5G-05

RMT Processorの割込み起床機構を用いた低遅延分散リアルタイム実行

大沢 幸平 †　羽鳥 雄介 †　溝谷 圭悟 ‡*　千代 浩之 ‡　山﨑 信行 ‡
† 慶應義塾大学 大学院理工学研究科　‡ 慶應義塾大学 理工学部　*現在，任天堂株式会社

1 はじめに

組込みリアルタイムシステムでは，デッドラインまでにタスクの実行が完了しなかった場合システムに致命的な障害を与えるハードリアルタイムタスクを持つシステムが多く存在する．特にヒューマノイドロボットに代表される分散リアルタイムシステムでは，ノード内の処理だけでなく，ノード間の通信により実行するタスクについてもデッドラインまでに処理を完了する事が求められる．通常，タスクの実行はスケジューラなどのオーバヘッドにより ms 単位の周期実行は可能だが，10μs 単位の周期実行は困難である．しかしながら，高精度な分散制御を実現するためには 10μs から 100μs 単位の短い周期で，かつジッタの小さい周期実行及び周期通信が求められる．特に短い周期の通信を行う場合，受信ノードのタスクの応答時間を短くする事も求められる．

10μs 単位の周期実行を実現した手法として，Responsive Multithreaded Processor (RMT Processor) [1] の割込み起床機構を用いた Responsive Task[3] がある．本研究では，Responsive Task を周期通信に応用することでオーバヘッドの小さい周期通信を実現する．通信には RMT Processor 上に実装されているリアルタイム通信規格 Responsive Link[2] を使用する．また，周期実行用に提案された Responsive Task を I/O 等の外部割込みによる非周期実行向けに拡張を行うことでパケット受信処理の応答時間を短縮する．

2 設計及び実装

本章では Responsive Task を外部割込みによって起床させる為の設計について述べる．

Responsive Task は RMT Processor が持つ 8 スレッド分のハードウェアコンテキストの 1 個を専有し，ハードウェアコンテキスト毎に設けられた内部タイマの割込みによって起床することで小さい起床オーバヘッドを実現した周期タスクである．RMT Processor は内部

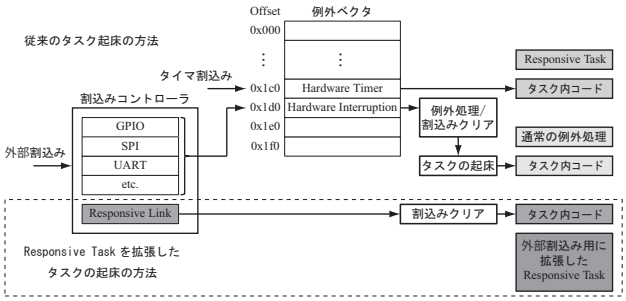

図 1: 割込み発生からタスクの起床までの流れ

タイマの割込みが発生した場合，ハードウェアが割込みのクリアを行う設計となっている．Responsive Task を外部割込みによって起床するように拡張する場合，ソフトウェアで割込みコントローラを制御して割込みのクリア処理を行わなければならない．そこで，タスクの起床時に割込みのクリアを行う専用の wait 関数を作成する．この関数を実行すると，次の外部割込みが発生するまでタスクの実行が停止する．そして外部割込みが発生すると，関数内の続きの処理である割込みのクリアを行い，本来の処理を開始する．また，RMT Processor ではハードウェアコンテキスト毎にどの外部割込みによってタスクを起床させるか設定することが可能である．1 種類の外部割込みによって複数のタスクを同時に起床させることも可能であるが，本研究では 1 種類の外部割込みで起床するタスクは 1 個とし，タスク生成時に既に起床タスクが設定されている外部割込みを指定した場合はタスクの生成に失敗する．

図 1 に通常の例外処理と Responsive Task，本章で外部割込み用に拡張を行った Responsive Task の割込み発生からタスクの起床までの流れを示す．図 1 の例では Responsive Link の割込み発生時のみ外部割込み用に拡張した Responsive Task でタスクが起床し，それ以外の外部割込みは通常の例外処理を行う．

3 評価

Responsive Task を用いた Responsive Link の通信と，本研究で実装を行った Responsive Task を拡張した非周期実行機構の評価を行う．評価では Cadence 社の NC-Verilog による RTL シミュレーションを行い，プロセスは TSMC 130nm を用いる．動作周波数 62.5MHz の RMT

A Low Latency Distributed Real-Time Execution with Interrupt Wake-up Mechanism on RMT Processor
Kohei Osawa †　Yusuke Hatori †　Keigo Mizotani ‡*
Hiroyuki Chishiro ‡　Nobuyuki Yamasaki ‡
†Graduate School of Science and Technology, Keio University
‡Faculty of Science and Technology, Keio University
*Presently with Nintendo Co., Ltd.

表 1: 評価の組合せ

組合せ	送信タスクの種類	受信タスクの起床方法
(1)	通常の周期タスク	通常の例外処理
(2)	Responsive Task	通常の例外処理
(3)	Responsive Task	直接起床

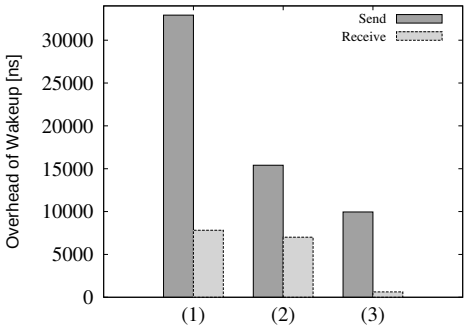

図 2: 起床オーバヘッドの評価

Processor 2 個を Responsive Link で接続したシミュレーションモデルを構築し，一方を送信ノード，もう一方を受信ノードとする．送信ノードでは Responsive Link のパケットを生成して送信するタスクを 1 個実行する．受信ノードでは Responsive Link のパケット受信割込みによって起床するタスクを 1 個実行する．ここで送信ノードのタスクの周期の開始から受信ノードのタスクが処理を開始するまでの時間を送信オーバヘッド，パケット受信割込み発生から受信ノードのタスクが処理を開始するまでの時間を受信オーバヘッドと定義する．送信ノードと受信ノードで動作するタスクの起床方法を表 1 に示す組合せで実行し，送信オーバヘッドと受信オーバヘッドを 100 周期測定した．受信ノードのタスクが処理を開始する時刻は，割込みのクリア等の処理を終えて本来の処理が開始される時刻とする．また，表 1 中の「通常の例外処理」は例外ベクタに処理を移して割込み処理を行う事を示し，「直接起床」は Responsive Task と同様にタスクがハードウェアコンテキストを専有し，対応する割込み発生時にタスクが直接起床する事を示す．図 2 に各条件における受信ノードのタスク起床時間の最大値を示す．図 2 中の「Send」は送信オーバヘッドを示し，「Receive」は受信オーバヘッドを示す．

図 2 より，送信ノードと受信ノードで 1 タスクずつ実行した場合において，スケジューラを使用する周期通信では送信オーバヘッドは 32.398μs であった．これに対して Responsive Task による周期通信の場合は，例外ベクタに処理を移す通常の手法で 15.408μs と，送信オーバヘッドを約半分に削減することが可能である．また，Responsive Task による周期通信を行う場合においても受信タスクの起床方法によって受信タスクの処理開始までの時間は大きく異なる．割込み発生時に例外ベクタに処理を移す通常の手法では受信オーバヘッドは 7.008μs である．一方，RMT Processor の割込み起床機構によって直接受信タスクを起床した場合は 0.624μs であった．

このことから，Responsive Task による周期通信や Responsive Task を外部割込み向けに拡張したタスクの実行機構を用いることで，ノード間通信の実行周期を最大で約 23μs 削減することが可能である．また，センサデータの取得などのノード内で完結する割込みについては通常の約 1/10 の起床オーバヘッドで実行可能である．

4 結論

本研究では，より高精度な分散制御を実現するために，従来よりも短い周期かつ小さいジッタでタスクを実行可能な Responsive Task による周期通信の性能調査を行った．また，分散制御においてはパケットの受信割込みやセンサデータの取得などの外部割込みの応答時間を短縮する事も重要なため，Responsive Task を非周期実行向けに拡張することで外部割込みの応答時間の短縮を行った．評価の結果，1 タスクのみ実行する環境において従来のスケジューラを用いた周期通信よりも最大で約 23μs 短い周期通信を実現した．

今後の課題として，より多くのタスクを実行した場合の評価を行う必要がある．また，実際のシステムに応用してどれだけ性能の向上が達成できるか調査する予定である．

参考文献

[1] N. Yamasaki. Responsive Multithreaded Processor for Distributed Real-Time Systems. *Journal of Robotics and Mechatronics*, Vol. 17, No. 2, pp. 130–141, April 2005.

[2] N. Yamasaki. Responsive Link for Distributed Real-Time Processing. In *Proceeding of the International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems*, pp. 20–29, January 2007.

[3] 渡邊大記, 溝谷圭悟, 羽鳥雄介, 千代浩之, 山﨑信行. 割込み起床機構を用いた低遅延リアルタイム実行. 組込みシステムシンポジウム 2015 論文集, pp. 74–83, October 2015.