オリジナルDSPの開発環境構築の事例研究

後閑 博　山口 啓太
三菱電機(株)

1　はじめに

現在さまざまなC言語ベースでのH/W設計が取り組みはじめられている。今回、敢えてその中で高位合成というものを意識せずH/W設計者がその経験と知識をフルに生かせるC言語ベースでのH/W, S/W設計への取り組みというものを考えてみたい。

本稿では現在スタンダードとなっているシステムC言語開発環境の問題点を挙げ、よりポータブルで使いやすいオリジナルのC言語環境を提案しその狙いと、それによりアプリケーション指向型のDSPコアの開発と実際の産業用コントローラへの適用を述べる。

2　C言語ベース設計

2.1　既存のC言語ベース設計の問題点

現在スタンダードとして使用されているシステム設計用C言語としてSystemCとSpec-Cなどがある。これらには特徴的な大きな違いがあり同一視すべきではないが、両方共通に見出される問題点を考察する。

(1) 信号処理データフローと実際のプロセッサを介した処理のデータの流れはまったく異なるものであり、信号処理データフローをあらわすシステム記述は実際の"システム"ではない。現実の処理は信号処理の数式どおりに演算器を配置するわけにいかず、複数の演算器とそれへデータの入出力を行うスケジューラからなるプロセッサとプロセッサとメモリ、または複数のプロセッサ間でデータの入出力を行うDMACや通信IOなどからなるハードウェアとその上のソフトウェアから成る。それらはシステム記述というものとはまったく異なるものである。

(2) 現時点では、高位合成能力が十分ではないため、それを無理に使おうとするのはあまり得策ではない。

(3) Cベースのシステムレベルやハードウェアレベルの各々のレベルの記述は理解し記述することができても、システム全体を動かそうとする場合、様々なレベルのオブジェクト指向的記述と関数の引数(=概念的ワイヤ記述)とハードウェアの引数(=物理的ワイヤ記述)の塊の中で自縄自縛に陥る。

2.2　オリジナルC言語ベースへの要求仕様

今回のLSIとそのソフト開発のためのオリジナルのC言語の開発のポイントを述べる。

Poc-C言語:今回開発したC記述(**Po**rtable **C**locked **C** Language)

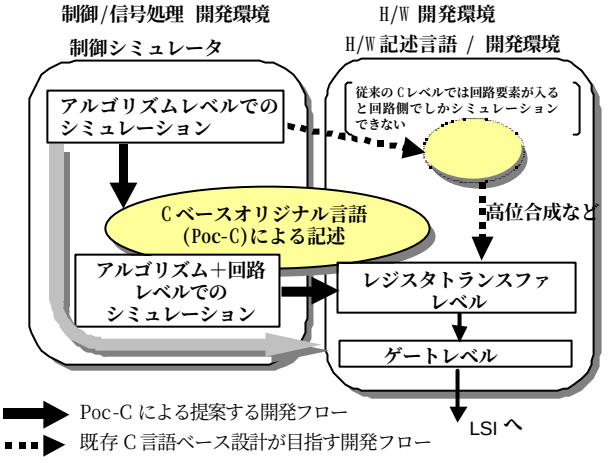

図1. 新しく提案する開発フローとPoc-C言語の関係

単純に(a)ハードウェアと等価なC記述があり、それがPCでもEWSにでもあるような(b)デバッガでデバッグすることでハードウェア設計ができ、かつ、汎用的な信号処理シミュレータの専用コンパイラでもコンパイルできて、(c)容易にリンクしてソフト込みで(d)高速にシミュレーションできるようなポータブルで使いまわしの良いC言語記述が必要であった。

従来のC言語ではアルゴリズム/ビヘイビアレベルの記述では上記は可能であるが、どうしてもクロックの概念を導入して、ソフトとハードでのシミュレーションを考えるとき、クロックを動かしながらの汎用的な信号処理シミュレータとのリンクは不可能ではないが、容易に実現しづらい。また、単にC言語で書けるといってもシステム屋にとってはハード設計専用の特別仕様のC言語環境には魅力なく、ハード設計者側の視点にしか立っていない。

図1に述べるH/W設計環境において他の言語が点線の

Design Methodology and Implementation of the original DSP
Hiroshi GOKAN, Keita YAMAGUCHI
Mitsubishi Electric Corporation
8-1-1, Tsukaguchi-honmachi, Amagasaki, Hyogo 661-8661, Japan

矢印に沿ったフローであるのに対し、ここでは黒矢印の回路もソフトも含めて信号処理の開発環境で評価することを経由することで設計の効率化を狙うものである。

2.3　オリジナル C 言語(Poc-C)の記述と方法論

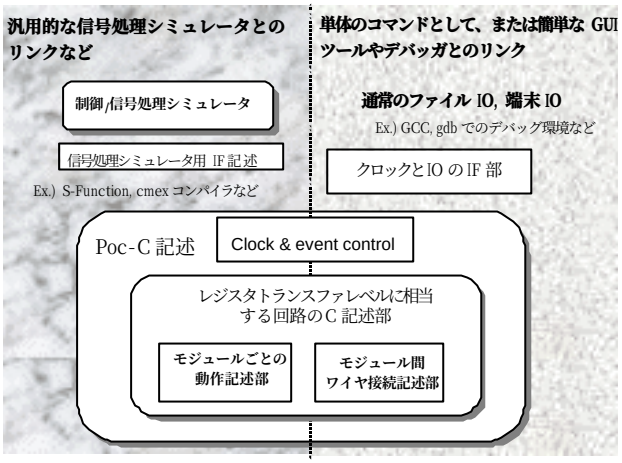

図 2. Poc-C 記述の階層構成

今回提案するオリジナルの C 言語は基本的に HDL のレジスタトランスファレベルに相当し、ルールに沿った回路モジュールとワイヤ接続を記述する。それと短いクロックコントロール部分とどこにでもある C コンパイラでコンパイル、リンクを行う。今回は RedHat Linux 7.0.1 上の GNU-C/gdb にて回路とソフト込みで基本動作の開発とデバッグを行った。

汎用的な信号処理シミュレータとして MathWorks 社の Matlab™を使用し、その C 言語リンクとして同ツール上の"cmex"コマンドによって Poc-C 言語記述部ごと全体をコンパイルして"S-function"で"Simulink"とインタフェースを取って"discrete 時間"でシミュレーションを実行させた。

3　オリジナル C 言語での DSP 設計適用

3.1　開発した LSI とコンパイラ

モータ制御用プロセッサで演算は一次の IIR 型フィルタ演算を多く使用し、一つの演算結果が次の演算や先の演算に用いる依存性が高い演算が特徴となる。汎用の高速 DSP だと局所的な演算は非常に速いが実際のプログラムとなるとメモリへの格納と読み出しを多くランダム的に使うことになり単純に期待通りには速度が上がらない。

短時間でランダムにアクセス可能なメモリを複数用意し、逆にレジスタ群を最小構成としてハードをコンパクトにすることや他との通信バッファ回路を高速のカスタム仕様とすることで、ハンドシェイクなどのソフト処理を簡素化した。消費電力の低減、レイアウト配線の収束性よくメモリアクセス速度が必要以上に長くなることを防ぐために、基本アーキテクチャのレジスタ構成とレジスタ間の配線はシンプルかつ単純化し、ゲート規模は小さく納め、動作周波数も下げることが求められた。加え処理上のハードウェアのフラグメントの処理省いて、専用のコンパイラ側で吸収しハードは不必要に処理を多くせず出来る限り規模を抑制した。

3.2　開発した LSI

開発した LSI
総命令数 42 種　ゲート規模 55KG　動作周波数 50MHz
32bit での加減算、16bit x 16bit 乗算器、limit など特殊命令あり

演算速度…三相誘導モータ制御(ベクトル制御演算)
20usec 未満で実行

試作ボード(部分拡大写真)

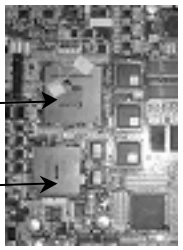

図 3. 開発した LSI と試作ボード

LSI の設計においては C による直接デバッグが主体で HDL のシミュレータは動作確認や記述の一致性の確認でしか使用していない。HDL に落とした後一致をチェックして直接 FPGA に焼いて実機の動作確認を行った（図3）。産業用コントローラとしての製品化をめざし ASIC 化した。

演算速度に関しても初期的要求の 3 相誘導モータ制御のためのベクトル制御演算として、モータの電圧/電流入力から PWM に出力するまでの一周期を 20μsec 未満で実行し、割込み処理で AD 入力や速度チェックなど別の処理を合わせて3つのタスクを同時に並行して動かしている。

4　まとめ

今回"高位合成"というものから離れて C 言語ベース設計を捉え、H/W 設計者が持つ能力を生かす形で H/W, S/W の同時設計をシステム設計の環境側での実現を行った。今後は SystemC や高位合成との協調などを検討したい。

参考文献

[1] SystemC v2.0.1 User's Guide, Open SystemC Initiative http://www.systemc.org/
[2] Using Simulink ver3 & Writing S-Functions ver3, Matlab/Simulink Manuals, MathWorks, Inc.
[3] Virtex-E 1.8V Field Programmable Gate Arrays Data Sheet: DS022-1 (v2.2) November 9, 2001, XILINX