Vol.2017-SLDM-181 No.41
Vol.2017-EMB-46 No.41
2017/11/8

イーサネットコントローラ共有機構

石野 正敏[†1,a)]　本田 晋也[†1]

概要：近年，自動車の多機能化が進み，求められる信頼度が異なる機能が混在しているシステムが増加している．信頼度の異なる機能間は影響を防ぐため，パーティショニングが必要となる．プロセッサやメモリに関しては，RTOS の提供するメモリ保護機能等により，パーティショニングすることが可能である．一方，周辺回路（デバイス）に関しては，信頼性確保のために機能ごとに用意し，それぞれの機能に占有して使用させる方法が一般的である．しかしながら，ネットワークデバイス等はコスト等の問題で機能ごとに用意することが不可能であり，信頼度の異なる機能間で共有する必要がある．そのため，単一のデバイスを信頼度の異なる機能間で安全に共有する手法が必要となる．本論文ではイーサネットコントローラを機能間で安全に共有する手法を提案する．提案手法では，既存のイーサネットコントローラに対して，ProtectionWrapper というデバイス保護機構の追加および共有のためのハードウェア拡張を行うことで，パーティショニングを維持したまま，それぞれの機能が直接イーサネットコントローラを操作して送受信を行うことを可能とした．提案手法を既存のイーサネットコントローラに対して適用し評価した結果，それぞれの機能に対してイーサネットコントローラを割り付けた場合と比較して，ハードウェア面積が23%の減少となり，実行オーバーヘッドを 1.7%に抑えられた．

Ethernet controller sharing mechanism

Ishino Masatoshi[†1,a)]　Honda Shinya[†1]

Abstract: In recent years, as the number of functions of automobiles has increased, a system with different reliability functions are increasing. In such a system, partitioning is necessary in order to prevent the influence between functions having different reliability. Regarding the processor and memory, partitioning can be realized by using the memory protection function of RTOS. On the other hand, regarding peripheral devices, it is common to prepare for each function to ensure reliability. However, it is difficult to prepare network devices for each function due to its cost, therefore, it is necessary to share network devices between functions with different reliability. Hence, it is necessary to have a method for securely sharing a single network device between such functions. This paper, we propose the method to share Ethernet controller between functions. In the proposed method, we apply the device protection mechanism called Protection Wrapper to the existing Ethernet controller, and expand controller for sharing, then it becomes possible for each function to use one Ethernet controller independently. As a result of using the proposed method, compared to the case where Ethernet controller is assigned to each function, echo back time was increased by 1.7%, and hardware area was decreased by 23%.

1. はじめに

近年，自動車は多機能化が進み，求められる信頼度の異なる機能が混在しているシステムが増加している．このように求められる信頼度が異なる機能が混在しているシステムをミックスドクリティカルシステムという [1]．多くの機能はソフトウェアによって実現され，そのソフトウェアは ECU 上で実行される．機能を追加するごとに ECU を追加するとその数が膨大になり，限られた自動車の空間内では ECU を搭載するスペースが不足してしまう．そのため，将来的にマルチプロセッサを利用するなどして，複数の機能を 1 つの ECU に実現したいという要求がある．そのような ECU では求められる信頼度の異なる機能が混在することになる．

一方，車載システムには高い信頼性が求められ，ISO26262 に基づく安全規格に従う設計が必要である．安全規格に従うために，求められる信頼度が異なる機能を信頼度ごとにパーティショニングするような手法がとられている．シングルコアプロセッサであればメモリ保護機能付きの RTOS を用いることによって，使用するメモリ領域を機能ごとに制限することでパーティショニングすることが可能である．マルチプロセッサであれば信頼度の異なる機能を別々の CPU に実装することによってパーティショニング可能である．しかし，デバイスについてはハードウェア面積や消費電力といった観点から共有せざるを得ない場合がある．その最たる例としてネットワークデバイスが挙げられる．

信頼度の異なる機能間でネットワークデバイスを共有する方法として，それぞれの機能が直接ネットワークデバイスを操作する直接操作方式が挙げられるが，この方式では，低信頼機能によって高信頼機能の信頼性が損なわれてしまう危険性が存在する．そのため，高信頼機能のみがネットワークデバイスにアクセス可能とし，低信頼機能がデバイ

[†1] 現在，名古屋大学
Presently with Nagoya Uniersity
[a)] ishino@ertl.jp

Vol.2017-SLDM-181 No.41
Vol.2017-EMB-46 No.41
2017/11/8

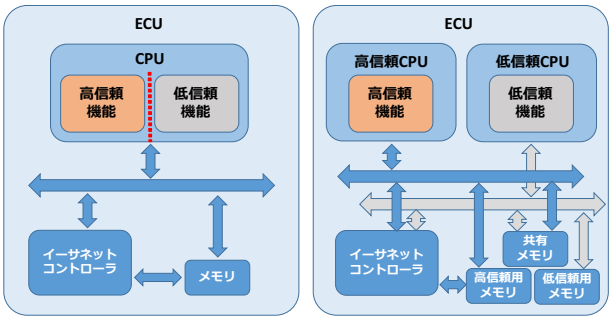


図 1　信頼度の異なる機能の実現方法

スを使用したい場合は処理を高信頼機能に依頼する方式があり，これを依頼方式と呼ぶ．しかし，この方式を用いると，高信頼機能が低信頼機能からの依頼を受け取る処理や，高信頼機能と低信頼機能間でのデータの受け渡し処理などが必要になるためオーバーヘッドが増加してしまう．その他にも高信頼機能が低信頼機能からの連続的な送信依頼により Dos 攻撃を受けるという問題がある．

この問題を解決し，直接操作方式において機能間のパーティショニングを実現する方法として，ProtectionWrapper（PW）というデバイス保護機構を提案し，CAN コントローラを信頼度の異なる機能間で安全に共有する機構を実装している [2]．

本論文では，昨今の車載システムにおけるイーサネット使用の需要が高まっていることから [3][4]，直接操作方式によってイーサネットコントローラを安全に共有する手法を提案する．直接操作方式では，低信頼機能がイーサネットコントローラに直接アクセスして送受信を行う．そのため，低信頼機能がイーサネットコントローラを不正に操作しないように，イーサネットコントローラに PW を適用しアクセス制御を行った．しかしながら，CAN コントローラと異なり，イーサネットコントローラは PW の追加だけでは，パーティショニングされた直接操作方式を実現できなかったため，イーサネットコントローラを拡張した．

提案手法を既存のイーサネットコントローラに適用し評価した結果，それぞれの機能に対してイーサネットコントローラを割り付けた場合と比較して，ハードウェア面積は 23%減少し，実行オーバーヘッドの増加は 1.7%に抑えられた．

2. 要件

車載システムにおいて，信頼度の異なる機能を混在させてシステムを実現する方法として図 1 に示すような実現方法が考えられる．1 つはシングルプロセッサで信頼度の異なる機能ごとにパーティショニングして実現する方法で，もう 1 つはマルチプロセッサで CPU ごとに信頼度の異なる機能を実現する方法である．

本論文では両方のケースを想定するが，説明を簡略化するため，これ以降は図 1 の（2）のマルチプロセッサを例に説明する．マルチプロセッサにおいて，高信頼機能を実行する CPU を高信頼 CPU，低信頼機能を実行する CPU を低信頼 CPU と呼ぶ．

イーサネットコントローラを CPU 間で共有する機構の実現方法を検討するにあたり，以下に挙げる 5 項目に留意した実現を目指した．

（要件 1）低信頼機能の動作不良により，高信頼機能の送受信が阻害されないこと．

（要件 2）イーサネットコントローラを共有することにより生じる実行オーバーヘッドの増加を抑えること．

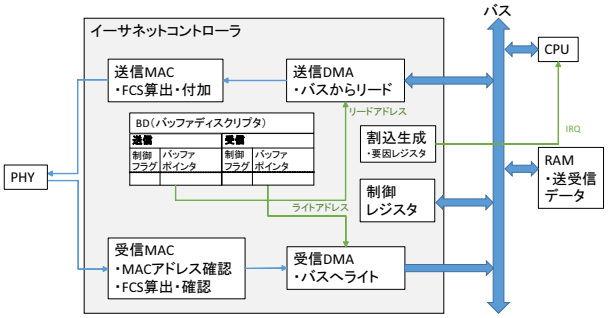


図 2　イーサネットコントローラの構成

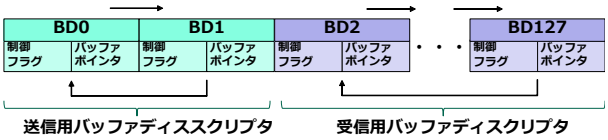


図 3　バッファディスクリプタテーブル

（要件 3）高信頼機能と低信頼機能間でのインタラクションを少なくすること．

（要件 4）低信頼機能がネットーワーク帯域を占有しないこと．

（要件 5）プロトコルスタックとドライバの変更量を少なくすること．

なお，本論文での実現方法では低信頼 CPU と高信頼 CPU 間でイーサネット通信を行うことは想定しないものとする．

イーサネットコントローラを 2 つ用意して，高信頼 CPU と低信頼 CPU にそれぞれ占有させた場合は上記の要件の内（要件 1）・（要件 2）・（要件 3）・（要件 5）は満たすが，低信頼 CPU が割り当てられた自由にイーサネットコントローラを使用可能であるため，（要件 4）については充足しない．また，その他にも消費電力やハードウェア面積の増大，ケーブルやスイッチのポート等が必要となり，コストが増加してしまう問題がある．

3. ハードウェア仕様

この章では，本論文で使用したイーサネットコントローラと ProtectionWrapper（PW）について説明を行う．

3.1 イーサネットコントローラ

本論文では OpenCores にて公開されている “Ethernet MAC 10/100 Mbps” を利用した [5]．構成は図 2 のようになっており，送受信のフレームはバッファディスクリプタ（BD）によって管理される．

3.1.1 バッファディスクリプタ（BD）

BD は送受信の際に使用する機構で，送受信フレームを格納するメモリアドレスを保持するバッファポインタ部と送受信のステータスを管理する制御フラグ部で構成されている．BD は 128 個あり，先頭から任意の数を送信用に割り付けることができ，残りの BD は受信用に割り付けられる（図 3）．BD は送受信の際にラウンドロビンで使用する仕様になっている．

3.1.2 制御レジスタ

制御レジスタには送受信開始/停止を設定するレジスタや送信 BD 数を設定するレジスタ等が存在する．これらのレジスタ設定値が不正に書き換わると通信に問題が発生する可能性があるため，低信頼機能からのアクセスを制限する必要がある．

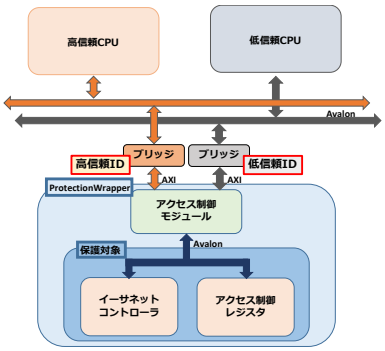


図 4　ProtectionWrapper のハードウェア構成

3.1.3　プロトコルスタック

プロトコルスタックとしては，オープンソースの uIP を使用した [6]．本研究で関連する BD の初期化処理および送受信処理を以降に記す．

- BD に関する初期化処理
  (1) 送信/受信 BD 数の設定を行う．
  (2) 各送信/受信 BD について，バッファポインタに各 BD で使用するメモリアドレスを割り当てる．
  (3) 各送信/受信 BD について，制御フラグの値を設定し，BD を使用可能にする．
  (4) 送受信フレームを格納するメモリをクリアする．
- 送信処理
  (1) 使用する BD のバッファポインタの指し示すアドレスに，送信フレームを書き込む．
  (2) 制御フラグの送信 READY フラグに 1 をセットする．
  (3) イーサネットコントローラがフレームを送信完了する (送信 READY フラグを 0 にする) のを待つ．
  (4) 制御フラグの送信 READY フラグが 0 になったのを確認して送信終了する．
- 受信処理
  (1) 受信割り込みをイーサネットコントローラから受け取り，受信処理を開始する．
  (2) 使用する受信 BD の制御フラグの EMPTY フラグが 0 であることを確認する．
  (3) BD のバッファポインタの指し示すアドレスに格納されているフレームを読みこんで受信する．
  (4) 受信が完了したら，BD の制御フラグの EMPTY フラグに 1 をセットする．

3.2　ProtectionWrapper（PW）の仕様

PW は文献 [2] にて提案されたデバイス保護機構で，デバイスへのアクセス制御を行うハードウェア機構である．

3.2.1　ハードウェア構成

PW のハードウェア構成は図 4 のようになっている．アクセス制御レジスタで保持される保護領域の設定によって，アクセス制御モジュールでアクセスを制御を行う．バスから送られてくるデータにはアクセス元の ID が付与されており，この ID によってアクセス可能/不可能の制御やアクセス頻度の制限を行うことができる．アクセス違反が起きた際にはアクセス元にアクセス違反を割り込みで通知する．

3.2.2　保護領域

アクセス保護領域の設定はアクセス制御レジスタで設定することができる．アクセスを許可/拒否することができるアクセス属性には READ/WRITE の他，通常/特権アクセス，セキュア/非セキュアアクセス，データ/命令アクセス，アクセス元パーティション，アクセス周期，書き込み値マスク，書き込み値範囲が存在する．

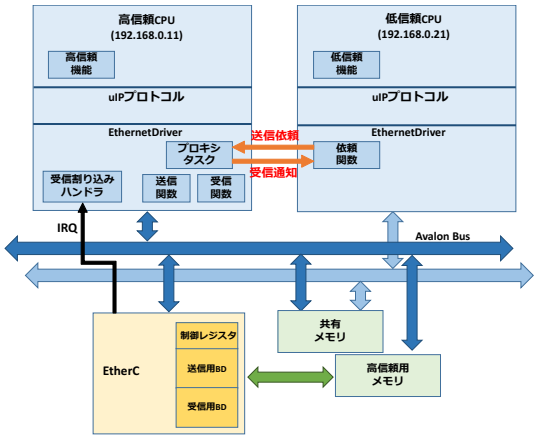


図 5　依頼方式のハードウェア構成

4.　実現方法

本論文では，2 章で述べた要件を満たすため，以下の共有方式を検討した．

- **依頼方式**：既存のイーサネットコントローラ（EtherC）を使用
- **直接操作方式（P-EtherC 版）**：PW を適用したイーサネットコントローラ（P-EtherC）を使用
- **直接操作方式（P-EtherC-Ex 版）**：機能追加したイーサネットコントローラ（P-EtherC-Ex）を使用

4.1　依頼方式

依頼方式は低信頼 CPU が送受信を行う際に，高信頼 CPU に処理を依頼する方式である．

そのため，ハードウェア構成は図 5 のようになっており，低信頼 CPU はイーサネットコントローラとバスで繋がっていない．高信頼 CPU と低信頼 CPU に別々の IP アドレスを割り当て，各 CPU の uIP スタックによって通信を行う．低信頼 CPU が通信を行う際には，送受信のデータを高信頼 CPU と低信頼 CPU の間で受け渡す作業が必要になるため共有メモリが用意されている．以降に具体的な低信頼 CPU の送受信の手順を述べる．

4.1.1　低信頼 CPU の送信方法

(1) 低信頼 CPU が共有メモリに送信フレームを書き込む．
(2) 低信頼 CPU が高信頼 CPU に割り込みを入れて送信を依頼する．
(3) 高信頼 CPU のプロキシタスクが起床し，低信頼 CPU の送信フレームを共有メモリから高信頼用メモリにコピーして，送信関数を呼び出して送信を行う．

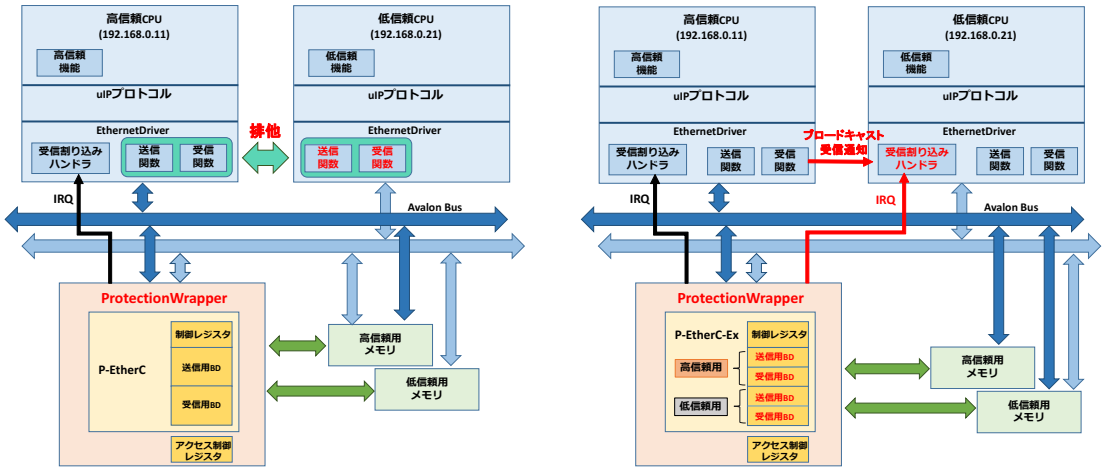
4.1.2　低信頼 CPU の受信方法

(1) 高信頼 CPU がイーサネットコントローラから受信割り込みを受けて受信関数を呼び出し，高信頼用メモリにある受信フレームを確認する．
(2) 受信フレームの宛先 IP アドレスを確認し，低信頼 CPU 宛もしくはブロードキャストであれば受信フレームを高信頼用メモリから共有メモリにコピーする．
(3) 高信頼 CPU のプロキシタスクが低信頼 CPU に割り込みで受信を通知する．
(4) 低信頼 CPU が共有メモリに格納された受信フレームを読み込み受信する．

4.1.3　要件充足

依頼方式を用いた際に，2 章で挙げた要件を充足するか確認を行う．

（要件 1） 低信頼 CPU がイーサネットコントローラにアクセスできないことから，低信頼 CPU がイーサネッ



(1) P-EtherC版　　(2) P-EtherC-Ex版

図 6　直接操作方式のハードウェア構成

トコントローラを不正に操作する危険性が存在しないために要件を満たす．

（要件 2）低信頼 CPU の送受信の際に高信頼 CPU に処理を依頼する必要があり，その際にオーバーヘッドが生じてしまうため要件を満たさない．

（要件 3）低信頼 CPU が送受信を行うたびに，高信頼 CPU とやり取りを行う必要があり，インタラクションの頻度が高い．

（要件 4）高信頼 CPU が送受信を行っている際には低信頼 CPU から送受信の依頼があっても処理を待たせれば良いので，低信頼 CPU によってネットワーク帯域が専有されることはなく，要件を満たす．

（要件 5）受信処理の際に宛先 IP アドレスの確認処理が高信頼 CPU の受信関数内で必要になるため，若干のソフトウェア変更が必要になる．

4.2　直接操作方式（P-EtherC 版）

4.1 節で述べた手法では 2 章で挙げた要件をほとんど満たさない．そのため，直接操作方式によって安全にイーサネットコントローラを共有するために，PW を EtherC に適用した P-EtherC を実現した．

4.2.1　ハードウェア概要

P-EtherC のハードウェア構成は図 6 の（1）のようになっている．具体的なハードウェア変更点は以下に記す 3 点である．

(1) 低信頼 CPU とイーサネットコントローラをバスで接続する．
(2) PW によって低信頼 CPU によるイーサネットコントローラの制御レジスタへのアクセスを不可能とする．
(3) 低信頼 CPU は BD にアクセス可能だが，PW によってアクセス頻度が制限されるようにする．

（1）は，低信頼 CPU が直接イーサネットコントローラを操作できるようにするためである．（2）により，制御レジスタの不正な書き換えによる送受信停止は起こらなくなり，（要件 1）を一部充足するようになる．また，制御レジスタへの操作は高信頼 CPU のみが可能なため，イーサネットコントローラの初期化は高信頼 CPU が行い，低信頼 CPU は初期化が完了するまでイーサネットコントローラの使用を待つ必要がある．（要件 3）は，送受信の処理の際に BD は高信頼 CPU と低信頼 CPU で共用されるため，低信頼 CPU からのアクセス頻度の制限を行うことによって，ネットワーク帯域を低信頼 CPU が占有できないようにした．これにより（要件 4）を満たすことができる．

4.2.2　低信頼 CPU の送信方法

各 CPU が別々に送信関数を呼び出して送信を行う．しかし，各 CPU が同時に同じ BD を使用しないようにするために，スピンロックによる排他制御が必要になる．

4.2.3　低信頼 CPU の受信方法

受信も送信と同様に CPU 間で排他制御をする必要である．以下に具体的な受信処理の手順を示す．

(1) 高信頼 CPU がイーサネットコントローラから受信割り込みを受け，受信関数を呼び出す．
(2) 受信関数内で宛先 IP アドレスを確認し，ブロードキャストもしくは低信頼 CPU 宛であれば低信頼 CPU に割り込みで受信を通知する．
(3) 低信頼 CPU が割り込みを受け，受信関数を呼び出して高信頼用メモリから受信フレームを読み込む．

4.2.4　要件充足

P-EtherC を用いて直接操作方式でイーサネットコントローラを共有した際に，2 章で挙げた要件を充足するか確認を行う．

（要件 1）この要件は十分に満たさない．PW によりイーサネットコントローラの制御レジスタを低信頼 CPU が不正に操作する危険性は存在しないが，BD については低信頼 CPU もアクセス可能にしないと送受信が行えないため，高信頼 CPU と低信頼 CPU で BD を共有している．そのため，BD を低信頼 CPU が不正に操作すると，高信頼 CPU の送受信が滞ってしまう危険性がある．

（要件 2）低信頼 CPU 送信の際のオーバーヘッドはなくなったが，低信頼 CPU 受信の際には高信頼 CPU が一旦フレームを受信してから宛先 IP アドレスを確認する処理が必要になるため，オーバーヘッドが生じてしまう．そのため要件は満たさない．

（要件 3）送信の際には高信頼 CPU と低信頼 CPU 間でやり取りを行う必要がなくなったため，依頼方式と比べて CPU 間のインタラクションの頻度は低くなった．しかし，受信の際には高信頼 CPU と低信頼 CPU 間でやり取りを行う必要がある．

（要件 4）低信頼 CPU から BD へのアクセス頻度を PW によって制限することによって，低信頼 CPU がネットワーク帯域を占有するのを防ぐことができる．

（要件 5）ドライバの受信関数にて宛先 IP アドレスの確認が必要になるため，若干のソフトウェア変更が必要になる．

4.3 直接操作方式（P-EtherC-Ex 版）

P-EtherC 版では 2 章で挙げた要件を十分に満たさない．この問題を解決するために P-EtherC の機能を拡張する．この拡張したイーサネットコントローラを以降では P-EtherC-Ex と呼ぶ．

4.3.1 ハードウェア概要

P-EtherC-Ex のハードウェア構成は図 6 の（2）のようになっている．以降で具体的なハードウェア変更点について述べる．

(1) バッファディスクリプタ構成の変更
図 6 の（2）に示すように，BD を高信頼用と低信頼用に分割した．このようにすることで，PW によるアクセス制御が可能になり，低信頼 CPU は高信頼用 BD 領域にアクセスできなくなる．また，低信頼 CPU の低信頼用 BD にアクセスする頻度を PW で制限することにより，ネットワーク帯域を低信頼 CPU が占有しないようにした．

(2) 宛先 IP アドレス判別処理の追加
外部からイーサネットフレームを受信した際の宛先 IP アドレスの判別をイーサネットコントローラで行うよう変更を行った．イーサネットフレームの 3 つの部分について値の確認を行う．まず受信したフレームのプロコトルタイプが 0x0800(IP プロトコル) であるかどうかの確認を行う．そして受信したフレームが IP プロコトルであれば，IP プロコトルのバージョンが 0x0100（IPv4）であるかの確認を行う．そして最後に，宛先 IP アドレスが低信頼 CPU の IP（レジスタ設定値）と一致するかの比較を行う．受信したフレームの宛先が低信頼 CPU 宛であった場合は低信頼用のディスクリプタを使用し，低信頼 CPU に受信割り込みを入れる．

(3) 低信頼 CPU 用の割り込みの追加
イーサネットコントローラから低信頼 CPU に向けた割り込みを追加し，宛先 IP によって割り込み先の CPU を決定するようにした．

(4) PW のマスタ側の保護の追加
既存の PW はスレーブポート側の保護機能しか持っていなかった．しかし，イーサネットコントローラは外部のメモリに送受信フレームを保持するため，マスタ側の保護が必要である．そのため PW を拡張してマスタ側のアクセス保護を追加し，使用している BD が低信頼用 BD であれば高信頼のメモリにアクセスできないようにした．

4.3.2 低信頼 CPU の送信方法

P-EtherC-Ex を用いた際の直接操作方式による送信では，送信方法は P-EtherC を用いた場合と変わらないが，BD が信頼度ごとにパーティショニングされたため，CPU 間で排他制御する必要がなくなった．

4.3.3 低信頼 CPU の受信方法

受信についても送信と同様に排他制御の必要がなくなった．具体的な受信処理の手順は以下の通りである．

- 宛先 IP アドレスが低信頼 CPU 宛の場合
  (1) イーサネットコントローラが受信したフレームの宛先 IP アドレスを確認し，宛先 IP アドレスが低信頼宛であれば，低信頼用 BD を使用し，低信頼用メモリに受信したフレームを書き込む．
  (2) イーサネットコントローラが低信頼 CPU に受信割り込みを入れる．
  (3) 低信頼 CPU が受信割り込みを受け，受信関数を呼び出して受信を行う．
- 宛先 IP アドレスがブロードキャストの場合
  (1) イーサネットコントローラが受信したフレームの宛先 IP アドレスを確認し，ブロードキャストであれば高信頼用 BD を使用し，高信頼用メモリに受信したフレームを書き込む．
  (2) イーサネットコントローラが高信頼 CPU に受信割り込みを入れる．
  (3) 高信頼 CPU が割り込みを受け，受信関数を呼び出して受信を行う．
  (4) 高信頼 CPU の受信が完了したら，受信したフレーム低信頼用メモリにコピーする．
  (5) 高信頼 CPU が低信頼 CPU に割り込みでブロードキャストの受信を通知する．
  (6) 低信頼 CPU が割り込みを受け，自身のメモリからブロードキャストのデータを読み込む．

4.3.4 要件充足

直接操作方式 (P-EtherC-Ex) で，2 章で挙げた要件を充足するかを確認する．

（要件 1）バッファディスクリプタが信頼度ごとにパーティショニングされたため，PW により低信頼 CPU が高信頼 BD を不正に操作することはなくなり，高信頼 CPU の送受信が低信頼 CPU の動作不良により滞る危険性もなくなった．

（要件 2）低信頼 CPU 宛のフレーム受信時に高信頼 CPU が行う処理は一切なくなったため，その分のオーバーヘッドは解消された．しかし，ブロードキャスト受信時には依頼方式と同じ方式で低信頼 CPU が受信するためオーバーヘッドが発生する．そのため，要件を一部満たさない．

（要件 3）ブロードキャスト受信時以外で高信頼 CPU と低信頼 CPU が処理のやりとりをすることがなくなった．そのため，他の方式と比べ CPU 間でのインタラクションの頻度が低い．

（要件 4）低信頼 CPU が低信頼用 BD にアクセスする頻度を PW によって制限することによって，高信頼 CPU のネットワーク帯域を保証することができる．

（要件 5）イーサネットコントローラの仕様が一部変更されたため，ドライバでイーサネットコントローラを初期化する処理に一部変更がなされた．具体的には低信頼 CPU の IP アドレスの設定や，高信頼用 BD 数・低信頼用 BD 数の割当て処理が新たに必要になる．

5. 評価

前述の各方式についてハードウェアを FPGA 上に実装し，通信におけるエコーバック時間・ハードウェア面積・ソフトウェア変更量・ハードウェア変更量について評価を行った．評価に用いた環境は次の通りである．

- 評価ボード　Terasic 社 DE2-115（Cyclone IV）
- CPU　NiosII/f プロセッサ（50MHz）× 2
- OS　TOPPERS/ATK2-SC1-MC（AUTOSAR 仕様）

エコーバック時間およびハードウェア面積の評価の際には以下の 4 パターンについて測定を行った．

- 依頼方式（EtherC）
- 直接操作方式（P-EtherC）
- 直接操作方式（P-EtherC-Ex）
- 非共有方式（EtherC + P-EtherC）

非共有方式では，各 CPU にイーサネットコントローラを割り当てたデザインを使用し，高信頼機能の帯域保証のために低信頼 CPU のイーサネットコントローラには PW を適用してアクセス頻度の制限を行っている．

5.1 エコーバック時間

エコーバック時間の測定では，PC と FPGA が 100Mbps で通信を行い，PC から FPGA 上の各 CPU 宛に PING 要求を送りエコーバックするまでの時間を測定した．使用した PC は Windows7 の 64bitOS で．測定の際にはネット

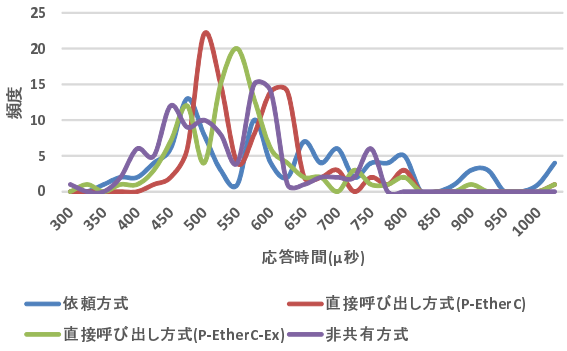


図 7　高信頼 CPU のエコーバック時間測定結果

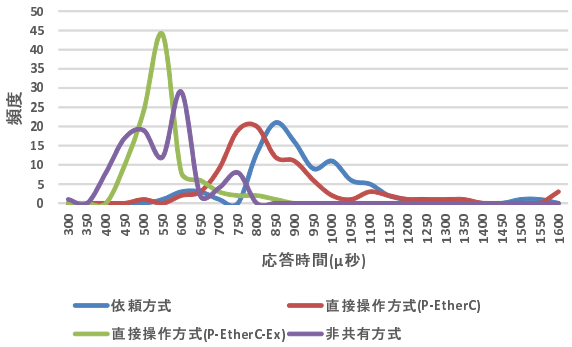


図 8　低信頼 CPU のエコーバック時間測定結果

ワーク・アナライザ・ソフトウェアである WireShark を使用した．測定結果は図 7・図 8・表 1 のようになった．測定した結果から非共有方式と各共有方式を用いた時のエコーバック時間の変化に関する考察を行う．

高信頼 CPU の送受信の処理は，依頼方式で行う場合も直接操作方式で行う場合も非共有方式と同じ手順によって行われる．しかし，受信の際に宛先 IP アドレスの確認処理が依頼方式および直接操作方式の両方で必要になるため多少オーバーヘッドが生じている．

低信頼 CPU のエコーバック時間については，依頼方式を用いた場合のエコーバック時間は非共有方式と比較して 78%増加している．この増加分が依頼処理ために発生するオーバーヘッドに相当する．また，非共有方式と比較して，エコーバック時間の分散が大きくなっている．これは低信頼 CPU が処理を依頼する際に CPU 間で割り込みが発生し，割り込み先 CPU の動作状況によっては処理を待たされるためである．

直接操作方式（P-EtherC）を用いた場合の低信頼 CPU のエコーバック時間は非共有方式と比較すると 63%増加しており，オーバーヘッドが削減しきれていない．これは送信処理に関しては改善されているが，受信処理に関しては，依頼方式と同じように高信頼 CPU が先に受信して宛先 IP アドレスを確認する処理が必要になるためである．また，エコーバック時間の分散も依頼方式と同じく大きくなっており，これは受信処理の際に各 CPU 間で割り込みが発生するためである．

直接操作方式（P-EtherC-Ex）を用いた場合の低信頼 CPU のエコーバック時間は非共有方式と比較して 1.7%しか増加していない．イーサネットコントローラの機能拡張により，ブロードキャスト以外は低信頼 CPU のみで独立して送受信が行えるようになったため，エコーバック時間の増加を抑えることができた．

表 1　平均エコーバック時間（μ秒）

	高信頼 CPU	低信頼 CPU
依頼方式	627	933
直接操作方式（P-EtherC）	564	854
直接操作方式（P-EtherC-Ex）	544	533
非共有方式	524	524

表 2　ハードウェア面積比較

	LUT 数
EtherC	2,941
P-EtherC	5,849
EtherC-Ex	3,133
P-EtherC-Ex	6,794

5.2　ハードウェア面積

ハードウェア面積の評価は FPGA に実装する際に使用されたルックアップテーブル数（LUT 数）の値を比較することにより行った．それぞれのイーサネットコントローラのモデルについて測定を行った結果は表 2 のようになった．

測定した結果から非共有方式のハードウェア面積は 8790LUT 数（EtherC + P-EtherC）である．これと各共有方式でのハードウェア面積を比較すると，依頼方式（EtherC）では約 67%，直接操作方式 (P-EtherC) では約 33%，直接操作方式 (P-EtherC-Ex) では約 23%減少した．

5.3　ソフトウェア変更量

各方式での送受信を実現する際に行ったソフトウェア変更量および変更箇所の詳細は以下の通りである．

(1) 依頼方式：38 行
- 受信フレームの宛先 IP 確認
- 依頼に伴う低信頼 CPU と高信頼 CPU 間のデータ受け渡し処理

(2) 直接操作方式 P-EtherC 版：47 行
- 受信フレームの宛先 IP 確認
- 受信フレームがブロードキャストか低信頼 CPU 宛かによる分岐処理

(3) 直接操作方式 P-EtherC-Ex 版：34 行
- 受信フレームがブロードキャストかどうかの確認
- イーサネットコントローラ拡張に伴う初期化処理の追加

5.4　ハードウェア変更量

イーサネットコントローラを機能拡張するべく，343 行を変更した．具体的には受信フレームの宛先 IP アドレス判別を行う回路の追加と BD 分割に伴う回路変更である．

謝辞

本研究の一部は，(株) 半導体理工学研究センターとの共同研究による．

参考文献

[1] Sanjoy Baruah, Haohan Li, Leen Stougie: Towards the design of certifiable mixed-criticality systems（2010），
[2] 加藤祐輔：ミックスドクリティカルシステム向けデバイス共有機構（2016），
[3] Peter Hank, Ovidiu Vermesan, Steffen Muller,Jeroen Van Den Keybus: Automotive ethernet: in-vehicle networking and smart mobility (2013)，
[4] Lucia Lo Bello: The case for Ethernet in Automotive Communications (2011)，
[5] OpenCores:Ethernet_MAC_10/100Mbps（https://opencores.org/project,ethmac），
[6] Adam Dunkels:The uIP Embedded TCP/IP Stack,（2006）