RTOS 利用システムのフルハードウェア化における ミューテックスの実装

Design of Mutex Module in Full Hardware Implementation of RTOS-Based Systems

南口 比呂 [1]　　石浦 菜岐佐 [2]
Hiro Minamiguchi　　Nagisa Ishiura

関西学院大学 大学院理工学研究科 [1]　　関西学院大学 工学部 [2]
Graduate School of Science and Technology, Kwansei Gakuin Univ.　　School of Engineering, Kwansei Gakuin Univ.

1　はじめに

RTOS を用いたシステムの応答性能を大幅に向上させる手法として, RTOS の機能とタスク/ハンドラの全てをハードウェア化する手法が提案されている [1] [2]. 本稿では, 文献 [2] のハードウェア構成においてミューテックスの実装を行う.

2　ミューテックスと前提とするハードウェア構成

ミューテックスはタスク間の排他制御機能を提供する. タスクがミューテックスをロックすると, そのタスクの優先度はミューテックスに設定されている上限まで引き上げられる. タスクがミューテックスをアンロックするとタスクの優先度は元に戻され, ロック待ちのタスクがあればそのうち優先度最大のタスクがロックを取得する.

本稿で前提とするハードウェア構成 [2] を図 1 に示す. T0~T3 はタスクをハードウェア化したもので, 実行可能状態のタスクは全て並列実行される. Manager は RTOS の機能をハードウェア化したもので, S0~S3 は RTOS のサービスを実行するサービスモジュールである. STATUS レジスタはタスクの状態, ベース優先度, カレント優先度, タイマーを格納している. タスクはレジスタ ti_regs にサービス ID と引数を書き込むことでサービスを要求する. サービスは 1 度に 1 つのみ実行される. Request Arbiter (RA) は優先度最高のタスクの要求をレジスタ x_regs に書き込むことでサービスモジュールを起動する. サービスモジュールは処理を終えると x_regs に返り値を書き込み, その返り値が ti_regs に書き込まれると, タスクに制御が戻る. 本稿で実装するミューテックスはサービスモジュールの 1 つである.

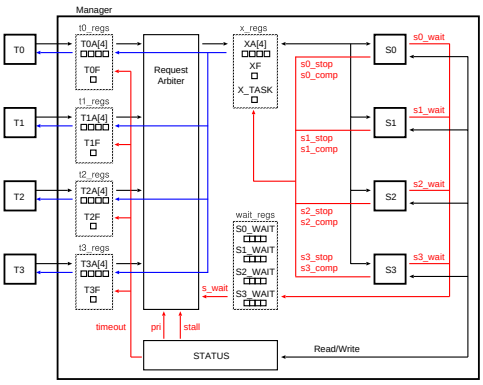

図 1　前提とするハードウェア構成 [2]

3　ミューテックスの実装

本稿で実装するミューテックスは 1 モジュールで複数ミューテックスを管理する.

タスクがアンロック状態のミューテックスにロックを要求すると, 1 サイクル目でロック状態にすると同時に, STATUS からタスクのカレント優先度を読み出し, 2 サイクル目で上限値と読み込んだ値の大きい方をタスクのカレント優先度として STATUS に書き込み, 正常終了を表す返り値を返す.

ロックを要求したミューテックスがロック状態の場合には, STATUS を書き換えてタスクを待ち状態にする. 待ちタスクの管理はキューで行うのではなく, タスクの要求を RA で処理待ち状態にすることにより行う.

タスクがアンロックを要求すると, ロックを解除し, 優先度を元に戻すとともに, ロック待ちのタスクがあれば RA にそれら全てのタスクの待ちを一旦解除するよう指示する. これにより, RA で選択された優先度最高のタスクがミューテックスに再送されてロックを取得し, それ以外のタスクは再度 RA で処理待ちとなる.

タイムアウト機能は, STATUS レジスタ内にタスクごとのタイマーを用意することにより実装する. タイマーが 0 になると, RA は優先的にその要求をサービスモジュールに送り, タイムアウト処理が行われる.

4　実装結果

提案手法に基づくミューテックスモジュールを Verilog HDL で設計し. Xilinx Vivado (2016.4) で Artix-7 をターゲットに論理合成を行った. 回路規模と実行サイクル数を表 1 に示す. 回路規模は LUT 数が 216, FF 数が 47, 遅延が 6.734ns となった. また, ロック/アンロックはいずれも 6 サイクルで行うことができた.

表 1　合成結果

(a) 回路規模

#LUT	#FF	delay (ns)
216	47	6.734

(b) 実行サイクル数

service call	cycle
ロック	6
アンロック	6

5　むすび

本稿では, RTOS 利用システムのフルハードウェア化においてミューテックスの回路構成法を提案した. タスクのサービス要求順でのロック待ち解除処理の実装が今後の課題である.

謝辞 本研究は一部 JSPS 科研費 19H04081 の助成による.

参考文献

[1] Y. Oosako, N. Ishiura, H. Tomiyama, and H. Kanbara: "Synthesis of full hardware implementation of RTOS-based systems," in *Proc. RSP 2018*, pp. 1–7 (Oct. 2018).

[2] 六車, 石浦, 安堂, 冨山, 神原: "RTOS 利用システムのフルハードウェア化におけるサービス処理機能の集約," 信学技報, VLD2020-75 (Mar. 2021).

Copyright © 2021 IEICE