

The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest General Guide and Contest Rule

コンテスト実行委員会コアチーム Version 2014-06-08

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



- 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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- ・ 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)
 - 技術情報揭示板
 - Google Group: HpCpsyDC2014
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>



ドキュメントガイド

- 本ファイルには以下のドキュメントが含まれます
 - P30 General Guide and Contest Rule
 - P32 Reference Design Test Manual (ATLYS board)
 - P40 Design Guide for Altera DE-2 board
 - P42 Reference Design Test Manual (DE2 board)
 - P33 Architecture of Reference Design Processor
 - P34 DRAM Memory Interface
 - P35 MIPS Cross Compiler Setup Manual
 - P36 SDK Setup Manual
 - P37 Application Specification of Processor Design Category
 - P51 Application Specification of Computer System Design Category
 - 【後ほど作成予定】 Design Data Submission
- これらのドキュメントは、第1回コンテストから大部分を流用しています
 - 参考)第1回IPSJ-ARC高性能プロセッサ設計コンテストのWEBサイト
 - <u>http://www.arch.cs.titech.ac.jp/contest/</u>

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.





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



コンテストルール 1/5

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



コンテストルール 2/5

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



コンテストルール 3/5

4. コンピュータシステム設計部門の競技内容

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



コンテストルール 4/5

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

【後程公開】



コンテストルール 5/5

7. 実行時間の計測方法

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





- 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

ネットワーク 192.168.10.0/255.255.255.0



192.168.10.X



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





- ・ 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





exStickBridgeでの動作確認済みスイッチー覧

- BUFFALO LSW3-TX-5EPL
- BUFFALO BSL-WS-G2116M
- (今後追加します)



設計を始めましょう

- Atlysボード、DE2-115ボード、USBケーブルなどの必要なハードウェアが揃っていない場合
 - コンテストホームページを参照の上,必要なハードウェアを揃えてください.
 - Atlysボードは、貸し出し出来る可能性があります.希望者はお問い合わ せください. contest_support@virgo.is.utsunomiya-u.ac.jp
- Atlysボード, DE2-115ボード, USBケーブルなどの必要なハードウェアが揃っている場合
 - ドキュメント "P32 Reference Design Test Manual" を参考に, リファレンスデザインの動作テストをおこなってください.



配付ライセンスについて

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





The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest Reference Design Test Manual (ATLYS board)

> コンテスト実行委員会コアチーム Version 2014-06-08

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



- 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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- ・ 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)
 - 技術情報揭示板
 - Google Group: HpCpsyDC2014
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>



exStickの接続方法

- ・ 入出カピン
 - 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	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

Digilent Adept	
Connect: Atiys	
Config Flash Test Power Register I/O File I/O I/O Ex Settings	1
FPGA XC6SLX45 mierusys.bit Browse Program	3
Initialize Chain Device 1: XC6SLX45 Set Config file for XC6SLX45: "C:¥Users¥ohkawa¥Desktop¥System_Atlys_t46¥fpga¥mierusys.bit" Preparing to program XC6SLX45	
Programming	



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、LDO 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

exStickBridgeを用いた リファレンスデザインの動作検証 (1)

- 以下のファイルを用いて、exStickBridgeの動作検証を行います。
 - exStickBridge_v05.bit.zip
 - System_Atlys_t63.bit.zip
 - UDP_Client_v01.jar
 - DesignCon_SDK.1.1.1.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_v01.zip (Eclipseプロジェクト)



exStickBridgeを用いた リファレンスデザインの動作検証(2)

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

ネットワーク 192.168.10.0/255.255.255.0



192.168.10.X



exStickBridgeを用いた リファレンスデザインの動作検証 (3)

- ・ DesignCon_SDK1.1.1.zip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_v01.jarと同じディレクトリに置きます
 - 310sort512.bin, 320mm512.bin, 330stencil512.bin, 340spath512.bin
- 以下のコマンドで、4つのアプリを順次実行します。

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

10.00.0	1	1007.10 2 M			
😋 🕞 🔻 📔 « Data (K:) 🕨 cy	gwin	home > ohkawa > contest		ntestの検索	Q
整理 ▼ ライブラリに追加 ▼	共有	頁▼ 書き込む 新しいフォルダー			• 🔳 🔞
🐌 Tracing	*	名前	更新日時	種類	サイズ
VirtualBox VMs		⊯ 199transfer512.bin	2014/06/03 7:12	BIN ファイル	512 KB
🔒 work			2014/06/03 7:12	BIN ファイル	512 KB
鷆 workspace		🖻 320mm512.bin	2014/06/03 7:12	BIN ファイル	512 KB
i workspace-2013		330stencil512.bin	2014/06/03 7:12	BIN ファイル	512 KB
🔋 workspace-ise		🖼 340spath512.bin	2014/06/03 7:12	BIN ファイル	512 KB
鷆 workspace-vivado		UDP_Client_v01.jar	2014/06/03 10:13	Executable Jar	7 KB
鷆 Xilinx					
퉬 Zedboard_dmastream					
🔓 アドレス帳					
┣ お気に入り	-				
6 個の項目					





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

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





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
 - LDO(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</pre>





The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest Design Guide for Altera DE-2 board

コンテスト実行委員会コアチーム Version 2014-06-08

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



- 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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- ・ 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
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>



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)





- 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

ネットワーク 192.168.10.0/255.255.255.0



192.168.10.X



DE2-115ボードの開発環境セットアップ

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





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

コンテスト実行委員会コアチーム Version 2014-06-08

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



- 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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- 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)
 - 技術情報揭示板
 - Google Group: HpCpsyDC2014
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>


exStickBridgeの接続方法

- ・ 入出カピン
 - 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)	GPIODGND	
INIT_FPGA_X(Pmod10番)	GPIO[9]	DE2-115 8
RST_X(Pmod9番)	GPIO[7]	
UART-RX (Pmod8番)	GPIO[5]	
UART-TX (Pmod7番)	GPIO[3]	asic
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_v05.bit.zip
 - MieruSys10.sof.zip
 - UDP_Client_v01.jar
 - DesignCon_SDK.1.1.1.tgz
- 上記ファイルに対応するプロジェクトファイルは以下の通りです
 - MieruSys10.zip (Aletra Quartus 13.1プロジェクト)
 - UDP_Client_v01.zip (Eclipseプロジェクト)



DE2-115ボード FPGAのコンフィギュレーション

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





DE2-115ボード サンプルアプリケーションの実行(1)

- Programmerで回路データ(sof)を書き込むと、LEDGO 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.





参考:LEDの説明

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



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



exStickBridgeを用いた リファレンスデザインの動作検証 (1)

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

ネットワーク 192.168.10.0/255.255.255.0



192.168.10.X



exStickBridgeを用いた リファレンスデザインの動作検証(2)

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

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

10.00.0	1.4.57.15 2.45		-	
😋 🔵 🔻 📙 « Data (K:) 🕨 cy	gwin ▶ home ▶ ohkawa ▶ contest		ntestの検索	Q
整理 ▼ ライブラリに追加 ▼	共有 ▼ 書き込む 新しいフォルダー		:	• 🔟 🔞
🐌 Tracing	^ 名前 [^]	更新日時	種類	サイズ
퉬 VirtualBox VMs	199transfer512.bin	2014/06/03 7:12	BIN ファイル	512 KB
🛯 🐌 work		2014/06/03 7:12	BIN ファイル	512 KB
Workspace	🖬 320mm512.bin	2014/06/03 7:12	BIN ファイル	512 KB
Workspace-2013	330stencil512.bin	2014/06/03 7:12	BIN ファイル	512 KB
Workspace-ise	🖬 340spath512.bin	2014/06/03 7:12	BIN ファイル	512 KB
🐌 workspace-vivado	UDP_Client_v01.jar	2014/06/03 10:13	Executable Jar	7 KB
🐌 Xilinx				
퉬 Zedboard_dmastream	=			
🔓 アドレス帳				
💦 お気に入り	-			
6 個の項目				





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

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

🛍 127.0.0.1:20000 - /cygdrive/k/cygwin/home/ohkawa/contest VT
ファイル(E) 編集(E) 設定(S) コントロール(Q) ウィンドウ(W) ヘルプ(H)
ohkawa@ohkawa-PC /cygdrive/k/cygwin/home/ohkawa/contest \$
<pre>chkawa8chkawa-PC /cygdrive/k/cygwin/home/ohkawa/contest \$ java -cp UDP_Client_v01.jar jp.ac.utsunomiya.is.UDP_Client 192.168.10.64 targetHost:192.168.10.64 targetPort:8100 UDP Socket created. Forward thread started! Backward thread started! 64KB sent 128KB sent 128KB sent 226KB sent 320KB sent 320KB sent 320KB sent 512KB sent sort n=307200</pre>
9418396 3774594 6594987 7448053 4782202 2429027 9454881 14506908 15022358 387226 8 67313 7727276 2958226 10505339 15852362 14194959 167736 4313118 683746 7448221 2926680 6149217 986845 9436079 2247151 14741336 9536931 8745721 3470875 1457566 0 12926306 1289274 1573040 2744079 3560112 6355242 5173105 13014990 4084930 341 8242 110037 4152237 11145510 3068254 14657566 10220646 445985 14825290 1453750 1169716 5496279 683197 7318916 6483106 10119257 95666046 4447804 28784949 1534523 7918655 677368 14460808 4039665 2250379 427641 7590767 8605590 5600714 3282507 1 2690486 9018921 3938508 65469 3387176 7006723 14722995 13607781 7492665 12771025 11364270 866238 1480042 12047420 15981203 7973098 5389410 8769982 12420850 826 8306 10304455 3562233 8945617 7987989 7592860 11195937 8415570 15183565 3024249 14016221 2234792
8512 330842 497609 661656 831568 999831 1164222 1331537 1504335 1677153 1841675 2012798 2186457 2357431 2523217 2684523 2848635 3016839 3181530 3350741 3523526 3669216 386042 4024926 4192497 4360337 4525864 4687753 4851843 5025996 5196469 5366266 5533259 5693348 5857749 6025910 6193306 6354003 6519132 6689238 6859754 7028396 7198680 7368066 7537342 7705375 7874728 6041066 8202271 886611 8537383 8707216 8871567 9034328 9198988 9367251 9542528 9710514 9870079 10034760 1020024 7 10373717 10542836 10704715 10870130 11048655 11230251 11405343 11579278 117471 82 11910929 12070824 12236611 12407825 <td< td=""></td<>
END finished! after-before = 19090204811 (ns) = 19.090204811(s) UOP Socket created. started! Forward thread started! Backward thread started! 64KB sent 128KB sent



Document P43

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 2014-06-06

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



- 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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- 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
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>





 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





- 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 access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





メモリフ		
開始アドレス	終了アドレス	
0×0000_0000	0x0003_FFFF	on-chip memory
0×0004_1000	0×0004_1000	UART_TX
0×0004_1020	0×0004_102F	MIPS
0×0800_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/unvde2-115-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



New Project Wizard Directory, Name, Top-Level Entity [page 1 of 5] What is the working directory for this project? /cadhome/kazuya/2014ProcessorDesignContest/Altera/MieruSys11 What is the name of this project? MieruSys name of the top-level design entity for this project? This name is case the entity name in the design file. MieruSys Use Existing Project Settings.

Name filterでデバイス名の候補を filteringできる

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation You can install additional device support with the Install Devices command on the Tools menu

Device family	Show in 'Available devices' list
Eamily: Cyclone IV E	Pac <u>k</u> age: Any
Devices: All	Pin <u>c</u> ount: Any
Target device	Sp <u>e</u> ed grade: Ay

Pac <u>k</u> age:	Any			
Pin <u>c</u> ount:	Any			
Sp <u>e</u> ed grade:	ану			
Name filter	ep4ce115F29			
Show advanced devices				

- C Auto device selected by the Fitter
- Specific device selected in 'Available devices' list

C Other: n/a

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Em
EP4CE115F29C7	2V	114480	529	3981312	532
EP4CE115F29C8	1.2V	114480	529	3981312	532
EP4CE115F29C8L	1.0V	114480	529	3981312	532
EP4CE115F29C9L	1.0V	114480	529	3981312	532
EP4CE115F29I7	1.2V	114480	529	3981312	532
EP4CE115F29I8L	1.0V	114480	529	3981312	532
4	-				



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



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

<u>k</u>	SDRAM Controller - new_sdram_contro
SDRAM Col altera_avalon_ne	ntroller ew_sdram_controller
Block Diagram Show signals show signals ew_sdram_controller_(clk conduit avaion_new_sdram_controller	Memory Profile Timing Data Widt: Bits: 32 * Architecture Chip select: 1 Banks: 4 * Address Width Row: 13 Column: 10 * Generic Memory model (simulation only) Include a functional memory model in the system testbench Memory Size = 128 MBytes 33554432 x 32 1024 MBits



SDRAM ControllerのTiming設定

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

SDRAM C altera_avalon	SDRAM Controlle ontroller new_sdram_controller	er - new_sdram_	controll
Block Diagram Show signals Image: sev_sdram_controller_(clk clock reset s1 avaion wire conduit avaion_new_sdram_controller	Memory Profile Timing CAS latency cycles:: Initialization refresh cycles: Issue one refresh command every: Delay after powerup, before initialization Duration of refresh command (t_rfc): Duration of precharge command (t_rp): ACTIVE to READ or WRITE delay (t_rcd): Access time (t_ac): Write recovery time (t_wr, no auto precharge)	 1 2 3 7.8125 200.0 70.0 20.0 20.0 5.5 14.0 	us us ns ns ns ns ns



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)

][ţΞ.	System	o Contents	🛛 🛛 🛛 Address Map				
	+	Use	Connecti	Name	Description	Export	Clock	
		~		🗆 clk_0	Clock Source			
	×		⊳	clk_in	Clock Input	clk	exported	
			⊳	clk_in_reset	Reset Input	reset		
			\longrightarrow	clk	Clock Output	Double-click to export	clk_0	
				clk_reset	Reset Output	Double-click to export		
		 		🛛 sdram	SDRAM Controller			
	-	($\bullet \longrightarrow$	clk	Clock Input	Double-click to export	clk_0	
			$\rightarrow \rightarrow$	reset	Reset Input	Double-click to export	[clk]	
	×			s1	Avalon Memory Mapped Slave	Pounte-citcle to export	[clk]	Ш°.
			\sim	wire	Conduit	sdram_wire		
	7		¢0	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clk_0のclkと接続する

<u>k</u>		On-Chip Memory (RAM or ROM) - onchip_mem
🚈 On-Chip Mei	mory (RAM or ROM)	
MegoCore altera_avalon_onch	ip_memory2	
* Block Diagram		
Show signals	Type:	RAM (Writeble)
onchip_memory2_0	Dual-port access	
dk 1	Single clock operation	
clock	Read During Write Mode:	DONT_CARE -
31 avalon	Block type:	AUTO 🖵
reset		
altera_avalon_onchip_memory2	▼ Size	
	Data width:	
	Total memory size:	262144 bytes
	Minimize memory block usa	in the impact for
	Read latency	
	Slave ST Latency.	1 🔻



- 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

F	ile <u>T</u> emplates
Kenter (Component Type Files Parameters Signals Interfaces
<u>File E</u> dit <u>Sy</u> stem <u>G</u> enerate <u>V</u> iew <u>T</u> ool:	About Component Type
Library S - Clock and Reset	Name: mips_avalon_interface Display name: mips_avalon_interface /ersion: 1.0 Group: My Own IP Core Description: Created by: Con: Title URL + -



- 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

	Lomponent Editor - mips_avalon_interface_hw.tcl*							
	<u>F</u> ile <u>T</u> emplates							
	Component Type Files Darar	neters Signals Interfaces						
	About Files							
	Synthesis Files							
	These files describe this component's implementation, and will be created when a Quartus II synthesis model is generated. The parameters and signals found in the top-level module will be used for this component's parameters and signals.							
	Output Path	Source File	Type	Attributes				
	mips_avalon_interface.v	mips_avalon_interface.v	Verilog HDL	Top-level File				
この不ダン	MipsCore.v	MipsCore.v	Verilog HDL	no attributes				
を抽す事で								
6 JI) + C								
ファイルの								
追加	+ - Analyze Synthesis Fil	es Create Synthesis File from	Signals					





- ・ 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.

Component Type Files Parameters Signals Interfaces Component Type Files Parameters Signals Interfaces							
About Signals About Signals							
Name	Interface	Sig	n Nama	Interface	Signal Type	Width	Direction
clock	clock reset	reset n	clock	clock sink	cll/	1	input
reset_n	new Avalon Memory Mapped Tristate Slave	reset_n		reset sink	LIK reset n	1	input
avs_s1_address	new AXI Master	address		resecsnik	I esec II	1	input
avs_s1_read	new AXI Slave	read	dvs_s1_duuress		road	1	input
avs_s1_write	new AXI4 Master new AXI4 Slave new Clock Output	write	dvS_S1_redu	51	reau	1	input
avs_s1_chipselect		chipselect	avs_s1_write	51	write	1	input
avs_s1_readdata		readdata	avs_s1_cnipselect	SI	chipselect	1	Input
avs_s1_writedata	new Clock Input	writedata	avs_s1_readdata	\$1	readdata	32	output
avm_imem_auuress	new Conduit	auuress	avs_s1_writedata	\$1	writedata	32	Input
avin_inteni_teau	new HSSI Ronded Clock Output	reau	avm_imem_address	Imem	address	32	output
clock_sinkを選択している所			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





- ・ 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





Component Type	Files Parameters	Signals	Interfaces					
About Interfaces								
r "s1" (Avalon Memory Mapped Slave)								
Nam	ne: s1			<u></u> oc				
Тур	e: Avalon Memory Ma	oped Slave		-				
Associated Cloo	:k. clock_sink			-				
Associated Res	et reset_sink			-				
Assignmen	ts: Euit							

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
 - ー 「mish(スンをクリック) 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			
	clk_reset	Re		
	🗆 sdram	SD		
\mapsto	clk	Clc		
$\diamond \longrightarrow$	reset	Re		
$\rightarrow \rightarrow \rightarrow$	s1	Av		
	wire	Co		
	onchip_memory	On		
\rightarrow	clk1	Clo		
$\bullet \bullet \rightarrow$	sl	Av condu	lit_ena_00export	
\diamond \rightarrow	reset1	Re.sec. mpar	Representation and the second	
	🗆 mips_avalon_inter	miips_avalon_interface		
	sl	Avraion Memory Mapped Slave	Double-click to export [c	
	imem	Avralon Memory Mapped Master	Double-click to export [c	
	dmem	Avralon Memory Mapped Master	Double-click to export [c	
••	conduit_end_0	Conduit	mips_led [c	
\mapsto	clock_sink	Clock Input	Double-click to export cl	
$\diamond \longrightarrow$	reset_sink	Reset Input	Double-click to export [c	



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

	•	r		L CIK_U	Clock Source		
	~		⊳-	clk_in	Clock Input	clk	ex
	-2		⊳-	clk_in_reset	Reset Input	reset	
				clk	Clock Output	Double-click to export	clk
				clk_reset	Reset Output	Double-click to export	
		~		🖻 sdram	SDRAM Controller		
ll i	-		\bullet \rightarrow	clk	Clock Input	Double-click to export	cH
	_		$\diamond \longrightarrow$	reset	Reset Input	Double-click to export	[Cl
	-		$ \uparrow \uparrow \uparrow \uparrow \rightarrow$	s1	Avalon Memory Mapped Slave	Double-click to export	[Cl
				wire	Conduit	sdram_wire	
11	Ш	2		🗆 onchip_memory	On-Chip Memory (RAM or ROM)		
			$\bullet \longrightarrow \bullet$	clk1	Clock Input	Double-click to export	cH
			$ \bullet \bullet \bullet \to \rightarrow$	s1	Avalon Memory Mapped Slave	Double-click to export	[CI
			$ \diamond \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow $	reset1	Reset Input	Double-click to export	[Cl
		~		🗆 mips_avalon_inter	mips_avalon_interface		
			$ \uparrow \uparrow \bullet \bullet \bullet \rightarrow$	s1	Avalon Memory Mapped Slave	Double-click to export	[Cl
				imem	Avalon Memory Mapped Master	Double-click to export	[Cl
				dmem	Avalon Memory Mapped Master	Double-click to export	[Cl
				conduit_end_0	Conduit	mips_led	[Cl
			$+$ + + + + \rightarrow	clock_sink	Clock Input	Double-click to export	cH
			\diamond	reset_sink	Reset Input	Double-click to export	[Cl
		2		🗆 contest_uart_avalo	contest_uart_avalon_interface		
			$ \diamond \diamond \diamond \rightarrow \rightarrow$	s1	Avalon Memory Mapped Slave	Double-click to export	[CI
				m1	Avalon Memory Mapped Master	Double-click to export	[CI
				conduit_end_0	Conduit	export	[CI
			$\bullet \longrightarrow$	clock_sink	Clock Input	Double-click to export	cH
			$\diamond \longrightarrow$	reset sink	Reset Input	Double-rlick to export	[r]



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





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 cO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





PLLの追加		
MegaWizard Plug-in Manager [page 8 of 14] (Wish L) X ALTPLL About Documentation		
Parameter PLL Curconfiguration Cit co cit cl cit cl cit co cit cl cit cl cit co cit cl cit cl pil cit co cit co incluido fraquency 50 000 Mbr cock cit co cit co cit co pil cit co cit co cit co cit co cit co	チェックを外す チェックを外す LegaWiz Construction	画面4 ard Plug-In Manager [page 14 of 14] (wish上) × About Documentation
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	pll inclk0 areset Operation Mode: Normal Cik Ratio Ph. (dg) DC (%) c 0.4/5 0.00 50.00 Cyclone IV E	Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates an optional file. Clck Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions. The MegaWizard Plug-In Manager creates the selected files in the following directory: /cadome/kazuya/2014ProcessorDesignContest/Altera/MieruSys11/ File Description Image: Pli N Variation file Image: Pli N Variation template file Image: Pli N Verilog HDL black-box file



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

1	Settings - MieruSys	
Category:		Device
General	Files	
 Files Libraries Operating Settings and Conditio Voltage 	Select the design files you want to include in the project. Click Add All to add all design files in project directory to the project.	n the
 Temperature Compilation Process Settings 	File Name Tune Library Design Entry/Synthest Teel UDL Version	Add
Early Timing Estimate	mips Qsys System File	Add All
 Physical Synthesis Optimizat EDA Tool Settings Design Entry/Synthesis 	MieruS Verilog HDL File <none> Default Default Pll.qip IP Variation File (.gip) <none></none></none>	<u>R</u> emove Up
ここをクリッ	っつして追加するファイルを選択し、その後addボタンを押	时



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







<	Named: *GPIO* 💌 🐇	» Edit: 💢 🗹								
7 7	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	:urrent Strengtl	Slew Rate	Difi
	GPIO_FLUSH_X	Output	PIN_AE16	4	B4_N2	3.3-V LVCMOS		2mA (default)	2 (default)	
	B_ GPIO_INIT_X	Input	PIN_AE15	4	B4_N2	3.3-V LVCMOS		2mA (default)		
	B_ 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)	
	③ GPIO[0]	Unknown	PIN_AB22	4	B4_N0	3.3-V LVTTL		8mA (default)		
		Unknown	PIN_AC15	4	B4 N2	3 3-V I VTTI		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 2014-06-08

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



- このドキュメントでは、リファレンスデザインに含まれるプロセッサの構成(アーキテクチャ)について説明します。
- また、Xilinx ISEを用いて、リファレンスデザインの回路ファイル(bitファイル) を生成する方法を示します。
- ・ 設計コンテストのWEBサイト
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- 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
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>



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O番アドレスから順に格納します。
 - すなわち, 0x00000000 ~ 0x0007FFFF に, 512KBのデータが格納されます.
- リファレンスデザインでは、プロセッサが実行する命令を格納するために、 64KBの命令メモリを実装しています。
 - 512KBのデータの先頭の64KBのみが,この命令メモリに格納されます.



Memory Mapped I/O of the Reference Design

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

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

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

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



リファレンスデザインに含まれるプロセッサ

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



89

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

If stage





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

Id stage



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'hla:	begin	IdOPN='DIV	; IdDST=IdRD;		end
196	6′h1b:	begin	IdOPN='DIVU	; IdDST=IdRD;		end
197	endcase					
198	6'h01: case	e (IdR	Г)			
199	5′h00:	bègin	IdOPN='BLTZ	; IdDST=0;		end
200	5′h01:	begin	IdOPN='BGEZ	; IdDST=0;		end
201	endcase					
202	6′h02:	begin	IdOPN='J	; IdDST=0;		end
203	6′h03:	begin	IdOPN='JAL	; IdDST=31;		end
204	6′h04:	begin	IdOPN='BEQ	; IdDST=0;		end
205	6′h05:	begin	IdOPN='BNE	; IdDST=0;		end
206	6′h06:	begin	IdOPN='BLEZ	; IdDST=0;		end
207	6′h07:	begin	IdOPN='BGTZ	; IdDST=0;		end
208	6′h08:	begin	IdOPN='ADDI	idDST=IdRT;		end
209	6′h09:	begin	IdOPN='ADDIU	; IdDST=IdRT;		end
210	6′h0a:	begin	IdOPN='SLTI	; IdDST=IdRT;		end
211	6′h0b:	begin	IdOPN='SLTIU	; IdDST=IdRT;		end
212	6′h0c:	begin	IdOPN='ANDI	; IdDST=IdRT;		end
213	6'h0d:	begin	IdOPN='ORI	; IdDST=IdRT;		end
214	6'h0e:	begin	IdOPN='XORI	; IdDST=IdRT;		end
215	6'h0f:	begin	IdOPN='LUI	; IdDST=IdRT;		end
216	6′h20:	begin	IdOPN='LB	; IdDST=IdRT;	IdATTR='LD_1B;	end
217	6′h21:	begin	IdOPN='LH	idDST=IdRT;	IdATTR='LD_2B;	end
218	6′h23:	begin	IdOPN='LW	; IdDST=IdRT;	IdATTR='LD_4B;	end
219	6′h24:	begin	IdOPN='LBU	; IdDST=IdRT;	IdATTR='LD 1B;	end



データパス - Idステージ - (2/5)

フェッチした命令をデコードします.

```
220
                 6'h25:
                            begin IdOPN='LHU
                                                 ; IdDST=IdRT; IdATTR='LD 2B;
                                                                                         end
                 6′h28:
                            begin IdOPN='SB
                                                   ; IdDST=0;
                                                                 IdATTR='ST 1B;
221
                                                                                         end
222
                 6′h29:
                            begin IdOPN='SH
                                                  ; IdDST=0; IdATTR='ST 2B;
                                                                                         end
223
                 6'h2b:
                            begin IdOPN='SW
                                                   ; IdDST=0;
                                                               IdATTR='ST 4B;
                                                                                         end
224
             endcase
225
         end
226
227
         wire ['ADDR] IdBPC = IfId npc + ({{16{IfId ir[15]}}}, IfId ir[15:0]} << 2);
         always @(*) begin ///// branch & jump resolution unit
228
             \{IdTPC, IdC, IdB\} = 0;
229
230
             case (IdOP)
                 6'h04: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS == IdRRT);
231
                                                                                      end
                                                                                                 // BEO
                 6'h05: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS != IdRRT);
232
                                                                                      end
                                                                                                 // BNE
                 6'h06: begin IdB=1; IdTPC = IdBPC; IdC = ( IdRRs[31]||(IdRRs==0)); end
                                                                                                 // BLEZ
233
                 6'h07: begin IdB=1; IdTPC = IdBPC; IdC = (~IdRRS[31]&&(IdRRS!=0)); end
                                                                                                 // BGTZ
234
                 6'h01: begin IdB=1; IdTPC = IdBPC; IdC = (IdRT) ? ~IdRRS[31] : IdRRS[31]; end // BGEZ,BLTZ
235
                 6'h02: begin IdB=1; IdTPC = IfId ir['ADDR]<<2; IdC = 1; end
236
                                                                                                 // J
                                                                                                 // JAL
237
                 6'h03: begin IdB=1; IdTPC = IfId ir['ADDR]<<2; IdC = 1; end
238
                               (IdFCT==6'h08) begin IdB=1; IdTPC = IdRRS; IdC = 1; end
                                                                                                 // JR
                 6'h00: if
239
                        else if (IdFCT==6'h09) begin IdB=1; IdTPC = IdRRS; IdC = 1; end
                                                                                                 // JALR
240
             endcase
241
         end
242
243
         always @(posedge CLK or negedge RST X) begin ///// update pipeline registers
             if(!RST X) {IdEx npc, IdEx rrs, IdEx rrt, IdEx dst, IdEx ir, IdEx opn, IdEx attr} <= 0;
244
245
             else if(!PSTALL) begin
246
                 IdEx npc <= (bstall) ? 0 : IfId npc;</pre>
                 IdEx rrs <= (bstall) ? 0 : IdRRS; // data from general-purpose register file
247
248
                 IdEx rrt <= (bstall) ? 0 : IdRRT;</pre>
                                                         // data from general-purpose register file
249
                 IdEx dst <= (bstall) ? 0 : IdDST;</pre>
250
                 IdEx ir <= (bstall) ? 0 : IfId ir;</pre>
251
                 IdEx opn <= (bstall) ? 0 : IdOPN;</pre>
252
                 IdEx attr <= (bstall) ? 0 : IdATTR:
253
             end
254
         end
255
```



データパス - Exステージ - (3/5)

命令操作の実行またはアドレス生成を行います

Ex stage



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) : begin ExRSLT = RRS U + RRT U; 286 'ADD end : begin ExRSLT = RRS U + SET32I; 287 'ADDI end : begin ExRSLT = RRS U + SET32I; 288 'ADDIU end ADDU begin ExRSLT = RRS U + RRT U; 289 end : begin ExRSLT = RRS U - RRT U; 290 'SUB end begin ExRSLT = RRS_U - RRT_U; 291 'SUBU end 292 'AND begin ExRSLT = RRS U & RRT U; end 293 ANDT begin ExRSLT = RRS_U & {16'h0, IMM}; end 294 INOR begin ExRSLT = ~(RRS_U | RRT_U); end : 295 begin ExRSLT = RRS U | RRT U; 'OR end 296 ORI begin ExRSLT = RRS_U {16'h0, IMM3: end 297 ' XOR begin ExRSLT = RRS_U ^ RRT_U; end : begin ExRSLT = RRS_U ^ {16'h0, IMM}; 298 'XORI end begin ExRSLT = RRT U << SHAMT; 299 SLL : end 300 SRL begin ExRSLT = RRT U >> SHAMT; end 301 : begin ExRSLT = RRT S >>> SHAMT; 'SRA end 302 'SLLV : begin ExRSLT = RRT U << RRS U[4:0]; end : begin ExRSLT = RRT U >> RRS U[4:0]; 303 'SRLU end 304 'SRAV begin ExRSLT = RRT_S >>> RRS_U[4:0]; end begin ExRSLT = (RRS_U[31] ^ RRT_U[31]) ? RRS_U[31] : (RRS_U < RRT_U); begin ExRSLT = (RRS_U[31] ^ IIMM[15]) ? RRS_U[31] : (RRS_U < SET32I) 305 'SLT end 306 'SLTI IMM[15]) ? RRS_U[31] : (RRS_U < SET32I);</pre> 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 : begin ExRSLT = IdEx npc + 4; 310 JALR end 311 'LUI : begin ExRSLT = {IMM, 16'h0}; end : begin ExRSLT = ExA; 312 'LB end 313 'LBU begin ExRSLT = ExA; end . begin ExRSLT = {ExA[31:1], 1'b0 }; begin ExRSLT = {ExA[31:1], 1'b0 }; 314 'LH end 'LHU 315 end begin ExRSLT = {ExA[31:2], 2'b00}; 316 1 L.W . end begin ExRSLT = ExA; 317 1SB ExWE = {4'b0001<<ExA[1:0]}; end begin ExRSLT = {ExA[31:1], 1'b0 }; ExWE = ExA[1] ? 4'b1100 : 4'b0011; 318 4 SH end 319 'SW begin ExRSLT = {ExA[31:2], 2'b00}; ExWE = 4'b1111; end : begin ExRSLT = hi; 320 'MFHI 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 if(!RST_X) {hi, lo} <= 0; else if(!PSTALL) begin 326 327 328 if(IdEx_opn == 'MULT {hi, lo} <= RRS_S * RRT_S;</pre> 329 if(IdEx_opn == 'MULTU_ {hi, lo} <= RRS_U * RRT_U;</pre> 330 if (IdEx_opn == 'DIV_ {hi, lo} <= (RRT_U) ? DURSLT : 0;</pre> 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; ExMa_std <= (IdEx_opn=='SB_ _) ? {4{RRT_U[7:0]}} : 340 (IdEx_opn=='SH 341) ? {2{RRT_U[15:0]}} : RRT_U; 342 ExMa mwe <= ExWE; 343 ExMa oe <= (IdEx_attr & 'LDST_ANY) ? 1 : 0; <= (IdEx opn=='LH) ? 2 : // Load selector 344 ExMa lds) ? 1 : (IdEx_opn=='LHU (IdEx_opn=='LB 345) ? 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

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 •

Wb stage

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」

Console 🔕 Errors 🔔 Warnings 😹 Find in Files Results

Document P34

The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest DRAM Memory Interface (ATLYS board)

> コンテスト実行委員会コアチーム Version 2014-06-08

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

- 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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- ・ 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
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>

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								
	Pin Compatible EPGAs							
	Pin Compatible FPGAs Pin Compatible FPGAs include all devices with package do not have the same bonded pins. B device and all selected devices. Use the defau UCF in the compatible uct folder. If you do r the generated UCF may not work for th if the package and speed grade matches to th	the same package and speed grade as the targe v selecting Pin Compatible FPGAs, MIG will only a ilt UCF in the par folder for the target part. If you tot choose a Pin Compatible FPGA now ar e new device and a board spin may be re a target part. MIG only ensures that MIG generate	t device. Different FPGA devices with the same select pins that are common between the target change the target part, use the appropriate d need to use a different FPGA later , juried . A device is considered compatible only d pin out is compatible among the selected					
	compatible FPGA devices. Unselected devices will not be considered for compatibility during the pin allocation process.							
	Target FPGA xc6slx45-csg324 -3	ne parts exist for the selected target part and this	arget part and this page can be skipped.					
	Pin Compatible FPGAs	Available MCBs						
	✓ spartan6							
	▲ 6s	MEMC1,MEMC3						
Managur	xc6slx16-csg324	MEMC1, MEMC3						
wemory	xc6slx25-csg324	MEMC1, MEMC3						
Interface								
Generator								
EXILINX .								
User Guide MCB User Guide Versi	on Info		< Back Next> Cancel					

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 Generato	r 🕞 🖬 🐱						
	Memory Selection						
DESIGN 🔛	Select the memory interface type from the Memory Type selection box provided for each bank. Hardware verified devices are listed in the User Guide.						
Controller Options Memory Options Multi-port Configuration AXI Parameter Arbitration FPGA Options Summary Memory Model PCB Information Design Notes	The MCB in Bark 3 (marked with an asterisk below) has fewer multi-purpose IO pins and is therefore the preferred location for designs with a single controller. The other MCB locations have more multi-purpose pins. Check your design to make sure there are no conflicts with MCB interface pins. AXI interface in the face for All MCBs Extended MCB performance range Extended MCB performance node requires a different Vocint specification to achieve higher maximum frequencies for DDR2. Consult the Spartan-5 datasheet (DS152) table 2 and 24 for more information. Memory Type Image: second						
User Guide MCB User Guide Version Info							

Xilinx Memory Interface Generator (5)

・ 3000 ps, EDE1116AXXX-8E を選択

🏹 Xilinx Memory Interface Generator		- • •
REFERENCE DESIGN 🗄	Options for C3 - DDR2 SDRAM	
	Frequency: The allowed frequency range(%1 - %2) is a function of the selected FPGA part, FPGA speed grade and Memory Controller type. Choose the clock period for the desired frequency. Refer to User Guide for supported frequency range.	3000 🌩 ps 333.33 MHz
Controller Options Memory Options Multi-port Configuration	Memory Part: Select the memory part. Parts marked with a warning symbol are not compatible with the frequency selection above. Find an equivalent part or create a part using the "Create Custom Part" button if the part you want is not listed here.	EDE1116AXXX-8E
AXI Parameter Arbitration FPGA Options		
Summary Memory Model PCB Information Design Notes		
	L Memory Details: 1Gb, x16, row:13, col:10, bank:3, data bits per strobe:8, with data mask, single rank	
User Guide MCB User Guide Versio	n Info	ack Next> Cancel

Xilinx Memory Interface Generator (6)

・ Fullstrength, 50ohms, Enable, Disable を選択

🏹 Xilinx Memory Interface Generator		J
REFERENCE	Memory Options for C3 - DDR2 SDRAM	
DESIGN 🖽	Choose the Memory Options settings for the memory device. Settings are restricted to those supported by the controller.	
Controller Options Memory Options Multi-port Configuration AXI Parameter Arbitration FPGA Options Summary Memory Model PCB Information Design Notes	Choose the Memory Options settings for the memory device. Settings are restricted to those supported by the controller. Output Drive Strength Selecting reduced strength will reduce all outputs to approximately 60 percent of the drive strength. Fullstrength •• RTT (nominal) - ODT This feature allows to apply internal termination resistance of the memory module for signals DQ, DQS/DQS#, LDQS/LDQS#, UDQS/UDQS# and LDM/UDM. This improves the signal integrity of the memory channel. DQS# Enable Crosstalk and simultaneous switching output impact on the strobe output driver can be reduced with this option ON. When Enabled DQS is differential and when disabled DQS is single-ended. High Temparature Self Refresh Rate Set this bit to enable self-refresh rate in case of higher than 85 C temperature self-refresh operation. Application Disable value selfrefresh L/F to enter self refresh state.	
User Guide MCB User Guide Versi	on Info	
User Guide MCB User Guide Versi	or Info	

Xilinx Memory Interface Generator (7)

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

Xilinx Memory Interface Generator REFERENCE	Port Configuration f	or C3 - DDR2 SDR	AM	
	Select one of five cor figure and table will g Configuration Selec	nfigurations from the co et updated. You can se tion	onfiguration menu and t lect the number of port	the ports from the table. As you select the port configuration, the below ts in a configuration, and data port settings from the table.
Controller Options	Port Selection	Interface	Direction	
Memory Options 🖌	Port0	NATIVE	Bi-directional The second s	
Multi-port Configuration C3	Port2	NATIVE	none →	
AXI Parameter C3	Port3	NATIVE	v none v	
Arbitration C3	Port4	NATIVE	<pre>v none v v none v v</pre>	
FPGA Options C3		L		
Summary Memory Model	- Memory Address M. Us	apping Selection er Address		
PCB Information Design Notes	A 2 9		A 0	
	ROW	BANK C	DLUMN	
	BANK	ROW CO	DLUMN	
EXILINX .				h
User Guide MCB User Guide Versio	on Info			< Back Next> Cancel

Xilinx Memory Interface Generator (8)

・ Next で次に進む

🖞 Xilinx Memory Interface Generator				
REFERENCE	Arbitration for C3 - DDR2 SDRAM			
DESIGN 🖽	Select either round robin or custom for port arbitration. You can alter the port priorities in custom arbitration. For each time slot, the leftmost port number has the highest priority. The order of port priority decreases from left to right. Each port should be given highest priority in at least one time slot. Below ports will be set to a warning symbol if the port is not given highest priority in at least to a solution.			
Controller Options C3 Memory Options C3 Multi-port Configuration C3 AXI Parameter C3 AXI Parameter C3 Arbitration C3	one time slot. Below ports will be set to a warning symbol if the port is not given h Select Arbitration Algorithm Timeslot 0 Timeslot 1 Timeslot 2 Timeslot 3 Timeslot 4 O	ighest priority in at least one time slot. Ind Robin - rt3 ② Port4 ③ Port5 ③		
FPGA Options C3 Summary Memory Model PCB Information	Timeslot 5 0 Timeslot 6 0 Timeslot 7 0			
Design Notes	Timeslot 8 0 Timeslot 9 0			
E XILINX _®	Timeslot 10 0 Timeslot 11 0			
User Guide MCB User Guide Versio	n Info	< Back Next> Cancel		



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	
REFERENCE	FPGA Options for C3 - DDR2 SDRAM
REFERENCE DESIGN	FPGA Options for C3 - DDR2 SDRAM Class I is recommended for all SSTL signals in memory interfaces. However, better signal integrity may sometimes be achieved with class I is superior for your application, select Class I below. This can be changed after generation by modifying the UCF. This option changes the drive strength for Data, Address & Control. Class for Address & Control. If IBIS simulations indicate that Class I is superior for your application, select Class I below. This can be changed after generation by modifying the UCF. This option changes the drive strength for Data, Address & Control. Class for Data Class II ▼ Memory Interface Pin Termination: Memory Interface Pin Termination: Provides calibrated on-die input termination resistors. Calibration requires two extra pins to be added to the interface: R20 and ZDO. An external resistor with a value 2x trace impedance needs to be connected from R20 pin to provide J pins need to be left unconnected. These additional pins and their locations will be listed in the generated UCF constraints file. Un-calibrated Input Termination: Provides un-calibrated (approximated) on-die input termination resistors to Vcco and Ground. Dd/DOS 25 Ohms ▼ External Input Termination: Provides discrete termination resistors for the controller on the PCB. The selected design's timing has not been verified with non-default R2Q locations Correct (NO). The selected design's timing has not been verified with non-default R2Q locations. Correct this prin (no-cornect)
S XII INX	System Clock: Choose the desired input clock configuration.
User Guide MCB User Guide Vera	sion Info Cancel



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

JSE 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	_ <i>8</i> ×		
Design ↔ □ ₱ × 4 58 //	A		
View @ Windowentation © Simulation E 59 // Purpose : Template file containing code that can be used as a model			
Hierarchy Hierarchy Hierarchy Go // Revision History:			
Image:			
g ⊕ ∰ xc6six45-3csg324 63 ¹ / ₂ = 6 ∰ xc6six45-3csg324 63 ¹ / ₂ = 6 ∰ xc6six45-3csg324 63 ¹ / ₂ = 6 ∰ xc6six45-3csg324 63			
G G	// core to be instantiated. Change the instance name and port connections		
66 // (in parentheses) to your own signal names.	66 // (in parentheses) to your own signal names.		
Gr G	67 68 // Begin Cut here for INSTANTIATION Template// INST TAG		
📝 olockgen02 - clockgen2 (clock.v) 69			
□ · · · · · · · · · · · · · · · · · · ·			
mem0 - dram_con (dram_mem.)	7 1.05 F0 mASA 512E (3),		
U gram - oram (gram.xco) G 73 , C3 P1 MASK 372Z (4),	(3 .C3 P1_MASK_SIZE(4),		
w lodel + CONDER (system) 0 74 .C3_FI_AALA_CORL_SIZE (32),			
76 .C3_MEMCLK_PERIOD (3000),	-		
77 .C3 CALLE SOFT IP ("RUE"),			
No Processes Running 78 .CS_DIFICUATION ("FALDE"), 79 .CS RST ACT LOW (0),			
Image: State of the state o			
% CORE Generator % CORE Generator % CORE Generator			
Manage Cores 83 .C3 MEM ADDR WIDTH (13) ,			
Regenerate Core 84 .C3_MEM_BANKADDR_WIDTH (3)			
Update Core to Latest Version 85)			
View HDL Functional Model 87			
We while installation reinplace 88 .c3_aya_clk (c3_aya_clk),			
89 .cs_aya_rat_1 (c3_aya_rat_1), 90			
91 .mcb3_dram_dq (mcb3_dram_dq),			
92cb3 dram a (mcb3 dram a),			
94 .mcb3_tram_ras n (mcb3_tram_tas n),			
95. mcb3 dram cas n (mcb3 dram cas n).			
🕞 Start 🗮 Desien 🖒 Files 🖒 Libraries 🕱 Desien Summary (out of date) 🗈 🖻 r/em ven 🖪 🖹 MieruSvsy 🗵 🖹 dram mem.v			
	⇔⊓ax		
	A		
	_		
4 m			
🔋 Console 📀 Errors 🔬 Wernings 🐹 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)





The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest MIPS Cross Compiler Setup Manual

コンテスト実行委員会コアチーム Version 2014-06-08

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



- 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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- ・ 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)
 - 技術情報揭示板
 - Google Group: HpCpsyDC2014
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>



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

and the

- RedHat Linux Version 5 (aquabase)
 - mips_procdesign_redhat5.tgz を次のURLからダウンロード <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
 - \$ cd /home/share/cad/
 - \$ tar xvfz mips_procdesign_redhat5.tgz
 - \$ In -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 <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
 - \$ cd /home/share/cad/
 - \$ tar xvfz mips_procdesign_cent63.tgz
 - \$ In -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 <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
 - \$ cd /home/share/cad/
 - \$ tar xvfz mips_procdesign_ubuntu1204.tgz
 - \$ In -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.
 - <u>http://www.arch.cs.titech.ac.jp/mcore/buildroot.html</u>
 - 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。





Document P36

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

コンテスト実行委員会コアチーム Version 2014-06-08

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

リファレンスデザインのためのアプリケーション開発

- リファレンスデザインのためのアプリケーションをコンパイルするためにSDK(ソ フトウェア開発キット)を提供します.
- ホームページから DesignCon_SDK.1.0.2.tgz というファイルをダウンロードし 展開してください(バージョンアップによりファイル名が異なることがあります).
- ・ 展開したディレクトリの README.txt に使い方の説明があります.
- 設計コンテストのWEBサイト
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- 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
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>



プロセッサ設計部門の SDKに含まれるソースコードは 今後改善する予定です

The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest Application Specification of Processor Design Category

> コンテスト実行委員会コアチーム Version 2014-06-08

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

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
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- ・ 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
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>





- 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.





- 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 -シミュレータ用実行バイナリ(elf)
 - 340spath512.bin -FPGA用バイナリ(512KB)
 - 340spath.bin -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 グラフ定義ファイル (2048ノード)
 - n2048.p2p 問題定義ファイル(始点・終点の指定)
 - n6.gr 【参考】単純なグラフ定義ファイル (6ノード)
 - n6.p2p 【参考】問題定義ファイル(始点・終点の指定)

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

The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest Application Specification of Computer System Design Category

> コンテスト実行委員会コアチーム Version 2014-06-08

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



- このドキュメントでは、コンピュータシステム部門のアプリケーションプログラムの仕様を説明します
- ・ 設計コンテストのWEBサイト
 - <u>http://aquila.is.utsunomiya-u.ac.jp/contest/</u>
- ・ 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
 - <u>https://groups.google.com/forum/?hl=ja#!forum/hpcpsy2014dc</u>







- ・ コンピュータシステム部門では、以下の処理を繰り返します
 - ホストから2つの入力画像を受け取る
 - ・ 画像は独自のJPEG圧縮されたフォーマットで渡されます.
 - オプティカルフロー以外の画像関係の処理に関しては、参考文献[1]に記載されている処理をベースに作成しています。
 - 参考文献[1]
 - 昌達 慶仁 著,「詳解 画像処理プログラミング」, ソフトバンククリエイティブ 株式会社, 2008.



run_contest.sh

SDK内のファイルについて

- コンピュータシステム部門のプログラムとして,下記の一連の処理をPC上で 仮想的に実現します.

- コンテストのホストで動作する処理,および,FPGAボードで実装する処理を

PC上で実行できるファイルを生成します.(Unix系のOSを想定しています)

- ・ホストでの前処理(FPGA入力データの生成)
- FPGAでの処理(入力画像の展開,オプティカルフローの計算,オプティカルフローの描画,出力画像の圧縮)
- ・ホストでの後処理(FPGAから受け取った画像の展開)
- ・ 以下を実行する事で、どのような処理が実行されるのか分かると思います.
 - make

Makefile

- ./run_contest.sh
- ・ 以降のページで、run_contest.shが行っている処理について説明します



ホストでの前処理

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



FPGAでの処理

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



ホストでの後処理

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







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.





- 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)

