動作合成ツールDesignPrototyperとCo-Emulation環境iPROVEでつくるC言語/SystemCからFPGAへのシームレスな設計、検証フロー

ローコスト量産用FPGA向けに
Motion JPEG CODECを実装する

■プロジェクトの目的

□要求性能、仕様

□方式検討

□設計検証評価環境

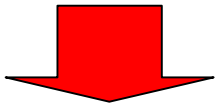
□Altera社FPGAへの実装結果

□デモンストレーション

□まとめ

プロジェクトの目的

- ローコスト量産対応FPGA向けMotion JPEG CODECコアをSystemC/C言語ベース設計で開発



動作合成およびSystemC/C言語ベース設計の適用範囲を見極める

□プロジェクトの目的

■要求性能、仕様

□方式検討

□設計検証評価環境

□Altera社FPGAへの実装

□デモンストレーション

□まとめ

Motion JPEG CODEC 目標性能、要求仕様

- 性能
  - 300万画素クラスCCDカメラに対応
  - フレームレートを満たすこと
- ローコストFPGAに搭載できること
  - Altera社 Cyclone2
  - Xilinx社 Spartan3
- 符号化処理,復号処理のデュアルモード
- ベースラインのみサポート

□プロジェクトの目的

□要求性能、仕様

■方式検討

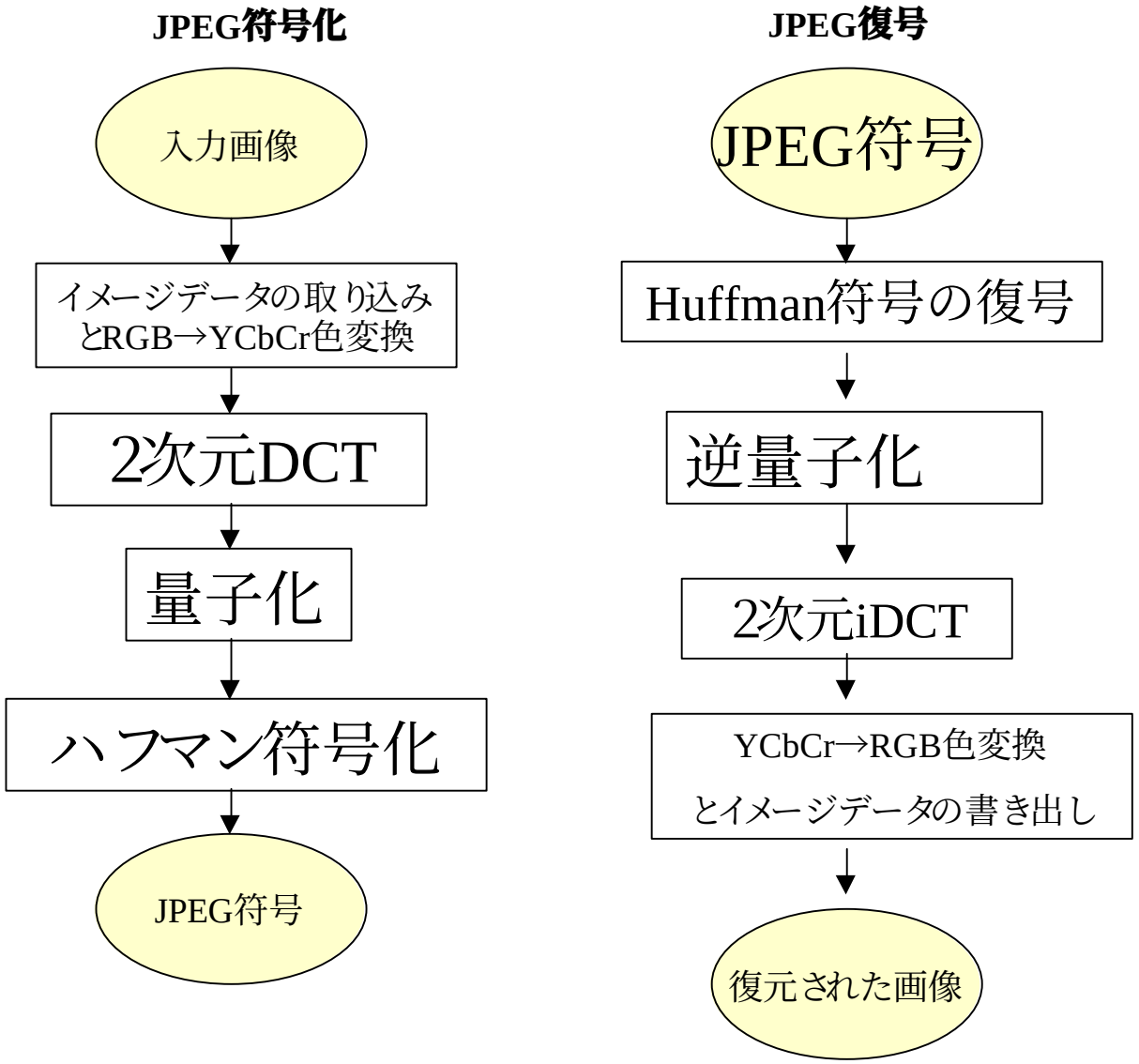
□設計検証評価環境

□Altera社FPGAへの実装

□デモンストレーション

□まとめ

JPEG CODEC の処理内容



JPEGエンコーダの実装

- 量子化
  - 除算を、逆数の乗算で実装
- Huffman符号化
  - 3個のブロックに分割
  - FIFOで接続しパイプライン化
- JPEG符号ヘッダ生成
  - 8 bit x 623語の RAMとして実装
    - 以下のパラメターのみ可変
      - ライン数
      - ラインあたりのサンプル数
      - コメント情報

JPEGデコーダの要素技術

- JPEGヘッダ解析
  - 本プロジェクトでは本CODECで符号化したコードのみ対応(保証)
- Huffman符号復号
  - Lookup tableによる高速逆引き
- 2次元高速DCT/IDCTブロック
- 量子化/逆量子化ブロック
- エラー検出
  - 入力されるJPEG符号が壊れていた場合への対応

符号化処理

- JPEG符号ヘッダは予め作成しておき、CPUのファームウェアで設定
- 量子化テーブルはCPUから変更可能
- Huffman符号は固定値を使用

高速化のための工夫

- 要求性能
  - 256万画素　= 2.56 M pixel
  - 16フレーム/秒
  - 40.96 M pixel/秒　　= 61.44　DCT M pixel/秒

1MCUブロック 16x16画素

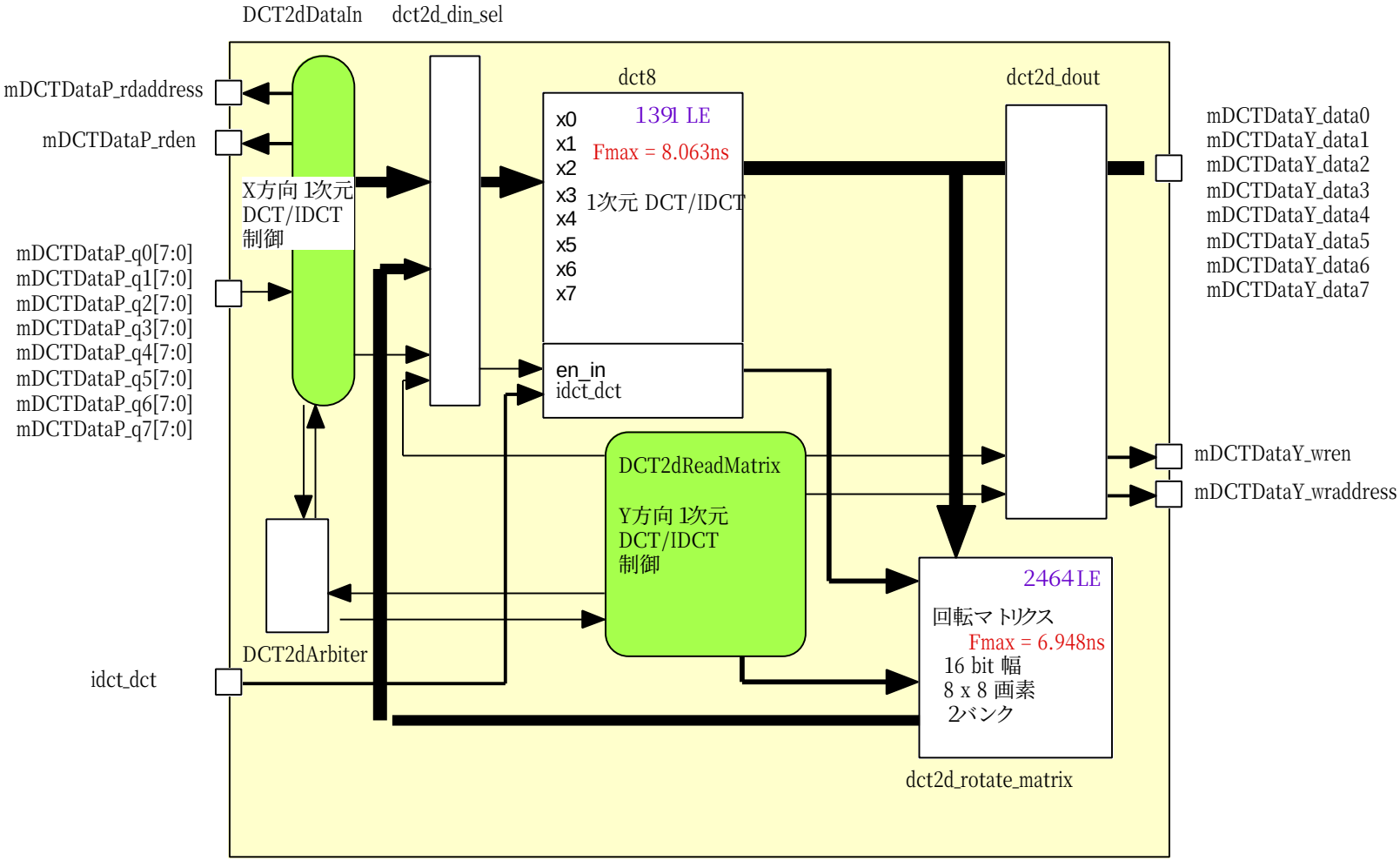
Y	4	DCTブロック
Cb	1	DCTブロック
Cr	1	DCTブロック

高速化のための工夫(2次元DCT/iDCT)

- パイプラインDCT/iDCT演算器
  - Chenのアルゴリズム
    - パイプライン段数 11サイクル
  - 8画素並列に入出力
  - 18bit x 18bit パイプライン乗算器　6個
  - DCT/iDCT兼用化

- DCT/iDCT演算器をX方向,Y方向の処理で共有

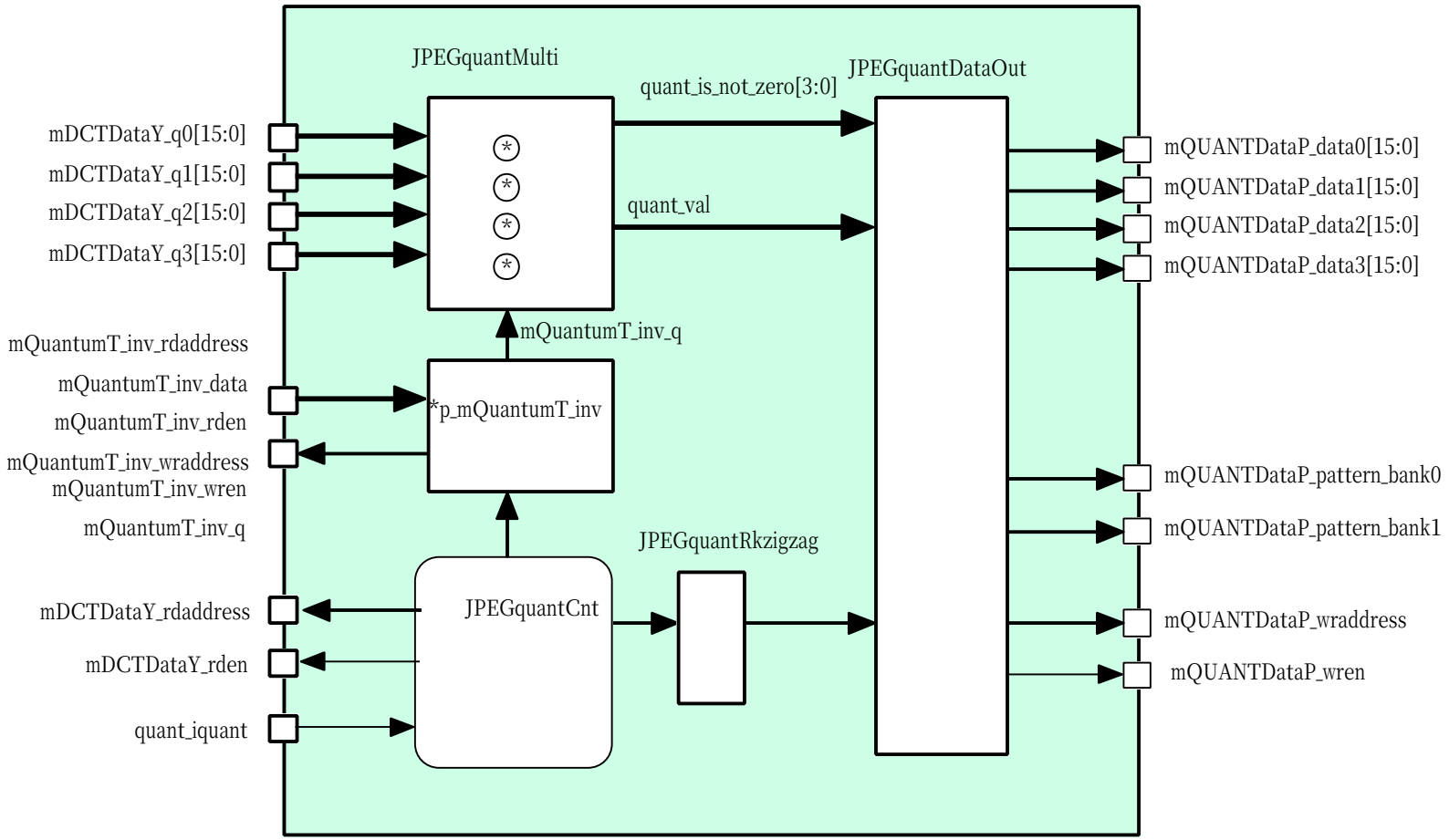
2次元DCT/IDCT



高速化のための工夫(量子化)

- 4画素　同時に量子化
- 量子化処理における除算を逆数の乗算で高速化
- 復号処理における逆量子化では、量子化係数を設定して使用

量子化/逆量子ブロック



復号処理

- JPEG符号ヘッダの解析
  - 最初のフレーム　ファームウェアで解析
    - 解析結果から、Huffmanテーブル、逆量子化テーブル、JPEG符号ヘッダの期待値テーブルをセット
  - 2番目以降のフレーム
    - ハードウェアで,入力されるJPEG符号ヘッダを期待値テーブルと比較
    - 不一致が発生した場合、割り込みを発生

高速化の工夫(Huffman復号)

- Look up table方式
  - 1bitから16bitの符号の候補を同時に検索
  - Look up tableはファームウェアにより、JPEGヘッダのHuffman符号テーブルを解析し、設定

- 出力バッファSRAMへのアクセス回数を削減
  - 0以外の値のみ出力SRAMに書き込み
  - 値が0か非0か否かの情報を64bitのパターンで逆量子化ブロックに渡す
    - 0: 量子化結果の値が0
    - 1:量子化結果が0以外の値

高速化の工夫(Huffman復号)

```
void DecodeHuffman::decode_vlc(
  unsigned short pattern ,
  int            *p_value ,
  sc_uint<6>     *p_size
  )
  {
  bool error_status ;
  unsigned short code ;
  sc_uint<6> length ;
  unsigned int index ;
  unsigned short ad ;

  bool  is_exist_0 ;
  bool  is_exist_1 ;

  bool  is_exist_14 ;
  bool  is_exist_15 ;

  error_status = false ;
  is_exist_0 = is_exist_code( pattern , 1 ) ;
  is_exist_1  = is_exist_code( pattern , 2 ) ;
    …
  is_exist_13  = is_exist_code( pattern , 14 ) ;
  is_exist_14  = is_exist_code( pattern , 15 ) ;
  is_exist_15  = is_exist_code( pattern , 16 ) ;
```

```
if ( is_exist_0 ) length = 1 ;
else if ( is_exist_1 ) length = 2 ;
else if ( is_exist_2 ) length = 3 ;
…
else if ( is_exist_14 ) length = 15 ;
else if ( is_exist_15 ) length = 16 ;
else
  {
  error_status = true ;
  }

code = pattern >> ( 16 - length ) ;
index = code2index( code , length ) ;

ad = index + gl_huffman_table_offset ;
set_address_header_table( ad ) ;

*p_value = read_header_table_b() ;
*p_size = length ;
}
```

高速化の工夫(Huffman符号化)

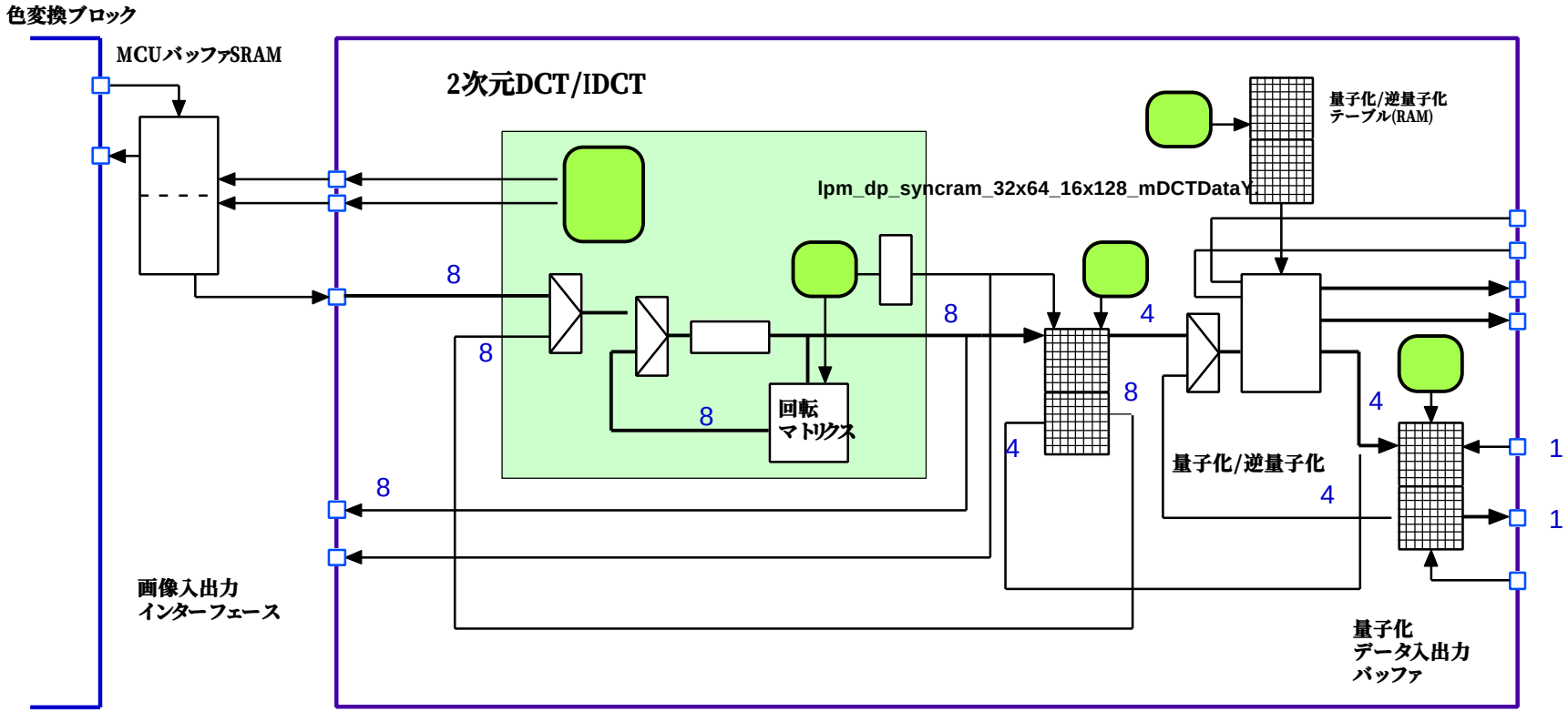
- 値0のAC成分のスキップ
- 量子化処理で、64bitのパターンを設定
  - 0: 量子化結果の値が0
  - 1:量子化結果が0以外の値

  - ビットパターンから連続した値ゼロのビットの個数(Run Length)を数える関数(プライオリティエンコーダ)を作成

エンコーダ/デコーダデュアルモード対応

- 演算ブロックの共用化
  - 2次元DCT/IDCTブロック
  - 量子化/逆量子化ブロック
- SRAMの共用化

JPEGPre (DCT,量子化/逆量子化/iDCT)



ハードウェアへの実装作業の方針

- ハードウェア化のためのアルゴリズムの検討はC++/ANSI-Cでおこなう
- ハードウェアでの処理とソフトウェアでの処理を使い分ける
- SystemC動作記述とVerilog RTL記述の混在デザイン
- ディバイスに依存しないデザイン

ハードウェアへの実装の方針

- デザインパターンの定義
  - 通信と機能の分離
- 合成可能なSystemCサブセットの定義

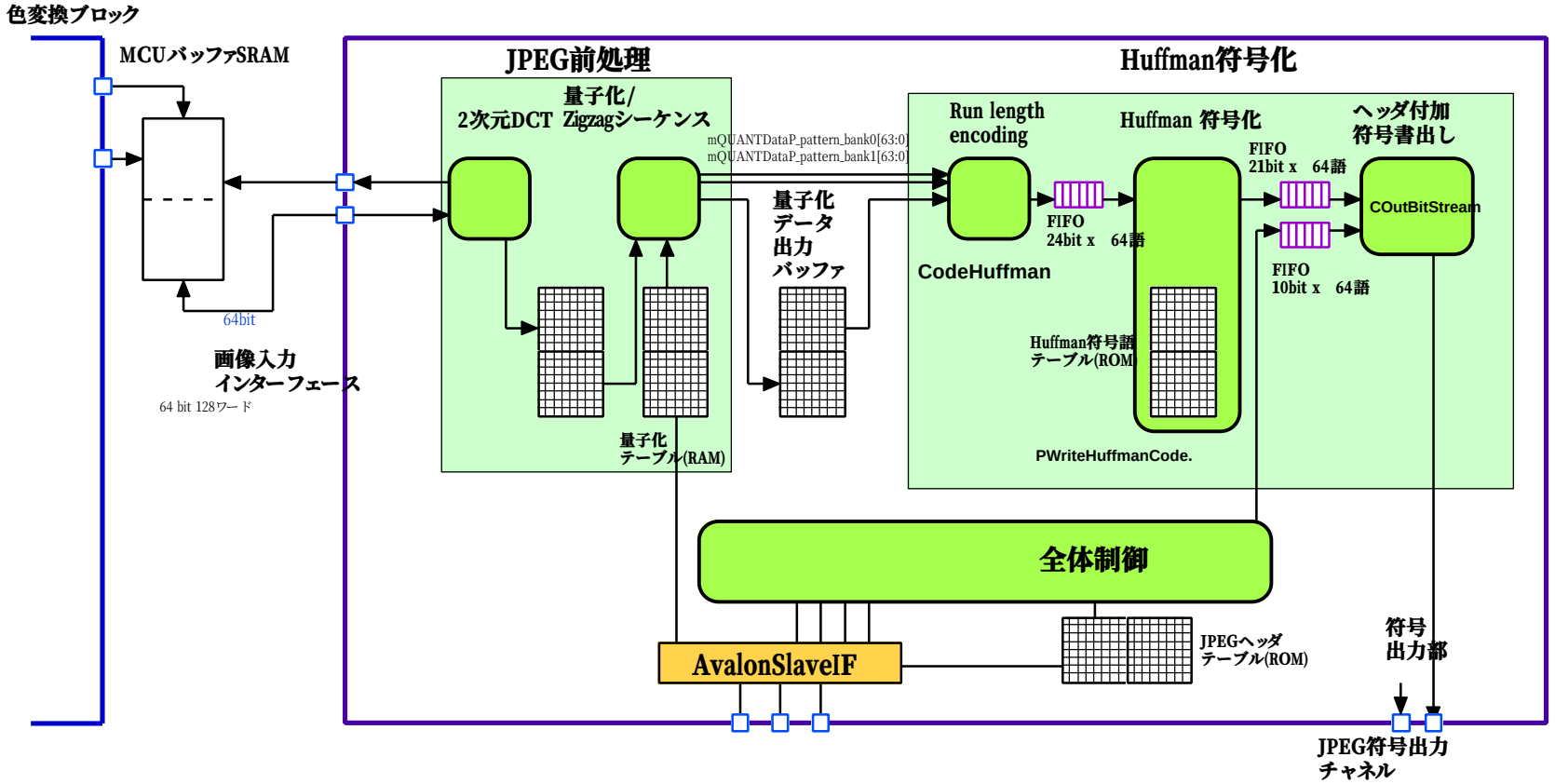
通信と機能の分離

- IOアクセスのためのメンバー関数を定義
  - アーキテクチャレベル高速検証用記述
    - sc_channelによるポートインターフェース接続
  - 動作合成用記述
    - BCAレベルのIOアクセスメンバー関数

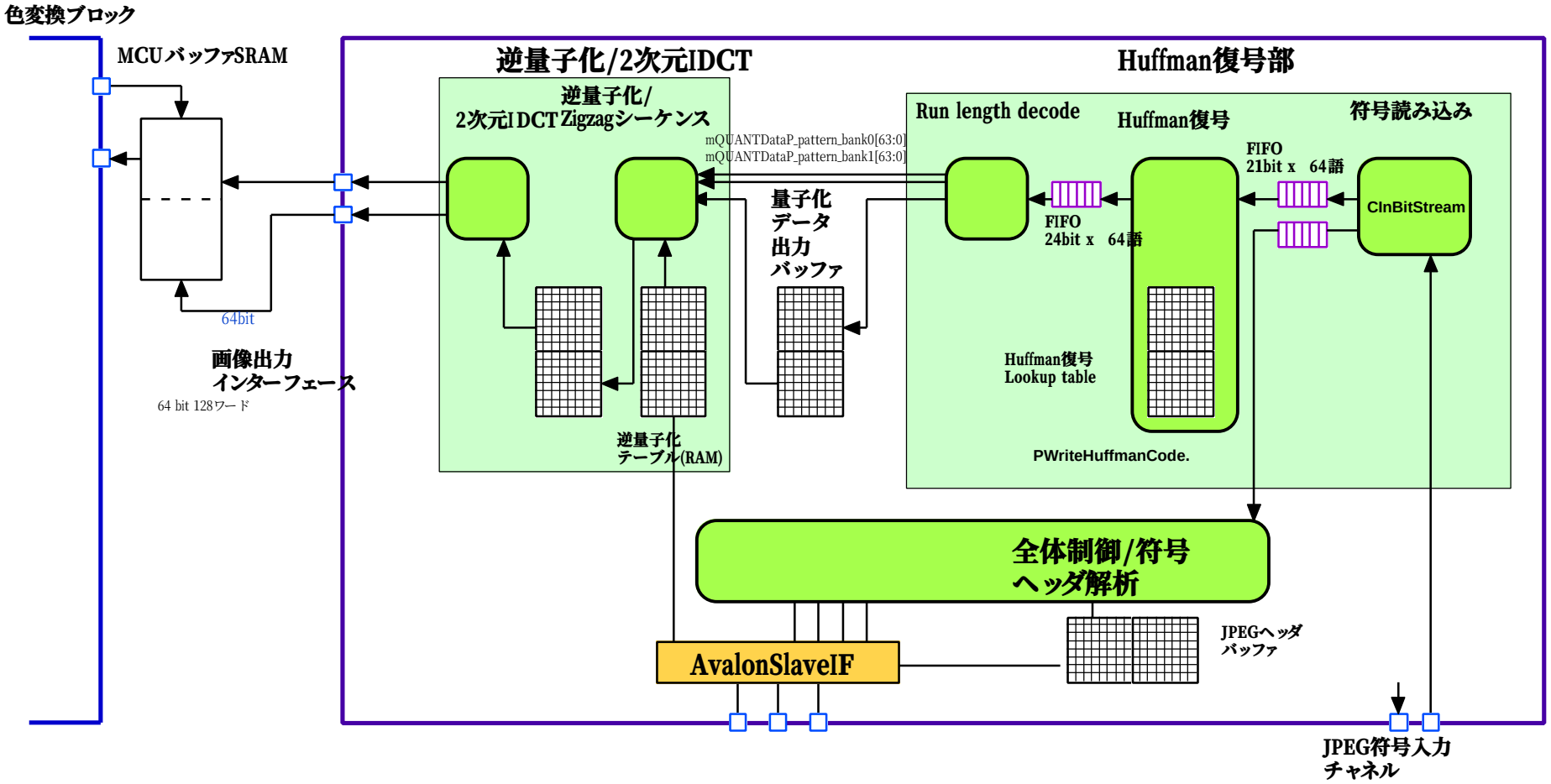
機能/アルゴリズムのモデリング

- 機能/アルゴリズムの記述はC++でANSI-Cのレベルで記述
- IOアクセスはメンバー関数をコール
- ビット幅の制限をおこなう場合
  - C/C++の基本型(short int,char,…)
  - SystemCの型(sc_int,sc_uint,...)
  - または関数(クリッピング,飽和関数等)を定義して使用

JPEGエンコーダ



JPEGデコーダ



□プロジェクトの目的

□要求性能、仕様

□方式検討

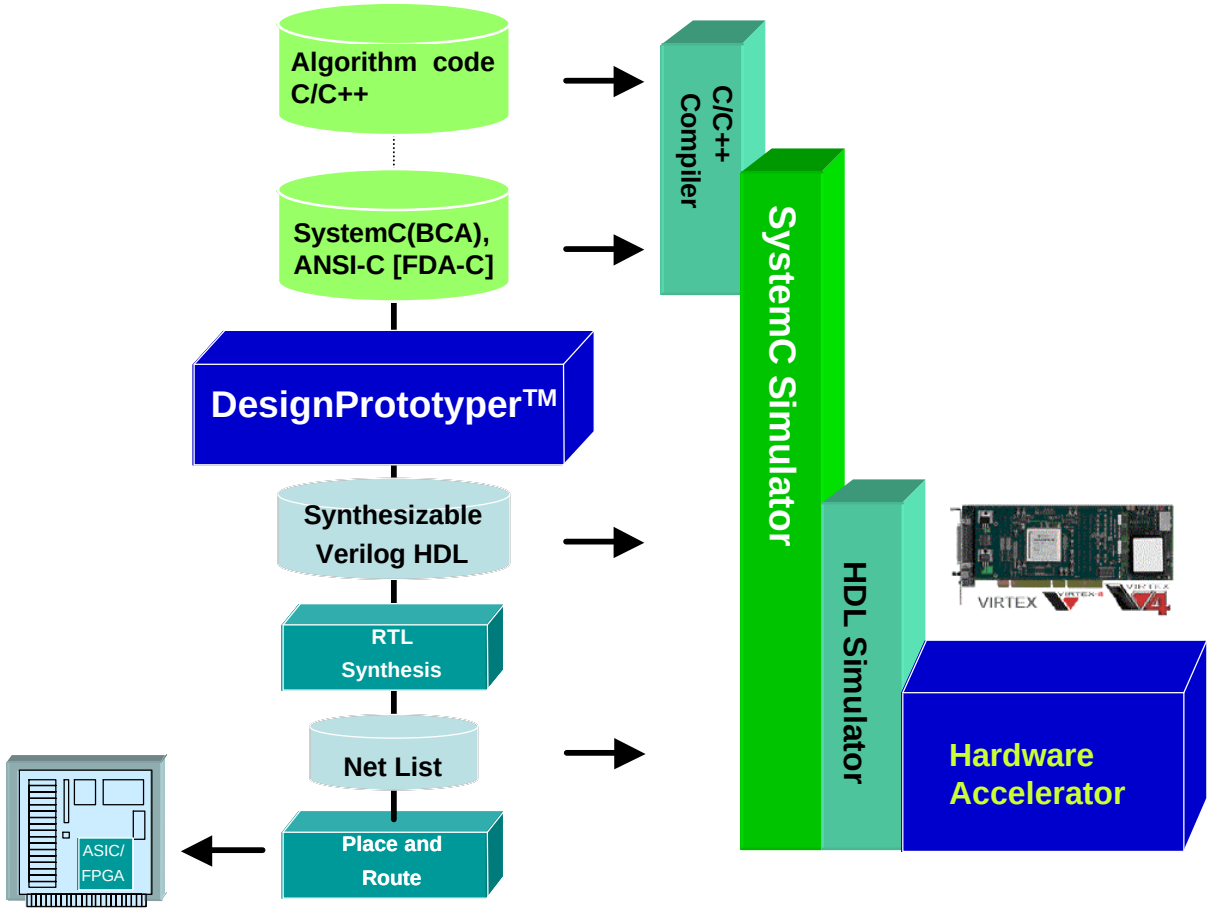
■設計検証評価環境

□Altera社FPGAへの実装

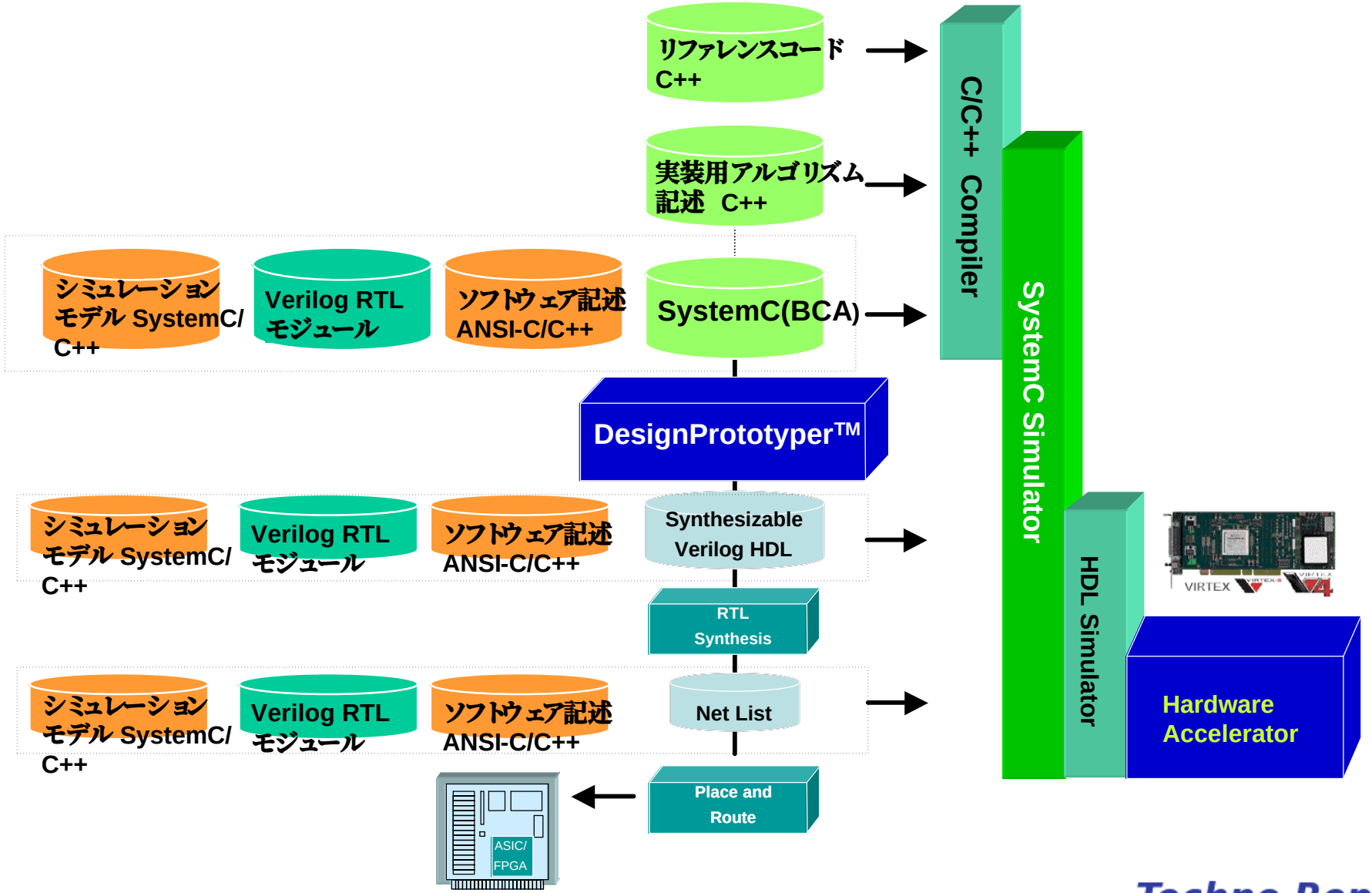
□デモンストレーション

□まとめ

高位合成を用いたC/C++からハードウェアへの設計検証のフロー



本プロジェクトの設計,検証のフロー



実行環境

ツール	製品/バージョン
OS	GNU Linux CentOS 3.8 , Vine Linux 3.2)
C++コンパイラ	GNU GCC (3.2.3 / 3.3.6 / 3.4.6)
SystemC シミュレータ	OSCI SystemC Referense Simulator Version 2.0.1
Verilog to C++ トランスレータ	Verilator 3.623 http://www.veripool.com/verilator.html
グラフィックス ライブラリ	EGGX/ ProCALL version 0.80 (Easy and Gratifying Graphics library for X11) http://phe.phyas.aichi-edu.ac.jp/~ cyamauch/eggx_pocall/
動作合成	DesignPrototyper Version 3.5.2c 20080828
論理合成/ 配置配線	Xilinx ISE 8.2i
ハードウェア アクセレータ	Dynalith iPROVE Version 4 Xilinx 社 Virtex2-8000 を搭載

DesignPrototyperとは？

◆SystemC/ANSI-C入力動作合成ツール

－アーキテクチャレベルのSystemC/C言語記述を短期間でFPGA化

◆現実の開発に適用できる動作合成ツール

既存のHDL設計資産,IPと容易にインターフェース可能

- DRAMコントローラ,DMAC,PCIインターフェース等

業界標準の標準言語による入力/出力

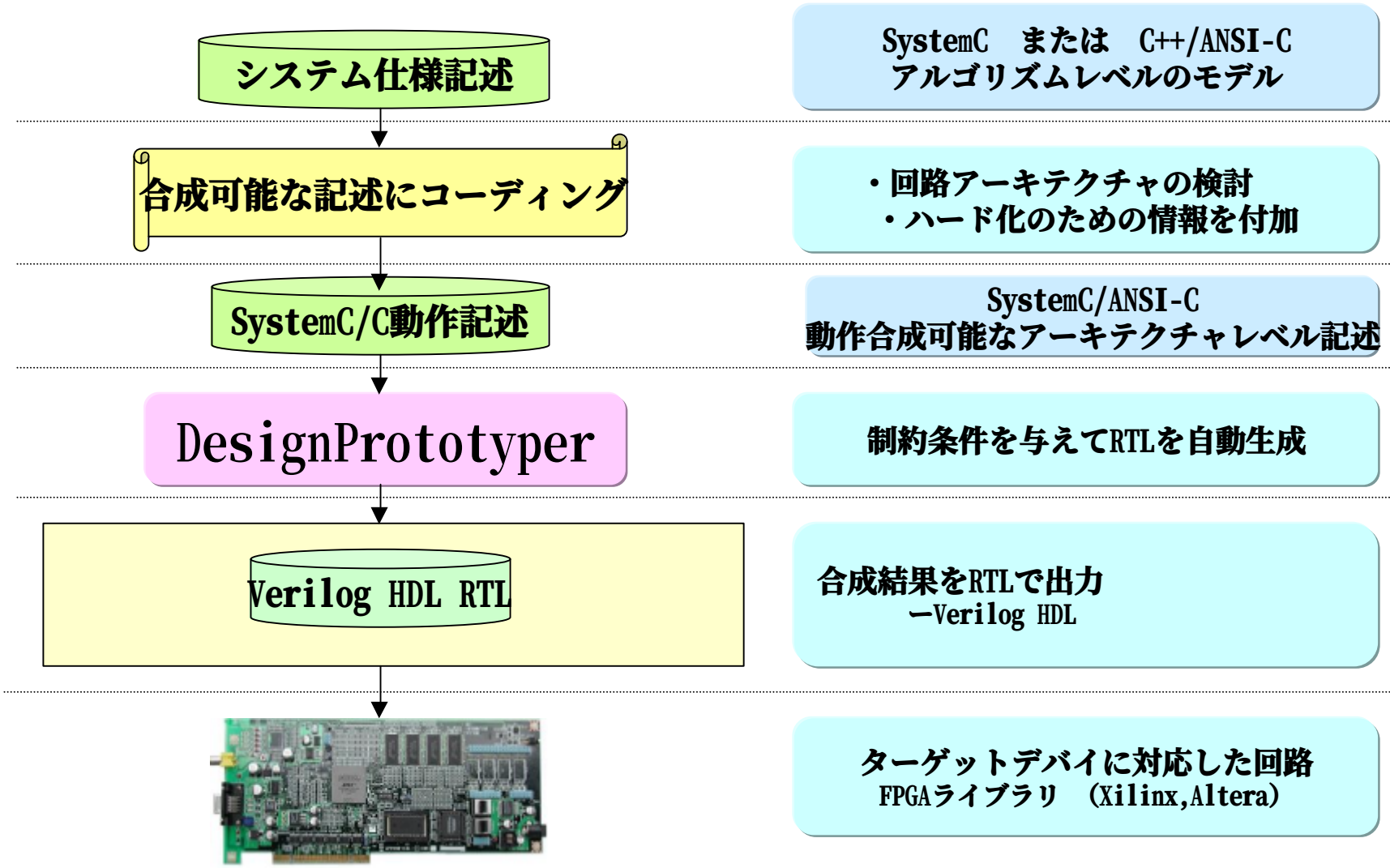
－入力形式 ANSI-C / SystemC

－出力形式 論理合成可能なVerilog HDL

ユーザが読解可能なRTLを生成

－拡張FSM形式の論理合成可能なRTL記述

DesignPrototyperの設計フロー



アルゴリズムレベルのCモデルを簡単にFPGA化

入力言語

SystemC

◆SystemC 2.0.1をサポート

◆入力可能な抽象モデルはBCA
(演算部分はアンタイムドモデル)

◆SC_CTHREAD

★なぜSystemCなのか？

－SystemCは既に国内標準
－ソフトとハードの協調設計・検証に対し有効
－SystemCの環境で検証が可能
(SystemC,HDL混在検証も可能)

ANSI-C

◆ANSI-Cフルセットサポート
構文レベルの拡張は一切行わない

◆ハード化に必要な情報は、コメントまたは
関数を用いて記述

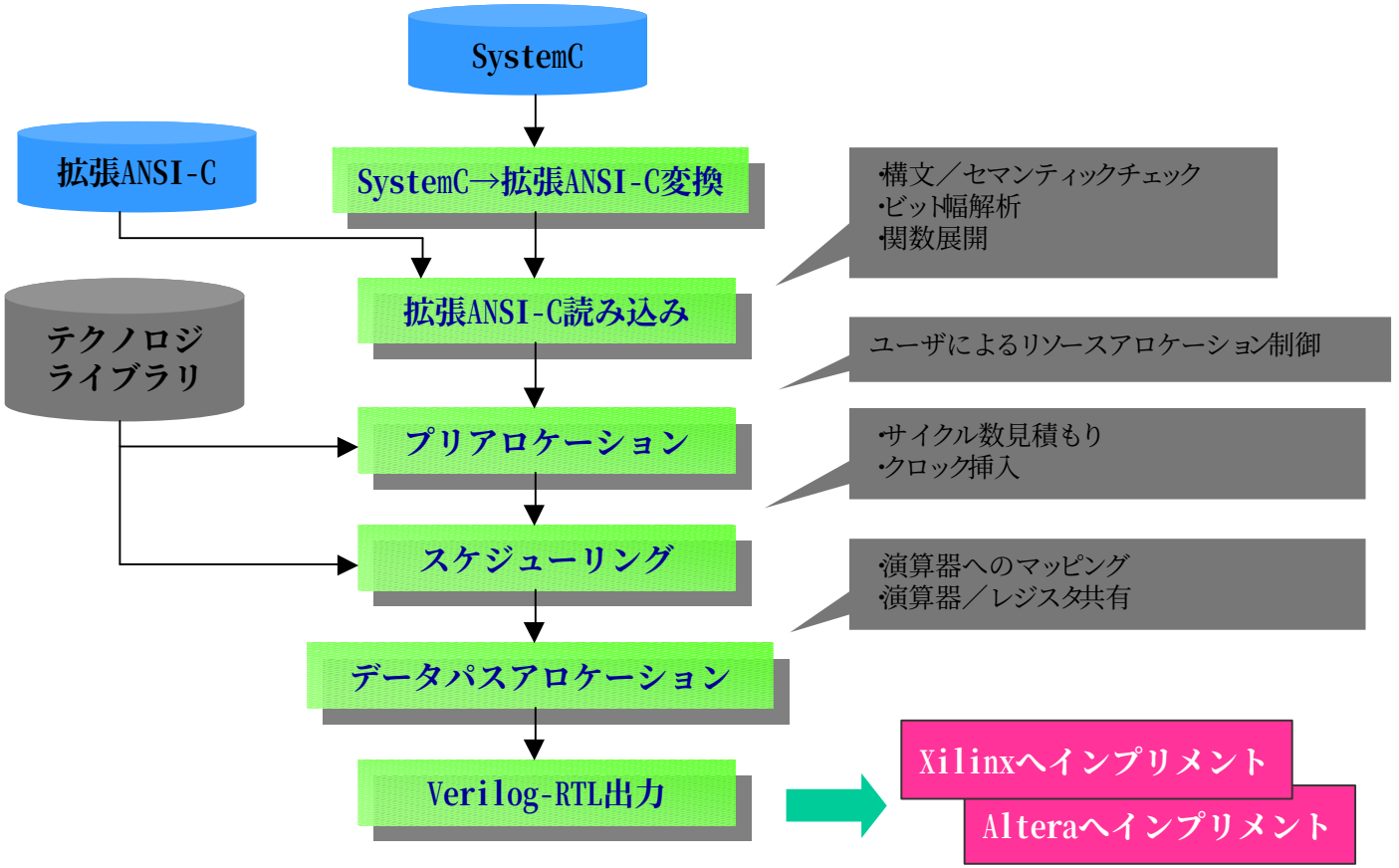
◆既存のCコンパイラで機能検証が可能

★なぜANSI-Cなのか？

－アルゴリズムやソフト資産の殆どはANSI-C
－ハードウェア設計者に対する敷居が低い
－ソフト開発とシームレスに繋がる

ユーザのニーズに合わせ、SystemC/ANSI-C双方のパスを提供

DesignPrototyperの内部処理



Verilog HDL to C++ Translator *Verilator*について

- オープンソースのVerilog HDLの高速シミュレータ
  - Verilog HDLのコードをC++/SystemCへ変換して高速にシミュレーションを実行
- 作者
  - *Wilson Snyder , Paul Wasson and Duane Galbi*
- ライセンス
  - GPL
- http://www.veripool.com/Verilator.html

Verilog HDL to C++ Translator *Verilator*について

http://www.veripool.com/verilator.html

“Do not download this program if you are expecting a full featured replacement for NC-Verilog, VCS or another commercial Verilog simulator for a little project! Don't get it if you expect a corporate support organization. However, if you are looking for a path to migrate synthesizable Verilog to C++ or SystemC, and writing just a touch of C code and Makefiles doesn't scare you off, this is the free Verilog compiler for you.”

VerilatorによるVerilog HDLからSystemCへの変換

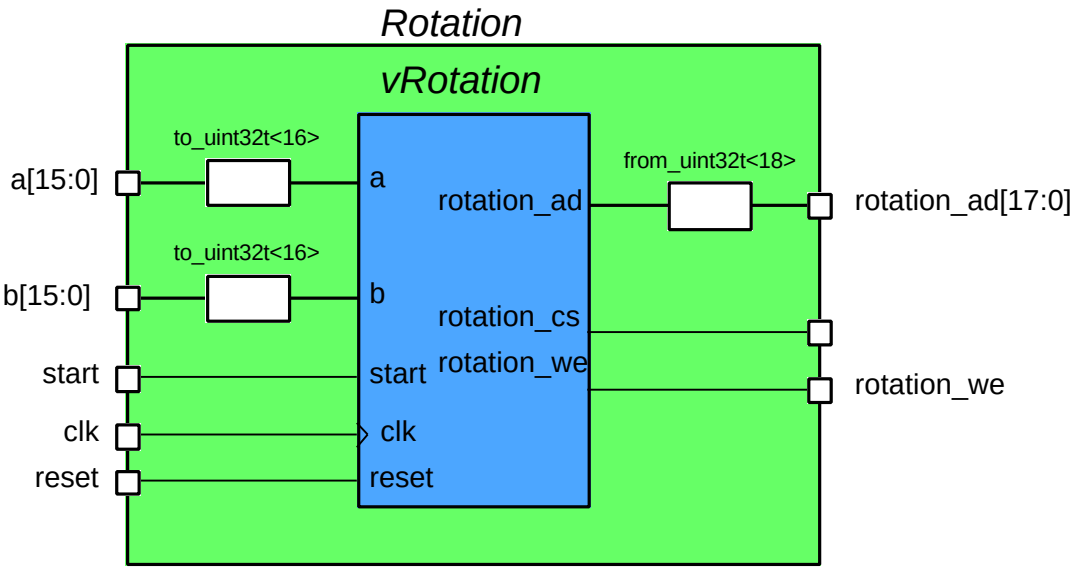
VerilatorによりVerilog HDLをSystemCに変換します。

```
% verilator.csh  --sc  Rotation.v
```

Veriratorにより変換されたSystemCのRTL記述およびラッパモジュールが生成されます。

```
./obj_dir/VRotation.h
./obj_dir/VRotation.cpp
```

```
./obj_dir/Rotation.h
./obj_dir/Rotation.cpp
```



VeliratorでSystemCのモジュールを生成した場合、2bitから32bitのポートは uint32_t 型 になります。

bit幅を変換するためのラッパーモジュール

Rotationを作成することにより,SystemCで記述された上位階層にインスタンスすることが可能になります。

Verilatorの制限事項

- SystemCとVerilog HDLの記述を混在したデザインにおける制限事項を以下に示します。
  - Verilog HDLのモジュールの下位の階層にSystemCのモジュールをインスタンスできません。
  - Verilog HDLの記述でサポートされている記述は論理合成可能なRTL記述のみです。
    - Task等は不可
  - Verilog HDLの記述において時間の遅延はδ遅延として評価されます。
  - プロテクトされたIPはSystemCに変換できません。

SystemC/Verilatorによる機能検証のメリット

- SystemC/Verilatorによる検証環境のメリット
  - 実画像でもシミュレーションが可能
  - 画像の可視化が可能
    - 幅広いソフトウェアの資産が活用可能
    - PLIによる接続は不要
  - BCAレベルとRTLで同一のテスベンチ、シミュレーションモデルを使用可能
  - 既存のVerilog HDLの設計資産の活用が可能

□プロジェクトの目的

□要求性能、仕様

□方式検討

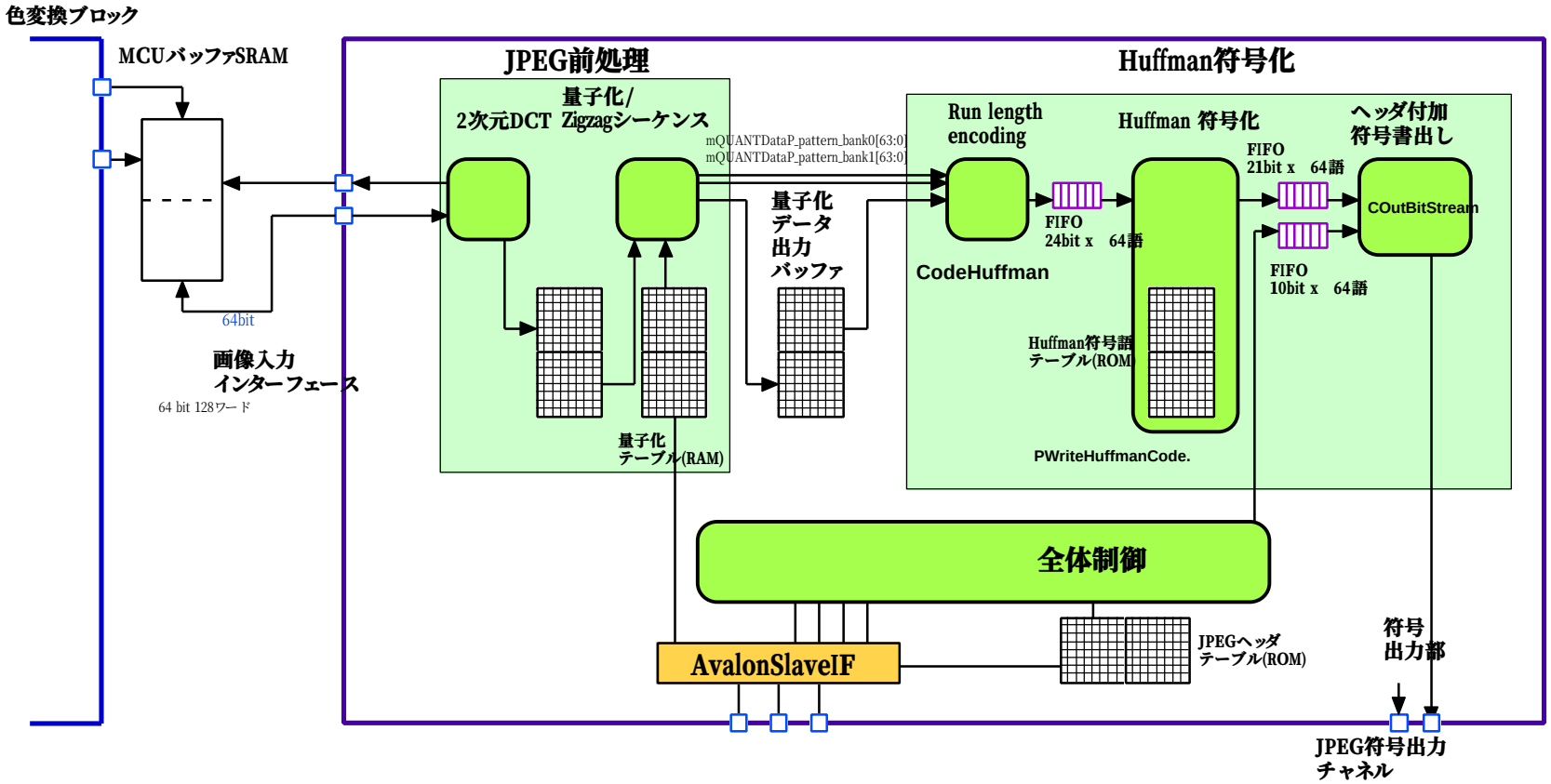
□設計検証評価環境

■ Altera社FPGAへの実装

□デモンストレーション

□まとめ

JPEGエンコーダコアのAltera社Cyclone2への実装



Cyclone II EP2C35F -8	
Total logic elements :	9,161 / 33,216 (28 %)
Total registers :	5773
Total memory bits :	31,488 / 483,840 (7 %)
Embedded Multiplier 9-bit elements :	42 / 70 (60 %)
FMAX	93.25 MHz (period = 10.724 ns)

要求性能
256万画素
16フレーム/秒
を達成

□プロジェクトの目的

□要求性能、仕様

□方式検討

□設計検証評価環境

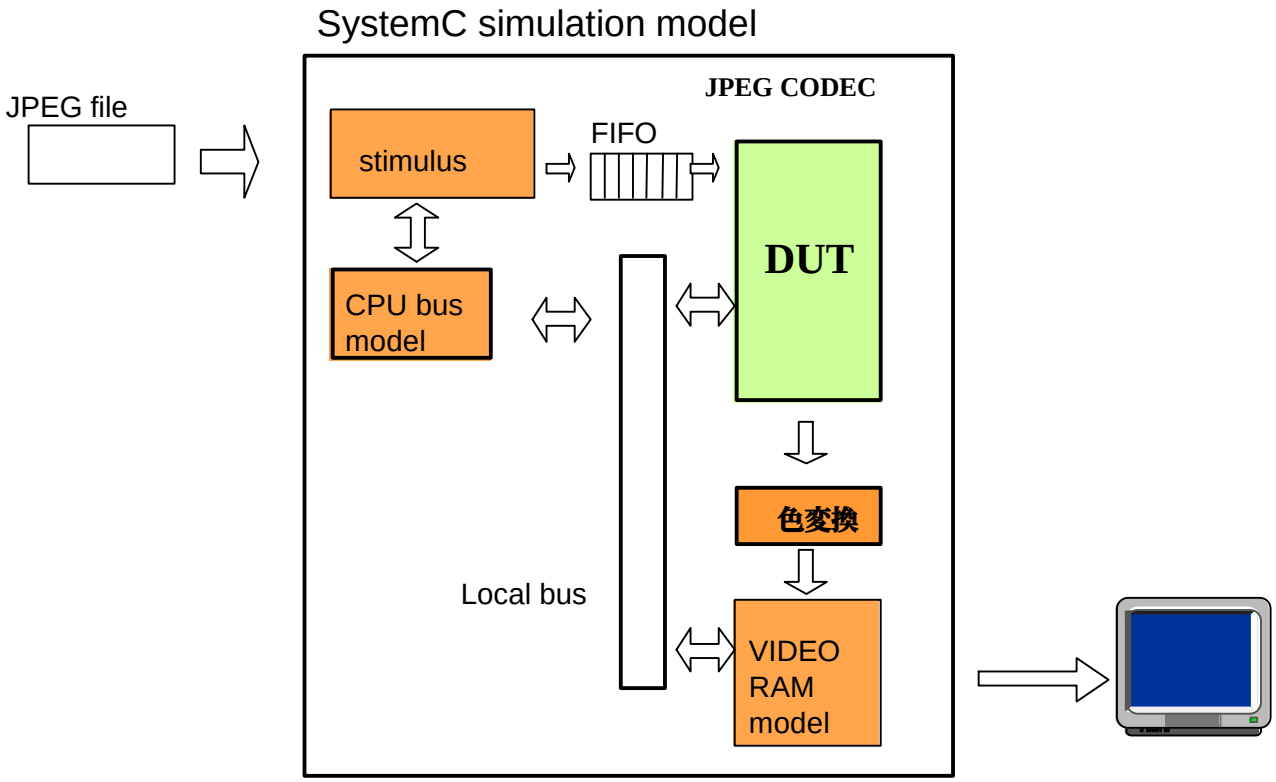
□Altera社FPGAへの実装結果

■デモンストレーション

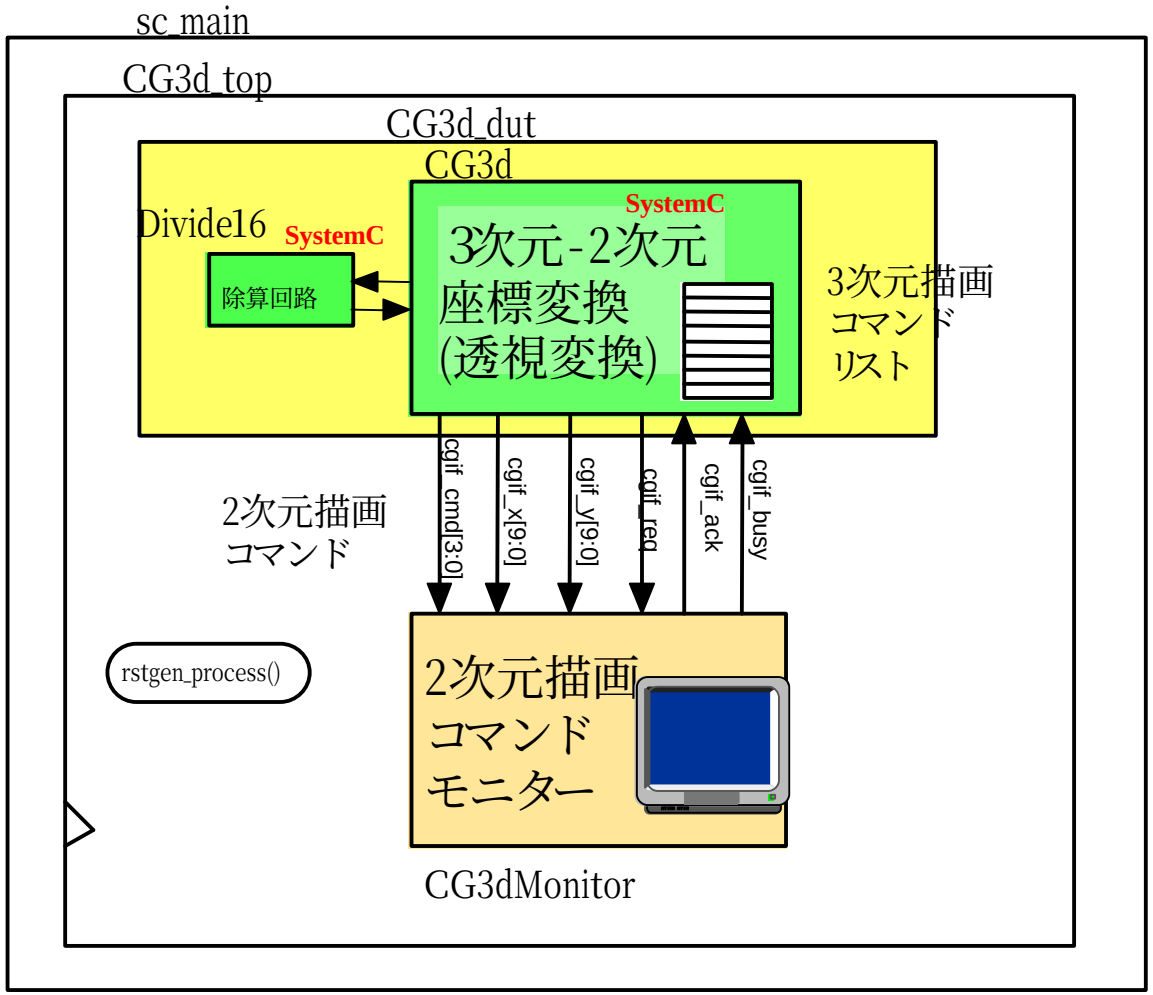
□まとめ

デモンストレーション　JPEG復号

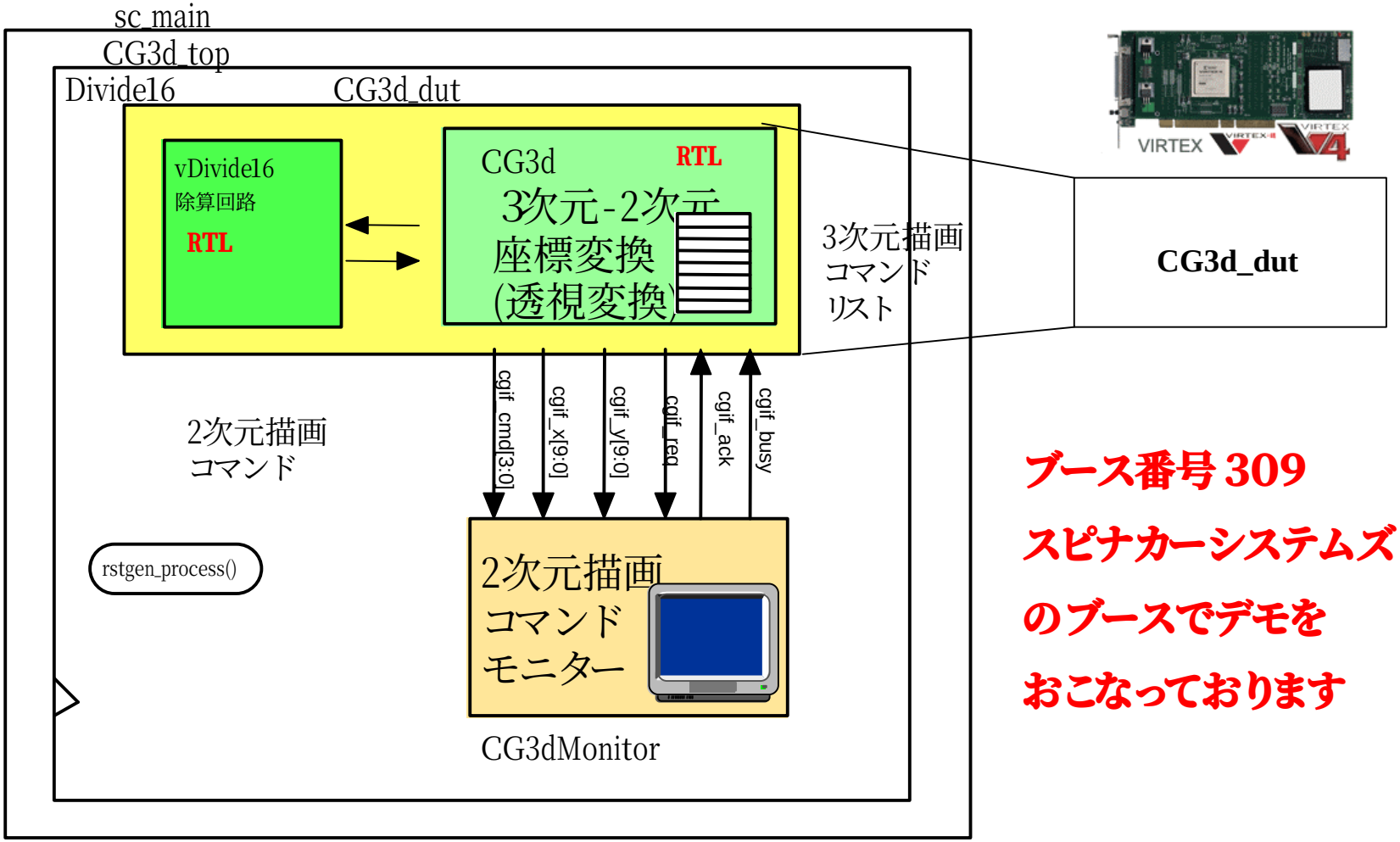
SystemC BCA/Verilog RTL/ ファームウェア(ANSI-C)混在デザイン



デモンストレーション3次元グラフィックス
単体機能検証(BCAレベル)



デモンストレーション3次元グラフィックス
iPROVEによりCG3_dutをFPGAに実装して検証



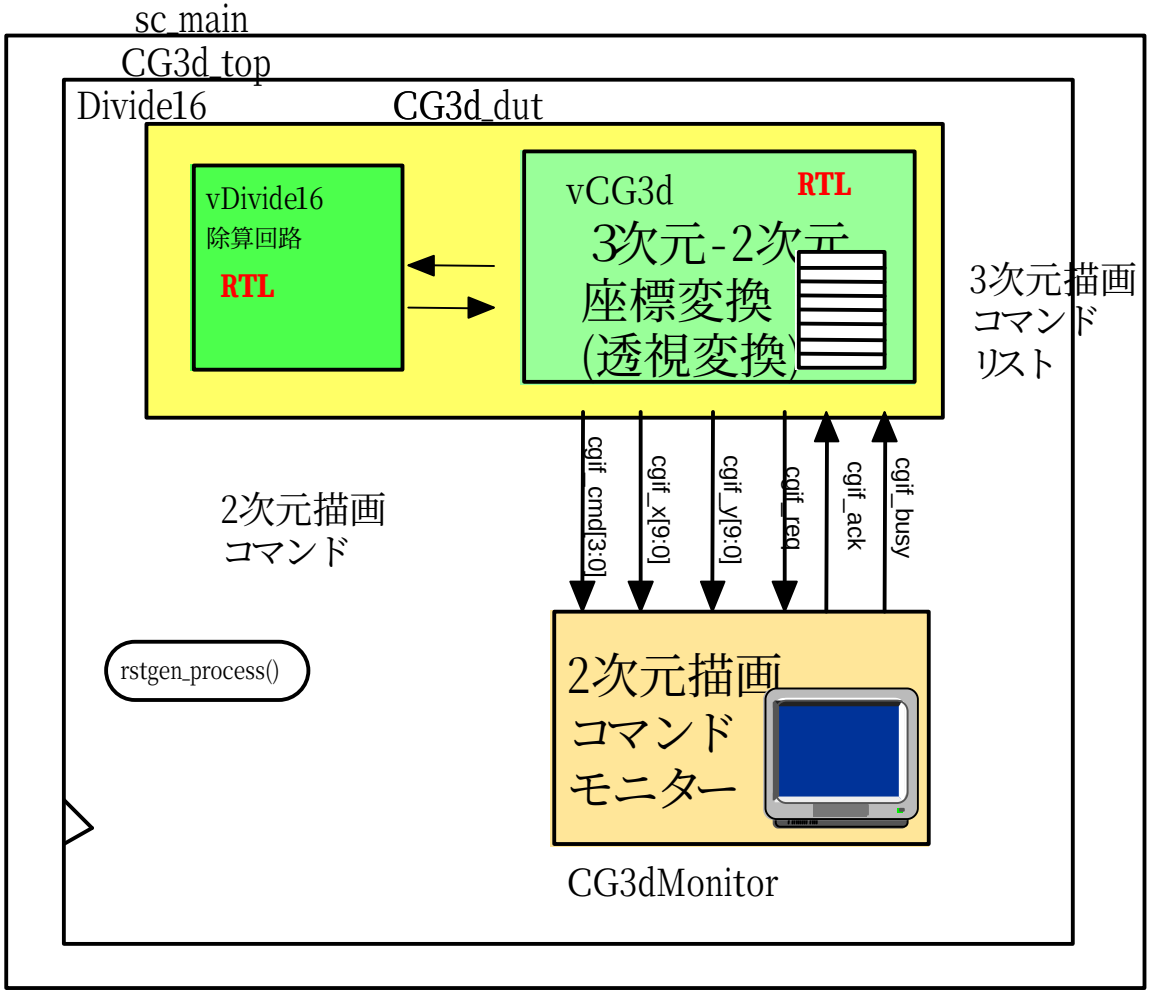
ブース番号 309

スピナカーシステムズ

のブースでデモを

おこなっております

デモンストレーション3次元グラフィックス
動作合成で生成されたRTLの単体機能検証



□プロジェクトの目的

□要求性能、仕様

□方式検討

□設計検証評価環境

□Altera社FPGAへの実装結果

□デモンストレーション

■まとめ

まとめ

- 動作合成ツール、SystemC/C++/ANSI-Cはハードウェア化の際の実装用のアルゴリズム、方式を検討するための道具として非常に有用
  - “実行可能なモデル”
- 動作記述,アルゴリズム記述により段階的な実装が可能
  - 最初からすべてRTLを用意するのは大変
- オープンソースのSystemC/Verilog RTL混在シミュレーション環境により、ANSI-C,C++/SystemC,Verilog RTLを混在して機能レベルの検証が可能
- ハードウェアアクセレータPROVEにより、ブロック単位にハードウェア化し、SystemC環境でC/C++/Verilog RTLとの混在検証が可能

有限会社テクノレポ　会社概要

- 有限会社テクノレポの事業内容
  - DesignPrototyperのユーザサポート
  - DesignPrototyperの改良に関する企画策定
  - C言語/SystemCによるハードウェア設計コンサルティング

- DesignPrototyper関連機能の改良
  - SystemC -> FDA-Cコンバータ 改良版
    - (2007年 3月 リリース予定)
  - SystemC 接続記述 -> Verilog HDL接続記述 トランスレータ
    - (version 3.5.2c 2006年 7月 リリース済)
  - 自動並列化機能の研究

- 開発実績
  - C言語/SystemCによるハードウェア実装設計検証セミナー
  - ローコスト量産向けFPGA用　Motion JPEGエンコーダ コア
  - 受託開発
    - USBコントローラ　SystemC TLM
    - SystemC上で稼働するμITRON互換 モデル

お問い合わせ先

動作合成ツールDesignPrototyper およびハードソフト混在検証環境,iPROVE/iNCITE に関する製品,お問い合わせ先

【DesignPrototyper販売代理店】

ブース番号 309

株式会社スピナカー・システムズ
〒222-0033 横浜市港北区新横浜2-3-12 新横浜スクエアビル11F
TEL :045-478-3803 FAX :045-478-3809 Email :info1@spinnaker.co.jp
http://www.spinnaker.co.jp

本セミナーに関する技術なご質問
動作合成ツールDesignPrototyperに関する技術関連のお問い合わせ先

【DesignPrototyper開発販売サポート,,ハードウェア設計コンサルティング】

有限会社 テクノレポ
Techno Repo
〒240-0023 横浜市保土ヶ谷区岩井町355-2
TEL：070-6611-6193 FAX：045-731-5392 E-Mail info@techno-repo.com
URL http://www.techno-repo.com

VerilatorによるSystemC/Verilog RTL混在検証環境の構築方法,
ラッパモジュール生成スクリプト,サンプルデザイン等のアーカイブは
弊社Webページにて下記のURLで、ご提供いたします。
(2月 1日より 公開予定)

http://www.techno-repo.com/doc　　C言語設計関連情報