RTOS 利用システムのフルハードウェア化における優先度継承ミューテックスの実装

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

志賀 光 [1]　　石浦 菜岐佐 [2]
Shiga Hikaru　　Nagisa Ishiura

関西学院大学 理工学部 [1]
School of Science and Technology, Kwansei Gakuin Univ.

関西学院大学 工学部 [2]
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]. 文献 [2] では文献 [1] のハードウェア構成においてミューテックスを実装しているが, それは優先度上限プロトコルに基づくものであった. 本稿では, 文献 [1] のハードウェア構成において優先度継承ミューテックスを実装する.

2　ハードウェア構成と優先度上限ミューテックス

文献 [1] のハードウェア構成を図 1 に示す. T0~T2 はタスクを, MANAGER は RTOS の機能をそれぞれハードウェア化したものである. タスクが RTOS のサービスを要求すると Request Arbiter (RA) がタスクのカレント優先度に応じた調停を行ってサービスモジュールを起動する. サービスモジュールにはミューテックスやデータキューの他, タスクの状態や優先度の変更等を行うタスク制御モジュール (CTRL) 等がある. タスクの状態や優先度は STATUS レジスタに格納されている. WAIT レジスタはサービス待ちのタスクの情報を管理しており, RA と連携してタスクの待ちと解除を行う.

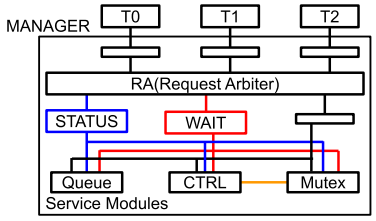

図 1　RTOS 利用システムのアーキテクチャ [1]

文献 [2] のミューテックスモジュールはタスク T からミューテックス M のロック要求があると, T が M をロックしたことをモジュール内に記録するとともに, STATUS 中の T のカレント優先度を M に定義されている優先度上限まで引き上げる. M が他のタスクによってロックされていた場合には, STATUS と WAIT に T が M 待ちであることを記録し, RA で T を待たせる.

3　優先度継承ミューテックスの実装

優先度継承プロトコルでは, M をロックしているタスク T のカレント優先度は M を待つ他タスクのカレント優先度の最大値に引き上げられる. 本稿では, ミューテックスモジュール内に M を待つタスクを記憶することによりこれを実現する.

ロック要求時, M がロックされていなければ優先度の変更は必要ない. 他のタスク T' が M をロックしていた場合には, STATUS にアクセスして T と T' の大きい方で T' のカレント優先度を更新する.

アンロック要求時には, T が他のミューテックスをロックしていなければ T のカレント優先度をベース優先度に戻す. T が他のミューテックス M' をロックしている場合には, M' を待つ全てのタスクのカレント優先度のうち最大値を T のカレント優先度とする.

優先度継承プロトコルでは, M をロックしているタスク T や, M を待っているタスク T' のベース優先度が変更された場合にも, T のカレント優先度を更新する必要がある. このためにタスク制御モジュールはベース優先度変更の要求を受け付けると STATUS の更新を行うと同時に, ミューテックスモジュールに T のカレント優先度を更新するように通知する. T のカレント優先度の更新は, アンロックと同様に行う.

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 (2020.2) で Artix-7 をターゲットに論理合成した. 結果を表 1, 表 2 に示す. 回路規模は文献 [2] と比較して LUT 数で 2.1 倍程度に増加した. アンロックや優先度変更の際には待ちタスクの数に比例するサイクル数が必要となった.

表 1 合成結果

LUT 数	FF 数	delay
450	147	5.78ns

表 2 実行サイクル数

機能	サイクル数 待ちタスク数 0 / n
ロック	5/　7
アンロック	5/ $n+6$
優先度変更	6/ $n+6$

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ける優先度継承ミューテックスを実装した. タスク数等に応じた設計記述の自動生成が今後の課題である.

参考文献

[1] T. Ando, et al.: "Full hardware implementation of RTOS-based systems using general high-level synthesizer," in *Proc. SASIMI 2022*, pp. 2–7 (Oct. 2022).

[2] H. Minamiguchi, et al.: "Hardware RTOS services for full hardware implementation of RTOS-based systems," in *Proc. SASIMI 2022*, pp. 14–19 (Oct. 2022).