SystemCを利用した組込みシステム開発事例

大日本印刷株式会社　電子デバイス研究所
梅海　勝浩、大澤　伊作

１．はじめに

大日本印刷では、ブラザーカンパニーである株式会社DNP エル・エス・アイ・デザイン（DLD）を通じて、お客様に設計サービスを展開している。電子デバイス研究所は、差別化技術開発・技術動向調査およびDLD業容拡大の観点から、主に独自IP開発と、LSI設計・評価技術、設計インフラの開発を行なっている。

数年前よりSoC型チップ設計が増加し、開発したIPをARM/MIPS/SuperH/MePに代表されるプラットフォームに組み込み、バス・インターフェイス用に最適化する取り組みと、設計サービスにおける顧客のデータ・エントリーを従来のRTL/NetlistからC/C++/SystemCに引き上げるための設計サービスフローの拡張を行なってきた。

2002年より本格的にシステム開発に着手し、これまでにMatlabを使ったアルゴリズム解析、システム言語SystemCを用いた設計、EDAツール導入によるHW/SW協調設計・検証環境の構築、並びに既存設計フローへの導入を行なってきた。本稿では、上記設計フローを用いて社内IPをハードウェア化した事例について、SystemC設計に関わる部分を中心に紹介する。

２．システム言語SystemCと開発環境

システム言語としてSystemCを選択した理由として、標準化団体であるOpen SystemC Initiative（OSCI）のサポート、言語のモデリングカバレッジ、SystemC Verification（SCV）Class Libraryの提供による検証手法のサポート、EDAベンダによる各種ツールのサポート体制を挙げることができる。

一般的なシステム言語を用いたシステム設計・検証フローを図１に示す。弊社では、システム／アルゴリズム設計においてはMathworks社のMatlab、プラットフォーム設計・検証においてはMentor Graphics社のSeamlessを用いている。またアーキテクチャ設計においては主に社内ツールを用いており、特にSystemCによる開発・検証においてはCATS社のXModelink SystemC Debuggerを適用している。HW仕様書からの論理設計は既存の設計フローを使っており、SW仕様書からのSW開発は社内のグループと共同で開発している。

３．SystemCに適用した社内IP

今回、上記設計フローを適用してハードウェア化した社内IPは、既存のCMOSイメージセンサ回路を高付加価値化するために独自開発したセンサIPである。

このIPの特徴は、画素回路そのものの高機能化ではなく、負帰還を用いたリセット方式にある。これによりフォトダイオードのリセット時に問題となる出力MOSFETの閾値バラツキをキャンセルすることができるだけでなく、中間値の非破壊読み出しを可能にし、機能的な画像処理へ応用することができる。

効果的な画像処理の一例として、画素アレイを制御して広ダイナミックレンジ化を行う処理の実装を行なった[1]。この処理では、画素を行列と見なし、一定の時間間隔で行選択を行ない、さらに各画素毎に飽和予想／リセット／データ読み出しを行なう。図２にシステムブロック、図３にタイミングチャートを示す。

---

1 電子情報通信学会 信学技法, ICD2003-83, Sept.2002

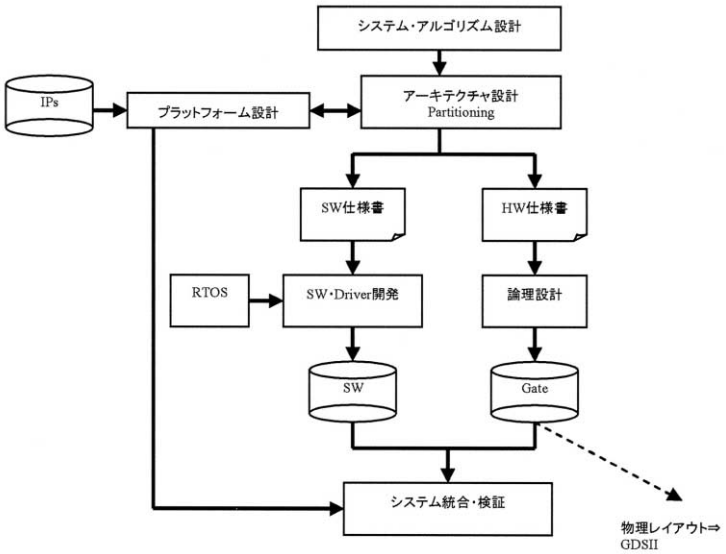

図1　システム設計・検証フロー

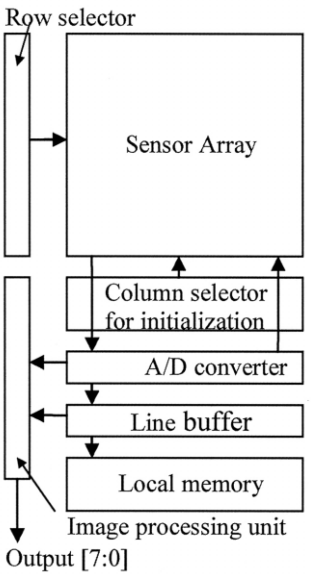

図2　システム・ブロック図

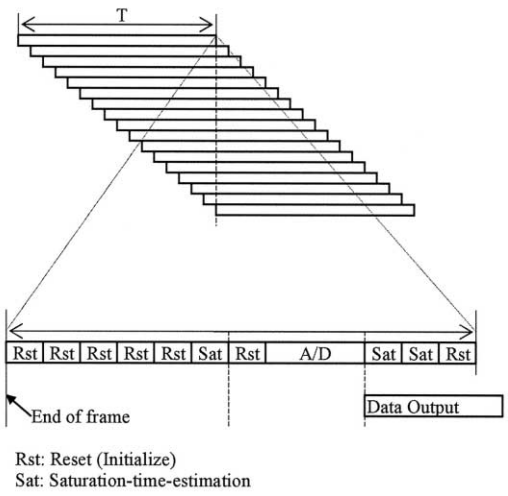

図3　タイミング・チャート

本センサ回路のアルゴリズム（特に圧縮計算）検討は、Matlabを用いて行なった。Matlabは、画像発生・データ検証にも利用した。

アルゴリズム検討を終えたMatlabモデルをC++モデルに変換し、整数演算化して精度検証を行なった。その後SystemCモデルへ変換し、図２／図３に示すアーキテクチャ検討を行なった。

以下、C++モデルからSystemCモデルへの変換について、詳細を述べる。

４．SystemCへの実装

Untimedモデルへの変換

まず、C++で記述されたセンサモデルを、SystemCのUntimedモデルに置き換えた（図４）。センサモデル部のみがハードウェアへのターゲット回路であり、この後RTL化への対象となる。入力画像データ発生と、センサ出力のモニタ・検証を行なうモジュールは、Regression Test用のゴールデン・モデルとして、これ以降も活用することができる。

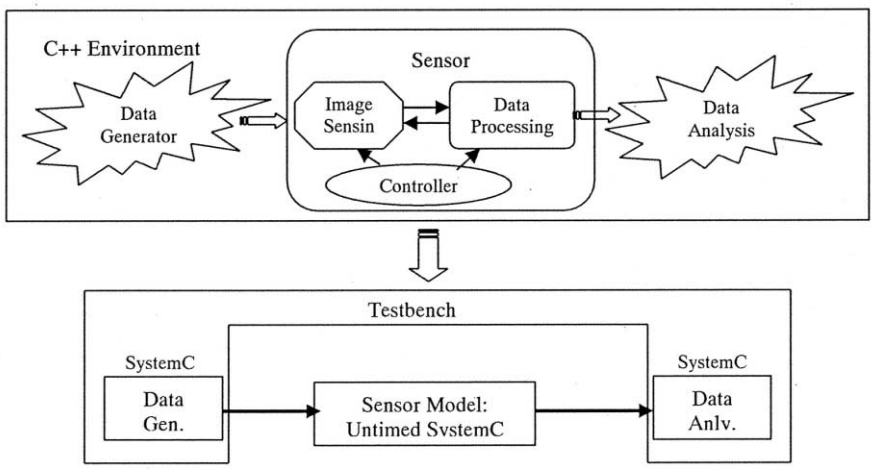

図４　SystemCへ変換

Timedモデルへの変換

次に、ハードウェア化の対象になるセンサ部を、UntimedモデルからTimedモデルに変換した（図５）。ここでは、画素アレイの遅延値を数量化したものをタイミングモデルとして入れることで、アーキテクチャの最適化を行なっている。なお、プロセス間通信にはFIFOを用い、配列部の処理は簡易メモリモデルを適用した。

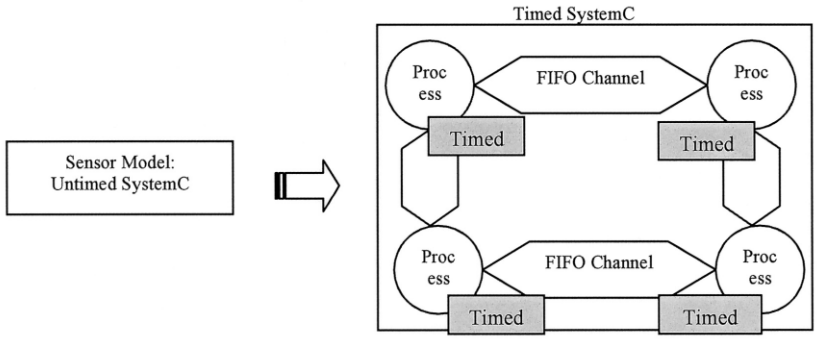

図５　UntimedモデルからTimedモデルへ

Cycle Accurateモデルへの変換

最後に、TimedモデルをCycle Accurate（CA）モデルに変換した。ここでは、後述するプロトタイプ・ボードでの検証用に、ターゲット・ハードウェアをAltera社のFPGAとした。Timedモデルで抽象化していたメモリモデルはAltera社のメモリリソースMegafunctionに沿ったビヘイビアに変更した（使用したメモリ・モデルはFIFO、同期メモリ）。

変換作業のまとめ

C/C++モデルからSystemC Untimedモデルへの変換は、抽象度に大きな差がないため問題なく完了した。

Timedモデルでは、ハードウェア特有のビヘイビアである複数プロセスの並列動作のモニタ・デバッグが不可欠となる。SystemC環境を導入した当初、開発環境のデバッグ機能が十分でなく、実作業において困難をきたす状況が容易に想像された。これを改善するため、CATS社のXModelink SystemC Debuggerを導入している。SystemC Debuggerでは、並列イベントのモニタ機能に加え、ウオッチ機能による変数の管理が容易にできるため、デバッグ環境を大幅に改善することができた。

Cycle Accurateモデルでは、ターゲットとなるFPGAのリソースを最小限にして高パフォーマンスを実現するための検討がキーであった。

最後に抽象度によるソースコード量とシミュレーション時間の違いを示す。

	コード量（Untimedの場合を1）	シミュレーション時間（Untimedの場合を1）
Timedモデル	5.2倍	7倍
CAモデル	6.5倍	20倍

ソースコード量の違いは可読性・デバッグ性に影響する。シミュレーション時間を見ると、抽象度が低い場合での高速性を活かして、上位レベルでの強固な検証を行うことが有効であることがわかる。

5．プロトタイプによるシステム検証

作成した上記Cycle Accurateモデルを、ARM社のシステム検証プラットフォームIntegratorに実装して動作検証を行なった。センサ部はIntegratorのFPGAに、テストベンチ部はARMプロセッサに実装した。この環境ではAMBA-AHBバスのインターフェイスが必要となるため、FPGAにこれを追加した（図６）。バス・インターフェイスがARMからの画像データを受け取り、センサにデータを渡し、センサからの出力値をARMプロセッサに転送する。ARMプロセッサにて受け取ったデータの整合性を検証する。

バス・インターフェイスの開発では、Transaction Level Model（TLM）を用いてAMBA-AHBのビヘイビアをモデル化した。TLMでは、インターフェイスのメソッドをライブラリとしてコールするだけでモジュール間通信ができる（図７）。図７ではCPUモデルのテストベンチはトランザクション・モデル（TLM）のインタフェース関数read/writeを用いてトランザクション・モデルとコミュニケーションを行なう。トランザクション・モデルはインターフェイス関数に従って通信プロトコルを発生させ、ターゲットのテスト・デバイスと信号の送受信を行なう。この動作は、図６におけるBUS I/Fに該当する。

トランザクション・モデルを使った通信は、インターフェイスに定義されたコール関数を使うので、例えば関数名とその引数が同じであれ

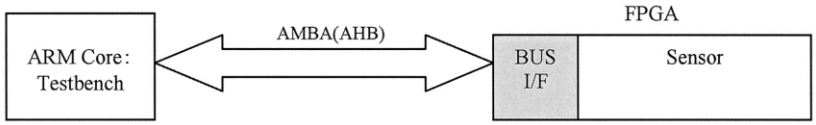

図６　プロトタイプ・システムの概要

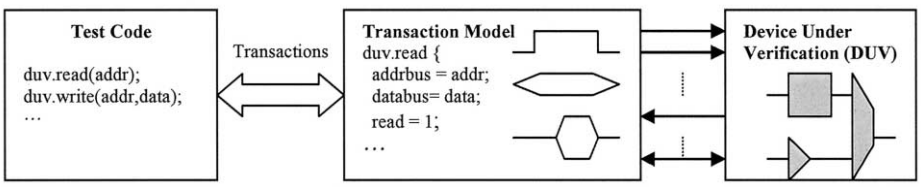

図７　トランザクション・モデルを使った検証

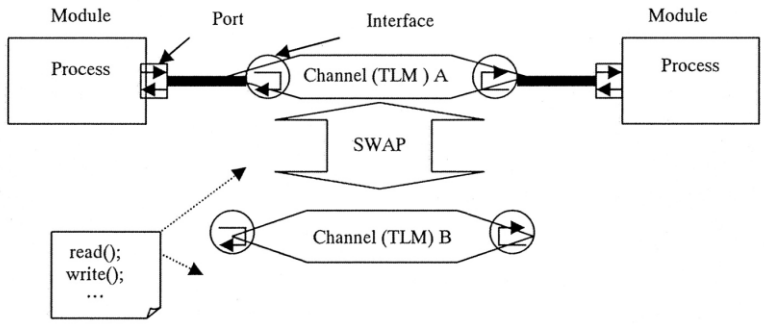

図８　トランザクション・モデルを使ったバス・チャネルの変更

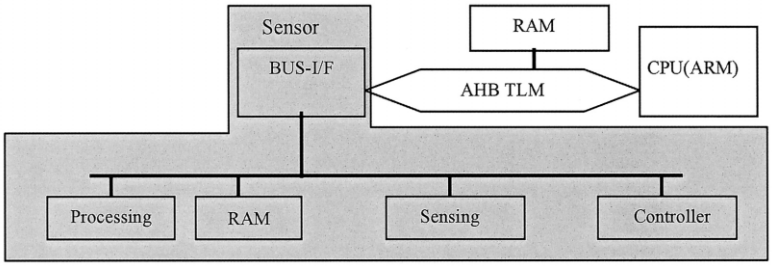

図９　プロトタイプ・システムのブロック・ダイアグラム

ば、異種のバス・モデルへの置き換えも容易にでき、アーキテクチャ検討に有効である（図８）。図８は双方のモデルにインターフェイスを提供している例で、右側のモジュールとのインターフェイスは信号でも可能である（図７を参照）。図９にプロトタイプ・システムのブロック・ダイアグラムを示す。

６．RTL化への取り組み

取り組みの最後として、SystemCからRTLへの変換について触れる。弊社では以前より、C言語からのRTL合成ツールとして、礎デザインオートメーション社のDesign Prototyper（DPT）を設計フローに取り入れている。DPT v3.4ではSystemCからのRTL合成がサポートされており、今回は評価を兼ねてDPTを適用した。

DPT v3.4でSystemCからのRTL合成を行なう場合、現状ではいくつか制約条件がある。主なものを挙げると

・プロセスタイプはSC_CTHREADのみ
・合成は１モジュール単位のみ

これらの制約により、CAモデルのモジュール分割と、それに伴うソースコードの書き換え、およびトップ・モジュールでの組み上げ作業が必要となった。

代表的なモジュールについてRTL合成後のソース・コード量を見てみると、SystemC記述に比較して3倍強になっており、高位合成フローの導入によって記述量の効率化が図られていることがわかる（RTLの場合ALTERA-Megafunctionは含んでいない）。

	SystemC記述	RTL（VerilogHDL）
ahb_if	134行/4kB	435行/19kB
Controller	99行/3kB	329行/14kB

７．成果

C++モデルからSystemCモデルへ変換を行い、RTL合成まで行なった。上位から抽象度を下げる作業においてアーキテクチャの検討を十分に行なうことができた。この過程でのポイントは、C++における配列をターゲットFPGA用の最適なメモリ・モデルに割り当て回路を最適化する作業（図10）と、ハードウェア特有の並列動作のデバッグ

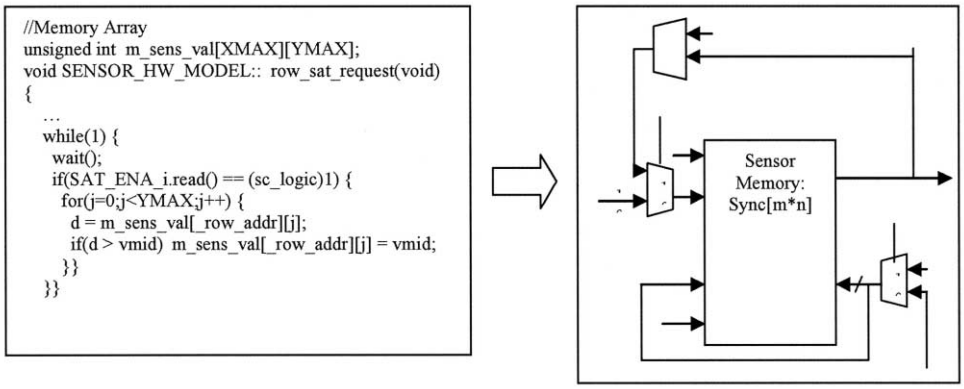

図10　配列（Untimedレベル）からCAモデル化

であった。

また、プロトタイプ・システムの開発において、特にSystemCのTLMの有効性を確認した。AMBA-AHBの動作を忠実に再現するためのモデル作成は時間もかかり容易ではなかったが、TLMを利用することでプロトタイプ・システムを作る前にソフトウェア環境でシステムをトータルで最適化することができた。この手法は、設計の早期に機能バグを見つけることができ有用である。

8．まとめ

社内のソフトIPであったイメージセンサIPをSystemC設計フローに沿ってハードIP化した。作成したIPはARMのバス・インターフェイスに対応させ、システム検証プラットフォームへ適用することができた。また、特にTLMモデルの作成においてノウハウを得ることができた。

同時に、SystemC設計フロー（プロトタイプ・システムへのパスも含む）を用いることで、他のソフトIPをハードIP化する目処がつき、顧客からのシステム設計受けの要望に応える体制が整ってきた。

今後の課題として、MatlabとSystemC環境のリンクによる検証環境の強化が挙げられる。RTL合成ツールのさらなる機能改善も今後に期待したい。また、SCVの取り入れによる強固な検証モデル作成も開始しており、設計フローに早期に取り入れたいと考えている。