

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 2014-08-17

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
 - P61 User Manual of Optical Flow System Reference Design (Atlys board)
 - P62 User Manual of Optical Flow System Reference Design (DE2-115 board)
 - P70 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 1/6

- 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







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







- ・ 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



- ・ 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。





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 2014-08-17

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

-		
方向	Atlys	
-	PMOD[6]: 3.3V	
-	PMOD[5]: GND	
\rightarrow	PMOD[4]	ANI INCH
\leftarrow	PMOD[3]	
\leftarrow	PMOD[2]	
\rightarrow	PMOD[1]	」 ■ PMONコネクタによろ接続
	方向 - - ← ← →	方向Atlys-PMOD[6]: 3.3V-PMOD[5]: GND \rightarrow PMOD[4] \leftarrow PMOD[3] \leftarrow PMOD[2] \rightarrow PMOD[1]

・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

A Digilent	Adept		-	1			-			x	
A	ΓLΥ	S			Connec Produc	ct: Atlys ct: Atlys				•	
Config	Flash	Test	Power	Regist	ter I/O	File I/C) /0 E>	c Set	tings		(1
FPC	3A L×45	mierusys,bit		2	/	• 8	rowse	Progra	m		3
				Initialize C	hain						
Device 1: Set Config t Preparing tr Programmin Verifying pr Programmin	XC6SLX45 file for XC6S o program X ig ogramming g Successfu	ILX45: "C:¥U C6SLX45 of device ıl.	isers¥ohkawa	¥Desktop¥	System_4	Atlys_t46¥f	pga¥mierus	ys.bit"		* 	



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.







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





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.†gz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

	awin	home hokewa hostat			
	ywin	Finome Fonkawa F contest	• +) cor	itestu))使杀	ر
整理 ▼ ライブラリに追加 ▼	共	有▼ 書き込む 新しいフォルダー			• 🔟 🕐
鷆 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
鷆 work		≝ 310sort512.bin	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
퉬 workspace-ise		ظ 340spath512.bin	2014/06/03 7:12	BIN ファイル	512 KB
퉬 workspace-vivado		UDP_Client_v01.jar	2014/06/03 10:13	Executable Jar	7 KB
퉬 Xilinx					
🐌 Zedboard_dmastream					
Ъ 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_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
 - 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 2014-08-17

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





- ・ 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 2014-08-17

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	10.4
GND(Pmod11番)	GPIOのGND	
INIT_FPGA_X(Pmod10番)	GPIO[9]	DE2-115
RST_X(Pmod9番)	GPIO[7]	
UART-RX (Pmod8番)	GPIO[5]	ISSI
UART-TX (Pmod7番)	GPIO[3]	asic

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、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)

- ・ 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

1.01.0	1. MORT. 1. M	10 () () () () () () () () () (- 0 ×					
😋 🖓 🗢 📙 « Data (K:) 🕨 cyc	win 🕨 home 🕨 ohkawa 🕨	contest - + co	ontestの検索	م					
整理 ▼ ライブラリに追加 ▼	整理 ▼ ライブラリに追加 ▼ 共有 ▼ 書き込む 新しいフォルダー 部 ▼ 1 0								
鷆 Tracing	▲ 名前 ~	更新日時	種類	サイズ					
퉬 VirtualBox VMs	199transfer512.bi	n 2014/06/03 7:12	BIN ファイル	512 KB					
)) work	310sort512.bin	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					
퉬 workspace-2013	i 330stencil512.bin	2014/06/03 7:12	BIN ファイル	512 KB					
퉬 workspace-ise	340spath512.bin	2014/06/03 7:12	BIN ファイル	512 KB					
퉬 workspace-vivado	UDP_Client_v01.ja	ar 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_v06.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		Ĩ
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)		
chkawa@ohkawa-PC /cygdrive/k/cygwin/home/ohkawa/contest \$	^	
chkawa@ohkawa-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. started! Forward thread started! Backward thread started! Backward thread started! 128KB sent 128KB sent 256KB sent 320KB sent 320KB sent 512KB sent 512KB sent 512KB sent		
Sort n=307200 9418396 3774594 6594987 7448053 4782202 2429027 9454881 14506908 15022358 387226 8 67313 7727276 2958226 10505339 15852362 14194959 167736 4313118 683746 7448221 2926806 6149217 966845 9436079 2247151 14741936 9536931 8745721 3470875 1457566 0 12926306 12889274 1573040 2744079 3560112 6355242 5173105 13014990 4084930 341 8242 110037 4152237 11145510 3068254 14657566 10220646 445985 14825280 14533750 1169716 5496279 68197 7318916 6483106 10119257 9566046 5447804 2878949 1534528 7918655 677368 14460808 4030685 2250379 427641 7590767 8605590 5600714 3828507 1 2890486 9018921 3938508 65469 3387176 7006723 14722995 13607781 749265 12771025 11364270 8662336 1490042 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 2484635 3016839 3181503 350741 3523526 3692116 3860642 4024926 4192497 4360337 4525864 4687753 4851843 5025936 5196469 556256 5532525 5683348 5857749 6025910 6193306 6354003 6519132 6689238 658754 7028396 7198680 7368066 753734 2705375 7874728 8041056 8202271 8366811 6537383 8707216 8871567 9034328 9198988 9367251 9542528 9710514 9870079 10034760 1020024 7 10373717 10542836 10704715 10870130 11048655 11230251 11405343 11579278 117471 82 11910929 12070824 12236471 12407925 12576223 1235610 12891874 13058709 13234 413 13408574 13582050 13750971 13913922 14082688 14250652 14416913 14581779 1475 1073 14912700 15075738 15246462 15417868 15584370 15753811 15921987 16090379 162 55590 16427659 16600045 16767460		
END finished! after-before = 19090204811 (ns) = 19.090204811(s) UDP 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 2014-08-17

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 What is the name of the top-level design entity for this project? This name is case me ently mane 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

Name filter ep4ce115F29

Show advanced devices

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 n 'Available devices' list-
amily: Cyclone IV E	•	Pac <u>k</u> age: Any
Dev <u>i</u> ces: All	Ψ.	Pin <u>c</u> ount: Any
Target device		Sp <u>e</u> ed grade: y

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

C Other: n/a

Available devices:

Na	me	Core Voltage	LEs	User I/Os	Memory Bits	Em
EP4CE11	5F29C7	0.2V	114480	529	3981312	532
EP4CE11	5F29C8	1.2V	114480	529	3981312	532
EP4CE11	5F29C8L	1.0V	114480	529	3981312	532
EP4CE11	5F29C9L	1.0V	114480	529	3981312	532
EP4CE11	5F29I7	1.2V	114480	529	3981312	532
EP4CE11	5F29I8L	1.0V	114480	529	3981312	532
•						



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





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

4	SDRAM Controller - new_sdram_control
SDRAM Co altera_avalon_i	ontroller new_sdram_controller
Block Diagram Show signals Show signals ew_sdram_controller_(clk clk clk clk clock reset reset s1 avalon wire conduit a_avalon_new_sdram_controller	Memory Profile Timing • Data Widtb 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

<u>å</u>	SDRAM Controlle	er - new_sdram	controll
SDRAM C altera_avalon	ontroller new_sdram_controller		
Block Diagram Block Diagram Show signals ew_sdram_controller_0 clk_clock reset reset reset reset reset 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 2 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)

00	ols <u>F</u>	<u>-</u> elp						
1	13	Systen	n Contents	🛛 🛛 🛛 Address Map	🕮 Project Settings 🕮			
	4	Use	Connecti	Name	Description	Export	Clock	
				🗆 clk_0	Clock Source			
	-	T	머	clk_in	Clock Input	clk	exported	
			머	clk_in_reset	Reset Input	reset		
				clk	Clock Output	Double-click to export	clk_0	
		T		clk reset	Reset Output	Double-click to export		
				a sdram	SDRAM Controller			
	-	i ($\bullet \rightarrow \rightarrow$	clk	Clock Input	Double-click to export	clk_0	
			\rightarrow	reset	Reset Input	Double-click to export	[clk]	
				s1	Avalon Memory Mapped Slave	Pour elicity export	[clk]	Ш°.
			•••	wire	Conduit	sdram_wire		
	1]						



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>å</u>		On-Chip Memory (RAM or ROM) - onchip	_mem					
On-Chip Memory altera_avalon_onchip_mem	(RAM or ROM)							
Block Diagram Show signals	Aemory type pe:	RAM (Writable)						
onchip_memory2_0] Dual-port access] Single clock operation ad During Write Mode:	DONT CARE						
s1avalon Blo	ock type:		-	ſ			N MMPTE STRUCT PROPERTY	1
altera_avalon_onchip_memory2	ize		_		< clk_reset	SDRAM Controllor	Double-click to export	
Dat	ata width:		_			Clock Input	Double-rlick to export	l
Tot	otal memory size:	262144 bytes		\$	> reset	Reset Input	Double-click to export	
	Minimize memory block usag	a may impact the			s1	Avalon Memory Mapped Slave	Double-click to export	l
R	lead latency			_ <u>~</u> ~	> wine	Conduit	sdram_wire	
Sia	ave si Latency.	1 🗸	Y		E onchip_memory	n-Chip Memory (RAM or ROM)		
					→ _1k1	Clock Input	Double-click to export	1
					s1	Avalon Memory Mapped Slave	Double-click to export	[
				0	reset1	Reset Input	Double-click to export	[



- 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

	🖕 Component Editor - mips_avalon_interi
	<u>F</u> ile <u>T</u> emplates
🗶 daha - mikaTahau	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 – C –	Name: mips_avalon_interface Display name: mips_avalon_interface Version: 1.0 Group: My Own IP Core Description:



- 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

	4	Component Editor - mips	_avalon_interface_hw.tcl*	•
	<u>F</u> ile <u>T</u> emplates			
	Component Type Files Parar	neters Signals Interfaces		
	About Files			
	Synthesis Files			
	These files describe this componen	t's implementation, and will be crea	ated when a Quartus II synthesis	model is generated.
	The parameters and signals found	in the top-level module will be use	d for this component's paramete	rs and signals.
	Output Path	Source File	Type	Attributes
ᆔᅶᅜ	mips_avalon_interface.v	mips_avalon_interface.v	Verilog HDL	Top-level File
ノホダン	MipsCore.v	MipsCore.v	Verilog HDL	no attributes
畑す重で				
ァイルの				
	+ - 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 Name	Interface	Signal Type))/(idth	Direction
clock	clock reset	reset n	clock	clock sink	clk	1	input
reset_n	new Avalon Memory Mapped Tristate Slave	reset_n	reset n	reset sink	reset n	1	input
avs_s1_address	new AXI Master	address	avs s1 address	icscc sink	address	1	input
avs_s1_read	new AXI Slave	read	avs s1 read	s1	read	1	input
dvs_s1_write	new AXI4 Master	chincoloct	avs s1 write	s1	write	1	input
avs_st_thipselett	new AXI4 Slave	readdata	avs s1 chinselect	s1	chinselect	1	input
avs s1 writedata	new Clock Output	writedata	avs_s1_cmpselect	s1	readdata	32	output
avm_imem_address	new Clock Input	address	avs s1 writedata	s1	writedata	32	input
avm_imem_read	new Conduit	read	avm imem address	imem	address	32	output
Inew USSI Ronded Clock Output - avi			avm_imem_read	imem	read	1	output
clock_sinkを選択している所			avm imem write	imem	write	1	output
			avm imem waitrequest	imem	waitrequest	1	innut
			avm_imem_readdata	imem	readdata	32	input
			avm_imem_writedata	imem	writedata	32	output
			avm_imem_byteenable	imem	hyteenable	4	output
			avm_mem_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 hyteenable	dmem	hyteenable	4	output
			coe led stall	conduit end 0	export	1	output
			coe led state	conduit end 0	export	3	output
			coc_ica_state			-	leadear.





- ・ 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)								
Name: s1	<u> </u>							
Type: Avalon Memory Mapped Slave	-							
Associated Clock_sink	-							
Associated Reset reset_sink	-							
Assignments: Eut								

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





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		
₫	<u>-</u>		⊳-	clk_in	Clock Input	clk	ex
l E	2		⊳-	clk_in_reset	Reset Input	reset	
				clk	Clock Output	Double-click to export	clk
	<u> </u>			clk_reset	Reset Output	Double-click to export	
4	≤	1		🖻 sdram	SDRAM Controller		
			\bullet \rightarrow	clk	Clock Input	Double-click to export	cH
			$\phi \longrightarrow$	reset	Reset Input	Double-click to export	[Cl
ΠĒ	× 1		$ \circ \bullet \bullet \to$	s1	Avalon Memory Mapped Slave	Double-click to export	[Cl
	2		⊳ ⇔	wire	Conduit	sdram_wire	
	M	~		onchip_memory	On-Chip Memory (RAM or ROM)		
			$\bullet \longrightarrow$	clk1	Clock Input	Double-click to export	cH
			$ \bullet \bullet \bullet \to$	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	[CI
		~		🗆 mips_avalon_inter	mips_avalon_interface		
			$ \diamond \phi \phi \rightarrow \phi \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
			$\blacklozenge \qquad \qquad$	clock_sink	Clock Input	Double-click to export	cH
			\diamond \rightarrow	reset_sink	Reset Input	Double-click to export	[cl
		~		🗆 contest_uart_avalo	contest_uart_avalon_interface		
1			$ \diamond \bullet \bullet \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	cII
			\rightarrow \rightarrow	reset sink	Reset Input	Double-rlick to export	[c]



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		
ALTPLL		
Ended Ended Stanmary Cik c0 cik c1 cik c2 cik c3 cik c4	チェックを外す 更面4	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 inclk0 frequency 50.000 MHz Operation Mode: Normal City Cot areset Operation Mode: Normal City Cot Cot <td>kmark indicates a file that is automatically potional file. Click Finish to generate the selected files. equent MegaWizard Plug-In Manager sessions. cted files in the following directory: litera/MieruSys11/ </td>	kmark indicates a file that is automatically potional file. Click Finish to generate the selected files. equent MegaWizard Plug-In Manager sessions. cted files in the following directory: litera/MieruSys11/
		Cancel < Back Next > Finish



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 P Operating Settings and Conditio	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
Voltage Temperature	<u>F</u> ile name:	Add
Early Timing Estimate	File Name Type Library Design Entry/Synthesis Tool HDL Versior	Add All
Incremental Compilation Physical Synthesis Optimizal EDA Tool Settings	mips Qsys System File <none> MieruS Verilog HDL File <none> ⊡ pll.qip IP Variation File (.gip) <none></none></none></none>	<u>R</u> emove
Design Entry/Synthesis		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	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

Named: *	Named: *								
Node Name	Loca	Direction ate	Location						6 7 8
🕀 🏷 DRAM1	Edit		•	ic)	<u>U</u> ndo	Ctrl	I+Z	∘ <u>A</u> QooQ	000
⊕ 膋 DRAM3 ⊕ ∯ KEY[00]	🛃 <u>E</u> arly	y Pin Planning		C	<u>R</u> edo	Ctrl	I+Shift+Z		00
EDG[70]	✓ <u>A</u> II Pi	ins List		¥	Cu <u>t</u>	Ctrl	I+X		
	✓ <u>G</u> rou	ups List		Þ	<u>С</u> ору	Ctrl	I+C		
	Crea	ate Group		ß	Paste	Ctrl	I+V		700
	 Add	to Gr <u>o</u> up		×	<u>D</u> elete	Del			
•	Add	<u>M</u> embers			Select <u>A</u> ll	Ctrl	I+A		
Groups Repo	Cust	tomize Columns.		Lin	Assign Do <u>w</u> n	ı			
Tasks	Cust	tomize <u>F</u> ilter		tin	Assig <u>n</u> Up			00000	000
🗄 🦾 Early Pi	Show	w <u>A</u> ssignable Pins	5	<u>منہ</u>	Assign <u>L</u> eft				∕ ● ●
Early	Show	w <u>D</u> evice Pins		i n	Assign Right		Sandida an Sandigad		
🛛 🕨 Run	Expo	ort		ie	🖳 Assian One				000
Expe	Back	<- <u>A</u> nnotate		Π	·····			~ 00000/	00,
🖃 📴 Highligh	D Rad	Pad View		11					900
- = I/O E	Paul								000
···· 📰 VRE	Boar	Board Trace Model						1 2 3 4 5	6 7 8
Edge	🥦 Pin <u>N</u>	gration window		ΙL					20 and 10 bit
× Named: *GPI	Pin L	egend Window							
료 및 Node Na	<u>R</u> eso	ources Window		h	I/O Bank		VREF Group	I/O Standard	Res
	Live	I/O Chec <u>k</u> Status	Window				•	2.5 V (default)	
	搦 <u>P</u> in F	inder			-			2.5 V (default)	
GPIO[0]	Node	e Properties		\vdash	4		B4_N0 B4_N2	3.3-V LVTTL 3.3-V LVTTL	
GPIO[1]	🚳 Find	Swappable Pins			4		B4_N2 B4_N0	3.3-V LVTTL	
GPIO[3]	2	OIKIOWI	TIN_117		4	I	B4_N0	3.3-V LVCMOS	
③ GPIO[4]		Unknown	PIN_AC21		4	1	B4_N0	3.3-V LVTTL	





ĸ	Named: *GPIO* 💌 🐇	Edit: 💢 🗸								
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)	
		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)		
		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 2014-08-17

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、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
 - <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(こ表示されます.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

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'hlb:	begin	IdOPN='DIVU	;	IdDST=IdRD;		end
197	endcase						
198	6'h01: case	e (IdR	Г)				
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	;	<pre>IdDST=IdRT;</pre>		end
209	6′h09:	begin	IdOPN='ADDIU	;	<pre>IdDST=IdRT;</pre>		end
210	6′h0a:	begin	IdOPN='SLTI	;	<pre>IdDST=IdRT;</pre>		end
211	6′h0b:	begin	IdOPN='SLTIU	;	<pre>IdDST=IdRT;</pre>		end
212	6′h0c:	begin	IdOPN='ANDI	;	<pre>IdDST=IdRT;</pre>		end
213	6'h0d:	begin	IdOPN='ORI	;	<pre>IdDST=IdRT;</pre>		end
214	6'h0e:	begin	IdOPN='XORI	;	<pre>IdDST=IdRT;</pre>		end
215	6′h0f:	begin	IdOPN='LUI	_;	<pre>IdDST=IdRT;</pre>		end
216	6′h20:	begin	IdOPN='LB	;	<pre>IdDST=IdRT;</pre>	IdATTR='LD_1B;	end
217	6′h21:	begin	IdOPN='LH	;	IdDST=IdRT;	IdATTR='LD 2B;	end
218	6′h23:	begin	IdOPN='LW	_;	<pre>IdDST=IdRT;</pre>	IdATTR='LD_4B;	end
219	6′h24:	begin	IdOPN='LBU	;	<pre>IdDST=IdRT;</pre>	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.

```
begin IdOPN='LHU
                                                 ; IdDST=IdRT; IdATTR='LD 2B;
220
                 6'h25:
                                                                                        end
                 6′h28:
                            begin IdOPN='SB
                                                 ; IdDST=0;
221
                                                              IdATTR='ST 1B;
                                                                                        end
                                                 ; IdDST=0; IdATTR='ST_2B;
222
                 6′h29:
                          begin IdOPN='SH
                                                                                        end
                            begin IdOPN='SW
223
                 6'h2b:
                                                  ; IdDST=0;
                                                              IdATTR='ST 4B;
                                                                                        end
224
             endcase
225
         end
226
227
        wire ['ADDR] IdBPC = IfId npc + ({{16{IfId ir[15]}}}, IfId ir[15:0]} << 2);
228
         always @(*) begin ///// branch & jump resolution unit
229
             \{IdTPC, IdC, IdB\} = 0;
             case (IdOP)
230
231
                 6'h04: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS == IdRRT);
                                                                                                // BEO
                                                                                    end
                 6'h05: begin IdB=1; IdTPC = IdBPC; IdC = (IdRRS != IdRRT);
232
                                                                                     end
                                                                                                // BNE
233
                 6'h06: begin IdB=1; IdTPC = IdBPC; IdC = ( IdRRs[31]||(IdRRs==0)); end
                                                                                                // BLEZ
234
                 6'h07: begin IdB=1; IdTPC = IdBPC; IdC = (~IdRRS[31]&&(IdRRS!=0)); end
                                                                                               // BGTZ
235
                 6'h01: begin IdB=1; IdTPC = IdBPC; IdC = (IdRT) ? ~IdRRS[31] : IdRRS[31]; end // BGEZ,BLTZ
                 6'h02: begin IdB=1; IdTPC = IfId ir['ADDR]<<2; IdC = 1; end
                                                                                               // J
236
                                                                                               // JAL
237
                 6'h03: begin IdB=1; IdTPC = IfId ir['ADDR]<<2; IdC = 1; end
                              (IdFCT==6'h08) begin IdB=1; IdTPC = IdRRS; IdC = 1; end
238
                 6'h00: if
                                                                                               // JR
                        else if (IdFCT==6'h09) begin IdB=1; IdTPC = IdRRS; IdC = 1; end
239
                                                                                                // JALR
240
             endcase
241
         end
242
         always @(posedge CLK or negedge RST X) begin ///// update pipeline registers
243
             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;</pre>
                                                      // data from general-purpose register file
247
                 IdEx rrt <= (bstall) ? 0 : IdRRT;
                                                        // data from general-purpose register file
248
249
                 IdEx dst <= (bstall) ? 0 : IdDST;</pre>
250
                 IdEx ir <= (bstall) ? 0 : IfId ir;</pre>
                 IdEx opn <= (bstall) ? 0 : IdOPN;
251
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 'NOR 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; 340 ExMa_std <= (IdEx_opn=='SB_ _) ? {4{RRT_U[7:0]}} :
_) ? {2{RRT_U[15:0]}} : RRT_U; 341 (IdEx_opn=='SH_ ExMa_mwe <= ExWE: 342 343 ExMa oe <= (IdEx_attr & 'LDST_ANY) ? 1 : 0; (IdEx_opn=='LH) ? 1 : (IdEx opn=='LHU) ? 2 : // Load selector 344 ExMa lds <= 345) ? 4 : (IdEx opn=='LBU (IdEx opn=='LB) ? 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]
- 作成した回路データファイルは SEE Project Navigator (Rédd) C. Wilsons Workshow Muberitory System_Atys_146/PgayArmal fpga/mierusys.bit View a 10 Inclementation Hierarchy 🔁 main です xc6slx45-3csg324 Automatic "includes 1
 - **README.txt** も参考にしてください.





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 2014-08-17

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 FPGAs					
	Pin Compatible FPGAs include all devices with the same package and speed grade as the target device. Different FPGA devices with the same package do not have the same bonded pins. By selecting Pin Compatible FPGAs, MIG will only select pins that are common between the target device and all selected devices. Use the default UCF in the par folder for the target part. If you change the target part, use the appropriate UCF in the compatible ucf folder. If you do not choose a Pin Compatible FPGA now and need to use a different FPGA later. the generated UCF may not work for the new device and a board spin may be required. A device is considered compatible only if the package and speed grade matches to the target part. MIG only ensures that MIG generated pin out is compatible among the selected compatible FPGA devices. Unselected devices will not be considered for compatibility during the pin allocation process.					
	Blank list indicates that there are no compatib Target FPGA xc6slx45-csg324 -3	le parts exist for the selected target part and	this page can be skipped.			
	Pin Compatible FPGAs	Available MCBs				
	▲ spartan6					
	xc6slx9-csg324	MEMC1, MEMC3				
	xc6slx16-csg324	MEMC1, MEMC3				
Memory	xc6slx25-csg324	MEMC1, MEMC3				
Interface						
Generator						
User Guide MCB User Guide Version Info						



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						
REFERENCE	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 Bank 3 (marked with an asterisk below) has fewer multi-purpose 10 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 AVI interface The AVI interface for All MCBs Extended MCB performance range Extended MCB performance mode requires a different Vocint specification to achieve higher maximum frequencies for DDR2. Consult the Spartar-5 datasheet (DS162) table 2 and 24 for more information.					
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 "Oreate Custom Part" button if the part you want is not listed here.	EDE1116AXXX-8E Create Custom Part
AXI Parameter Arbitration FPGA Options		
Summary Memory Model PCB Information Design Notes		
EXILINX ®	Memory Details: 1Gb, ×16, row:13, col:10, bank:3, data bits per strobe:8, with data mask, single rank	
User Guide MCB User Guide Versio	in Info	ack Next> Cancel



Xilinx Memory Interface Generator (6)

・ Fullstrength, 50ohms, Enable, Disable を選択

嶺 Xilinx Memory Interface Generator	•						
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.						
	Output Drive Strength Selecting reduced strength will reduce all outputs to approximately 60 percent of the drive strength. Fullstrength						
Controller Options	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.						
Multi-port Configuration AXI Parameter Arbitration	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.						
FPGA Options Summary Memory Model	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 can use selfrefresh L/F to enter self refresh state.						
PCB Information Design Notes							
E XILINX.							
User Guide MCB User Guide Vers	sion Info	Cancel					


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

4	Xilinx Memory Interface Generator					3
		Port Configuration f	or C3 - DDR2 SDRA	м		
Select one of five configurations from the configuration menu and the ports from the table. As you select the port configuration, the figure and table will get updated. You can select the number of ports in a configuration, and data port settings from the table. Configuration Selection Two 32-bit bi-directional and four 32-bit unidirectional ports				and the ports from the table. As you select the port configuration, the below f ports in a configuration, and data port settings from the table.		
	Controller Options 🛛 🖌	Port Selection	Interface	Direction		
	C3	Port0	NATIVE -	Bi-directional 👻		
	C3	Port1	NATIVE -	none 👻		
	Multi-port Configuration C3	Port2	NATIVE	none 👻	•	
	AXI Parameter	Port3	NATIVE -	none 👻	×	
	C3 Arbitration	Port4	NATIVE -	none 👻	*	
	C3	Port5	NATIVE -	none 👻	•	
	FPGA Options C3 Summary	Mamani Addina M	ing Salastian			
Summary Memory Address Mapping Selection Memory Model Liter Address						
PCB Information Design Notes						
		ROW	BANK CO	LUMN		
		BANK ROW COLUMN				
	EXILINX ®				th.	
User Guide MCB User Guide Version Info						



Xilinx Memory Interface Generator (8)

・ Next で次に進む

🖏 Xilinx Memory Interface Generator				
REFERENCE	Arbitration for C3 - DDR2 SDRAM			
	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 G3 Memory Options G3 Multi-port Configuration G3 AXI Parameter G3 AXI Parameter G3 Arbitration G3 FPGA Options G3 Summary Memory Model PGB Information Design Notes	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 one time slot. Select Arbitration Algorithm Round Robin Timeslot 1 0 Timeslot 2 0 Timeslot 3 0 Timeslot 4 0 Timeslot 5 0 Timeslot 6 0 Timeslot 7 0 Timeslot 9 0 Timeslot 10 0			
E XILINX.	Timeslot 11 0			
User Guide MCB User Guide Version Info 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

	FPGA Options for C3 - DDR2 SDRAM
DESIGN 🔛	
Controller Options Memory Options Multi-port Configuration AXI Parameter Arbitration FPGA Options Summary Memory Model PCB Information Design Notes	Class II is recommended for all Sol Is signals in memory interfaces. However, better signal interrity may sometimes be achieved with Class I for Address & Control. If IBS simulations indicate that Class II selevice in the class I below. This can be changed after generation by modifying the UCF. This option changes the drive strength for Data, Address & Control. Classs for Address and Class II ▼ Class for Data Class II ▼ Memory Interface Pin Termination: Provides calibrated on-die input termination resistors. Calibration requires two extra pins to be added to the interface. F2Q and ZD, An external resistor with a value 2x trace impedance needs to be connected from R2Q pin to ground, and the ZD pins need to be left unconnected. These additional pins and their locations will be listed in the generated UCP constraints file. Un-calibrated Input Termination: Provides un-calibrated (approximated) on-die input termination resistors to Vcco and Ground. DQ/DQS 25 Ohms ▼ External Input Termination: Provides discrete termination resistors for the controller on the PCB. Select R2Q pin location: This pin is required for all MCB designs. If Calibrated Input Termination is used, this pin must have a resistor of value 2R to ground, where R is the desired input termination value. Otherwise, it may be left as a no- connect (NO). The selected design's timing has not been verified with non-default RZQ locations Select ZIO pin location: ZIO must be placed on a bonded pin in the package, but there should be no board trace connected to this pin (no-connect). The selected design's timing has not been verified with non-default RZQ locations. Debug Signals for Memory Controller: This allows the debug signals to be monitored on the ChipScope tool. Selecting this option will port map the debug signals to the ChipScope modules in the design to monitored on the ChipScope tool. Selecting this option will port map the debug signals to the ChipScope modules in the design to monitored on the ChipScope tool. Selecting this op
	System Clock: Choose the desired input clock configuration.



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	_ <i>8</i> ×		
- P Z 🖌 🖓 🕹 🖇 🖓 😕 🖉 🔍 🔕 🖉 🖉 💽 🔁 🕒 😐 🖉 💙 🖓			
Design ↔ 0 #X Inclamantation → 0 #X = 58 //	^		
Hierardy 60 // and 60 // a			
a le main a la contractory: a contractory:			
The distribution of the following must be inserted into your Verilog file for this			
E Automatic includes E 9 65 // core to be instantiated. Change the instance name and port connections			
Image: Big			
- V dockgen0: - dockgen (dock.v) 68 // Begin Cut here for INSTANTIATION Template// INST_TAG			
→ O cockgence - Lockgenc (clocky) % 70 dram ‡ (
□ 1 mem0 - dram_con (dram_mem. 71 .CS_P0 Mass, SIZE (3), CS_P0 DaTA_PORT_SIZE (32),	_		
U gram - gram (gram.xco) 373 . C3 P1 MASK SIZE (4),			
Serc - serialc (system.v) 75 .DEBUG_EN (0),			
<pre></pre>	E		
► ₹ No Processes Running 78 .C3_SIMULATION ("FALSE"), .C3_SET ACT LOW(0).			
Trocesses: u_dram - dram B0 .C5_INFUT_CLK_TYPE("SINGLE_ENDED"),			
Image: CORE Generator 81 .CS_MDB ("ROUBL ("ROUBL ("ROUBLE")"), 82 .CS_MDB ("ROUBL ("ROUBL ("ROUBLE")),			
Anage Cores Bas - C3 MEM ADDR WIDTH (13), Reparate Core Section 2010 - Section			
Update Core to Latest Version 85)			
Idew HDL Functional Model B6 u_oram (View HDL Functional Model B7			
Wew HDL instantiation template 88 .c3 ays_clk (c3 ays_clk), 89 .c3 ays_rti (c3 ays_clk),			
90 90 90 90 90 90 90 90 90 90 90 90 90 9			
91 .mobs_dram_d (mcbs_dram_dq), 92 .mobs_dram_a (mcbs_dram_a),			
93 .mob3_dram_ba (mob3_dram_ba), 04 .mob3_dram_rag n (mob3_dram_rag n).			
95 mcb3 dram cas n (mcb3 dram cas n).	-		
🍃 Start 📭 Design 🐚 Files 🐚 Libraries 📡 Design Summary (out of date) 🖂 📄 dram.veo 💌 📄 MieruSysx 💌 📄 dram.mem.v 💌	·		
Console	+□ 8 ×		
	*		
The Concole of France IV Warnings 100 France 100 F			
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 2014-08-17

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>

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 2014-08-17

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 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
 - <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 ソースコードは SDK-1.1.2を使用して下さい

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 2014-08-17

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。要素数 n の上限は 4M (データサイズ 16MB) です。

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。行列サイズ n の上限 は 4096です。

```
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 CRC32 を計算して表示します.(2014/8/27変更)



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. 配列サイズ n の上限は 4096 です. イタレーション回数 iter の上限は 100000 です.

```
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. n は2のべき乗, m は n の倍数です. ノード数 n の上限は 2048 です. エッジ数 m の上限 は 8192 です.

・ 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.

<pre>struct data_t {</pre>	
unsigned int b	uf[SIZE-4]; // 256KB -16(4×4)Byte buffer
int n;	// the number of nodes
int m;	<pre>// the number of edges</pre>
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 -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 -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 -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 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)

Document P51

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 2014-07-28

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、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
 - <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





- ・ 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ー覧
 - 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,	addressmap.h addressmap.c	アドレスマップ
xilinx, altera)	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




- 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





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。



- ・ 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)
```





- ・ 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



411_anim



整理 ▼ 📕 ブレビュー ▼ 印刷	副 書き込む 新しいフォルダー			• ==	(
🛀 nas 🔷	名前	更新日時	種類	サイズ	
퉬 workspace	🌉 412_star1.y.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
🎍 ohkawa (fs1.ced.is.utsunom	413_rectangle0.bmp	2014/06/25 12:53	BMP ファイル	118 KB	
	413_rectangle0.y.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
デフクトップ	🧮 413_rectangle1.bmp	2014/06/25 12:53	BMP ファイル	118 KB	
	413_rectangle1.y.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
	RESULT.410_tux.64KB.decoded.bmp	2014/06/27 2:36	BMP ファイル	24 KB	
Subversion	RESULT.411_anim.64KB.decoded.bmp	2014/06/27 2:36	BMP ファイル	49 KB	
▶ ドキュメント	RESULT.412_star.64KB.decoded.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
≧ ピクチャ	RESULT.413_rectangle.64KB.decode	2014/06/27 2:37	BMP ファイル	118 KB	
🛃 ビデオ	410_tux0.encoded	2014/06/27 2:36	ENCODED ファ	5 KB	
♪ ミュージック	410_tux1.encoded	2014/06/27 2:36	ENCODED ファ	5 KB	
🙀 ohkawa	411_anim0.encoded	2014/06/27 2:36	ENCODED ファ	5 KB	
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	411_anim1.encoded	2014/06/27 2:36	ENCODED ファ	5 KB	
	412_star0.encoded	2014/06/27 2:36	ENCODED ファ	6 KB	
• ネットワーク	412_star1.encoded	2014/06/27 2:36	ENCODED ファ	6 KB	
I DEGIN-PC	413_rectangle0.encoded	2014/06/27 2:36	ENCODED ファ	6 KB	
IN EP49AF3A	413_rectangle1.encoded	2014/06/27 2:36	ENCODED ファ	6 KB	
IN LYNX	UDP_Client_v06.jar	2014/06/26 19:55	Executable Jar	8 KB	
NAS	₩ 410_tux.sh	2014/06/26 23:09	SHファイル	1 KB	
N OHKAWA-PC	e 411_anim.sh	2014/06/26 23:10	SHファイル	1 KB	
VELA	9 412_star.sh	2014/06/26 23:10	SHファイル	1 KB	
WINS1	413_rectangle.sh	2014/06/26 23:10	SHファイル	1 KB	
- WINCO	e all.sn	2014/06/26 23:09	5日 ファイル	1 KB	
I≟ wilw52	макепіе	2014/06/26 23:02	ファイル	1 KB	



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。
 す。注意してお使いください。
 - 一元画像(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)







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.



Document P61

The 2nd ARC/CRSY/RECONF High-Performance Computer System Design Contest User Manual of Optical Flow System Reference Design (Xillinx ATLYS)

コンテスト実行委員会コアチーム Version 2014-07-28

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、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
 - <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>





- ・ 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.
 - <u>http://japan.xilinx.com/support/university.html</u>
- ・ 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。
 - <u>http://www.digilentinc.com/</u>
- 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





メモリア	ドレス	
開始アドレス	終了アドレス	
0×0000_0000	0x0000_7FFF	Block RAM (on-chip)
0×4060_0000	0x4060_FFFF	RS232_Uart (115.2kbps USB-UART)
0x77A0_0000	0x77A0_FFFF	Axi_ploader_uart (1Mbps UART via PMOD)
0×A800_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 \\ A + lys-oflow
 - http://aquila.is.utsunomiya-u.ac.jp/contest/toolkit.html





- 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-bridgingsolutions/usb-uarts/xr21v1410
 - Software Drivers Windows 2000, XP, Vista, 7, and 8 Drivers Version 2.0.0.0 August 2013

(R21V1410		CPUInterface	US	
Ch Sell Second USB LIADT		CH	1	
-cirrui-spied 050 0AK1		Data Rate@5/3.3/2.5V	na	
eatures	Applications	Max Data Rate @ 5/3.3/2.5/1.8V (Mbps)	na/	
USB 2.0 Compliant Interface	Portable Appliances	Tx/RxFIFO(Bytes)	120	
 Supports 12 Mbps USB full-speed data rate 	 External Converters (Dongles) 	AutoRTS/CTS	Yes	
 Supports USB suspend, resume and remote wakeup 	operations • Battery-Operated Devices	AutoRS-485	Yes	
Enhanced UART Features	Cellular Data Devices	5VTolloovts	Yes	
Data rates up to 12 Mbps Eractional Based Pate Conservoir	 Factory Automation and Process Controls Industrial Applications 	S VI Onipus	100	
 128 byte TX FIFO 		Sup V	2.9	
 384 byte RX FIFO 	Product Change Notification, Revision B to D	Pkgs	QF	
 7, 8 or 9 data bits, 1 or 2 stop bits Automatic Hardware (RTS/CTS or DTR/DSR)Flow Control Automatic Schware (Xen/Xen) Flow Control 	PCN_12-0305-01.pdf	► Documents	► Documents	
Multidrop mode w/ Auto Half-Duplex Transceiver Con Multidrop mode w/ Auto TX Enable	trol	Block Diagram		
 Half-Duplex mode 		Datasheets		
 Selectable GPIO or Modern I/O Internal 48 Mills clock 				
Internal 40 Minz Clock Single 2 97.3 63V power supply		Datasheet		
5V tolerant inputs		Version 1.3.1		
Virtual COM Port Drivers		July 2013		
 Windows 2000, XP, Vista, 7, and 8 		279 57 KB		
 Windows CE 4.2, 5.0, 6.0, and 7.0 		444.07 110		
- Linux		Software Drivers		
* Mac		Software Univers		
lescription		Windows 2000, XP,	Vista, 7, ar	
b. VD2024440 is an exhaust of the second distance of the	Dearline and Teacardine (1407) with a 100 latertain. The 100 l	Drivers		
is fully compliant to Full Sneed USB 2.0 specification that sun	norts 12 Mhos USB data transfer rate. The USB Interface also supports U	Version 2.0.0.0		
uspend, resume and remote wakeup operations.	serve in maps were said annote raid. The were interface into supports of	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.

▶ ■ ポータブル デバイス
 ▲ 常 ポート (COM と LPT)
 ▲ 常 XR21V1410 USB UART (COM5)
 ▶ ♪ ペ マウスとそのほかのポインティング デバイス

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.

Tera Term: シリアルポー	ト設定	X
ボート(<u>P</u>):	COM16 -	ок
ボー•レート(<u>B</u>):	115200 -	
データ(<u>D</u>):	8 bit 🔹	キャンセル
バリティ(<u>A</u>):	none 🔹	
ストップ(<u>s</u>):	1 bit 🔹	ヘルプ(円)
フロー制御(<u>F</u>):	none 🔹	
送信遅延 0 ミリ利)/字(<u>c</u>) 0 ≷!	リ秒/行(∟)



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





Workspace Launcher	×
Select a workspace	
Xilinx SDK stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: K:¥workspace-edk147¥Atlys-oflow¥SDK¥SDK_Export	✓ <u>B</u> rowse
Use this as the default and do not ask again	OK Cancel



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	Hello World Memory Head Perpheral Tests SREC Bootlooder Xilikernel POSIX Threads Demo
Application Project Create a managed make application project.	
Project name: 400_oflow	
Location: K:¥workspace-edk147¥Atlys-oflow¥SDK¥SDK_Export¥400_oflow Browse Choose file system: default v	? < gack Best > Emish Cancel
Target Hardware	
Processor microblaze_0	
Target Software OS Platform standalone Language C C ++ Board Support Package C Create New 400_oflow_bsp Use existing v	
? < Back Next > Finish Cancel	

New Proje Templates

Available Ter

Create one of the available templates to



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

G

A blank C project

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



C/C++ - 400_oflow_bsp/system.mss - Xilinx SDK

File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help



- 0 X

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0

r	am froal (rev.	Configure	e JTAG Settings					
		Specify t These se	the JTAG cable to us attings affect how XI	e for comm ID connects	unication and JTAG Device to the FPGA.	Chain configuration of the tai	rget board.	₽ +4
Γ)WindowでProgramをクリック	JTAG Cab	le					
-			Type. Digilent USB	Cable				
s -	Xilinx SDK	Hostn	ame: Xilinx Hardw	are Server				
te	Search Run Project Xilinx Tools Window Help		Port: Digilent USB	Cable				
	📸 🗣 📸 🗣 🚱 🛐 Generate linker script	Freque	ency: Xilinx Paralle	III Cable				
-	Board Support Package Settings	Other Opt	3rd Party Ca	ole, Xilinx Pl	lug-in			
	system.xml	JTAG Dev	vice Chain atically Discover De	icos on 1TA	C Chain			
~	401 off	 Manual 	l Configuration of JT/	G Chain	ig chain			
	4012ν_C ρ BC							
	Modify this BSP's Se	FPGA?	Device Name		ID Code	IR Length		
	XMD Console							
	Target Information							
	This Prove support Pite Configure JTAG Settings							
	Hardware Specification							
	Target Process							
	Operating System	0						Cancol
Ξ	Board Support Package OS.							Concer
	Name: standalone							
	Version: 3.11.a Description: Standalone is a simple low-level software layer. It provides access to							
	as caches, interrupts and exceptions as well as the basic features of a				Program FPGA	W. Barris and and the Rest		a de astro o re
	standard input and output, profiling, abort and exit.				Program FPGA			
	Documentation: <u>standalone v3 11 a</u>				Specify the bitstneam and	the ELF files that reside in BRAM memo	ny .	
					Hardware Configuration Hardware Specification: K:¥	workspace-edk147¥Atlys-oflow¥SDK¥S	DK_Export#Atlys-oflow_hw_platform#system.x	ml
	Overview Source				Bitstream: system.bit			Search
	📳 Problems 🕢 Tasks 📮 Console 🛛 🔲 Properties 🖉 Terminal 🛛 🕹 🔂 🔚 👪	🖹 🖌	📮 🕶 📑 💌 🗖		BMM File: system_bd.bm	m		Search] [B
	CDT Build Console [401_oflow]				Processor ELF File to	Initialize in Block RAM		
	elfcheck passed.				microblaze_0 bootloop			
	Thisned building, 401_0110W.EIT.EITCHECK				0		Program	m Cascal
	02:08:56 Build Finished (took 4s.710ms)				U U		Progra	Carder
	(,							
-	[

Onfigure JTAG Settings



.∎≎

C/C++ - 401_oflow_bsp/system.m File Edit Source Refactor Naviga 📑 🗝 🔚 🖻 🗁 😽 🕶 💼 🔲 🕥 🗄 🖢 🕶 😓 🗸 🖛 🦕 🖛 Project Explorer 🔀 Binaries Includes Debug Image: 1 400_oflow_bsp 📂 401_oflow 🐰 Binaries Includes Debug 🔺 👝 src

> In addressmap.c b addressmap.h codec.c b h codec.h communication.c b in communication.h

fileio.c h fileio.h image.c In image.h

b 🖻 main.c oflow.c In oflow.h b c platform.c b h platform.h 🙀 lscript.ld README.txt 😂 401_oflow 0-0

rowse,

×

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

@ C/C++ - 401_oflow	_bsp/system.mss - Xilinx SDK	AL 17	
File Edit Source F	Refactor Navigate Search Run	Project Xilin:	x Tools Window Help
	छ र % र ि, (a + a + a + a) र % (- र -) र	•• •••	\$\$ ▼ 0 ▼ 0 <u>↓</u> ▼ ≥ # 2 6 2 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
🎦 Project Explorer 🖇	3 📃 🗖 🕞 system.xm	nl 🛛 👔 syste	em.mss 👔 system.mss 🛛 🖓 🖓 🐨 🖓 👘 Ma 🖓 🖓
	😑 🔄 🔻 🎽 401 oflow	bsp Board	Support Package
⊳ 📂 400_oflow			
▷ 100_oflow_bsp	Modify this	BSP's Settings	
▲ ₩ Binarie	New	•	=
Þ 🕸 40:	Go Into		
⊳ 👘 Includ	Open in New Window		is compiled to run on the following target.
Debug	Сору	Ctrl+C	¥workspace-edk147¥Atlys-oflow¥SDK¥SDK_Export¥Atlys-ofl
≥ sic	Paste	Ctrl+V	
🛛 🕞 ado 🎽	Delete	Delete	
⊳ 💽 coc	Move	,	
⊳ <u>h</u> coc	Rename	F2	e
⊳ ic cor	Import		o is a simple, low level software layer. It provides access to
🛛 🕞 file 🛃	Export		, interrupts and exceptions as well as the basic features of a
⊳ 庙 file	Build Project		input and output, profiling, abort and exit.
⊳ <u>ie</u> ima	Clean Project		<u>e v3 11 a</u>
⊳ ia ma 🌯	Refresh	F5	
⊳ 🖻 ofic	Close Project		
⊳ 🕞 ofic	Close Unrelated Projects		pnsole 🔅 🔛 Properties 🧬 Terminal 🛛 🕹 😯 🌄 🔜 🖓 🐨 🗳 🗸 🔽 🗸
⊳ <u>ic</u> pla	Build Configurations	+	
N Isci	Make largets	*	low.elf.elfcheck
RE/	Changin Demote Contenes dans		
▲ 101_oflov	Convert To		took 4s.710ms)
	Run As	+	1 Launch on Hardware (GDB)
1 🗠 😂 4	Debug As	+	C 2 Local C/C++ Application
	Profile As	•	2 Remote ARM Linux Application
	Team	*	Run Configurations
Phillippine and the second	Restore from Local History		
*	Run C/C++ Code Analysis		
N CON	Generate Linker Script		
A State	Change Referenced BSP		
	C/C++ Build Settings		
	Properties	Alt+Enter	



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)





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

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。
- 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。

😡 Generate a linker script		9			×
Generate linker script					6
Control your application's memory	y map.				2
Project: 401_oflow Output Script: Jace-edk147¥Atlys-oflow¥SDK¥SD	DK_Export¥401_c	oflow¥src¥lscript	.Id Browse	Basic Advanced Place Code Sections in: m Place Data Sections in: m	0_ddr2_S0_AXI_BASEADDR cb_ddr2_S0_AXI_BASEADDR
Modify project build settings as fol	llows:			Place Heap and Stack in mi	icroblaze_0_i_bram_ctrl_microblaze_0_d_bram_ctrl 👻
Set generated script on all project	t build configurat	ions	•	Heap Size: 1 H	KB
<u></u>				Stack Size: 16	КВ
Memory	Base Address	Size			
microblaze_0_i_bram_ctrl_mi	0x00000000	32 KB			
mcb_ddr2_S0_AXI_BASEADDR	0xA8000000	128			
 Fixed Section Assignments 					
?					Generate Cancel



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

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!

COM16:115200baud - Tera Term VT							
ファイル(<u>E</u>) 編集(<u>E</u>) 設定(<u>S</u>) コントロール(<u>O</u>) ウィンドウ(<u>W</u>) ヘルプ(<u>H</u>)				-	_		
Waiting recv 32KB at ac100000	^	COLOR DE LA COL	- 10 F		-		
firstFrame : finished reading/decoding img0 width=128, height=128		🕒 🗸 🖉 🕹 🖉 🕹 🖓 🕹	home > ohkawa > contest > 400_offc	w 🕨 client	▼ 4 clientの検	索	Q
secondFrame: finished_reading/decoding_img1_width=128, height=128		整理 ▼ ■ プレビュー ▼ 印刷	書き込む 新しいフォルダー				
The number of flows: 58			A ==	西 公口:由	10047		
The Tength is bood bytes.		i workspace	合則 ■ 412 star1 v hmn	更新口时 2014/06/27 2:36	催焼 BMP ファイル	118 KB	
send 04ND From aC200000		hohkawa (fe1 cod is utsunom	413 rectangle0.bmp	2014/06/25 12:53	BMP ファイル	118 KB	
Einiched 400 of Low (batchID:411) ===		Olkawa (Ist.ceu.is.ucsulom)	413_rectangle0.y.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
Infisited 400_0110W (batchib:411)			413_rectangle1.bmp	2014/06/25 12:53	BMP ファイル	118 KB	
Starting 400 of Low (batchID:412) ===		■ テスクトップ	413_rectangle1.y.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
Waiting recy 32KB at ac000000			RESULT.410_tux.64KB.decoded.bmp	2014/06/27 2:36	BMP ファイル	24 KB	
Waiting recy 32KB at ac100000		Subversion	RESULT.411_anim.64KB.decoded.bmp	2014/06/27 2:36	BMP ファイル	49 KB	
firstFrame : finished reading/decoding img0 width=200, height=200			RESULT.412_star.64KB.decoded.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
secondFrame: finished reading/decoding img1 width=200, height=200		■ ピクチャ	RESULI.413_rectangle.64KB.decode	2014/06/27 2:37	BMP ファイル	118 KB	
The number of flows: 77		■ ビデオ	410_tuxt.encoded	2014/06/27 2:30	ENCODED 77	5 KB	
The length is 7454 bytes.		🍙 ミュージック	411 anim0.encoded	2014/00/27 2.30	ENCODED 97	JKB	
send 64KB from ac200000		🚺 ohkawa	411 anim1.encoded		100		
send 1KB from a801587c		🍢 コンピューター 💡	412_star0.encoded	The state of the second			
Finished 400_oflow (batchID:412) ===		🗣 ネットワーク	412_star1.encoded	A 10 10 1			
		I≣ DEGIN-PC	413_rectangle0.encoded				
Starting 400_otlow (batchID:413) ===		P49AF3A	413_rectangle1.encoded	And the Party of the last	HALL IN THE		
Waiting recv 32KB at acUUUUUU		I LYNX	UDP_Client_v06.jar	1 1			=
Waiting recv 32KB at ac100000		NAS	2 410_tux.sh	1			_
firstFrame : finished reading/decoding imgU width=200, height=200		N OHKAWA-PC	411_anim.sh	and the second second	and the second second		
secondFrame: finished reading/decoding imgl width=200, height=200 The number of flavor, 100		IN VELA	412_star.sn				
The humber of flows; IVU		WINS1	all ch				
The Tength IS 10001 bytes.		WINS2	Makefile	Contraction (man -		
cond 1/R from a01597c		RESULT 410 tur	64KB decoded hmp + ++++ 1				
Finished 400 of Low (batchID:413) ===		BMP ファイル	サイズ: 2		100		
I III SIEd 400_0110# (batch10;410)		状況:	33 共有 作成日時: 2				
Starting 400 of low (batchID:414) ===		更新日時:	2014/06/27 2:36				
やマはいといと言法ル	1-4128	化ナフ 印発		± 1	L =		
ソル (ごよいよいよ高1米11.	10 1719	149 今日3	\mathbf{E} \mathbf{T} \mathbf{T} \mathbf{E} \mathbf{A} \mathbf{D}	モレノ	よつ		
						-	-



Document P62



コンテスト実行委員会コアチーム Version 2014-07-28

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、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
 - <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.








- 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





メモリフ		
開始アドレス	終了アドレス	
0×0000_0000	0x0003_FFFF	on-chip memory
0×0004_1020	0x0004_102f	Contest UART
0×0005_0000	0x0005_07FF	Nios II/e
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。
 - 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/unvde2-115-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)
 - 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

Name filter ep4ce115F29

Show advanced devices

· · · · · · · · · · · · · · · · · · ·						
Directory, Name, Top-Level Entity [page 1 of 5]						
What is the working directory for this project?						
/cadhome/kazuya/2014ProcessorDesignContest/Altera/ContestSys05						
What is the name of this project?						
ContestSys						
ContestSys						
Dec and in reject Settings						
< <u>Back</u> <u>Next</u> <u>Finish</u> <u>Cancel</u> <u>H</u> elp						

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 n'Available devices' list-
Eamily: Cyclone IV E	Pac <u>k</u> age: Any
Dev <u>i</u> ces: All	Pin <u>c</u> ount: Any
Target device	Sp <u>e</u> ed grade: Any

- 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		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 cO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ペロジ先の画面4)e Computer System Design Contest

PLLの追加





画面2



PLLの追加

MegaWiza	ard Plug-In Manager [page 7 of 1]	1] (wish上) ×
Parameter 2 PLL Settings Reconfiguration 3 Output 4]EDA	
clkc0 $>$ $clkc1$ $>$ $clkc2$ $>$ $clkc3$	> clk c4 >	
ALTPLL1403687282944402	c1 - Core/External Output Able to implement the requested PLL	t Clock
inclk0 inclk0 frequency: 50.000 MHz c0	Clock Tap Settings	mequested Settings Actual Settings
Clik Ratio Photo Photo	C Enter output clock frequency: C Inter output clock parameters: Clock parameters:	50.0000000 MHz 50.000000
c1 1/1 -54.00 50.00	Clock division factor	1 << Copy 1
Cyclone IV E	Clock phase shift	-3.00 - ns3.00
	Clock duty cycle (%)	50.00
		Description Va
	Note: The displayed internal settings of the PLL is recommended for use by	Modulus for M counter
	advanced users only	Modulus for N counter
		Per Clock Feasibility Indicators
		c0 c1 c2 c3 c4
		Cancel < Back Next > Finish

画面4

	<u>^</u> -?=		clk_in	Clock Input	clk	exported	
		머	clk_in_reset	Reset Input	reset		
			clk	Clock Output	Double-click to export	clk_0	
			clk_reset	Reset Output	Double-click to export		
V			Epil	Avalon ALTPLL			
(┣┓┫┝┥┢	\rightarrow	Inclk_interface	Clock Input	Double-click to export	clk_0	
		\rightarrow	inclk_interface_reset	Reset Input	Double-click to export	[inclk_inte	
	\smile		pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_inte	÷.
		<	c0	Clock Output	Poulde click to export	pll_c0	
			c1	Clock Output	sdram_clk	pll_c1	
		8	areset_conduit	Conduit	Double-circk to export		
		\leftarrow	locked_conduit	Conduit	Double-click to export		
		\sim	phasedone_conduit	Conduit	Double-click to export		

画面3



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

	Epll	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
<u>}</u>	c0	Clock Output	Double-click to export	pll_c0
	- c1	Clock Output 🛛 🌔	sdram_clk	pll_c1
	areset_conduit	Conduit	power of the second	
0	locked_conduit	Conduit 🤇	locked	



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

4	SDRAM Controller - new_sdram_control
SDRAM Co altera_avalon_i	ontroller new_sdram_controller
Block Diagram Show signals Show signals ew_sdram_controller_(clk clk clk clk clock reset reset s1 avalon wire conduit a_avalon_new_sdram_controller	Memory Profile Timing • Data Widtb 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

<u>å</u>	SDRAM Controlle	er - new_sdram	controll
SDRAM C altera_avalon	ontroller new_sdram_controller		
Block Diagram Block Diagram Show signals ew_sdram_controller_0 clk_clock reset reset reset reset reset 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 2 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)
- pllのcO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)

ġ.	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	
		\frown	clk	Clock Output	Double-click to export	clk_0
			clk_reset	Reset Output	Double-click to export	
	~		🗆 pll	Avalon ALTPLL		
-		$ \rightarrow \rightarrow$	inclk_interface	Clock Input	Double-click to export	clk_0
_		$ \uparrow \longrightarrow$	inclk_interface_reset	Reset Input	Double-click to export	[inclk_inte
-			pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_inte
\bigtriangledown			cO	Clock Output	Double-click to export	pll_c0
<u>u</u>			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	
			🗆 Edram	SDRAM Controller		
		\rightarrow	clk	Clock Input	Double-click to export	pll_c0
		\bigcirc	reset	Reset Input	Double-click to export	[clk]
			s1	Avaion Memory Mapped Slave	couble-click s export	[CIK]
		•••	wire	Conduit	Isdram_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

<u> </u>		On-Chip Memory (RAM or ROM) - onchip_mem
On-Chip Mem	o_memory2	
* Block Diagram		
Show signals	Type:	RAM (Writable) 👻
onchip_memory2_0	Dual-port access	
	Single clock operation	
clk1 clock	Read During Write Mode:	DONT CARE
s1avalon reset1rsset	Block type:	
altera_avalor_onchip_memory2	▼ Size	
	Data width:	
	Total memory size:	262144 bytes
	Minimize memory block usa	ge may impact fme
	🔻 Read latency	
	Slave s1 Latency.	1 🗸





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

. [Use	Connections	Name	Description	Export	Clock	Bas
-	~		🗆 clk_0	Clock Source	· · · · ·		
×			clk_in	Clock Input	clk	exported	
2		D-	clk_in_reset	Reset Input	reset	-	
		\longrightarrow	clk	Clock Output	Double-click to export	clk_0	
			clk_reset	Reset Output	Double-click to export		
	V		🖾 pll	Avalon ALTPLL			
		\bullet \bullet \diamond \leftrightarrow \rightarrow	inclk_interface	Clock Input	Double-click to export	clk_0	
		$\bullet \longrightarrow$	inclk_interface_reset	Reset Input	Double-click to export	[inclk_inte	
			pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_inte	1
2			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		
	V		🗆 sdram	SDRAM Controller			
		$\diamond \rightarrow \phi \rightarrow \phi$	clk	Clock Input	Double-click to export	pll_c0	
		\bullet	reset	Reset Input	Double-click to export	[clk]	
			s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	ef.
		<u>~~</u>	wire	Conduit	sdram_wire		
	v		🗆 onchip_memory	On-Chip Memory (RAM or ROM)			
		$\uparrow \rightarrow \rightarrow$	clk1	Clock Input	Double-click to export	pll_c0	
			s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	÷
		\bullet	teset1	Reset Input	Double-click to export	[clk1]	
	V		🗧 jtag_uart 💦 刘	JTAG UART			
		\rightarrow	UIK	Clock Input	Double-click to export	pll_c0	
		$(\rightarrow \rightarrow$	reset	Reset Input	Double-click to export	[clk]	
			avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	m ²



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

	$ \uparrow + \uparrow \uparrow \rightarrow \rightarrow$	clk	Clock Input		Double-click to expo	τI
	$ + \rightarrow$	reset	Reset Input		Double-click to expo	21
		avalon_jtag_slave	Avalon Memory Mapped Slave		Double-click to expo	21
~		🗆 contest_uart	contest_uart_avalon_interface			
		s1	Avalon Memory Mapped Slave		Double-click to expo	τlí
		conduit_end_0	Conduit	C	export	
	$ \diamond \downarrow \bigcirc \diamond \longrightarrow$	clock_sink	Clock Input		Double-click to expo	τI
	$ \bigcirc \longrightarrow$	reset_sink	Reset Input		Double-click to expo	t I



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

	, , L		ICIUCK IIIDUC		EXPRICES			
	→ D-	clk in reset	Reset Input	reset				
		clk	Clock Output	Double-click to export	clk 0		clk_0.c	:lk_in
		clk reset	Reset Output	Double-click to export			Clock	Input [clock_s
V			Avalon ALTPLL					
	\bullet	inclk interface	Clock Input	Double-click to export	cik 0			
	$ \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$	inclk interface reset	Reset Input	Double-click to export	finclk inte			
		pli slave	Avalon Memory	Double-click to export	finclk inte	÷		
		cO	Clock Output	Double-click to export	DJ IIa			
		c1	Clock Output	sdram_clk	pll_c1			<u> </u>
	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	areset_conduit	Conduit	Double-click to export				
		locked_conduit	Conduit	Double-click to export				
		phasedone_conduit	Conduit	Double-click to export				
		🗆 sdram	SDRAM Controller					
_	\diamond \bullet \diamond \rightarrow \rightarrow	clk	Clock Input	Double-click to export	pll_c0			
	$ \downarrow \downarrow$	reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory	Double-click to export	[clk]		0x07ff_ffff	
		wire	Conduit	sdram_wire				∽
×		🗆 onchip_memory	On-Chip Memor					
	$\diamond + \phi + + + \rightarrow$	clk1	Clock Input	Double-click to export	pll_c0			
		s1	Avalon Memory	Double-click to export	[clk1]		0x0003_ffff	
	+ + + + + + + + + + + + + + + + + + +	reset1	Reset Input	Double-click to export	[clk1]			
2		🗆 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_0007	⊢(∩)
~		🗆 contest_uart	contest_uart_av					Y
	$ \bigcirc + \rightarrow$	s1	Avalon Memory	Double-click to export	[clock_sink]		0x0000_000f	
		conduit_end_0	Conduit	export				
	$\diamond \downarrow \phi \phi \downarrow \downarrow \downarrow \longrightarrow$	clock_sink	Clock Input	Double-click to export	pll_c0			
	$ \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$	reset_sink	Reset Input	Double-click to export	[clock_sink]			
V		🗆 nios2	Nios II Processor					
		clk	Clock Input	Double-click to export	pll_c0			
		reset_n	Reset Input	Double-click to export	[clk]			
	- <u>-</u>	data_master	Avalon Memory	Double-click to export	[clk]	IRQ (I	RQ 31 🛹 🗌
		instruction_master	Avalon Memory	Double-click to export	[clk]			
		jtag_debug_modu	Reset Output	Double-click to export	[clk]			
	• • · · · · · · · · · · · · · · · · · ·	itaa debua module	Avalon Memory	Double-click to export	[clk]		0x0000 Offf	



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

Core Nios II Caches	and Memor	ry Interfaces	Advanced Fe	atures	MMU and MPU Settings	JTAG Debug M
" Select a Nios II Cor	e					
Nios II Core:		Nio	s II/e			
		🔾 Nio	s II/s			
		⊖ Nio	s II/f			
	٩	lios II/e		Nios	ll/s	Nios II/f
Nios II Selector Guide	R	NSC 12-bit		RISC 32-bit Instru Branch Hardw Hardw	ction Cache 1 Prediction rare Multiply rare Divide	RISC 32-bit Instruction Cache Branch Predictior Hardware Multipi Hardware Divide Barrel Shifter Data Cache Dynamic Branch
Memory Usage (e.g.	Stratix IV) T	'wo M9Ks (or	equiv.)	Two M	9Ks + cache	Three M9Ks + ca
<u>,</u>	,			,	,	
* Hardware Arithmet	ic Operatio	n				
Hardware multiplicati	on type:	Ember	dded Multiplier	s 💌		
🗌 Hardware divide						
Reset Vector						
Reset vector memory.		onchip	_memory.s1	-		
Reset vector offset:		0x000	00000			
Reset vector:		0x000	00000			
Exception Vector						
Exception vector men	nory:	onchip	_memory.s1	-		
Exception vector offse	et:	0x000	00020	1		
Exception vector:		0x00C	00020			
MMU and MPU						
🗌 Include MMU						



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 y a Clock Output sdram_alk pl_a

н	c1	Clock Output	sdram_clk	pll_c1			а- I		
-	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]			いわせ	· MI +	よ能で
÷	s1	Avalon Memory	Double-click to export	[clk]	● 0x0800_0000	Oxofff_fff	ツンさ	ニットレノニイノ	へ怒て
\succ	wire	Conduit	sdram_wire				· · · ·]		
	🛛 onchip_memory	On-Chip Memor				「ア	ドレフ	くを 人 カー	
÷	clk1	Clock Input	Double-click to export	pll_c0		/			∠,
÷	s1	Avalon Memory	Double-click to export	[clk1]	● 0x0000_0000	00003_ffff	A 14	n ht.	ムルフ
÷	reset 1	Reset Input	Double-click to export	[clk1]		7	リ復	ロツクをフ	いける
	🗉 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	0x004_0007	⊢_0		
	🗆 contest_uart	contest_uart_av							
÷	s1	Avalon Memory	Double-click to export	[clock_sink]		0x0004_102f			
거	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 O	I	.RQ 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	0.0005_07ff			
\leq	custom_instruction	Custom Instructi	Double-click to export						



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

	Se	ungs - com	estsys					
						Device		
	Files							
nd Conditio	Select the design files project directory to th	s you want to i le project.	nclude in the project. C	lick Add All to	add all design file	es in the		
	<u>F</u> ile name:					. Add		
Settings	File Name	Type Libra	ry Design Entry/Syntl	hesis Tool	HDI Version	Add All		
pilation 5 Optimizal	nios2_sys.qsys ContestSys.v	Qs Ver	<none> <none></none></none>		Default	<u>R</u> emove		
hesis						<u>U</u> р		
n						Down		
Settings						Properties		
J	こをクリック	ルて追	加するファ	イルを	選択し, ^上	その後ad	dボタンを押す	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	報 の ir	nport	と割き	Ц					
×	Edges Named: *GPIO[?] ▼ ॐ	» Edit: 💥 🖌	_				() and post () and post () () and post () and post () () in post () and () and () () post () and () ()	Constant, S. J. Constant, S. J. Constant, Society and	ala an	Fil	ter: Pir
8	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Jurrent Strengt	Slew Rate	Differential Pair	
		Unknown	PIN AB22	4	B4 N0	3.3-V LVTTL		8mA (default)			
	③ 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)			
	③ GPIO[3]	Unknown	PIN_Y17	4	B4_N0	3.3-V LVCMOS		2mA (default)			
	③ GPIO[4]	Unknown	PIN_AC21	4	B4_N0	3 - VEVILE		8mA (default)			
	OPIO[5]	Unknown	PIN_Y16	4	B4_N0	3.3-V LVCMOS	>	2mA (default)			
	<pre> GPIO[6] GPIO[7] </pre>	Unknown	PIN_AD21	4	B4_N0	3.3-		8mA (default)			
		Unknown	PIN_AE16	4	B4_N2	3.3-V LVCMOS	>	2mA (default)			
		Unknown	PIN_AD15	4	B4_N2	3 3-V I VCMOS		2mA (default)			
	< <new node="">></new>			•				and (actually			



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)

ŧ	Workspace Launcher	×
Select a wo	rkspace	
Eclipse store Choose a wo	s your projects in a folder called a workspace. Inkspace folder to use for this session.	
<u>W</u> orkspace:	14ProcessorDesignContest/Altera/ContestSys05/software]
▶ <u>C</u> opy Sett	ings	
?	Cancel OK	



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

Nios II Software Examples	
Create a new application and board support package based on a software example template	
Target hardware information SOPC Information File name: /cadhome/kazuya/2014ProcessorDesignContest/A CPU name: nios2	
Application project Project name 400oflow Use default location Project location: /cadhome/kazuya/2014ProcessorDesignContest/Altera/Cont	
Project template Templates Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance Image: Template description Image: Template description Image: Template description Blank Project Template description Blank Project creates an empty project to which you can add your code. For details, click Finish to create the project and refer to the readme.txt file in the project directory. The BSP for this template is based on the Altera HAL operating system. To use a BSP based on a different	
? < Back Next > Cancel Finish)

Nios II Application and BSP from Template





BSPを保存するときの

File>___

- 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

Linker Section Mappings Linker Section Name .ess .entry .exceptions .beap .rodata .stack .text	Linke or fip_memory reset onchip_memory onchip_memory onchip_memory	Reg. o Name	Memory Device	: Name	Add
Linker Section Name .bss .entry .exceptions .heap .rodata .stack .text	Linke openip_memory feset onchip_memory onchip_memory onchip_memory	• Keg. no Name	Memory Device onchip_memory onchip_memory	2 Name	Add
bss .entry .exceptions .heap .rodata .rwdata .stack .text	orenip_memory feset onchip_memory onchip_memory onchip_memory		onchip_memory onchip_memory		Domous
.entry .exceptions .heap .rodata .rwdata .stack .text	reset onchip_memory onchip_memory onchip_memory		onchip_memory		a cuulue
.exceptions .heap .rodata .rwdata .stack .text	onchip_memory onchip_memory onchip_memory				De la composición
.heap .rodata .rwdata .stack .text	onchip_memory		onchip_memory		Restore Default
.rodata .rwdata .stack .text	I onchip memory		onchip_memory		
.rwdata .stack .text			onchip_memory		
.stack .text	onchip_memory		onchip_memory		
.text	onchip_memory		onchip_memory		
	onchip_memory		jonch1p_memory		
	reset				
l	onchip_memory				
Linker Memory Regions	Isdram]		
Linker Region Name	Address Sange 🔻	Memory Device Name	Size (bytes)	Offset (bytes)	Add
sdram 0:	x08000000 - 0x0EFFFFFF	sdram	134217728	0	Remove
onchip_memory 0:	x00000020 - 0x000siiii	onchip_memory	262112	32	
reset 0:	x00000000 - 0x0000001F	onchip_memory	32	0	Restore Default
					Add Memory Dev
					Remove Memory D
					Memory Usage
					Memory Map.
[]					p
Crownd out optries are outomatically	created at generate time. The	v are not editable or parci	icted in the PSP cettings file		
Grayed out entries are automatically	y created at generate time. The	y are not cultable or persi	sted in the bor settings file.		
Information Problems Processi	ing				
Cinished las ding drivers from another	an la la caractería				
Finisheu loaunig unvers from ensei	mble report.				
Loading BSP settings from settings	Tile.				
Finished loading SOPC Builder syste	em into tile "//nios2_sys.sopcir	to [relative to settings file]"			
Changed mapped section ".bss" fro	m memory region "sdram" to men	1 ory region "onchip memory	الر		



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

	📉 Tutoria
🖫 Problems 🖉 Tasks a Console 🗞 🗉 Properties 🗖 🗖	🥸 Sample
CDT Build Console [400oflow_bsp]	
↓ ↓ </th <th>🔶 What's</th>	🔶 What's
<pre>alt_software_exception.0 obj/HAL/src/alt_stat.0 obj/HAL/ src/alt_tick.0 obj/HAL/src/alt_times.0 obj/HAL/src/ alt_uncached_free.0 obj/HAL/src/alt_uncached_malloc.0 obj/ HAL/src/alt_unlink.0 obj/HAL/src/alt_usleep.0 obj/HAL/src/</pre>	🗇 Workb
alt_wait.o obj/HAL/src/alt_write.o obj/HAL/src/	
alt svs init.o obj/drivers/src/	
altera_avalon_jtag_uart_fd.o obj/drivers/src/	
altera_avalon_jtag_uart_init.o obj/drivers/src/	
altera_avalon_jtag_uart_ioctl.o obj/drivers/src/	
altera_avalon_jtag_uart_read.o obj/drivers/src/	
[BSP build complete]	
**** Build Finished ****	
v	



bsp

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_STA TUS,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);



Document P70

The 2nd ARC/CPSY/RECONF High-Performance Computer System Design Contest Submission Rule of Design Data

コンテスト実行委員会コアチーム Version 2014-07-28

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

プロセッサ設計部門 予選のための設計データの提出方法

- 予選データの提出期限は、日本標準時2014年8月4日(金)午後1時です。
- 参加チームは、上記期限までに参加登録をおこなってください.
- 提出方法: <u>contest_support@virgo.is.utsunomiya-u.ac.jp</u> までメール
 - メールには、登録チーム名・使用したFPGAボード名を記入してください。
- 提出するファイルは次の6種類のデータを1つのZIPファイルにまとめたものとします.
 - 1. FPGAの回路情報のファイル名はXilinxの場合は System.bit, Alteraの場合はSystem.sof としてください.
 - 2. アプリケーション 310_sort のためのユーザファイル. ファイルサイズ 256KB, ファイル名 310sort.bin としてください.
 - 3. アプリケーション 320_mm のためのユーザファイル. ファイルサイズ 256KB, ファイル名 320mm.bin としてください.
 - 4. アプリケーション 330_stencil のためのユーザファイル. ファイルサイズ 256KB, ファイル名 330stencil.bin としてください.
 - 5. アプリケーション 340_spath のためのユーザファイル. ファイルサイズ 256KB, ファイル名 340spath.bin としてください.
 - 審査の参考資料として、1~4ページの原稿(情報処理学会研究報告のフォーマット)を提出して ください. PDF形式、ファイル名は document.pdf としてください. https://www.ipsj.or.jp/kenkyukai/genko.html
 - 受け取った予選データを用いて, 直ちに, 動作検証をおこないます. 正しく動作しない場合には, その 旨を通知し, データの再提出を求めることがあります. データ提出後, しばらくは電子メールに直ちに 返信できるようにしてください.



コンピュータシステム設計部門 予選のための設計データの提出方法

- 予選データの提出期限は、日本標準時2014年8月4日(金)午後1時です。
- 参加チームは、上記期限までに参加登録をおこなってください.
- 提出方法: <u>contest_support@virgo.is.utsunomiya-u.ac.jp</u> までメール
 - メールには、登録チーム名・使用したFPGAボード名を記入してください。
- 提出するファイルは次の3種類のデータを1つのZIPファイルにまとめたものとします。
 - 1. FPGAの回路情報のファイル名は Xilinxの場合は System.bit, Alteraの場合はSystem.sof としてください.
 - 2. アプリケーション 400_oflow の動作のために必要な設計データー式と,動作手順書. 予選では実行委員の手によって提出デザインを実際に動作させるので,そのために必要なデーター式です. ソースコード等の提出は必要ありません.
 - 審査の参考資料として、1~4ページの原稿(情報処理学会研究報告のフォーマット)を提出して ください. PDF形式、ファイル名は document.pdf としてください. https://www.ipsj.or.jp/kenkyukai/genko.html

受け取った予選データを用いて, 直ちに, 動作検証をおこないます. 正しく動作しない場合には, その 旨を通知し, データの再提出を求めることがあります. データ提出後, しばらくは電子メールに直ちに 返信できるようにしてください.



両部門 FIT2014デザインコンテスト予稿集原稿

- 決勝に進出したチームには、原稿(1~4ページ)を執筆していただきます.
 - 提出日: 日本標準時 2014年8月22日(金)午後1時
 - 提出方法: <u>contest_support@virgo.is.utsunomiya-u.ac.jp</u> までメール
- ・ この予稿集は、FIT2014本体の予稿集とは別に、コンテスト独自に作成するものです。
- ・ 原稿は予選時に提出したものと同様に、情報処理学会研究報告のフォーマットで作成してください.
- ・ PDF形式, ファイル名は (予選通過時のチーム番号).pdf としてください 例)P1.pdf
 - <u>https://www.ipsj.or.jp/kenkyukai/genko.html</u>
- 以下の事項に同意いただいた上で、提出をお願いいたします。
 - ・ 著作権は著者に帰属するものとします。
 - ・ 作成した予稿集は、電子データで、コンテストの参加者に配布します。
 - ・ 提出されたPDFの、情報処理学会研究報告のテンプレートに含まれる著作権表示・ページ数等の表記を、実行 委員にて編集させて頂くことがあります。



両部門 決勝での競技方法

- 決勝では各自FPGAボードを持ち寄り、コンテスト実行委員が用意するexStick,およびホストPCと接続してその場で処理時間を計測する競技を行います。
 - 設計データの提出は必要ありません.
 - 処理時間の計測が可能なように、各参加者が準備を行ってください、
- 決勝当日の本番前に接続テストを行いますので、決勝に進出した方は参加してください。
 - 詳細は後ほど連絡します
- 2014年9月5日午後 開催の決勝戦イベントセッション@筑波大にてポスター発表していただきます.
- 以上の条件を満たせないチームは失格となりますので、注意してください。





- Ver.2014-08-17
 - プロセッサ設計部門のルールDocument P37を修正
 - ・ 320_mmの出力値をチェックサム→CRC32に変更しました
 - ・ 各アプリケーションのデータサイズ上限値を明記(第1回コンテストと同様)
 - Document P70にFIT2014デザインコンテスト予稿集原稿の提出方法を追加
- Ver.2014-07-28
 - 予選のための設計データ提出方法に関する記述(P.70)を追加
- 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)

