Document P51

AL

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

> コンテスト実行委員会コアチーム Version 2014-06-26

**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



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





出力画像(RGB成分を同様の形式で圧縮)

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

3



- ・ 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



#### SDKの使い方 (1) ディレクトリ構成

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



#### 主なソースファイルの構成

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



ユーティリティツール一覧

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





- 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



登埋 ▼ 🗧 ブレビュー ▼ 印	刷	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	
デスクトップ		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	
		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	
I 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	
💻 ОНКАWA-РС		@ 411_anim.sn	2014/06/26 23:10	SHファイル	1 KB	
IN VELA		• 412_star.sh	2014/06/26 23:10	SHJPTU	1 KB	
WINS1		413_rectangle.sh	2014/06/26 23:10	SH JPTU	1 KB	
I WINCO		e all.sn	2014/06/26 23:09	SH ファイル	1 KB	
Im ₩1N52	•		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。
  - 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)







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





- Ver.2014-06-26
  - 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(Xilinx,Altera)公開
- 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)



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 2014-06-26

**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>







- 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.
  - <u>http://japan.xilinx.com/ise\_eval/index.htm</u>



#### 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	U
Ch Sull Second LISS LIADT		СН	1
-carrenspeed 030 GART		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)	12
<ul> <li>Supports 12 Mbps USB full-speed data rate</li> </ul>	<ul> <li>External Converters (Dongles)</li> </ul>	AutoRTS/CTS	Ye
<ul> <li>Supports USB suspend, resume and remote wakeup</li> </ul>	operations • Battery-Operated Devices	AutoRS-485	Ye
Enhanced UART Features	Cellular Data Devices	5V/Tolloouts	Ye
Data rates up to 12 Mbps     Eractional Raud Rate Constrator	Factory Automation and Process Controls	Svitemputs	16
<ul> <li>128 byte TX FIFO</li> </ul>		Sup V	2.9
<ul> <li>384 byte RX FIFO</li> </ul>	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 Co	► Documents	► Documents	
Automatic Somware (Xon/Xon) Flow Control     Multidrop mode w/ Auto Half-Duplex Transceiver Cont     Multidrop mode w/ Auto TX Enable	Block Diagram	Block Diagram	
Half-Duplex mode		Datasheets	
<ul> <li>Selectable GPIO or Modern I/O</li> <li>Internal 48 Mild clock</li> </ul>			
Single 2 97-3 63V power supply		Datasheet	
<ul> <li>5V tolerant inputs</li> </ul>		Version 1.3.1	
Virtual COM Port Drivers		July 2013	
<ul> <li>Windows 2000, XP, Vista, 7, and 8</li> </ul>		229 57 KB	
<ul> <li>Windows CE 4.2, 5.0, 6.0, and 7.0</li> </ul>			
- Linux		Safaran Drivere	
* Mac		- Soitware Drivers	
lescription		Windows 2000, XP,	Vista, 7, a
he XR21V1410 is an enhanced 1-channel Universal Asynchro	nnous Receiver and Transmitter (LIART) with a LISR interface. The LISR inter	days Drivers	
s fully compliant to Full Speed USB 2.0 specification that sup	ports 12 Mbps USB data transfer rate. The USB interface also supports USB	Version 2.0.0.0	
uspend, resume and remote wakeup operations.		August 2013	



## ソフトウェアセットアップ Tera Term のインストール



- 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
 ▲ ⑦ ポート (COM と LPT)
 ▲ ⑦ XR21V1410 USB UART (COM5)
 ▶ ⑧ 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	×		
ボート( <u>P</u> ):	COM16 -	ок		
ボー•レ <i>ー</i> 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 ミリ秒/字( <u>c</u> ) 0 ミリ秒/行( <u>L</u> )				



## 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	→ Browse
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 ٠

- Finishをクリック	Memory read Perpheral Tests SRCE Bockoder Xilkernel POSIX Threads Demo
le New Project	
Application Project Create a managed make application project.	
Project name: 400_offow	v
Location: K:¥workspace-edk147#Atlys-oflow#SDK_Export#400_oflow Browse Choose file system: default	(?) < Back Next > Finish Cancel
Target Hardware Hardware Platform Atlys-oflow hw platform	
Processor microblaze_0	
Target Software OS Platform standalone	
Language	
Board Support Package @ Create New 400_oflow_bsp	
Image: Second	

New Proje Template

Create one of the available templates to generate a fully-function

A blank C project



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

G

## 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

grun i POAL C Peuro	Specify the JTAG settings Specify the JTAG seble to use for communication and JTAG Device Chain configuration of the target board.
のWindowでProgramをクリック mss - Xilinx SDK gate Search Run Project [Xilinx Tools] Window Help	These settings affect how XMD connects to the FPGA.  JTAG Cable  Type: Diglient USB Cable Hos name: Xilinx Hardware Server Pole: Diglient USB Cable UBglient USB Cable Encode USB Cable
Image: Second Secon	Frequency: Xilinx Parallel III Cable Other Options: Xilinx USB Cable 3rd Party Cable, Xilinx Plug-in TTAG Device Chain © Automatically Discover Devices on JTAG Chain © Manual Configuration of JTAG Chain ① Manual Configuration
Modify this BSP's Se       Important Flash         XMD Console       XMD Console         Target Information       Import And Settings         This Processing       System Generator Co-Debug Settings         Hardware Specification       Create Zynq Boot Image         Target Procession       Create Zynq Boot Image	FPGA?     Device Name     ID Code     IR Length
Operating System         Board Support Package OS.         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 standard input and output, profiling, abort and exit.         Documentation:       standalone v3 11 a         Image: Source       Image: Source         Image: Problems Image: Tasks Image: Console Image: Source       Image: Properties Image: Source         CDT Build Console [401_oflow]       Image: Source         CTT: Itae Image: Harding Hardi	OK         Cancel           Program FPGA         Specify the bitstream and the ELF files that reside in BRAM memory           Hardware Specification: KL*workspace edit47W4tys effow/EDKXEDK/EDeptrK4tys effow/Jws.platformWrystem.aml         Bitstram:           Bitstram:         program fPGA           Sochware Configuration         Restmin_M           Hordware Specification:         KL*workspace edit47W4tys effow/EDKXEDK/EDK/EDeptrK4tys effow/Jws.platformWrystem.aml           Bitstram:         program           Sochware Configuration         Freesor           Procesor         ELF file to Initialize in Block RAM           microtalaze_0         Sochware Configuration

Configure JTAG Settings



- ∎≎

C/C++ - 401\_oflow\_bsp/system File Edit Source Refactor Nav 📑 🗕 🖪 🕼 🗁 🚫 🕶 🍕 🕶 🔲 🕥 🗄 🖢 🕶 🖓 🕶 🌤 🦾 🕶 占 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

Imain.c oflow.c In oflow.h b c platform.c b h platform.h 🙀 lscript.ld README.txt 😂 401\_oflow ---

rowse..

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_oflov	v_bsp/system.mss - Xilinx SDK	a. 17	
File Edit Source	Refactor Navigate Search Run	Project Xilin:	x Tools Window Help
	⑧ ▼ � ▼ 局 │ @ ▼ @ ▼ @   ▼ \$> \$> ▼	• © •	糁 ▼ 🕢 ▼ 💊 ▼ 📐 🕞 🍀 🗹 🧭 🥙 ▼ 🛛 🛱 Œ C/C++
Project Explorer	🛪 👘 🗆 🕞 system.xm	l 👔 syste	m.mss 👔 system.mss 🛛 🖓 🖓 Ma 🖓 🖓
	🖻 🔄 🔻 🞽 401_oflow	_bsp Board	Support Package An outline is not available.
▷ 25 400_oflow		•	
▶ 100_oflow_bs	p Modify this	BSP's Settings	
⊿ 🐰 Binari∈	New	+	
Þ 🏇 40:	Go Into		
⊳ 👘 Includ	Open in New Window		is compiled to run on the following target.
Debug	Сору	Ctrl+C	¥workspace-edk14/¥Atiys-oflow¥SDK¥SDK_Export¥Atiys-ofl
b la ade	Paste	Ctrl+V	
🕞 庙 ado 🎽	Delete	Delete	
⊳ 💽 coc	Source	,	
⊳ <u>h</u> i cod	Rename	F2	e
	Import		e is a simple, level coffuere laver. It provides access to
🕨 🔂 file 🎿	Export		, interrupts and exceptions as well as the basic features of a
⊳ 🔓 file	Build Project		input and output, profiling, abort and exit.
⊳ <u>ic</u> ima	Clean Project		<u>e v3 11 a</u>
	Refresh	F5	III b
⊳ 🖻 ofic	Close Project		
⊳ 🔓 ofic	Close Unrelated Projects		pnsole 🛿 🗧 Properties 🦉 Terminal 🛛 🔱 🔐 🔛 🔝 🖬 🔹 🗹 🛨 🗌
⊳ <u>ic</u> ì pla	Build Configurations	•	
isci	Make Targets	*	low.elf.elfcheck
📄 RE/	Index	,	
⊿ 🏄 401_oflov	Show in Remote Systems view		took 4s.710ms)
▶ i BSP D	Run As	•	1 Launch on Hardware (GDB)
D* 😂 4	Debug As	•	c 2 Local C/C++ Application
	Profile As	•	X 3 Remote ARM Linux Application
	Team	•	Run Configurations
R. C.S	Compare With	<b>ب</b> ا	
100 H 100 H	Run C/C++ Code Analysis		
	Generate Linker Scrint		
	Change Referenced BSP		
6013	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

# スタック領域の大きさを設定するには、

スタック領域の設定 (1/2)

٠

プロジェクトを右クリックして、Generate Linker Script を選択します

- しかし、最初の画像を読み込んだ後、止まってしまうかもしれません。
  - XSDKのデフォルトでは、スタック領域が1KB(0x400)バイトしかありません ので、上手く動きません。 C/C++ - 401\_oflow/src/lscript.ld - Xilinx SDK File Edit Source Refactor Navigate Search Run Project





## 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	-		_		X
Generate linker script					8
Control your application's memory	y map.				2
Project: 401_oflow Output Script:				Basic Advanced Place Code Sections in:	m/b_ddr2_50_AXI_BASEADDR
pace-edk147¥Atlys-oflow¥SDK¥SE	OK_Export¥401_c	flow¥src¥lscript.ld	Browse	Place Data Sections in:	mcb_ddr2_SU_AXI_BASEADDR
Modify project build settings as fol	lows:			Place Heap and Stack in	microblaze_0_i_bram_ctrl_microblaze_0_d_bram_ctrl 👻
Set generated script on all projec	t build configurat	ions	•	Heap Size:	1 KB
				Stack Size:	16 KB
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(E) 編集(E) 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 firstFrame : finished reading/decoding img0 width=128, height=128 geografications/finished reading/decoding img1 width=129, height=129	*	🕒 🗸 🖌 K:) 🕨 cygwir	n + home + ohkawa + contest + 400_ofic	ow 🕨 client	▼ ↓ clientの検	*	
The number of flows: 58		整理 ▼ 🔤 プレビュー ▼ 印	刷 書き込む 新しいフォルダー			833 🔻	
The length is 6888 bytes.		📲 nas 🔺	名前	更新日時	種類	サイズ	
send 64KB from ac200000		Workspace	412_star1.y.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
send 1KB from a801587c		🖳 ohkawa (fs1.ced.is.utsunom	413_rectangle0.bmp	2014/06/25 12:53	BMP ファイル	118 KB	
Finished 400 oflow (batchID:411) ===			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	
Starting 400 of low (batchID:412) ===			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 beight=200		■ ドキュメント	RESULT.412_star.64KB.decoded.bmp	2014/06/27 2:36	BMP ファイル	118 KB	
secondFrame: finished reading/decoding img0 width=200 beight=200		🔤 ピクチャ	RESULT.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_tux0.encoded	2014/06/27 2:36	ENCODED ファ	5 KB	
The length is 7/5/ bytes		↓ ミュージック	410_tux1.encoded	2014/06/27 2:36	ENCODED ファ	5 KB	
cond 64/R from ac200000		ohkawa	411_anim0.encoded	1000			
cond 11/B from a801587c			411_anim1.encoded		C		
Send TND FTOM adorsore			412_star0.encoded				
r mished 400_0110w (batchib:412)			412_star1.encoded	- <b>19</b>	1 C 1 C 1		
Starting 400 of Law (batab ID: 412)		I DEGIN-PC	413_rectangle0.encoded	1000			
Starting 400_0110W (batch10.415)		IN EP49AF3A	413_rectangle1.encoded	1000			
Mailing recy SZND at action000		IN LYNX	UDP_Client_v06.jar	- <b></b>	and the second second		
Waiting recy 32Kb at actu00000 firstFrame , firishad aradian/daradian imp0 width=200, haidt=200		NAS	410_tux.sh	<b>7</b>			
TIRSTERAME FINISHED READING/DECODING IMBU WIDTN-200, HEIGHT-200		IN OHKAWA-PC	411_anim.sn	10 m 10 m	<b></b>		
secondFrame: finished reading/decoding imgl width=200, height=200		VELA	412_star.sn				
The number of flows: IUU		WINS1	I all all all all all all all all all al				
The Tength is TUUDI bytes.		WINC2	Malufia	Sec. Sec. Sec.	and and		
send 64KB from aczulluu		I≣ WIN32 ▼	Makelle	and the second s			-
Send IKB from abulbb/c		RESULI.410_tu	JX.64KB.decoded.bmp 大きさ: 1	a service a	1000		
Finished 400_oflow (batch1D:413) ===			2: 23 共有 作成日時: 2		100 C		
Starting 400_oflow (batchID:414) ===		更新日時	: 2014/06/27 2:36				
		R		_			_
わっしょう トレート 古法ル	1-4128	おうち て 日日		± 1	L 2	•	
んじょいよいよう迷化	17 4765	ex g sie:	$\mathbf{H}$		よつ		
					5 -	-	-



Document P62



コンテスト実行委員会コアチーム Version 2014-06-26

**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)
  - 4. EDA toolの設定でSimulationツールとしてModelsimを選んでいる場合は FormatをVerilog HDLにする
  - 5. その他はデフォルトでNextを押し, finishまでいく



## 新規プロジェクトの作成



## Name filterでデバイス名の候補を filteringできる

Name filter ep4ce115F29

Show advanced devices

Directory, Name, Top-Level Entity [page 1 of 5]									
What is the <u>w</u> orking directory for this project?									
/cadhome/kazuya/2014ProcessorDesignContest/Altera/ContestSys05									
What is the name of this project?									
ContestSys									
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the case of the design file.									
ContestSys									
Ose This Project Settings									
< <u>B</u> ack <u>N</u> ext> <u>Fi</u> nish Cancel <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 in '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: 🏹

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

### C Other: n/a

### Available devices:

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



## Qsysの起動と外部クロックの設定

- ・ ここではQsysのシステムとして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ページ先の画面4)e Computer System Design Contest

## PLLの追加





画面2



## PLLの追加

MegaWizard Plug-in Manager [page 7 of 11] (wish 上)								
Parameter PLL Settings Reconfiguration Clocks	EDA							
ALTPLL1403687282944402           Inciko         Inciko freguency 50 000 MHz           Operation Mode: Normal         Inciko           Exception of the methy incide to the methy incidence to t	Cock duty cycle (%)							
	Note: The displayed internal settings of the PLL is recommended for use by advanced users only         Description         Vit           Modulus for M counter           Modulus for M counter							
	Per Clock Feasibility Indicators c0 c1 c2 c3 c4							

画面4

	Ŷ <b>─</b> Ŷ <b>─</b>	- <b>C</b> -	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			Epll	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	÷.
			cO	Clock Output	Poulde click to export	pll_c0	
		-0-	c1	Clock Output	sdram_clk	pll_c1	
		<u>~</u>	areset_conduit	Conduit	Double-circk to export		
		$\diamond$	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

		[		Avalon ALTPLL		
$\bigcirc$	\$ <del>\$</del>	$\longrightarrow$	inclk_interface	Clock Input	Double-click to export	clk_0
	+ ÷	$\longrightarrow$	inclk_interface_reset	Reset Input	Double-click to export	[inclk_inte
		$\longrightarrow$	pli_slave	Avalon Memory	Double-click to export	[inclk_inte
	$\succ$		cO	Clock Output	Double-click to export	pll_c0
			c1	Clock Output 🛛 🌔	sdram_clk	pll_c1
		Q	areset_conduit	Conduit	power of the second	
			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

<b>4</b>	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 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):	<ul> <li>1</li> <li>2</li> <li>3</li> <li>2</li> <li>7.8125</li> <li>200.0</li> <li>70.0</li> <li>20.0</li> <li>20.0</li> <li>5.5</li> <li>14.0</li> </ul>	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c0をsdram controllerのclkに配線
- clk\_0のclk\_resetをsdramのresetに配線
- sdramのwireをExportするようにExport欄をダブルクリック.(Export名を sdram\_wireとする)

ġ.	System	Contents 🛛	Address Map 🛛	Project Settings 🛛		
<b>.</b>	Use	Connections	Name	Description	Export	Clock
-	~		🗆 clk_0	Clock Source		
~		~~ <del>-</del>	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	at they impact free
	🕆 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

•		clk_0     clk_in     clk_in_reset     clk     clk_reset     pll     inclk_interface     inclk_interface_reset     pll_slave     c0	Clock Source Clock Input Reset Input Clock Output Reset Output Avalon ALTPLL Clock Input Reset Input Avalon Memory Mapped Slave	cik reset Double-click to export Double-click to export Double-click to export Double-click to export	exported clk_0 clk_0 (inclk_inte	
		clk_in clk_in_reset clk_reset <b>□ pll</b> inclk_interface inclk_interface_reset pll_slave c0	Clock Input Reset Input Clock Output Reset Output Avalon ALTPLL Clock Input Reset Input Avalon Memory Mapped Slave	clk reset Double-click to export Double-click to export Double-click to export Double-click to export	exported clk_0 clk_0 [inclk_inte	
ſ		cik_in_reset cik cik_reset <b>D pii</b> incik_interface incik_interface_reset pii_slave c0	Reset Input Clock Output Reset Output Avalon ALTPLL Clock Input Reset Input Avalon Memory Mapped Slave	reset Double-click to export Double-click to export Double-click to export Double-click to export	clk_0 clk_0 [inclk_inte	
•		clk clk_reset inclk_interface inclk_interface_reset pll_slave c0	Clock Output Reset Output Avalon ALTPLL Clock Input Reset Input Avalon Memory Mapped Slave	Double-click to export Double-click to export Double-click to export Double-click to export	clk_0 clk_0 [inclk_inte	
ł		clk_reset pll inclk_interface inclk_interface_reset pll_slave c0	Reset Output Avalon ALTPLL Clock Input Reset Input Avalon Memory Mapped Slave	Double-click to export Double-click to export Double-click to export	<b>cik_0</b> [inclk_inte	
•		pll inclk_interface inclk_interface_reset pll_slave c0	Avalon ALTPLL Clock Input Reset Input Avalon Memory Mapped Slave	Double-click to export Double-click to export	<b>cik_0</b> (inclk_inte	
•		inclk_interface inclk_interface_reset pll_slave c0	Clock Input Reset Input Avalon Memory Mapped Slave	Double-click to export Double-click to export	<b>clk_0</b> [inclk_inte	
		inclk_interface_reset pll_slave c0	Reset Input Avalon Memory Mapped Slave	Double-click to export	[inclk_inte	
		pll_slave c0	Avalon Memory Mapped Slave	The solution of the test state of the		
		cO		Double-click to export	[inclk_inte	ef .
	- I - ) — œ		Clock Output	Double-click to export	pll_c0	
		c1	Clock Output	sdram_clk	pll_c1	
	Ŷ	areset_conduit	Conduit	Double-click to export		
	¢	locked_conduit	Conduit	Double-click to export		
	¢	phasedone_conduit	Conduit	Double-click to export		
		🗆 sdram	SDRAM Controller			
Ŷ		clk	Clock Input	Double-click to export	pll_c0	
	$\uparrow \rightarrow$	reset	Reset Input	Double-click to export	[clk]	
		sl	Avalon Memory Mapped Slave	Double-click to export	[CIK]	=i,
		wire	Conduit	sdram_wire		
		i onchip_memory	Un-Chip Memory (RAM or RUM)			
ľ			Clock Input	Double-click to export	pII_CU	
		51	Avaluri Memory Mapped Slave	Double click to export	[LIK1]	=.
		T itse uset	Keset input	DURABLE-CHEK TO EXPORT	[CIK1]	
		e jtay_uart	Clock Input	Double, click to expert	nii c0	
		rocot	Posot Input	Double click to export	pn_co felki	
6		avolon itaa clavo	Avolon Momony Monned Slove	Double click to export	[cik]	ے ا
	•		S1 tag_uart tik reset avalon_jtag_slave	Avaion Memory Mapped Slave Reset Input JTAG UART Clock Input Reset Input Reset Input Avaion Memory Mapped Slave	S1 Avaion Memory Mapped Slave Double-click to export Reset Input Tag_uart Clock Input reset avaion_jtag_slave Avaion Memory Mapped Slave Double-click to export Avaion Memory Mapped Slave Double-click to export	S1     Avaion Memory Mapped Slave     Double-click to export [clk1]       Double-click to export [clk1]     Reset Input     Double-click to export [clk1]       TAG UART     Clock Input     Double-click to export [clk]       Clock Input     Reset Input     Double-click to export [clk]       Avaion Memory Mapped Slave     Avaion Memory Mapped Slave     Double-click to export [clk]



contest uart avalon interfaceをComponentとして登録

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

## contest uart avalon interfaceをComponentとして登録



- ・ Signalsタブ内
  - clockのinterfaceの所を選択して、new Clock Inputを選択し、interface 欄がclock\_sinkとなるようにし、Signal Typeをclkとする
  - resetのinterfaceの所を選択して、new Reset Inputを選択し、 interface欄がreset\_sinkとなるようにする.
- ・ Interfaceタブ内
  - Remove Interfaces With No Signalsボタンを押す
  - s1のAssociated Clockをclock\_sinkに, Associated Resetをreset\_sink にする
  - conduit\_end\_0のAssociated Clock, Associated Resetも同様にする
  - reset\_sinkのAssociated Clockをclock\_sinkにする
- ・ Finishボタンを押し、保存するかどうかを訪ねるWindowが出たら、Saveを選ぶ

## contest uart avalon interfaceを追加

- ProjectのMy Own IP Core内のcontest\_uart\_avalon\_interfaceを選択して Add
  - Finishボタンをクリック
- 名前をcontest\_uartに変更
- contest\_uart\_avalon\_interfaceのclock\_sinkをpllのc0と接続
- contest\_uart\_avalon\_interfaceのreset\_sinkをclk\_0のclk\_resetと接続
- contest\_uart\_avalon\_interfaceのconduit\_end\_0のexport欄をダブルクリ ックして、export名をexportとする

	$  \uparrow + \bullet \uparrow \rightarrow$	clk	Clock Input		Double-click to expor	٤
	$    +   \rightarrow$	reset	Reset Input		Double-click to expor	8
		avalon_jtag_slave	Avalon Memory Mapped Slave		Double-click to expor	8
~		🗆 contest_uart	contest_uart_avalon_interface			
		s1	Avalon Memory Mapped Slave		Double-click to expor	εĮ
		conduit_end_0	Conduit	C	export	
	$ \diamond \downarrow \bigcirc \diamond \longrightarrow$	clock_sink	Clock Input		Double-click to expor	εĮ
	$  \bigcirc \longrightarrow$	reset_sink	Reset Input		Double-click to expor	εĮ



## 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

				1116	FANNIEN			
		clk in reset	Reset Input	reset	coporcea			<u> </u>
-		cik	Clock Output	Double-click to export	cik û		clk_0.clk_in	1
		cik reset	Reset Output	Double-click to export	circ_0		Clock Inpu	t [clock_s
		nll	Avalon ALTPLI	Double energy cappie				
		inclk interface	Clock Input	Double-click to export	cik 0			
		inclk interface reset	Reset Innut	Double-click to export	linclk inte			
		nll slave	Avalon Memory	Double-click to export	[inclk_inte	<b>₽</b>		
		rO	Clock Output	Double-click to export	nll r0			
		c1	Clock Output	sdram cik	pll_co			
	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	areset conduit	Conduit	Double-click to export				
		locked_conduit	Conduit	Double-click to export				
		phasedone_conduit	Conduit	Double-click to export				
1		sdram	SDRAM Controller					
	$\diamond + \phi \rightarrow$	clk	Clock Input	Double-click to export	pll_c0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory	Double-click to export	[clk]		0x07ff_ffff	
		wire	Conduit	sdram_wire				$\sim$
V		onchip_memory	On-Chip Memor					
		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	j	tag_uart	JTAG UART					
		clk	Clock Input	Double-click to export	pll_c0			
	$   \bullet   \to   \to  $	reset	Reset Input	Double-click to export	[clk]			
		avalon_jtag_slave	Avalon Memory	Double-click to export	[clk]	e 0x0000_0000	0x0000_0007	
~	🗡   🗆 🛛	contest_uart	contest_uart_av					
		s1 .	Avalon Memory	Double-click to export	[clock_sink]		0x0000_000f	
		conduit_end_0	Conduit	export				
		clock_sink	Clock Input	Double-click to export	pll_c0			
		reset_sink	Reset Input	Double-click to export	[clock_sink]			
~		nios2	Nios II Processor					
		clk	Clock Input	Double-click to export	pll_c0			
		reset_n	Reset Input	Double-click to export	[clk]			
		data_master	Avalon Memory	Double-click to export	[clk]	IRQ C	IRQ 3	
		instruction_master	Avalon Memory	Double-click to export	[clk]			
		jtag_debug_modu	Reset Output	Double-click to export	[clk]			
	$  \bullet \bullet \longrightarrow  $	itaa debua module	Avalon Memorv	Double-click to export	lfclk1		0x0000 0fff	



#### nios2の設定

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

Core Nios II Caches and Mer	mory Interfaces	Advanced Features	MMU and MPU Settings	[ JTAG Debug M
r Select a Nios II Core				
Nios II Core:	Nice	os II/e		
	🔾 Ni	os II/s		
	O Ni	os II/f		
	-			
	Nios II/e	Nio	s II/s	Nios II/f
Nios II Selector Guide	RISC 32-bit	RISC 32-E Instr Bran Hard Hard	it uction Cache ch Prediction ware Multiply ware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multipl Hardware Divide Barrel Shifter Data Cache Dynamic Branch
Memory Usage (e.g Stratix IV)	Two M9Ks (or	equiv.) Two	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV)	Two M9Ks (or	equiv.) Two	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV) Hardware Arithmetic Opera	Two M9Ks (or	equiv.) Two	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV)	Two M9Ks (or tion Embe	equiv.) Two	M9Ks + cache ·	Three M9Ks + ca
Memory Usage (e.g Stratix IV) THardware Arithmetic Opera Hardware multiplication type: Hardware divide	Two M9Ks (or tion Embe	equiv.) Two	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV) Hardware Arithmetic Opera Hardware multiplication type: Hardware divide Reset Vector	Two M9Ks (or tion Embe	equiv.) Two	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV; Hardware Arithmetic Opera Hardware multiplication type: Hardware divide Reset Vector Reset vector memory.	tion Embe	equiv.) Two dded Multipliers	M9Ks + cache	Three M9Ks + ca
Hardware Arithmetic Opera Hardware multiplication type: Hardware divide Reset Vector Reset vector memory. Reset vector offset:	tion Embe	equiv.) Two dded Multipliers  p.memory.s1	M9Ks + cache	Three M9Ks + ca
Hardware Arithmetic Opera Hardware Multiplication type: Hardware divide Reset Vector Reset vector memory. Reset vector offset: Reset vector:	tion Embe	equiv.) Two dded Multipliers  p_memory.s1  000000  000000	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV) THardware Arithmetic Opera Hardware multiplication type: Hardware divide TReset Vector Reset Vector Reset vector offset: Reset vector: Texception Vector	tion Embe	equiv.) Two dded Multipliers  p_memory.\$1  000000  000000	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV;  Hardware Arithmetic Opera Hardware multiplication type: Hardware divide  Reset Vector Reset Vector Reset vector offset: Reset vector:  Exception Vector Exception vector memory:	tion Embe Onchij Ox000 Ox000	equiv.) Two  dded Multipliers  p_memory.s1  000000  p.memory.s1	M9Ks + cache	Three M9Ks + ca
	tion Conchi Ox000 Ox000 Ox000 Ox000	equiv.) Two dded Multipliers  ↓  p_memory.\$1  000000  p_memory.\$1  000000	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV) THardware Arithmetic Opera Hardware multiplication type: Hardware divide TRESET Vector Reset Vector offset: Reset vector: TException Vector Exception vector memory: Exception vector offset: Exception vector iffset: Exception vector:	tion Embe Clonchin 0x000 0x000 0x000 0x000 0x000 0x000 0x000 0x000	equiv.) Two dded Multipliers  p_memory.\$1  000000  p_memory.\$1  000002  0000020	M9Ks + cache	Three M9Ks + ca
Memory Usage (e.g Stratix IV;    Hardware Arithmetic Opera Hardware multiplication type: Hardware divide  Reset Vector Reset Vector offset: Reset vector:  Exception Vector Exception Vector memory: Exception vector offset: Exception vector offset: Exception vector:  MMU and MPU	tion Conchin Ox000 Conchin Con	equiv.) Two dded Multipliers  p_memory.\$1  000000  p_memory.\$1  0000020  0000020	M9Ks + cache	Three M9Ks + ca



#### 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]					
$\leq$	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, <sup>-</sup>	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ì	Ц					
×	-	» Edit <sup>.</sup> 🗶 🖌	<b>_</b>				(classe), ) () an approximation of provide () an approximation of the state ()	Silication of a state of the st	rada adat	Fil	Iter: Pir
8	Node Name	Direction	Location	I/O Bank	VREE Group	I/O Standard	Reserved	`urrent Strengtl	Slew Bate	Differential Pair	
Τ		Upknown		4	P4 NO		Reserved	RmA (default)	Siew Hate	Sirrerendarran	
		Unknown	PIN_AC15	4	B4_N0 B4_N2	3.3-V I VTTI		8mA (default)			
	GPIO[2]     GPIO[2]     GPIO[2]	Unknown	PIN AB21	4	B4_N0	3.2-V LV11L		8mA (default)			
	GPIO[3]	Unknown	PIN Y17	4	B4 N0	3.3-V LVCMOS		2mA (default)			
	GPIO[4]	Unknown	PIN_AC21	4	B4_N0	3 - VEVILE		8mA (default)			
	③ GPIO[5]	Unknown	PIN_Y16	4	B4_N0	3.3-V LVCMOS		2mA (default)			
	③ GPIO[6]	Unknown	PIN_AD21	4	B4_N0	3.2-111		8mA (default)			
	③ GPIO[7]	Unknown	PIN_AE16	4	B4_N2	3.3-V LVCMOS	)	2mA (default)			
	GPIO[8]	Unknown	PIN_AD15	4	B4_N2	3.3.41.41		8mA (default)			
	③ GPIO[9]	Unknown	PIN_AE15	4	B4_N2	3.3-V LVCMOS	>	2mA (default)			
	< <new node="">&gt;</new>										



#### 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 wor	kspace	
Eclipse stores Choose a wo	s your projects in a folder called a workspace. rkspace folder to use for this session.	
<u>W</u> orkspace:	14ProcessorDesignContest/Altera/ContestSys05/software 🔹 Browse	
▶ <u>C</u> opy Setti	ngs	
?	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 Template 3 Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance Image: A state of the seadme.txt file in 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 Image: Im
? < Back Next > Cancel Finish

Nios II Application and BSP from Template





BSPを保存するときの

File X \_ \_ \_

- 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

Manual Contraction of the Contract of the Cont	Drivers Linker Script	Enable File	Generation	Target BSP Dire	ectory			
Linker Section Mappings								
Linker Section N	ame 🔺	Linke	r Reg. n Nar	ne	Memory De	vice Name	Ad	d
.bss	07 -N	ip_memory			onchip_memory		Demo	
.entry	rese	t			onchip_memory		Kenre	
.exceptions	onch	ip_memory			onchip_memory		Restore D	efault
.heap	onch	ip_memory			onchip_memory			
.rodata	onch	ip_memory			onchip_memory			
rwdata	onch	ip_memory			onchip_memory			
.stack	onch	1p_memory		_	onchip_memory			
.text	onch	ip_memory			joncnip_memory			
	reset							
J	onen	ip_memory						
Linker Memory Regions	Isurar	n						
Linker Region Name	Address Pang	ie 🔺	Memo	y Device Name	Size (bytes)	Offset (bytes)	Ad	J
sdram	0x08000000 - 0x	PEFFFFFF	sdram	· · · · ·	134217728	3	0 Remo	W/P
onchip_memory	0x00000020 - 0x	00051155	onchip_m	emory	262112	2	32	
reset	0x00000000 - 0x	0000001F	onchip_m	emory	32	2	0 Restore L	erault:
							Add Memor	γ Devi
							Remove Mem	ory De
							Memory	Usage
							Memory	/ Map
Crowned out optring are outer	notically created at gang	rate time. The	ovara not a	ditable or porci	stad in the PSP sattings fi	lo.		
Grayed out entries are autor	natically created at gene	race time. The	cy are not e	anable of persi	stea in the bor settings in			
Information Problems	Processing							
	in occosing							
Inished loading drivers fri	om ensemble report.							
U Loading BSP settings from	settings file.							
Finished loading SOPC Build	der system info file "//ni	os2_sys.sopci	nfo [relative ·	o settings file]"				
Changed manned section *	bss" from memory region	"sdram" to me	mory region	"onchip_memory	".			



# **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
Rroblems 🖉 Tasks Console 🔊 🗉 Properties 🧧 🗖	🔮 Sample
CDT Build Console [400oflow_bsp]	
•          •          •	🔶 What's
alt_software_exception.o obj/HAL/src/alt_stat.o obj/HAL/	
<pre>src/alt_tick.o obj/HAL/src/alt_times.o obj/HAL/src/</pre>	
<pre>alt_uncached_free.o obj/HAL/src/alt_uncached_malloc.o obj/</pre>	🗇 Workb
HAL/src/alt_unlink.o obj/HAL/src/alt_usleep.o obj/HAL/src/	_
alt_walt.o obj/HAL/src/alt_write.o obj/HAL/src/	
altera_hiosz_qsys_ind.0 obj/HAL/sic/cit0.0 obj/	
altera avalon itag 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/	
altera_avalon_jtag_uart_write.o	
[BSP build complete]	
WWW Duild Finished WWW	
TATA DUTIO LIUTNIGO ALAL	



psp

#### 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\_v07.tar.gz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

