RTOS を用いたシステムの高位合成によるフルハードウェア化

大迫 裕樹 Yuuki Oosako
石浦 菜岐佐 Nagisa Ishiura
関西学院大学 理工学部

神原 弘之 Hiroyuki Kanbara
京都高度技術研究所

冨山 宏之 Hiroyuki Tomiyama
立命館大学 理工学部

本稿では, TOPPERS/ASP3 カーネルを用いたプログラムを高位合成によりフルハードウェア実装する手法を提案する. 本手法では, 1 つのタスクやハンドラを独立した 1 つの ハードウェアに合成し, 全てのタスク/ハンドラが並列に実行することを許す. これによって, 処理性能とレスポンスを向上させるとともに, RTOS のタスクスケジューリング機能を不要にし, 軽量なハードウェアでシステムを制御することを目指す.

1 はじめに

近年の情報通信技術の発展によって, 新しい製品やサービスが次々に提供されるようになっているが, これらに必要な機器を実装するために, 組込みシステムには益々高い機能が要求されるようになっている. 特に, 車載機器, 無人飛行機, ロボットの制御には, 高い機能と同時に高い応答性能が要求される.

リアルタイム OS (RTOS) を用いたシステムの応答性を向上させる手法としては, RTOS の機能をハードウェアとして実装する手法 [1][2][3] がある. しかし, これらの手法では, タスクやハンドラはソフトウェアとして実行されるため, その処理の計算量が多い場合には必ずしも応答性能は改善されない. 高位合成技術 [4] を用いれば, タスクやハンドラをハードウェア化し, 高速化することができる. これを利用してタスクや割込みハンドラをハードウェア化するシステムレベル設計手法 [5] が存在するが, RTOS を含めたシステム全体をハードウェア化することはできない.

本研究では, RTOS を用いたプログラムを入力とし, タスク, ハンドラおよび RTOS のサービスコールの機能全てをハードウェアに合成する手法を提案する. 本手法では, 1 つのタスクやハンドラを独立した 1 つの ハードウェアに合成し, 全てのタスク/ハンドラを並列に実行させることによって, 処理性能とレスポンスの向上を目指す. また, これによって RTOS のタスクスケジューリング機能を不要にし, 軽量なハードウェアでシステムを制御することを目指す.

2 高位合成

高位合成は, C 言語等で記述された動作記述からハードウェア設計記述を自動生成する技術である. これにより, 高い抽象度での設計や, ソフトウェア開発のノウハウを用いたデバッグが可能になる.

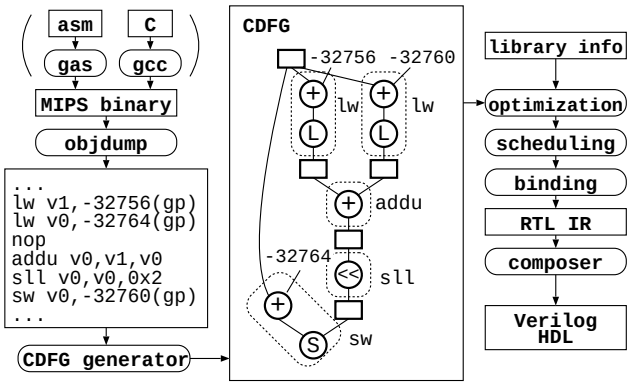

図 1 ACAP の高位合成処理の流れ [6]

高位合成システム ACAP [6] は, MIPS R3000 の機械語プログラムを入力として (アセンブラやコンパイラを用いることにより, アセンブリ言語や C 言語プログラムも入力可能である), レジスタ転送レベルの Verilog HDL を生成する. ACAP の処理の流れを図 1 に示す. ACAP は, 入力された機械語プログラムを CDFG (control dataflow graph) に変換した後, 最適化, スケジューリング, バインディングの処理を施して, これを Verilog HDL に変換する. また, ACAP は割込みハンドラを含むプログラムの合成も可能である [7].

3 TOPPERS/ASP3 カーネルを用いたプログラムのフルハードウェア化

3.1 概要

本手法では, TOPPERS/ASP3 カーネルを用いたプログラムを入力とし, これを実行するプロセッサと機能等価なハードウェアを合成する.

対象プログラムは, 複数のタスク, 周期ハンドラ, アラームハンドラ, 割込みハンドラからなるものとする. ただし, これらのタスク類は, 並列に実行され得るという前提で必要な排他制御が記述されているものとする. すなわち, 同時に 2 つ以上のタスクが実行されないことを前提とした排他制御を実装しているプログラムは, 本稿の手法の対象外である.

本手法で実装するハードウェアの構成を図 2 に示す. 1 つのタスク/ハンドラは 独立したハードウェアモジュールとして実装する. これらは並列に動作可能である. すなわち, 実行が可能な状態にあるタスクは, その優先度によらず全て実行される. ただし, 複数のメモリアクセスを同時

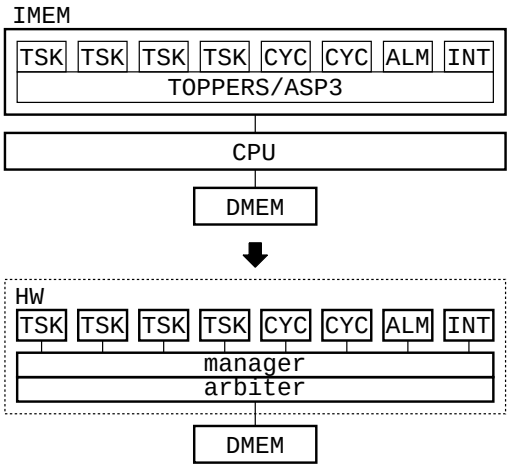

図 2　TOPPERS を用いたプログラムのフルハードウェア化

に行うことができないため, 調停が必要になる. タスクの優先度はこの調停に反映する.

3.2　ハードウェア構成と動作の仕組み

本手法で実装するシステムの構成を, 図 3 に示す. “TSKi” はタスク, “CYCi” は周期ハンドラ, “ALMi” はアラームハンドラ, “INTi” は割込みハンドラである. “manager” と “arbiter” は, それぞれタスク/スケジューラの実行制御, メモリアクセスの調停を行うモジュールである.

各タスク/ハンドラの状態は, manager モジュール内の status レジスタに格納されており, カーネルの状態は global status レジスタに格納されている. 周期ハンドラやアラームハンドラのタイマは status レジスタに含まれる. タスク/ハンドラは stall 信号によって停止するようになっており, manager モジュールは, status レジスタの値に基づいて stall 信号を変更することにより, タスク/ハンドラの実行 (起動/停止) を制御する. status レジスタおよび global status レジスタは, メモリ空間上にマップされており, 各タスク/ハンドラモジュールからロード/ストア演算によりアクセスできる. タスク/ハンドラからのサービスコールは, これらのレジスタを参照/更新する処理として実装する.

全てのタスク/ハンドラモジュールは 1 つのアドレス空間を共有しており, 全てのデータは共有メモリ (mem) に置かれる. メモリのポート数には限りがあるので, 同時に複数のメモリアクセスがあった場合は, arbiter モジュールが調停を行う. この調停は, タスク/ハンドラの優先度に基づいて行う. すなわち, 同時に起こった複数のメモリアクセスのうち, 優先度の最も高いタスク/ハンドラのメモリアクセスのみが許可され, それ以外のタスク/ハンドラモジュールには stall 信号が送られる.

3.2.1　タスク

タスクの実行は manager からの stall 信号によって制御される. 全タスクが並列実行可能であるため, ディスパッチが禁止されていない限り, 実行可能状態のタスクは即座に実行状態に遷移し, 実行される. 停止状態や待ち状態, 強制待ち状態では stall 信号が入力され, タスクモジュールの処理が停止する.

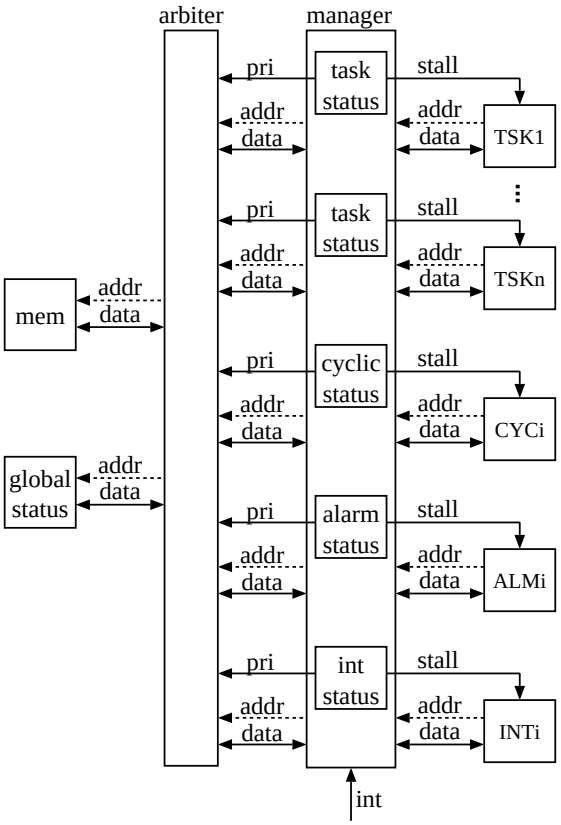

図 3　ハードウェア構成

タスクモジュールは, タスクを実行するプログラムから高位合成により生成する. タスクの状態を遷移させるサービスコールは, manager 内部の status レジスタを書き換える関数 (あるいはマクロ) に変換した上で高位合成する.

3.2.2　周期ハンドラ・アラームハンドラ

周期ハンドラ・アラームハンドラの実行もタスク同様, manager からの stall 信号によって制御される. 周期通知・アラーム通知が起動された場合, status レジスタ内部のタイマがセットされ, サイクル毎にタイマの値が減じられる. タイマの値が 0 になると周期ハンドラ/アラームハンドラモジュールへの stall が解除され, 実行が開始される. 周期ハンドラの場合は, タイマが 0 になった後, 自動的に次のタイマが再設定される.

3.2.3　割込みハンドラ

割込みハンドラも他のハンドラ同様, manager からの stall 信号によって管理される. manager に対して割込み信号 (int) が入ると, 割込みがマスクされていなければ割込みハンドラモジュールへの stall が解除され, 実行が開始される.

3.3　ハードウェア合成処理の流れ

ハードウェア合成処理の流れを図 4 に示す. まず, TOPPERS の静的 API を記述したコンフィギュレーションファイル (PROG.cfg) から, タスク/ハンドラの

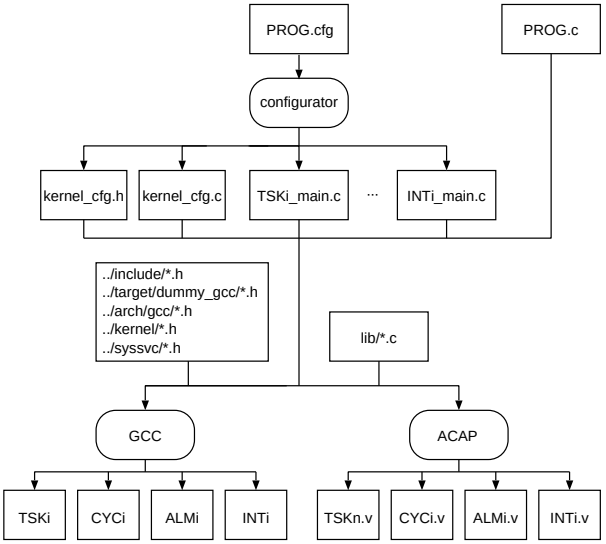

図 4　合成の流れ

status や関連するマクロ等の定義ファイル (kernel_cfg.c, kernel_cfg.h), および合成用の main 関数を含むファイル (TASKi_main.c 等) を生成する．TASKi_main.c 等のメインファイル, kernel_cfg.c, アプリケーションを記述したソースファイル (PROG.c), およびハードウェア合成用に実装したサービスコールを高位合成システム ACAP に入力し, Verilog HDL で記述されたハードウェアを生成する．また, 上記のファイルを GCC 等の x86 コンパイラに入力し, 実行バイナリ (TSKi, CYCi 等) を生成することにより, x86 上でテスト/デバッグを行うことができる．

4　実装と実験

現在, 本手法により, TOPPERS/ASP3 カーネル付属のサンプルプログラム “sample1” のハードウェア化を進めている．

このプログラムは, 全体を制御するタスク MAIN_TSK, 並行実行する 3 つのタスク TASK1, TASK2, TASK3 と周期ハンドラ・アラームハンドラ・割込みハンドラからなる．並行実行タスクは定期的にメッセージを出力し, 周期ハンドラでレディキューを回転している．メインタスクでは, シリアル通信から受け取った文字に対応してタスクやハンドラの状態を変化させる．

4.1　実装

コンフィギュレーションファイルから高位合成用のソースコード kernel_config.c, kernel_config.h を生成するためのコンフィギュレータを Perl5 で実装した．また, TOPPERS/ASP3 のサービスコールのうち, sample1 に含まれる 33 個を TOPPERS の統合仕様書 [8] をベースにハードウェア実装用に C 言語で実装した．

4.2　実験

sample1.c からタスク TASK1〜TASK3 のハードウェア実装を生成した．コンフィギュレータに sample1.cfg を入力して生成した main 関数を含むソースファイル, kernel_config.c, kernel_config.h および sample1.c をコンパイルし, MIPS アセンブリを生成した．これにサービルコールをリンクして MIPS の実行可能バイナリを生成した．生成したバイナリを ACAP を入力することにより, 各タスク/ハンドラおよびサービスコールの機能を持つハードウェアの Verilog HDL 記述を生成した．

動作確認としては, TASK1 に 17 個のサービスコールをテストするコードをリンクして生成したハードウェアで RTL シミュレーションを行い, 正常に動作することが確認できている．現在, arbiter, manager モジュールを実装中である．

5　むすび

本稿では, TOPPERS/ASP3 カーネルを用いたシステムを高位合成によりフルハードウェア実装する手法を提案した．タスク/ハンドラを並列動作可能なハードウェアに合成することにより, 処理速度とレスポンスを格段に向上させることができると考えている．残りのサービスコールの実装, arbiter, manager モジュールの実装が今後の課題である．

謝辞

本研究に関して有益な御助言を頂いた元立命館大学の中谷嵩之氏, 元関西学院大学の田村真平氏に感謝致します．本研究は一部科研費 16K00088, 16K01207, 15H02680 の助成による．

参考文献

[1] 森久直, 坂巻佳壽美, 重松宏志: “組込み制御システム向けリアルタイム OS のハードウェア化,” 東京都立産業技術研究所研究報告, 第 8 号 (2005).

[2] 丸山修孝, 石原亨, 安浦寛人: “仮想キューによる高性能ハードウエア RTOS の実現,” FIT2010 (Aug. 2010).

[3] Carl Stenquist: “HW-RTOS Improved RTOS Performance by Implementation in Silicon,” White Paper—Renesas R-IN32M3 Industrial Network ASSP (May 2014).

[4] D. D. Gajski, N. D. Dutt, A. C-H Wu, and S. Y-L Lin: *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers (1992).

[5] Yuki Ando, Shinya Honda, Hiroaki Takada, Masato Edahiro: “System-level Design Method for Control Systems with Hardware-implemented Interrupt Handler,” *Journal of Information Processing*, Vol. 23, No. 5, pp. 532-541. (Nov. 2014).

[6] Nagisa Ishiura, Hiroyuki Kanbara, and Hiroyuki Tomiyama: “ACAP: Binary Synthesizer Based on MIPS Object Codes,” in *Proc. ITC-CSCC 2014*, pp. 725–728 (July 2014).

[7] 伊藤直也, 石浦菜岐佐, 冨山宏之, 神原弘之: “割込みハンドラを独立したモジュールとして実装するバイナリ合成,” 信学技報, VLD2015-106 (Jan. 2016).

[8] TOPPERS project: “TOPPERS 第 3 世代カーネル (ITRON 系) 統合仕様書 Release 3.0.0,” `https://www.toppers.jp/docs/tech/tgki_spec-300.pdf` (Feb. 2016).