Vol.2017-ARC-225 No.42
Vol.2017-SLDM-179 No.42
Vol.2017-EMB-44 No.42
2017/3/10

組込みマルチコアシステムにおける リアルタイム性保証機構の評価

北原 祐[1]　本田 晋也[1]　高田 広章[1]

概要：
近年，組込みシステムにおいて高信頼性・リアルタイム性と多機能性を両立するための手段として，RTOS と汎用 OS の双方を用いてシステムを構築する方法が利用されるようになっている．しかしながら，汎用 OS と RTOS で処理の分離を行った場合でも，ハードウェア資源を共有するため，汎用 OS の実行によって RTOS の処理性能が変動し，RTOS 側の予測可能性やリアルタイム性に影響を与えてしまうという問題がある．本論文では，汎用 OS と RTOS を用いたシステム構築方法における OS 間の干渉及び，リアルタイム性保証機構の評価を行いその有用性を示す．

Evaluation of Real-time performance guarantee mechanism in embedded multicore system

Yu Kitahara[1]　Shinya Honda[1]　Hiroaki Takada[1]

Abstract:
Recently, embedded system needs to fulfil requirement of various functions. The requirement makes development costs and complexity of emvedded system increased. To solve that problem and fulfil real-time constraint and high reliability, often used method GPOS and RTOS execute on the same system. We call the this configuration HDOSP. On the multicore system built by these configuration, execution time of the tasks on the RTOS fluctuates by sharing memory and bus. The fluctuation makes development costs higher and the function limited. In this resarch, we evaluate interference between GPOS and RTOS on the HDOSP and usefulness of real-time constraint guarantee mechanism.

1. はじめに

近年組込みシステムにおいて，IoT(Internet of Things) が着目されるようになり，組込み機器はネットワークに接続され，センサーから得た情報の活用，遠隔からの機器の操作，組込み機器同士の協調動作等の機能がさらに多様化し利便性を高めている．また，自動運転技術やトラッキングのように，組込み機器その物においてもプロセッサ等のハードウェア資源の高性能化や要求される機能の多様化に伴い， 画像処理，知識処理など様々な処理を求められるようになっている．よって組込みシステムを設計する際， 開発者は新たなハードウェアや様々な技術に対し迅速に対応する必要があり，設計の複雑化や開発コストの増大が問題となっている．このような設計の難化やコストの増大を解決する手段として，組込み機器に対して汎用 OS を用いる方法がある．しかし，汎用 OS を組込み機器に対して用いるにはリアルタイム性と信頼性の確保が課題となる．

この課題を解決する方法として，組込み機器の制御など高信頼性，リアルタイム性が求められる制御系処理を RTOS で行い，画像処理，知識処理など多機能性が求められる情報系処理を汎用 OS で行うように処理を分離させて実現する方法がある（**図 1**）．このようなシステムを Heterogeneous Dual Operating System Platform（HDOSP）呼ぶ．制御系処理と情報系処理を 2 つの OS により分離させることによって，リアルタイム性と高信頼性を実現することが可能となる．しかしながら，メモリシステムなどのハードウェ

[1] 名古屋大学 大学院情報科学研究科
愛知県名古屋市千種区不老町

Vol.2017-ARC-225 No.42
Vol.2017-SLDM-179 No.42
Vol.2017-EMB-44 No.42
2017/3/10

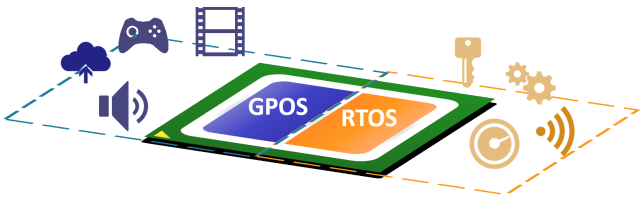

図 1 Heterogeneous Dual Operating System Platform（HDOSP）

ア資源の共有から汎用 OS と RTOS 間で干渉が発生し，汎用 OS の実行により RTOS 側の実行時間が変動する可能性がある．そのため，RTOS 上のプログラムの時間制約を解析する際に悲観的に見積もらなければならず，その結果，余裕のあるハードウェアを用いることとなり，コストの増加に繋がる．

本研究では，HDOSP を実現するマルチコアシステムにおける OS 間の干渉を評価と RTOS のリアルタイム性を保証するための機構（リアルタイム性保証機構）の有用性を評価する．マルチコアシステムとしては，対象型と非対称型のマルチプロセッサを用いて，汎用 OS の実行により RTOS の実行時間がどの程度悪化するか評価する．リアルタイム性保証機構としては，4 種類の機構を対象に，RTOS の実行時間の悪化が緩和されるか評価する．これらの機構の多くはハードウェア機構であり，市場の製品も備えているが，その効果に対する定量的な評価がなされていない．

2. HDOSP おける各系の要件

HDOSP において制御系及び情報系に求められる要件について述べる．

2.1 制御系（RTOS）の要件

RTOS が実行される制御系に求められる機能要件について述べる．

- **機能要件 R1** ： リアルタイム性が確保されていること．制御系の処理は，結果の正確性だけでなく時間制約も存在する．そのため，予測可能性が高いプログラムの実行が必要となる．
- **機能要件 R2** ： 高信頼性が保証されていること．人命に関わるようなシステムでは，制御の不具合により重大な事態が生じる恐れがあり，高い信頼性が求められる．汎用 OS は規模やネットワークに接続されることが多いため，信頼性の確保が困難であるため，汎用 OS の実行から RTOS を保護する機構が必要である．

2.2 情報系（汎用 OS）サブシステム

次に汎用 OS が実行される情報系に求められる機能要件・非機能要件について述べる．

- **機能要件 G1** ： 多機能性を提供すること．汎用 OS の利用の主な目的として，多数のライブラリの利用や搭載された既存のデバイスドライバを用いることで開発効率を向上させることが挙げられる．
- **非機能要件 G2** ： ソースコードの変更が少ないこと．Linux に代表される汎用 OS に用いられる OS は，複雑でその規模が大きい．HDOSP 向けに汎用 OS を変更するのは工数が大きくなるため，可能な限りソースコードを変更せずに用いることが要求される．

3. HDOSP の実現方法

本章ではマルチコアハードウェアによる HDOSP の実現方式について述べる．

3.1 マルチコアハードウェア

HDOSP を実現するマルチコアハードウェアの構成は次の 2 種類が挙げられる．

- 対照型マルチコアプロセッサ（SMP-HW）
- ヘテロジニアスマルチコアプロセッサ（AMP-HW）

SMP-HW は，複数キャッシュを共有する同一命令セットのコアにより構成されている．プログラムは基本的にどのプロセッサでも実行可能であるが，多くのハードウェアリソースは各コアで共有される．

AMP-HW は，メインコアとサブコアの複数種類のプロセッサにより構成されている高機能で演算性能の高い Application Processing Unit（APU）とリアルタイム性と低消費電力に優れた Real-time Processing Unit（RPU）による構成が一般的となっている．一部バスや外部メモリは両者のコアで共有する．

3.2 システム構成

AMP-HW/SMP-HW を用いた HDOSP の実現する構成として，**図 2** に示す 4 種類が挙げられる．

3.2.1 構成 1：各 OS を SMP-HW に配置

RTOS と汎用 OS を SMP-HW の各コアに配置する方法である．各 OS は，配置されたコアを占有する． よって，それぞれの OS における最大処理性能は割り当てられたコアに制限される．さらに，供給クロックがコア毎ではなくプロセッサ全体で同一の場合，汎用 OS 側で DVFS を行うと，RTOS の実行コアの供給クロックも落ちてしまい，実行時間の見積もりが難化してしまい消費電力の削減が難しいことがあげられる．また，各コア間の保護も実現されていない．

3.2.2 構成 2：仮想化を用いて各 OS を SMP-HW に配置

RTOS と汎用 OS を SMP-HW 上に仮想化を用いて配置する構成であり，2 種類の配置方法がある．

配置方法 1 コア共有なし

構成 1 と同様に RTOS と汎用 OS は特定のコアに配置さ

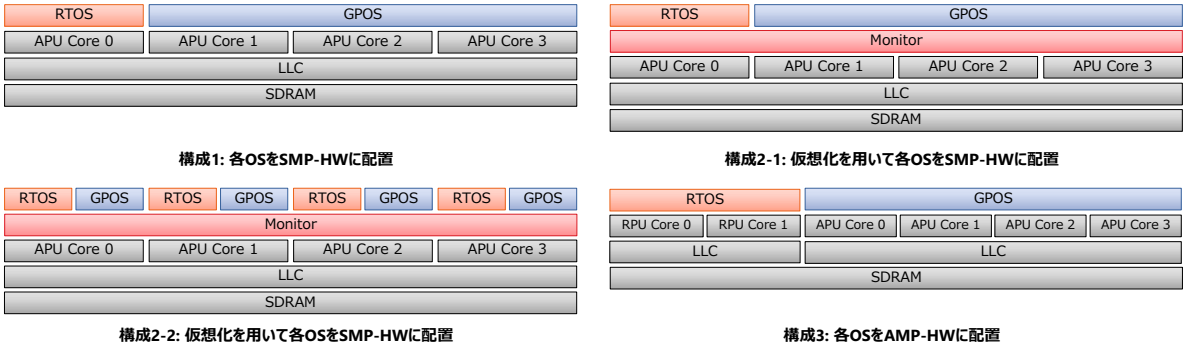

図 2 HDOSP のシステム構成

比較内容	RTOS の処理性能	OS 間通信オーバヘッド	RTOS のリアルタイム性	プロセッサ利用率	RTOS の保護	消費電力
構成 1	○	△	×	△	×	×
構成 2-1	○	○	△	△	○	×
構成 2-2	○	◎	△	◎	○	×
構成 3	×	×	◎	×	◎	◎

表 1 システム構成の比較

れる．処理性能の制限と消費電力の削減が難しいことが構成 1 同様に課題として残る．仮想化を支援するハードウェア機構を利用することで RTOS のメモリ，デバイスの保護が可能となる．さらに，キャッシュの一貫性を維持することができ，構成 1 より OS 間通信を高速で実現が可能である．しかし，この構成では仮想化による実行オーバヘッドが割込みなどの際に生じることが欠点である．

配置方法 2 コア共有あり

汎用 OS と RTOS でコアを共有する．リアルタイム性の確保のため，RTOS の実行が汎用 OS より優先的に扱われ，汎用 OS の実行は RTOS がアイドル状態になっている間に処理が行われる．構成 1 や構成 2-1 と比べると，プロセッサ全体の利用効率は高い．さらに，同一コア内の OS 間通信においては，コア内部のキャッシュを利用するため構成 2-1 より高速な通信を実現することが可能となる．欠点としては，同じコアで各 OS が実行されるため OS 間のキャッシュの影響が大きく RTOS 側の実行時間が大きく変動する可能性があることである．また，構成 2-1 同様に実行オーバヘッドの問題がある．

3.2.3 構成 3：各 OS を AMP-HW に配置

AMP-HW を用いて，RTOS を RPU に，汎用 OS を APU に配置する方法である．特徴として，各 OS で異なるプロセッサを利用するため． 汎用 OS のクロックを変更しても RTOS の実行に影響がなく消費電力の削減が容易に可能である．また，共有するハードウェア資源が少ないため干渉を受けにくいことが考えられる．さらに，RPU が機器の保護や故障耐性に対応するための機構を持つ場合は，RTOS 側の信頼性を確保することが可能である．

3.3 システム構成の比較

前述の 4 種類の構成について，RTOS の処理性能，OS 間通信におけるオーバヘッド，RTOS のリアルタイム性，プロセッサの利用率，RTOS の保護の 6 つの項目の比較を**表 1** に示す．

4. リアルタイム性保証機構

機能要件 R1 で述べたように，HDOSP においては，RTOS のリアルタイム性を実現する必要がある．一方，3 章で述べたように，SMP-HW/AMP-HW 共に各コアはリソースを共有するため，汎用 OS から RTOS への影響が小さくなるよう，何らかの方法でリソースを分割する必要がある．本章では，現状のシステムで利用可能なリアルタイム性保証機構について説明する．

4.1 キャッシュロックダウン機構

この機構はデータをキャッシュに固定化する．その結果，固定化されたデータに対するデータアクセスは常にキャッシュヒットとなるため，アクセス時間が変動しない．また，キャッシュパーティショニングと併せて利用することで性能向上が見込める [1].

4.2 アドレスフィルタリング機構

アクセスするデータのアドレスによって利用するバスマスタを決定することができる機構である．同時に複数のトランザクションが発行された場合でも特定のアドレスに対するアクセス時間を保証することが可能となる．

Vol.2017-ARC-225 No.42
Vol.2017-SLDM-179 No.42
Vol.2017-EMB-44 No.42
2017/3/10

4.3 QoS 機構

QoS 機構はトランザクションに対して優先度を付与し，優先度ベースのメモリアクセスを提供する機構である．優先度を高く設定したトランザクションは低優先度のトランザクションより優先的に処理されるため，メモリアクセスが集中した場合においても，高優先度のメモリアクセスのアクセス時間を保証することが可能となる．また，メモリアクセスのレイテンシに合わせて動的に優先度の変更を行うものもある．

4.4 レギュレーション機構

インターコネクトにおける，時間当たりのデータアクセスを制限する機構である．この機構を用いることでメモリコントローラ等の複数のデバイスからトランザクションが発行されるデバイスにおいて，それぞれのデバイス毎にデータアクセスを個別に制限することで，低優先度のデバイスから高優先度のデバイスに対するメモリアクセスの影響を減らすことが可能となる．

5. リアルタイム保証機構の評価

まず，SMP-HW と AMP-HW における，汎用 OS の実行が RTOS の実行に与える影響について評価する．次に，前章で述べたリアルタイム性実現のための機構を評価する．具体的な評価項目は次の通りである．

- 評価 1：OS 間干渉評価
  - 評価 1-1：SMP-HW
  - 評価 1-2：AMP-HW
- 評価 2：リアルタイム保証機構の評価
  - 評価 2-1：キャッシュロックダウン機構の評価
  - 評価 2-2：アドレスフィルタリング機構の評価
  - 評価 2-3：QoS 機構の評価

5.1 評価環境

SMP-HW として，NXP 社の i.mx6 quad SABRE 評価ボード（i.mx6）を用いた．構成を**図 3** に示す．i.mx6 は，アウトオブオーダ実行の 4 コアで構成される．リアルタイム性保証機構としては，SCU と L2 キャッシュにアドレスフィルタリング機構を持ち，L2 キャッシュはキャッシュロックダウン機構も持つ．

AMP-HW としては，Xilinx 社の Zynq UltraScale+ MPSoC ZCU102 評価キットを用いた． 構成を**図 4** に示す．APU はインオーダー実行の 4 コア構成である．RPU はロックステップ動作可能な 2 コア構成であり，コア毎にデータと命令の L1 キャッシュメモリと TCM を有する．リアルタイム性保証機構として，APU には CCI400 に QoS 機構を持ち，RPU では QoS400 に QoS 機構とレギュレーション機構を持つ．DDR コントローラは QoS 機構に対応しており通常はトランザクションは各ポートに対するラウンドロビンで処理を行うが，QoS 優先度が負荷されているトランザクションは優先的に処理される．

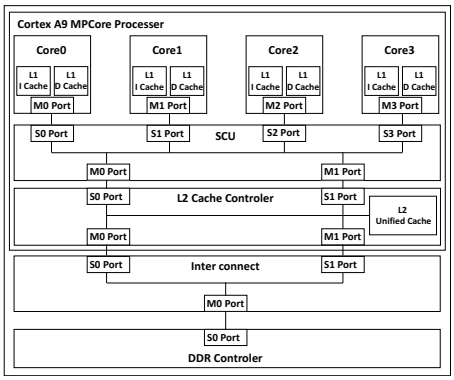

図 3 i.mx6 quad SABRE ブロック図

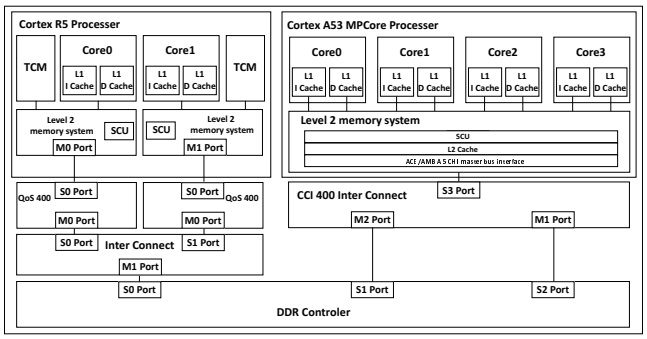

図 4 Zynq UltraScale+ MPSoC ブロック図

OS は TOPPERS/FMP カーネル（FMP カーネル）を用いた． 仮想化環境としは TOPPERS/SafeG（SafeG）を利用した．構成 2-1 と 2-2 の評価においては，RTOS と汎用 OS としてそれぞれ FMP カーネルを実行した．構成 3 の評価においては，APU，RPU それぞれに FMP カーネルを配置し実行する．

5.2 評価 1：OS 間干渉評価

5.2.1 評価 1-1：SMP-HW

コア 0 において 1ms 周期で RTOS に対してタイマ割込みを生成し，割込みの発生から割込みハンドラの起動までの時間を測定した．汎用 OS の実行による影響を図るために，汎用 OS のデータアクセスの実行を行うコアの数を 0 から 4 まで増加させ汎用 OS の実行コア数の変化による影響をみた．RTOS は割込みハンドラの実行を除いて常にアイドル状態となっており，この間は汎用 OS が割り当てられている場合は汎用 OS が実行される．汎用 OS 側のコアではデータアクセスのためにレジスタからメモリに対してを 1024KB データのストアを行うプログラムを実行し続ける．RTOS の試行は 10000 回とした．

評価結果を**図 5** に示す．汎用 OS の実行コアが増加しアクセスが増加すると実行時間が増加して分散が大きくなっている．最悪実行時間についてはコア数の増加に合わせて汎用 OS を実行しない場合と比較して，コアを増加させる

Vol.2017-ARC-225 No.42
Vol.2017-SLDM-179 No.42
Vol.2017-EMB-44 No.42
2017/3/10

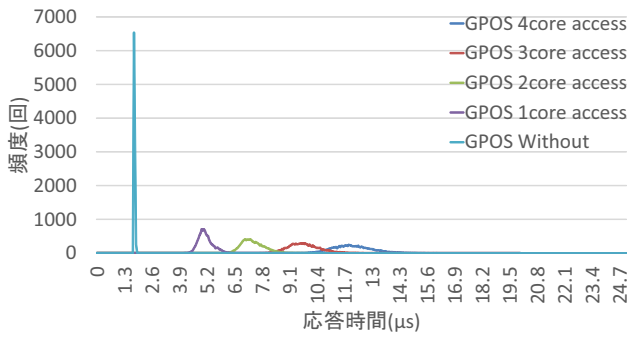

図 5　評価 1-1：SMP-HW ： 割込み応答時間

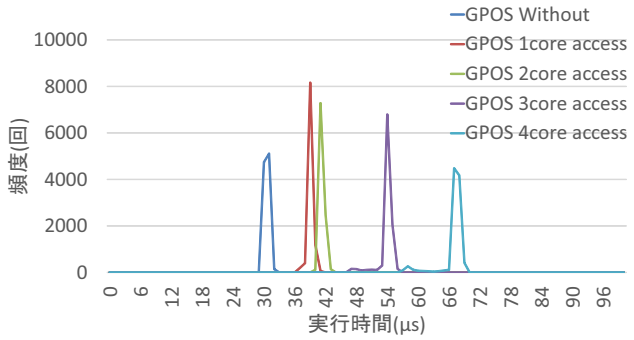

図 6　評価 1-2：AMP-HW ： データアクセス時間

毎に 2 倍増加している．

5.2.2　評価 1-2：AMP-HW

RPU で実行される RTOS による DDR 上のデータへのアクセス時間測定した．汎用 OS による干渉として，汎用 OS によりデータへアクセスするコアの数を 0 から 4 に変更して計測した．条件の一定化のため RTOS は TCM に，汎用 OS は L1 キャッシュメモリにテキストデータを配置し命令のデータアクセスによる実行時間の変動を無くした．また，データキャッシュを無効として計測した．

評価結果を図 **6** に示す．汎用 OS のコア数の増加に合わせて，3 割から 4 割ほど最悪実行時間が増えており，APU と RPU 間で干渉が生じることが確認できた．

5.3　評価 2：リアルタイム保証機構の評価

5.3.1　評価 2-1：キャッシュロックダウン機構の評価

評価 1-1 と同様の評価を行い，RTOS がアクセスするデータはロックダウン機構を用いてライン単位で L2 キャッシュに固定する．

キャッシュロックダウン機構を用いた場合と用いない場合の RTOS の割込み応答時間のヒストグラムを図 **7** に示す．ロックダウン機構を用いた場合は，わずかに応答時間が短くなってはいるが，大差はない．

5.3.2　評価 2-2：アドレスフィルタリング機構の評価

評価 1-1 と同様の評価を汎用 OS が 4 コアの場合のみ行い，データ要求はアドレスフィルタリング機構を用いて

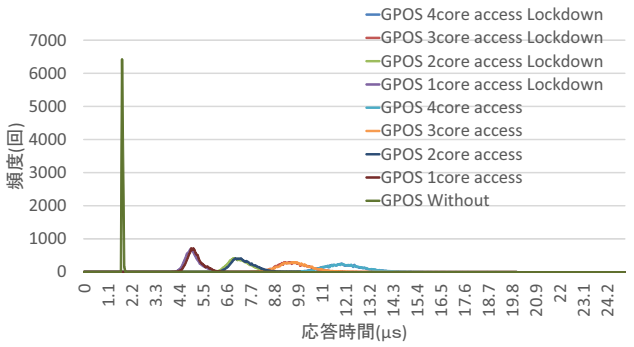

図 7　評価 2-1：キャッシュロックダウン機構の評価：割込み応答時間

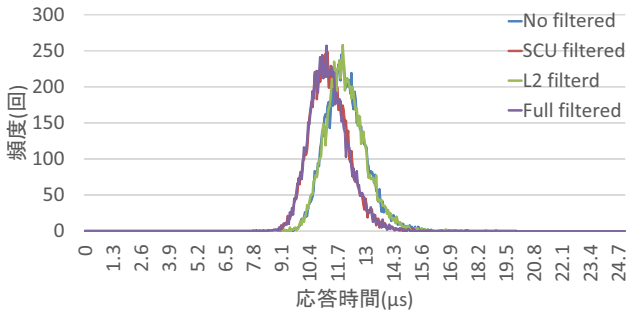

図 8　評価 2-2：アドレスフィルタリング機構の評価：割込み応答時間

RTOS と汎用 OS 利用するポートの分離を行った．

L2 キャッシュと SCU にアドレスフィルタリング機構を設定し，RTOS のテキストやデータがあるアドレスへのデータ要求を M0 ポート，汎用 OS のテキストやデータがあるアドレスへのデータ要求を M1 ポートを用いるようにする．その上で，次の 4 種類のパターンについて評価した．

- パターン 1：SCU，L2 フィルタリング有効
- パターン 2：SCU フィルタリング有効
- パターン 3：L2 フィルタリング有効
- パターン 4：フィルタリング無効

各パターンにおける評価結果を図 **8** にに示す．ヒストグラムのピークは 2 つの組に結果が分かれており，全フィルタリングを有効にした場合と SCU のみ有効にした場合をグループ 1，L2 キャッシュフローのみフィルタリングを有効にした場合と全フィルタリング無効をグループ 2 とすると，汎用 OS の実行数の全パターンにおいてグループ 1 の方がグループ 2 より応答時間が短くなっている．グループ 1 はどちらも SCU に対してアドレスフィルタリングを行っており，今回用いた環境では SCU がボトルネックであることが推測できる．

5.3.3　評価 2-3：QoS 機構の評価

評価 1-2 と同様の評価を行い，RTOS の性能保証のため QoS400 の QoS 機構を用いて APU，RPU の各トランザクションに対して優先度の付与を行った．各プロセッサのト

Vol.2017-ARC-225 No.42
Vol.2017-SLDM-179 No.42
Vol.2017-EMB-44 No.42
2017/3/10

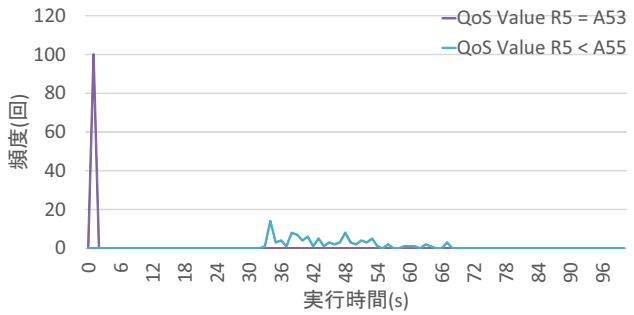

図 9　評価 2-3：QoS 機構の評価：データアクセス時間 1

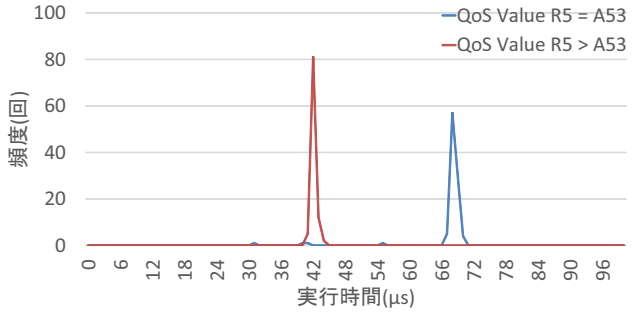

図 10　評価 2-3：QoS 機構の評価：データアクセス時間 2

ランザクションに対して，優先度を次のように付与した．

- パターン 1：QoS Value R5 = A53
- パターン 2：QoS Value R5 > A53
- パターン 3：QoS Value R5 < A53

パターン 1 とパターン 2 の結果を**図 9** に，パターン 1 と 3 の結果を**図 10** に示す．それぞれスケールが異なることに注意されたい．汎用 OS が 2 コア実行時においては優先度に関わらずほぼすべての山が一致しており QoS 優先度による実行時間の差がない．また，QoS 優先度を APU より高優先度に設定した場合汎用 OS の実行が 2 コア，3 コア 4 コアの全ての場合においてピーク値が 42μs となりデータアクセスの有無にかかわらず一定の処理時間を保証できることが分かった．しかし，汎用 OS の実行が 3 コア以上の場合パターン 3 のように APU が高優先度になると 60 倍以上の性能悪化が見られた．

5.4　考察

構成 2 におけるボトルネックは，評価 2-2 よりアドレスフィルタリング機構を用いて SCU のポートを分離することで性能が改善したことから SCU と言える．階層構造のメモリシステムにおいて，それぞれのインターコネクトは下位レベルのメモリシステムに対するデータ受け渡しを非同期に実現するため，ポート毎に下位レベルのメモリシステムに対する未処理のトランザクション (アウトスタンディングトランザクション) の保持を行う．保持できる未処理のトランザクション数は上限があり，上限を超えた場合，処理中の要求が完了するまで，ストールする [2]．よって上限超えた場合におけるデータアクセス時間は目的のデータの転送時間と処理要求に空きができるまでの時間となる．今回 SMP-HW として利用した APU は 4 コア，アウトオブオーダー実行で動作周波数も高いため，連続したデータ要求でバスの最大トランザクション数が埋まったのだと考えられる．またキャッシュロックダウンを利用しても性能がほとんど改善しなかった事も同じ原因だと予測される．AMP-HW では今回 APU と RPU が共有した接続は DDR コントローラのみであるためボトルネックは DDR コントローラであり，本研究では QoS 優先度を用いることで性能改善ができた，しかし QoS 優先度が高優先度の別のデバイスがある場合，アクセスは大きな遅延を生じさせる可能性を含んでいる．

次に各ハードウェア機構の有用性では，ロックダウン機構については評価 2-1 でもわかるように高速にデータ要求を行うマスタが多い，もしくは，キャッシュミスが多くプログラムを汎用 OS 側で動かすと RTOS がストールする可能性が高いため，リアルタイム性の保証のため利用するより平均性能の向上のために用いる方がよい．アドレスフィルタリング機構は評価 2-2 より性能の改善が確認できた．ただしリアルタイム性の保証のためにはボトルネックとなりうるポートを把握する必要があり，またアドレスの領域をもとにポートの分離を行うため，複数の不連続なデータ領域に対する分離は行えない，QoS 機構は評価 2-3 よりデータアクセスのレイテンシの保証ができることが分かった．ただし，低優先度のデバイスによるメモリアクセスが大きく阻害される可能性があるため，利用する際にはシステム全体を把握しながら設定を行わなければならない．

6.　まとめ

本研究では，HDOSP によるシステム構成において OS 間の干渉を評価を行い，RTOS のリアルタイムルイム性を保証するための機構を評価した．評価結果，HDOSP の各構成においてコア間の干渉が生じることを示し，リアルタイム性保証の機構の効果を示した．

今後の課題としては，動的な QoS 機構のパラメータ変更による，汎用 OS の効率的な実行が挙げられる．

参考文献

[1] Gracioli, Giovani, and Antnio Augusto Frhlich. ”On the Design and Evaluation of a Real-Time Operating System for Cache-Coherent Multicore Architectures.” ACM SIGOPS Operating Systems Review 49.2 (2016): 2-16.

[2] Yun, Heechul, and Prathap Kumar Valsan. ”Evaluating the Isolation Effect of Cache Partitioning on COTS Multicore Platforms.” OSPERT 2015 (2015): 45.