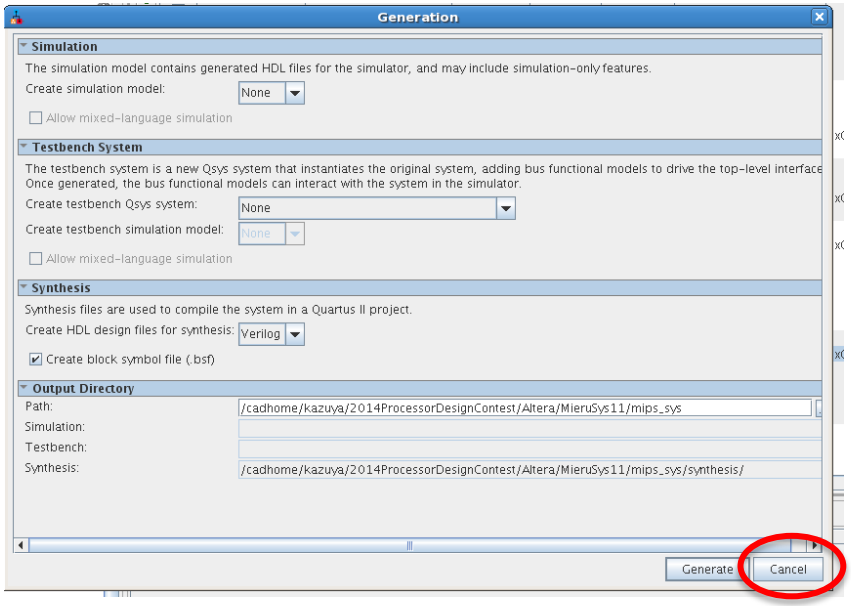
- メモリマップとしてbase addressを以下のように指定して，ロック
  - sdramのmemory: 0x0800_0000
  - onchip_memory:0x0000_0000
  - contest_uart: 0x0004_1020
  - nios2: 0x0005_0000
- jtag_uartはどこに割り当ててもよいので，System->Assign Base Addressで割り当てる

c1	Clock Output	sdram_clk	pll_c1		
areset_conduit	Conduit	Double-click to export			
locked_conduit	Conduit	Double-click to export			
phasedone_conduit	Conduit	Double-click to export			
⊟ sdram	SDRAM Controller				
clk	Clock Input	Double-click to export	pll_c0		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory ...	Double-click to export	[clk]	0x0800_0000	0x0fff_ffff
wire	Conduit	sdram_wire			
⊟ onchip_memory	On-Chip Memor...				
clk1	Clock Input	Double-click to export	pll_c0		
s1	Avalon Memory ...	Double-click to export	[clk1]	0x0000_0000	0x0003_ffff
reset1	Reset Input	Double-click to export	[clk1]		
⊟ jtag_uart	JTAG UART				
clk	Clock Input	Double-click to export	pll_c0		
reset	Reset Input	Double-click to export	[clk]		
avalon_jtag_slave	Avalon Memory ...	Double-click to export	[clk]	0x0004_0000	0x0004_0007
⊟ contest_uart	contest_uart_av...				
s1	Avalon Memory ...	Double-click to export	[clock_sink]	0x0004_1020	0x0004_102f
conduit_end_0	Conduit	export			
clock_sink	Clock Input	Double-click to export	pll_c0		
reset_sink	Reset Input	Double-click to export	[clock_sink]		
⊟ nios2	Nios II Processor				
clk	Clock Input	Double-click to export	pll_c0		
reset_n	Reset Input	Double-click to export	[clk]		
data_master	Avalon Memory ...	Double-click to export	[clk]	IRQ 0	IRQ 31
instruction_master	Avalon Memory ...	Double-click to export	[clk]		
jtag_debug_modu...	Reset Output	Double-click to export	[clk]		
jtag_debug_module	Avalon Memory ...	Double-click to export	[clk]	0x0005_0000	0x0005_07ff
custom_instruction...	Custom Instructi...	Double-click to export			

ロックを外した状態で
アドレスを入力し，
その後ロックをかける

nios2_sysの生成

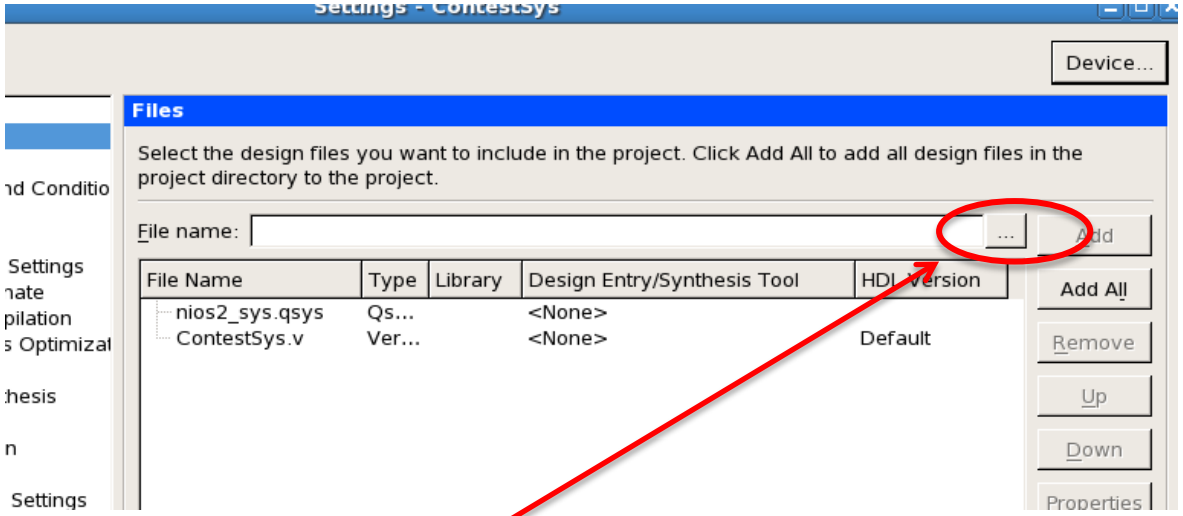
- メニューSystem -> Create Global Reset Networkをクリック
- この時点で，Qsys下部のMessageからエラー表示がなくなっているはず
- mips_sysを保存
- メニューGenerate -> Generateをクリック
  - Generateボタンをクリック
- ちなみに，mips_avalon_interface.vなどを変更する度に，QsysでGenerateする必要があります
- 以上で，Qsysは終了してOK



Generateでは
右下のGenerateボタンを
押すだけでよい

プロジェクトへのファイルの追加

- あらかじめ，ContestSys.vをプロジェクトディレクトリにコピーしておく
- Porject -> Add/Remove files in Projectで以下のファイルを追加する
  - ContestSys.v
  - nios2_sys.qsys
- 追加したらOKボタンを押してウィンドウを閉じる



ここをクリックして追加するファイルを選択し，その後addボタンを押す

ピン配置情報のimportと割当

- Assignments -> Import Assignmentsをクリック
  - DE2_115.qsfを選択し，importする
- Assignments -> Pin Plannerをクリック
  - 現れるウィンドウの下の方にある各ピンの設定で，GPIO[3]のI/O Standardを3.3V LVCMOSにする
  - 同様に GPIO[5], GPIO[7], GPIO[9]も同様に変更する
  - メニューFile -> closeをクリックして，ウィンドウを閉じる

Node Name	Location	I/O Standard
GPIO[3]	PIN_Y17	3.3V LVCMOS
GPIO[5]	PIN_Y16	3.3V LVCMOS
GPIO[7]	PIN_AE16	3.3V LVCMOS
GPIO[9]	PIN_AE15	3.3V LVCMOS

ピン配置情報のimportと割当

Edges

Named: *GPIO[?] Edit: Filter: Pin

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair
GPIO[0]	Unknown	PIN_AB22	4	B4_N0	3.3-V LVTTL		8mA (default)		
GPIO[1]	Unknown	PIN_AC15	4	B4_N2	3.3-V LVTTL		8mA (default)		
GPIO[2]	Unknown	PIN_AB21	4	B4_N0	3.3-V LVTTL		8mA (default)		
GPIO[3]	Unknown	PIN_Y17	4	B4_N0	3.3-V LVCMOS		2mA (default)		
GPIO[4]	Unknown	PIN_AC21	4	B4_N0	3.3-V LVTTL		8mA (default)		
GPIO[5]	Unknown	PIN_Y16	4	B4_N0	3.3-V LVCMOS		2mA (default)		
GPIO[6]	Unknown	PIN_AD21	4	B4_N0	3.3-V LVTTL		8mA (default)		
GPIO[7]	Unknown	PIN_AE16	4	B4_N2	3.3-V LVCMOS		2mA (default)		
GPIO[8]	Unknown	PIN_AD15	4	B4_N2	3.3-V LVTTL		8mA (default)		
GPIO[9]	Unknown	PIN_AE15	4	B4_N2	3.3-V LVCMOS		2mA (default)		
<<new node>>									

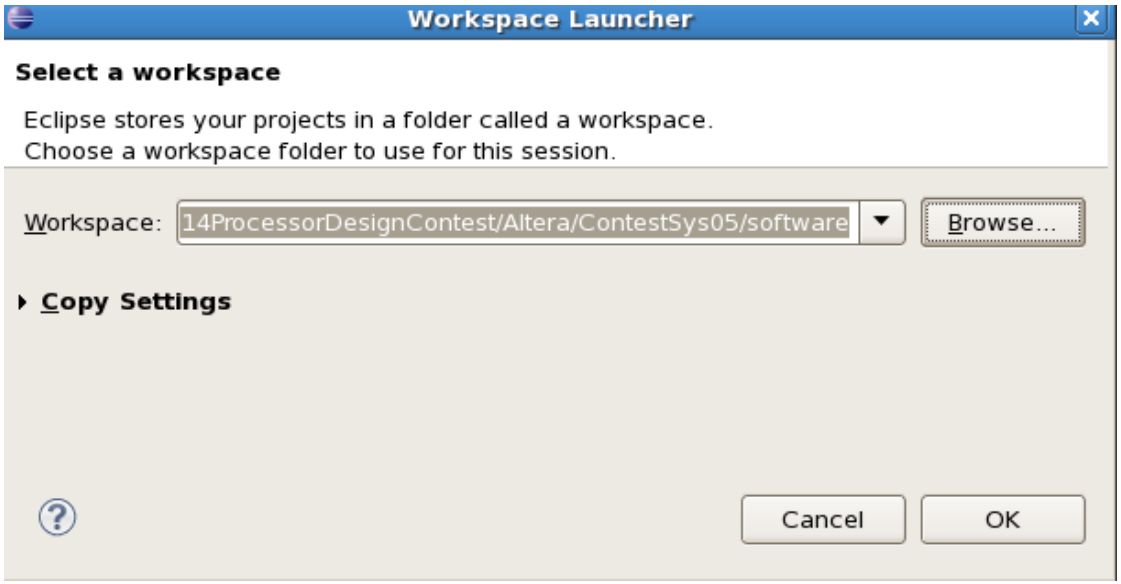
構成情報の生成

- Processing -> Start Compilationをクリックして，論理合成＆配置配線
- DE2-115ボードの電源をいれる.
- Taskウィンドウ内のProgram Device(Open Programmer)をダブルクリック
- Programmer内にて，Hardware Setupボタンをクリック.
  - No Hardwareとなっている所をUSB-Blaster USBを選択して，closeボタンをクリック
- Start ボタンを押して，Progress が100%(Successful)になればOK
- Programmerを閉じる

ソフトウェアデザイン

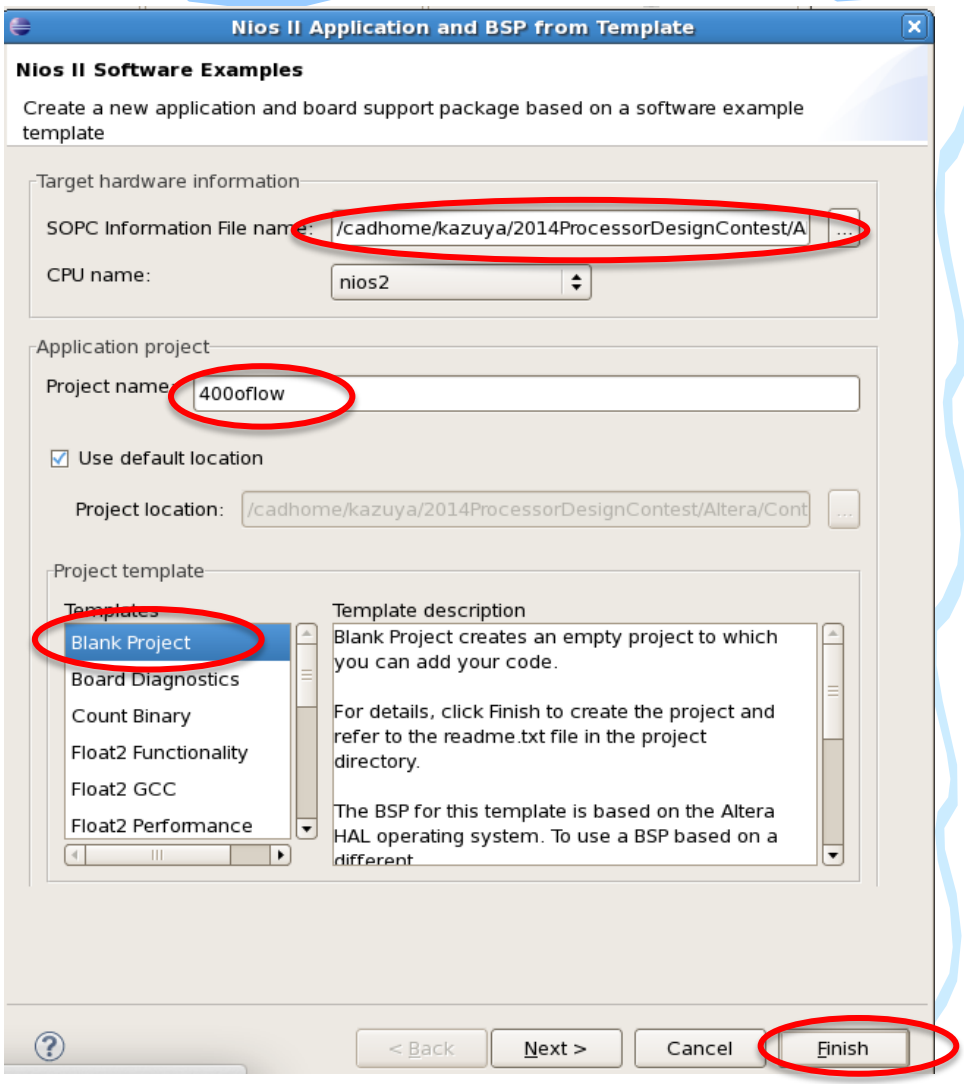
Workspaceの設定

- ハードウェアのプロジェクトディレクトリの下にsoftwareというディレクトリを作る.（例：ContestSys05/software）
- QuartusのメニューTools->Nios II Software Build Tools for Eclipseを選択する
- EclipseのメニューFile -> Switch Workspace -> Otherを選択し，上記で作成したContestSys05/softwareを選択する.（初めてEclipseを立ち上げた場合は，いきなりWorkspaceの選択から入るかもしれない）



新規プロジェクトの作成

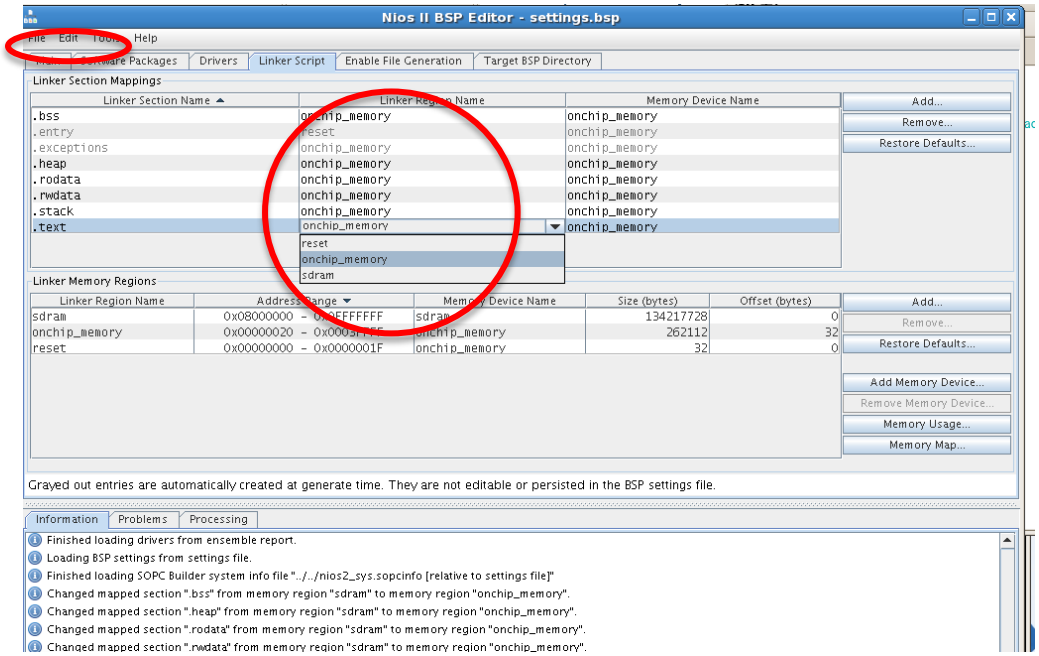
- File -> New -> Nios II Application and BSP from Templateを選択
- Target hardware informationのSOPC Information File nameには，Quartusのプロジェクトディレクトリにあるnios2_sys.sopcinfoを選択する
- Project nameを「400oflow」とする
- Project templateにはBlank Projectを選ぶ
- 以上で，Finishボタンを押す



BSPの設定

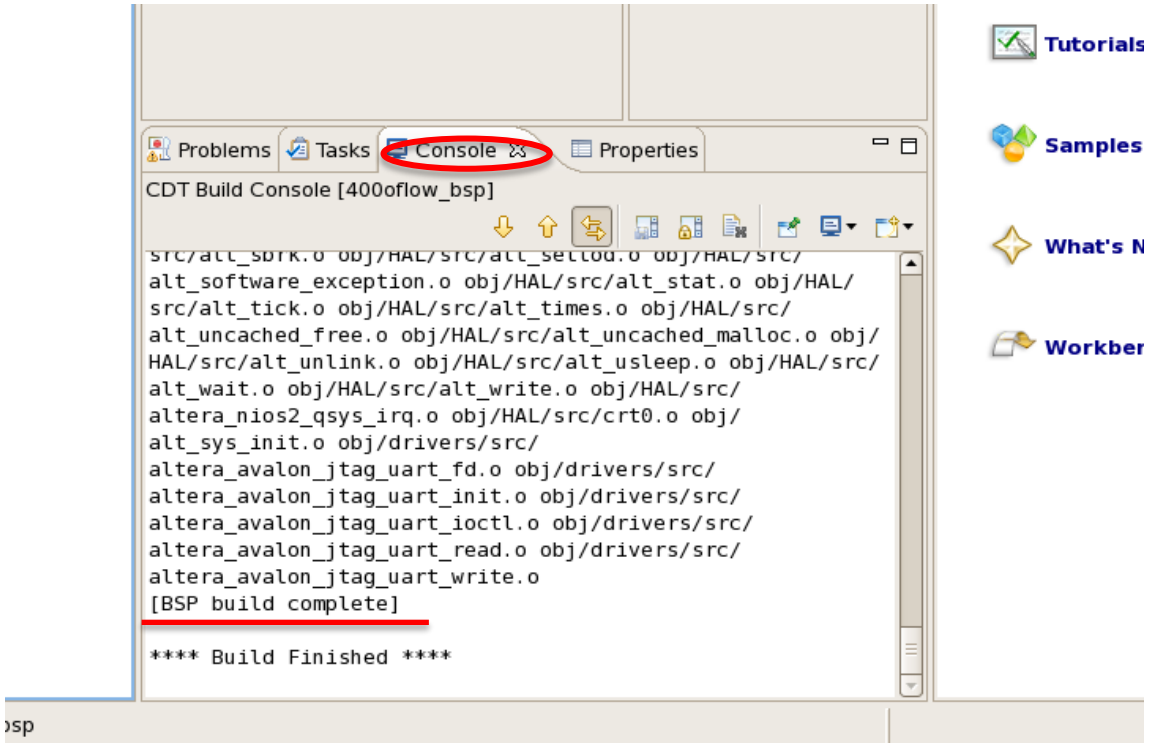
- 400oflow_bspを選択した状態で，右クリックし現れるメニューで，Nios II -> BSP Editorを選択
- 現れるWindowの中のLinker Scriptを選択し，".bss", ". heap", などを全てonchip_memoryに置くように設定する
- Nios II BSP EditorのFile メニューの中からSaveを選択する
- Generateボタンを押し，Exitボタンを押す

BSPを保存するときのFileメニュー



BSPのBuild

- 400oflow_bspを選択した状態で，右クリックし現れるメニューで，Build Projectをクリック
- Consoleにて，“[BSP build complete]”が出力されている事を確認



400oflowのBuild

- ファイルの準備
  - コンテストのサイトから配布されている400oflow_v10.tgzを展開し，400_oflowディレクトリに移動し，そこでmake exportをする．
  - 上記により，export/altera/src/の下にnios用のソースファイルが生成される
  - software/400oflowディレクトリに移動し，export/altera/srcディレクトリにあるファイルを全てコピーする．
- Eclipse上で，400oflowプロジエクトを選択した状態で，右クリックし現れるメニューで，
  - Refreshをクリックする
  - Build projectをクリックする
  - Run As -> Nios II Hardwareをクリックする
- 後はホストPCから画像の転送プログラムなどを動かせばOK

Alteraのリファレンスデザインの変更について

- PLLにより生成されるクロックを修正した時は，define.vにあるSYS_CLOCKの値も変更してください．
  - 例：50MHzを100MHzにしたら，SYS_CLOCKも100にする
  - 理由：この値をベースに，contest_uart内のuartの受信モジュールが1つのシリアルデータの長さを決めています．そのため，この値が実際のクロック周波数と異なっていると，UARTの受信データが期待したものになりません．
- Nios II/fを使用し，データキャッシュを有効にした場合は，communication.cでのUARTへのアクセスの記述を変更する必要があります．変更内容は次ページに示します．
  - 理由：volatile宣言された変数であってもコンパイラはキャッシュをバイパスするldio/stio命令を使用しません．そのため，キャッシュをバイパスさせたい時にはHALにより提供されるIORD，IOWRなどのマクロを使います．（io.hのincludeが必要）

データキャッシュを使用する場合のcommunication.cの変更点

変更前

- 40行目：
  while((*status & 0x01) != 0x01) {
- 51行目：
  data[i]=*val;
- 69行目：
  while((*status & 0x4) != 0x04) {
- 72行目：
  *tx = c;

変更後

- 追加：
  #include<io.h>
- 40行目：
  while((IORD(CONTEST_UART_STATUS,0) & 0x01) != 0x01) {
- 51行目：
  data[i] = IORD(CONTEST_UART_RX,0);
- 69行目：
  while((IORD(CONTEST_UART_STATUS,0) & 0x4) != 0x04) {
- 72行目：
  IOWR(CONTEST_UART_TX, 0, c);

改訂履歴

- Ver.2014-07-03
  - 参加可能ボードに関する情報を追記
  - コンテストルール9、ホストとの通信方法を追記
  - コンピュータシステム設計部門リファレンスデザインに関するドキュメントP51, P61, P62をマージ
  - その他微修正
- Ver.2014-06-08
  - DE2ボード版リファレンスデザインの説明追加
  - 各種配布物のバージョン情報の更新・その他微修正
- 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）