平成２１年度戦略的基盤技術高度化支援事業

「高級言語(C#)によるカスタマイズ可能なネットワークアプリケーションプロセッサのハードウェア論理開発環境」

研究開発成果等報告書

平成２２年３月

委託者関東経済産業局

委託先　株式会社イイガ

内容

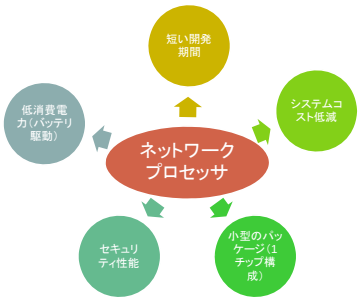
第１章研究開発の概要

１−１研究開発の背景・研究目的及び目標

１−１−（１）研究開発計画概要

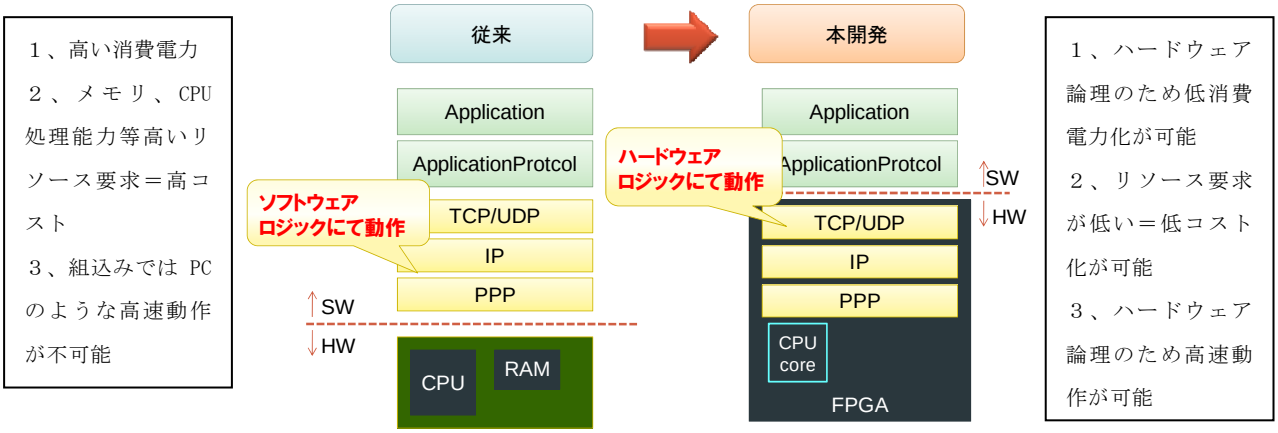
（１）計画名：高級言語（C#）によるカスタマイズ可能なネットワークアプリケーションプロセッサのためのハードウェア論理開発環境

【現状】ユビキタスアプリケーションの需要の高まりにつれて、ネットワークプロセッサへの様々な要望は増えているが、CPU ベースでは市場の要求にこたえられない。



【本提案】

1. 情報家電や携帯電話の低消費電力、高速化のためのネットワークプロトコルのハードウェア論理ブロックの実現。



2. 高級言語（C#）によって記述されたネットワークアプリケーションのソフトウェア・ハードウェアミクスド動作・開発環境の実現。

１−１−（２）研究開発の背景

ア．ネットワークサービスの多様化

これまで、弊社では、様々なネットワークアプリケーション機器のための組込み開発を各種ネットワークプロセッサ（SoC）等で行ってきました。しかし、ユビキタス時代となり、携帯電話との連携した機器等の製品を開発するに当たっては、様々なネットワークプロトコルを必要とし、またバッテリ長時間駆動を必要とする機器では、CPU ベースの SoC での実現では、コスト、消費電力と言う面で困難であることが少なくないと感じております。現在まで、弊社では、16bit/8KB RAM 等の環境に TCP/IP、HTTP、HTTPS 等のプロトコルを搭載するなど、低リソース化＝低コスト化を実現するために様々な研究開発を行ってきました。

このようなソフトウェアにより実現される TCP/IP などネットワークプロトコルの実装では、ソフトウェアで実現していることによる、CPU 性能等に依存した通信スループットの問題や、通信が行なわれ

る際の CPU の活動による消費電力への課題を克服しえないため、TCP/IP プロトコルをハードウェアで実装した論理ブロックが必要と考えております。

このような研究としては、電気通信大学情報工学科では、トランスポート層にTCPを持つプロトコルスタックをハードウェア化するためFPGAを用いた研究が行なわれています。実装の結果、TCPプロトコルスタック全体が、60万ゲートFPGAを用いれば、1チップに搭載できる規模であると、研究報告されております。東京都立大学大学院工学研究科では、TCP/IP処理用回路の設計とFPGAによる実装が行なわれ、設計した回路には、 TCP/IPの機能の一部とPOP3を処理するための機能が備えた状態で、約20,000ゲート規模のFPGAデバイスで動作を確認したと、研究報告されております。倉敷芸術科学大学大学院産業科学技術研究科では、FPGAを利用し、内部メモリなどのFPGA機能の有効利用と通信プロトコルの単純化により、ホストインタフェースから小規模スイッチまでのすべてのネットワーク機能を1個のFPGAで実現。リング型PCクラスタを構成して性能評価した結果、100baseT(MPICH,TCP/IP)による通信に対して約3倍の高バンド幅と1/6以下の低レイテンシ生態が得られたと、研究報告されております。

本提案では、15万ゲート程度にて、TCP/IPのフル機能を実現し、その他、情報家電や携帯電話に必要とされるプロトコルの搭載も行いながら、スループットの高速化、商品化可能な実用的なプロトコルの実装、低消費電力という3つ同時に実現することをひとつの目標とします。また、このネットワークプロトコルのハードウェア論理ブロックをソフトウェアエンジニアが容易に利用可能とするためにC#という高級言語で書かれたアプリケーションから利用可能な動作・開発環境を提供することを二つ目の目標とします。

１−１−（３）研究開発の内容

1.情報家電や携帯電話の低消費電力、高速化のためのネットワークプロトコルのハードウェア論理ブロックの実現。

(１)TCP/IP プロトコルをハードウェアで実装した IP コアの開発

TCP/IP プロトコルをハードウェアで実装した IP コアを開発します。開発新製品の技術的特徴としては、ハードワイヤードによる TCP/IP プロトコルの実装であることと、ハードワイヤードで実装を実現することで、低消費電力化が可能であることを特徴とする IP コアであることです。

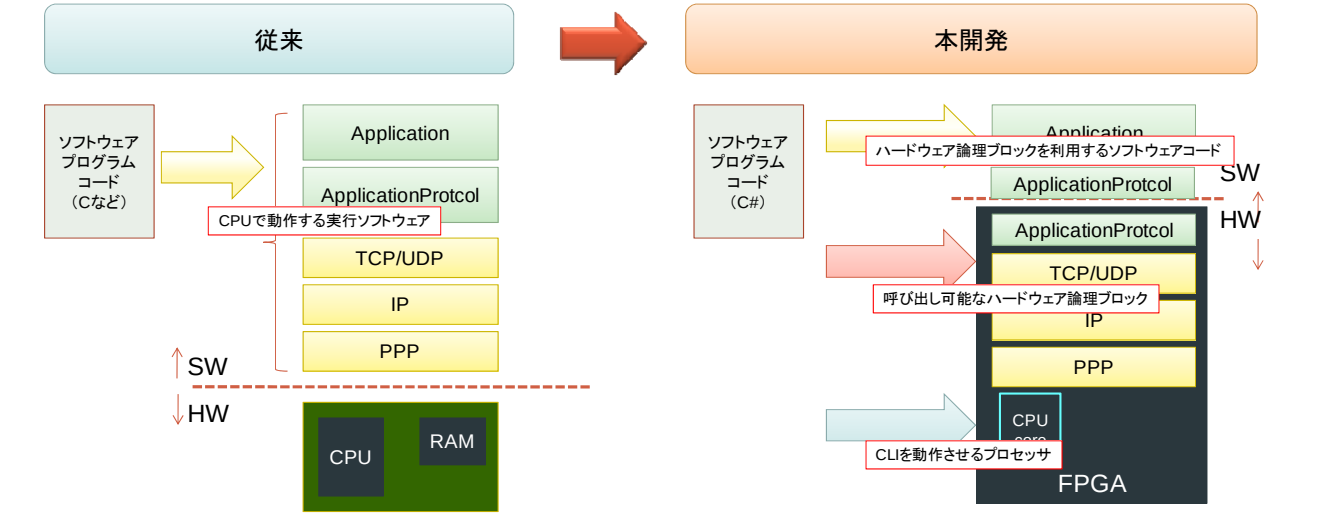
(２)HTTP や SNMP、UPnP といったプロトコルのハードウェアでの実装

アプリケーションとしては、TCP/IP のみでの通信が行なわれることは、希であるため、IP コアで TCP/IP レイヤーより上位のアプリケーションレイヤーのプロトコルが市場で需要があり、それらを含めた IP コアを実現可能かの検討課題があります。東京都立大学大学院工学研究科では、TCP/IP の他に、POP3 をプロトコルとして実装しておりますが、組み込み用とでは、ほとんど使用されていないプロトコルであるため、組込み用途として、一般的に必要となる HTTP や SNMP、UPnP といったプロトコルのハードウェアでの実装を行います。

(３)低レイテンシ、低消費電力化へのチューニング

従来品と比較した場合、ハードワイヤードにすることによる TCP/IP 高速データ通信における低レイテンシ、低消費電力化が期待されますので、倉敷芸術科学大学大学院産業科学技術研究科での研究報告である、100baseT(MPICH,TCP/IP)による通信に対して 1/6 以下の低レイテンシを目標としております。また、従来の微細プロセスを使った高集積な FPGA の開発では、リーク電流や消費電力の低減が重要な課題でしたが、本開発では、動作時に数十 mW(ミリワット)台を実現可能な IP コアを目指しています。

2. 高級言語（C#）によって記述されたネットワークアプリケーションのソフトウェア・ハードウェアミクスド動作・開発環境の実現。



C、C++などで記述されたソフトウェアを CPU 上で動作

- CPU アーキテクチャ毎にソフトウェアの変更が必要
- C、C++などの高度な組込みエンジニアが必要

C#という手軽な開発言語で開発が可能、

- ネットワークプロトコルのハードウェア論理ブロック、CLI プロセッサ　ハードウェア論理ブロックを結合可能なソフトウェアを生成

（１）CLI プロセッサのハードウェア論理での実装

ECMA-335、(JIS) X3016 にて標準化されている中間言語 CLI を FPGA で実行するためのスタックマシン型プロセッサ　ハードウェア論理ブロックを開発します。

（２）スタックマシン型プロセッサのチューニング

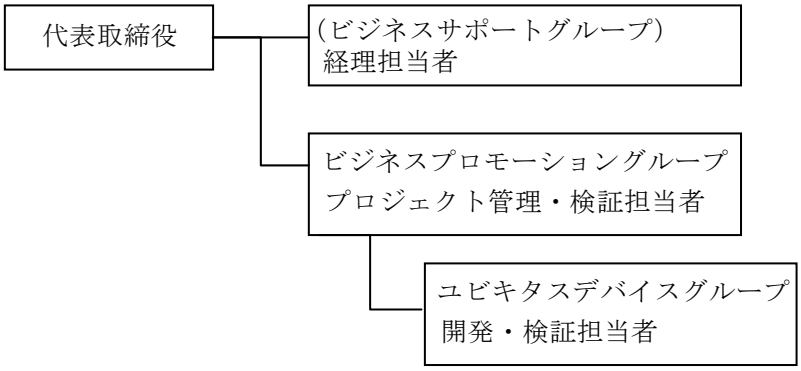
開発するスタックマシン型プロセッサは，スタック，ローカル変数領域をレジスタ上に実現し高速化をはかり、また入出力パスを用意してスタックへの読み書きが１クロックで行えるようにするなどの高速化を考慮し設計します。

（３）C#でハード/ソフトのミックスシステムの開発環境

高級言語（C#）によって記述されたネットワークアプリケーションのプログラムコードからネットワークプロトコルのハードウェア論理ブロックと、スタックマシン型プロセッサ　ハードウェア論理ブロック、その CPU ハードウェア論理の上で動作するアプリケーションソフトウェアコードを生成する開発環境の実現。

１−２研究体制

１−２−（１）研究組織



１−２−（２）責任者および研究者

研究員等氏名	役　職
安部則孝	代表者（代表取締役）
藤沼健太郎	主任研究員
谷脇　真人	副主任研究員
塩川　裕康	副主任研究員
平田優	研究員
吉田隆興	研究員
寺田　仁郎	研究員
駒井　良一	研究員
遠藤　央	研究員

１−２−（３）参画企業および法人所在地

株式会社　イイガ

東京都千代田区外神田３－５－２

１−３成果概要

１−３−（１）研究実施期間

平成 21 年 10 月 1 日～平成 22 年 3 月 31 日

１−３−（２）成果概要

本研究期間には、TCP/IP プロトコルをハードウェアで実装した IP コアの設計は、CPU と TCP/IP ハードウェア IP コアを FPGA 上にインテグレートした場合でも、CPU を外付け可能な IP コアを設計し、外部 CPU からの制御を可能にした場合でも、順調に機能し、動作検証することが出来た。また、CLI プロセッサは組み込み型 CPU 上で動作できることが確認できた。実装したライブラリはシュミレータ上の範囲ではアプリケーションの動作が確認できたことで、基本構想段階で目標にしていた部分は達成できたと考えている。

第２章　研究開発報告

２－１情報家電や携帯電話の低消費電力、高速化のためのネットワークプロトコルのハードウェア論理ブロックの実現。

２－１－（１）本テーマの目的

昨今のデジタル機器には、ネットワーク機能は必須であり、そのためデバイス全体のシステムにかかるネットワーク機能の負荷を減らすことはシステム設計の必須要件となっている。しかし、ネットワーク機能を実現する TCP/IP プロトコルスタックは OS の上で動作させるソフトウェアとして実現することが通例となっているため、消費電力・必要プロセッサ計算能力等が過大となっている。

２－１－（２） 研究の内容

２－１－（２）－①TCP/IP プロトコルをハードウェアで実装した IP コアの開発

２－１－（２）－①－[１]開発方針

開発方針としては、第一段階として、CPU と TCP/IP ハードウェア IP コアを FPGA 上でインテグレートした IP を設計し、他の開発テーマに利用しやすいプラットフォームとします。第 2 段階として、CPU を外付け可能な IP コアを設計・開発し、外部 CPU からの制御を可能にしたプラットフォームを開発します。その後の第三段階では、低消費電力の FPGA プラットフォームへのポーティングを行いう計画です。本期間では第 2 段階の動作検証まで行った。

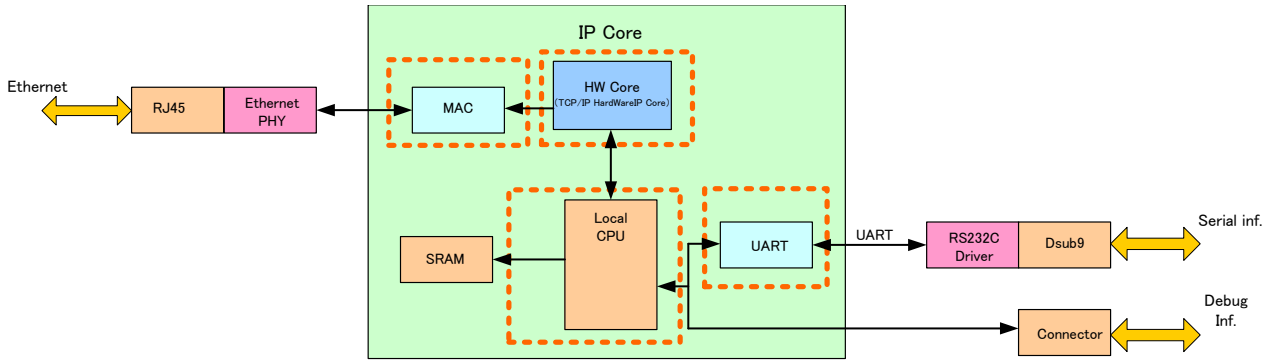
２－１－（２）－①－[２]設計仕様

２－１－２－①－２－１CPU と TCP/IP ハードウェア IP コアをインテグレートした IP コア

２－１－２－①－２－１－１CPU と TCP/IP ハードウェア IP コアをインテグレートした IP コア概要

以下は、CPU と TCP/IP ハードウェア IP コアをインテグレートした IP コアの概要である。

第一段階での IP コアの概要は以下。



- TCP/IP ハードウェア IP コア機能を搭載した HW Core を設計。
- 通信用 API を実行するための Local CPU を設計。
- デバッグ・各種 API 設定用のシリアルインターフェース (UART)を設計。

No	項目	内容	備考
1	ネットワークインターフェース	①Ethernet（10/100）x１	
2	汎用シリアルインターフェー	①UART（300～460.8Kbps） x １	各種設定用インターフェイ

	ス		ス
3	HW　Core　主要機能	①TCP 処理 ②Raw-IP 受信処理 ③IP 処理（ARP 送受信処理含む）	
4	On-Chip CPU SW 主要機能	①各種 API ②Configuration	

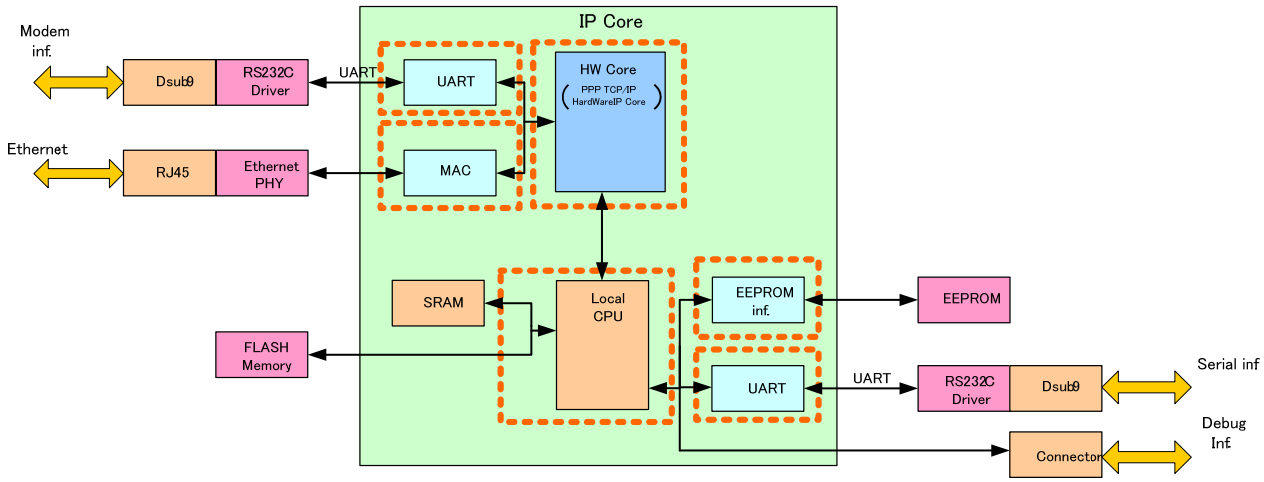
２－１－２－①－２－２CPU を外付け可能な TCP/IP ハードウェア IP コア

２－１－２－①－２－２－１CPU を外付け可能な TCP/IP ハードウェア IP コア概要

以下は、CPU を外付け可能な TCP/IP ハードウェア IP コアの概要である。外部 CPU と IP コアは UART にて通信を行えるように設計する。

また、第二段階では、第一段階で設計した CPU と TCP/IP ハードウェア IP コアを FPGA 上でインテグレートした IP コアの機能としての見直しも計る。

主な設計見直し点としては、情報家電や携帯電話を想定した PPP プロトコルの盛り込み、および、その設定情報保存のためのインターフェイス設計である。



- Modem（FOMA Module 等）を介した PPP 接続によりネットワーク接続を行うためのシリアルインタフェース(UART)を設計。
- PPP TCP/IP ハードウェア IP コア機能を搭載した HW Core を設計。
- 通信用 API、PPP ネゴシエーション、各種設定のソフトウェアを実行するための Local CPU を設計。
- 各種設定の保持用の EEPROM を接続するためのシリアルインターフェースを設計

カードリーダからの情報取得、および各種設定、外部 CPU を想定した通信用のシリアルインターフェース（UART)を設計。

No	項目	内容	備考
1	ネットワークインターフェース	①モデム（FOMA モジュール等）接続用 UART（300～460.8Kbps）×１	

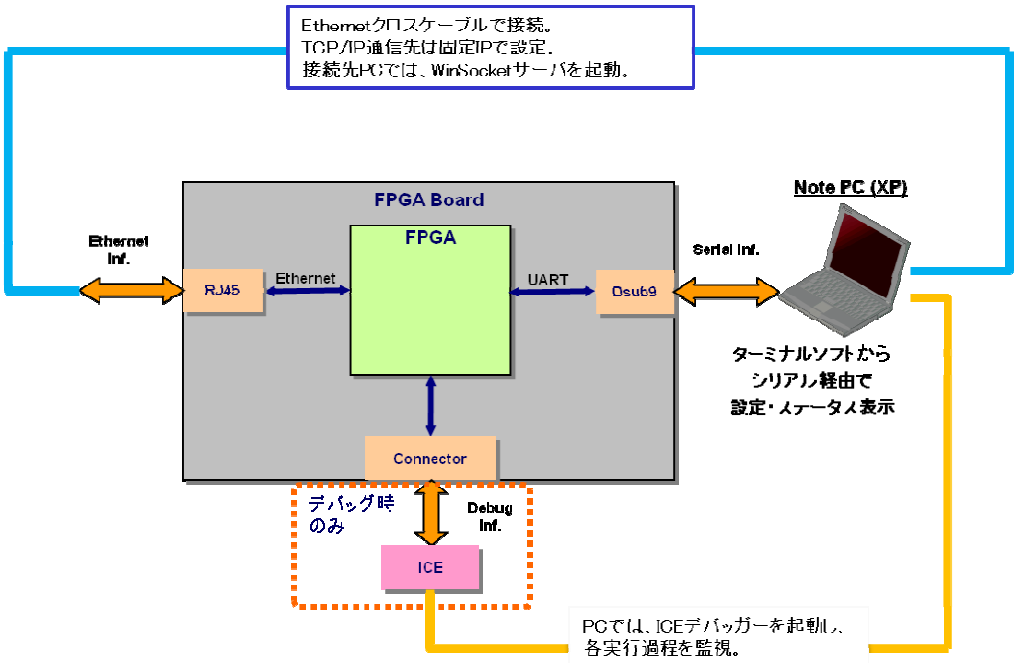
		②Ethernet（10/100）x 1	
2	汎用シリアルインターフェース	①UART（300～460.8Kbps） x 1 ②外付け FlashROM 用 SPI ③EEPROM 用 I2C	外部 CPU 用インターフェイス 各種設定・保存用インターフェイス
3	HW Core 主要機能	①TCP 処理 ②Raw-IP 受信処理 ③IP 処理（ARP 送受信処理含む） ④PPP 処理	PPP は第二段階で追加した機能。
4	On-Chip CPU SW 主要機能	①PPP 処理 ②各種 API ③Configuration	PPP は第二段階で追加した機能。

※赤字は、主な追加機能

２－１－２－①－２－２－２CPU を外付け可能な TCP/IP ハードウェア IP コア概要
本段階では、第一段階で設計した CPU と TCP/IP ハードウェア IP コアを FPGA 上でインテグレートした IP コアの機能の見直しを計り、以下で示す下線の機能を追加する。

２－１－（２）－①－[３]機能評価・測定

２－１－２－①－３－１CPU と TCP/IP ハードウェア IP コアをインテグレートした IP コア機能評価
第一段階では、FPGA 上に実装した API が正しく動作するか機能の評価を行なうため、以下のような評価環境を構築し、CPU と TCP/IP ハードウェア IP コアをインテグレートした IP コアでこれまで実装した各機能の単体での評価を行なった。



２－１－２-①-３－２CPU を外付け可能な TCP/IP ハードウェア IP コア機能評価
第二段階では、機能の評価を行なうため、以下のような評価環境を構築し、CPU を外付け可能な TCP/IP ハードウェア IP コアでこれまで実装した各機能の単体での評価を行なった。

２－１－（２）-②HTTP といったプロトコルの実装
２－１－（２）-②-[１]開発方針
開発方針としては、第一段階として、試作ボード①にて HTTP プロトコルを実装し、作成したコードの動作検証を行った後、第二段階として FPGA 開発ボードにて、Ｃ－ｔｏ Hardware コンパイラを用い、ハードウェア化する。第三段階として、TCP/IP ハードウェア IP コアとの結合、調整を行う。この手法にて、各種プロトコルの対応を行ったが、本期間では第二段階の実現方法の検証まで終了したところである。

２－１－（２）-②-[２]試作ボード①での実装
試作ボード①にてソフトウェア HTTP プロトコルを実装し、動作検証を行う。

２－１－２-②-２－１試作ボード①
試作ボード①では、ソフトウェア HTTP プロトコルおよび、ソフトウェア HTTP プロトコルのベースとなるプロトコルである IP、TCP、UDP といったプロトコルの実装を行なう

２-１-２-②-２-１-１試作ボード①仕様

項目	仕様
電源	DC5V、1.5A
使用環境	温度+5℃～＋45℃　湿度 70%RH 以下（結露なきこと）
保存温度	温度-20℃～＋60℃ 湿度 70%RH 以下（結露なきこと）
外部インターフェイス	Ethernet(1000BASE)x1
	UART（デバッグ・通信用）
音声出力	8Bit サンプリング音での出力が可能
押しボタン	電源用ボタンｘ１、アプリケーション用ボタンｘ１
外形寸法	約１０８mm×６６ｍｍ（ただし、突起部分は除く）
インジケータランプ	2 色 LED x 3 個
その他	予備 UART ポート（通信用）

２-１-（２）-②-[３]C-to Hardware による各種プロトコルのハードウェア化

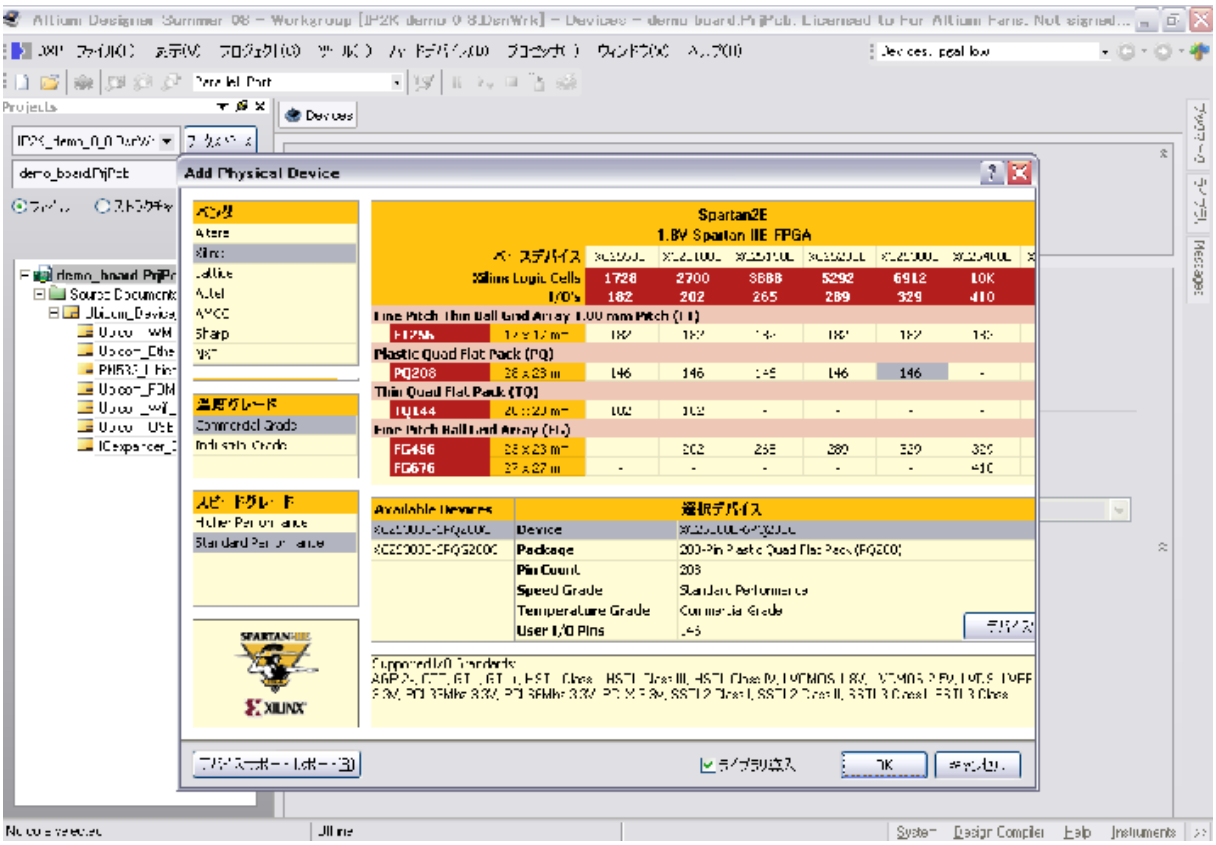
２-１-２-②-３-１C-to Hardware 概要

第一段階で、試作ボード①に実装したソフトウェア HTTP プロトコル、および、ソフトウェア HTTP を使用するために実装した IP、TCP、UDP といったプロトコルを、第二段階として FPGA 開発ボードにて、Ｃ－ｔｏ Hardware コンパイラを用い、ハードウェア化します。

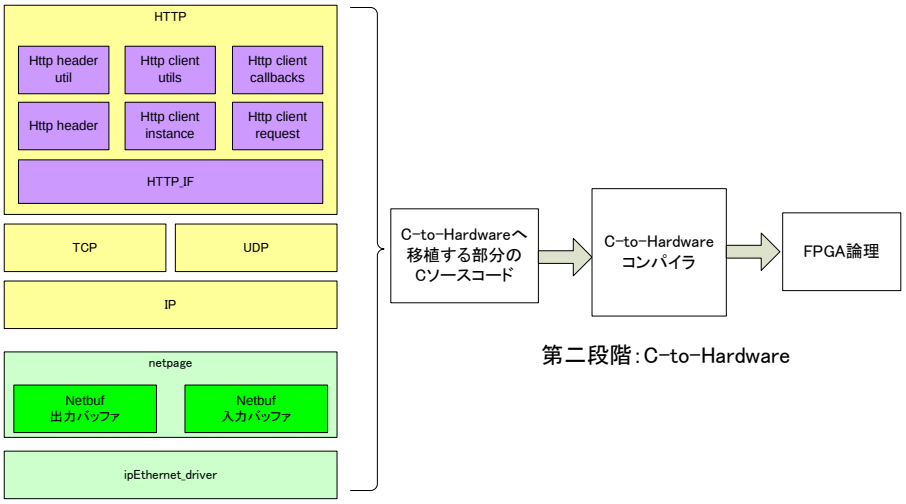
２-１-２-②-３-２C-to Hardware 開発環境

Altium Designer の FPGA 開発環境を用いて、C-to Hardware の実現を行ないます。

以下は、Altium Designer の画面です。



Ｃ－ｔｏ Hardware コンパイラを用い、ハードウェア化します。



第一段階:ソフトウェアプロトコル

第二段階:C-to-Hardware

２－１－２－②－３－３FPGA 開発支援ボード

Altium Designer 上で、TCP/IP ハードウェア IP コアと各種プロトコルの搭載を実現するために、以下のような Altium Designer と親和性の高く、且つローパワーな FPGA 環境であるメインボード NanoBoard NB2＋ドータボード EP3C40F780C8N にて行なう。本期間内に結合は完了しなかった。

メインボード NanoBoard NB2 仕様

タッチスクリーンを搭載した統合カラーTFT LCD パネル(320x240) により、アプリケーション間の連携を容易にします
高品質オンボードアンプ、ミキサ、ライン入出力およびステレオスピーカ付きステレオアナログオー

ディオシステム
各種標準通信インターフェイス - RS-232 シリアル、CAN、PS/2 ミニ DIN
各種ファイルのダウンロード機能など、I/O の柔軟性を付加する SD カードリーダ
4 チャンネル、8 ビット ADC と 10 ビット DAC、12C 互換
一般的な設計へのアプローチとして機能するユーザ定義可能な PDA スタイルの押しボタンスイッチ
各種汎用スイッチと LED
ターゲットの FPGA で使用できる、6～200MHz のプログラマブルクロック
パワーセンシングシステムにより、システムとデバイスの電力消費のリアルタイムモニタリングが可能
3V バッテリバックアップ付き SPI リアルタイムクロック
NanoTalk コントローラからアクセス可能なオンボードメモリ- 256K x 32 ビットコモンバス SRAM(1MB)、16M x 32 ビットコモンバス SDRAM(64MB)、16M x 16 ビットコモンバス 3.0V ページモードフラッシュメモリ(32MB)、256K x 32 ビット独立 SRAM(1MB)
デュアルユーザボード JTAG ヘッダによる量産ボード上での対話的設計と開発
Home/Reset ボタン - Home ボタンによりファームウェアで TFT パネルを制御可能、Reset により NanoBoard をリセット可能
NanoTalk コントローラ- JTAG アクセス可能な Flash コンフィギュレーション PROM を搭載した Xilinx® SpartanTM -3 (XC3S1500-4FG676C) を使用して、Altium Designer、ボード、および NanoBoard ファームウェアとのリアルタイムな独自の通信を管理します
開発ボードを複数接続するマスタ/スレーブコネクタにより、複数 FPGA システムの開発が可能
ボード ID メモリ - 1-Wire® ID システムがドータボードとペリフェラルボードを独自に識別します
電源- 電源スイッチ付きデュアル 5VDC 電源デイジーチェーンコネクタ、5VDC 電源出力コネクタ、ボード上で使用できるすべての電源レベルの電源テストポイント、4 つの GND ポイント
USB 2.0 による高速な PC との接続により、高速ダウンロードとデバッグを可能にします

ドータボード EP3C40F780C8N 仕様

パッケージ	780-Ball FineLine Ball Grid Array (FBGA780)
スピードグレード	8
温度グレード	商用向け
Pin 数	780
最大 User I/O ピン	535
最大個別 I/O ペア	219
Altera Logic Elements(LE 数)	39,600

搭載 RAM(Block 数表記)	1,161,216 bits (126 M9K RAM blocks: 8192 memory bits + 1024 parity bits per block)
Embedded Multipliers (18x18)	126 (each configurable as 9x9 multipliers)
Clock Managers (PLLs)	0
Global Clock Resources	20
必要設定メモリ	10,500,000 bits
On-Chip Termination Support	Yes
その他特徴	■On-board memories available for use by FPGA design: 256K x 32-bit common-bus SRAM (1MByte) 16M x 32-bit common-bus SDRAM (64MByte) 16M x 16-bit common-bus Flash memory (32MByte) Dual 256K x 16-bit independent SRAM (512KByte each) ■1-Wire® memory device used to store board ID and related information ■Three 100-way connectors for attachment to NB2DSK01 motherboard. These connectors provide: Interface to resources on the NB2DSK01 motherboard and plugged-in peripheral boards SPI bus interface I2C bus interface 1-Wire bus interface JTAG, power and additional control lines from the motherboard.

２－１－（２）-③低レイテンシ、低消費電力化へのチューニング

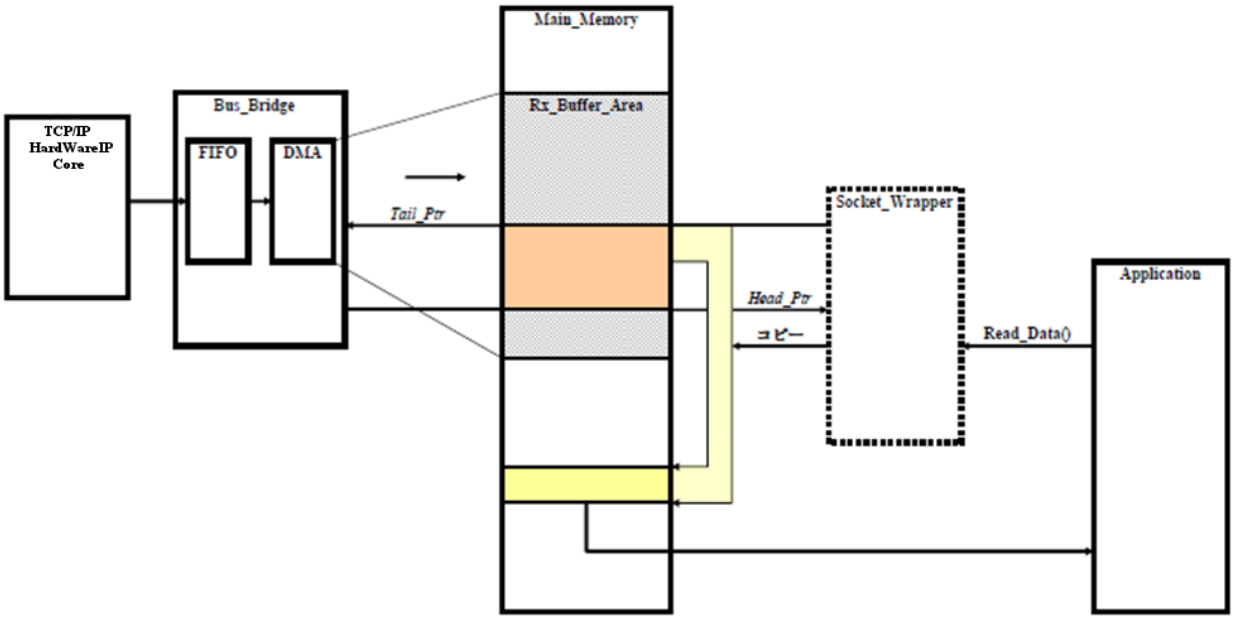
２－１－（２）-③-[１]実施方針

試作ボード①にソフトウェアにて実装された TCP/IP プロトコルスタックと TCP/IP ハードウェア IP コアとの性能検証を行いながら、各開発へフィードバックをかけました。

２－１－（２）-③-[２]試作ボード①と FPGA 開発支援ボード①での比較

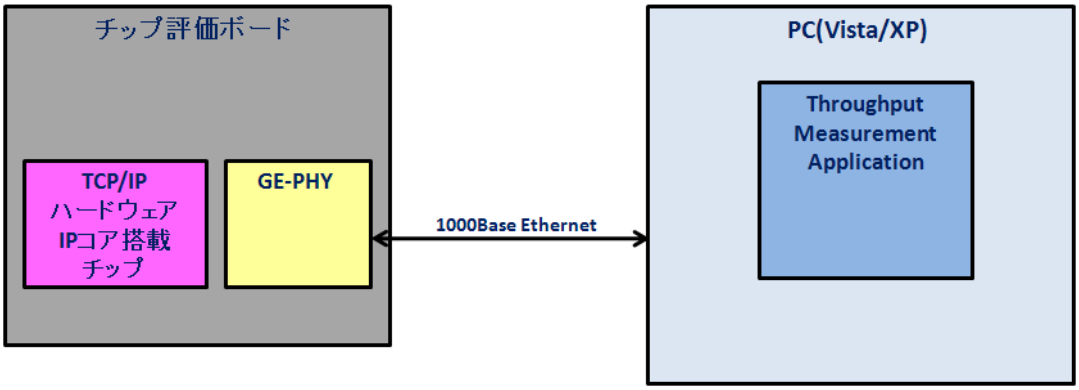
TCP/IP ハードウェア IP コアエンジンの効果を評価するため、以下のような評価環境を構築し、実システムにおけるスループット測定を行なった。

✧　スループット算出モデル(TCP/IP ハードウェア IP コアを用いた場合のモデル)



✧　実システムにおけるスループット測定

➢　評価システム構成



➢　評価方法

①PC 上にスループット測定 S/W を載せ、これを用いてデータ転送とスループット測定を実施。

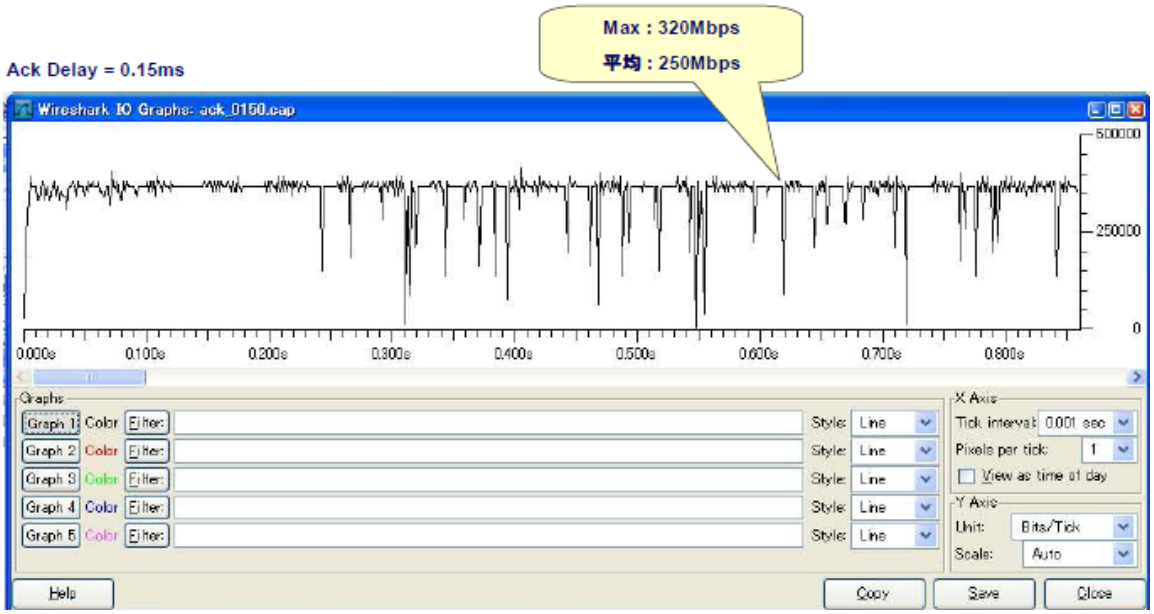
②５０M-Byte のデータ転送を行い、スループットを測定。これを 5 回繰り返し、平均を取る。

➢　評価結果

TCP/IP ハードウェア IP コア: 250Mbps（以下参照）

ソフトウェア TCP/IP　　: 56Mbps

XP 対向での測定時の設定（Window_Limit=16K, HW_Threshold=12K, SW_Threshold_L= 2K, SW_Threshold_H=2K）



✧　スループット測定値からの処理能力推定

①TCP/IP ハードウェア IP コアありの場合: M-Byte あたりの処理に要する時間は 32msec であった。

②ソフトウェア TCP/IP の場合: M-Byte あたりの処理に要する時間は 143msec であった。そのうち、111msec が TCP/IP 処理であった。

上記の結果より、実システムにおけるスループット測定では、111/143 = 77%の効果があるものと推測する。

２-１-（３）まとめ

情報家電や携帯電話の低消費電力、高速化のためのネットワークプロトコルのハードウェア論理ブロックのまとめとしては、TCP/IP プロトコルをハードウェアで実装した IP コアの開発は、CPU と TCP/IP ハードウェア IP コアを FPGA 上にインテグレートした場合でも、CPU を外付け可能な IP コアを開発し、外部 CPU からの制御を可能にした場合でも、順調に機能し、動作検証も行えたことと、また、TCP/IP ハードウェア IP コア動作検証も行え、スループットの測定までできたことから、順調に進んでいると考える。今後は、第三段階で、低消費電力の FPGA プラットフォームへのポーティングを行なうが、選定した FPGA である SiliconBlue 社の FPGA プラットフォームや、Low Power Reference Platform (LPRP)での検討をすすめることとする。

一方で、CLI プロセッサのハードウェアでの実装で使用した FPGA では、今後様々な FPGA 環境への対応も視野に入れる必要がある。例えば、現在のところは、32bitCPU アーキテクチャである ARM コアを搭載可能な FPGA は限られているが、今後はローパワーな FPGA でも機能を絞りつつも対応可能となることが予想される。

そこで第三段階では、現在、低消費電力の FPGA プラットフォームへのポーティングを行ないつつ、平行して、今後、ARM コアを搭載できる可能性のある FPGA 自体、または対応開発環境

の調査も行なうことが良いと考える。
　TCP/IP ハードウェア IP コアの測定が行えたことから、ソフトウェアで実装したプロトコルや、C-to-Hardware で実装したソースコード全体において、更に省略可能な変数が無いか、コードの記述方式による見直し点が無いかを検討し、更に高速な TCP/IP ハードウェア IP コアとする必要性もあると考えられる。

２-２高級言語（C#）によって記述されたネットワークアプリケーションのソフトウェア・ハードウェアミクスド動作・開発環境の実現。
２-２-（１）本テーマの目的と設計仕様
２-２-（２）研究の内容
２-２-（２）-①CLI プロセッサのハードウェアでの実装
２-２-（２）-①-[１]開発方針
開発方針としては、第一段階として、PC 上のシミュレータで動作する CLI プロセッサの実装をします。第二段階として FPGA に搭載可能な CPU コアと同一アーキテクチャの半導体で動作する CLI プロセッサを開発します。第三段階として FPGA に CPU コアを搭載し、その上で CLI プロセッサを動作させます。

２-２-（２）-①-[２]CLI プロセッサの実装
第一段階として、PC 上のシミュレータで動作する CLI プロセッサの実装をします。

２-２-２-①-２-１CLI プロセッサの構造と各機能の説明
CLI プロセッサの構造図は以下のようになっています。
CLI プロセッサは、大きく分けて、CPU に相当する Virtual Machine(以下では VM と呼ぶ)部とアプリケーション用のライブラリ部に分割されます。

- Application Framework

  .NET アプリケーションから VM を呼び出すことを可能にするためのインターフェイスおよび補助クラス郡です。ビルド時に使用される VM Interface for VS と実行時に使用される VM Interface by C の２つにわかれています。

  VM Interface for VS には VM API のラッパーライブラリが含まれています。

- CIL Bytecode Interpreter

  CIL(Common Intermediate Language)を解釈することで、.NET で書かれた VM アプリケーションを実行します。Application Framework で変換・生成された VM アプリケーションのバイトコードを実行する機能です。

- OO Engine

  OO Engine の OO とは Object Oriented の略であり、メモリ管理機能(ガベージコレクション：GC)、メソッド呼出し機構(Message Dispatcher)、VM で使用できるネイティブな C のクラスそのものを総称した機能です。

- Metadata Loader

  CLR の情報をロードする機能です。C 言語で記述された VM API の Meta 情報を集めた物(Interface Reflection by C)もこの中に含まれます。

  Metadata Loader は Metadata の物理的なレイアウトから論理的なコンテンツ(Metada テーブルやオブジェクト、それらの関係情報など)をメモリ領域へ展開します。

- COFF/PE Loader

COFF/PE は Common Object File Format / Potable Excutable の略であり、.Net が準拠しているオブジェクトファイルの形式です。この形式にしたがったデータはロードする機能をもつのが COFF/PE Loader です。

- HAL (Hardware Abstraction Layer)
  ハードウェアごと際を吸収する層(レイヤー)であり、物理的なデバイスの違いがあっても、振る舞いの違いは HAL が吸収しているため、アプリケーションソフトの製作者はその差を気にすることなく開発を行なうことができる。

２-２-２-①-２-２動作概要

下図に、.NET CLR on Virtual Machine の内部構成及び動作概要を示す。

1)ByteCode Interpreter
CIL を CPU ネイティブの命令に変換し実行する。
2)Garbage Collector
参照されなくなったオブジェクトが使用しているメモリを解放する。
3)Native Library
VM API を呼び出し、API からのイベントを受け取る。
4)Assembly Loader
実行対象のアセンブリをロードし、タイプチェックなどを行う。

① アセンブリローダーは CIL やメタデータを含むアセンブリを読み込む。
② アセンブリローダーはアプリケーションが依存している各種ライブラリを読み込む。
③ ①②で読み込んだ結果を Objects store& Metadata dictionary に反映する。
④ Objects store& Metadata dictionary に展開されたアプリケーションを実行する。
⑤ ByteCode Interpreter が必要に応じて Native Library を呼び出す。

⑥　Native Library は BREW API を呼び出す。
⑦　Native Library は BREW からの event を受け取る。
⑧　⑦で受け取った event のうち必要なものを byte code interpritor に通知する。
⑨　⑤⑥⑦⑧の過程で Objects store& Metadata dictionary にデータを読み書きする。
⑩　Objects store の中で不要となったオブジェクトを mark/sweep 方式で開放する。※１

※１CG の動作するタイミングは、実メモリの残容量が設定された値を下回った時である

２-２-（２）-①-[３]ARM コア CPU への搭載

第二段階として FPGA に搭載可能な CPU コアと同一アーキテクチャの半導体で動作する CLI プロセッサを開発します。
この段階では、FPGA に搭載可能な CPU コアアーキテクチャーを ARM と想定し、ARM ベースの CPU である Cortex-M3 を搭載した試作ボード②を製作しました。この試作ボード②上で CLI プロセッサを動作させることで、将来 FPGA に搭載する際に想定される問題点の洗い出しと、Virtual Machine の性能評価を行ないました。

２-２-２-①-３-１試作ボード②
２-２-２-①-３-１-１試作ボード②の仕様

項目	**仕様**
電源	DC5V、USB ポートからの給電
使用環境	温度+5℃～＋５０℃　湿度 70%RH 以下（結露なきこと）
保存温度	温度-20℃～＋60℃ 湿度 70%RH 以下（結露なきこと）
CPU	ARM Cortex-M3　96MHz
外部インターフェイス	Ethernet(100BASE-T)x1
	Mini-USB　Slave ポート x1　(PC 接続用、および本体への電源供給)
	SPI(通信用)
	GPIO インターフェイス(10 ポート)
	UART（デバッグ・通信用）
押しボタン	電源用ボタンｘ１、リセット用ボタンｘ１
外形寸法	約１１８mm×８０mm（ただし、突起部分は除く）
その他	予備 UART ポート（通信用）ｘ２

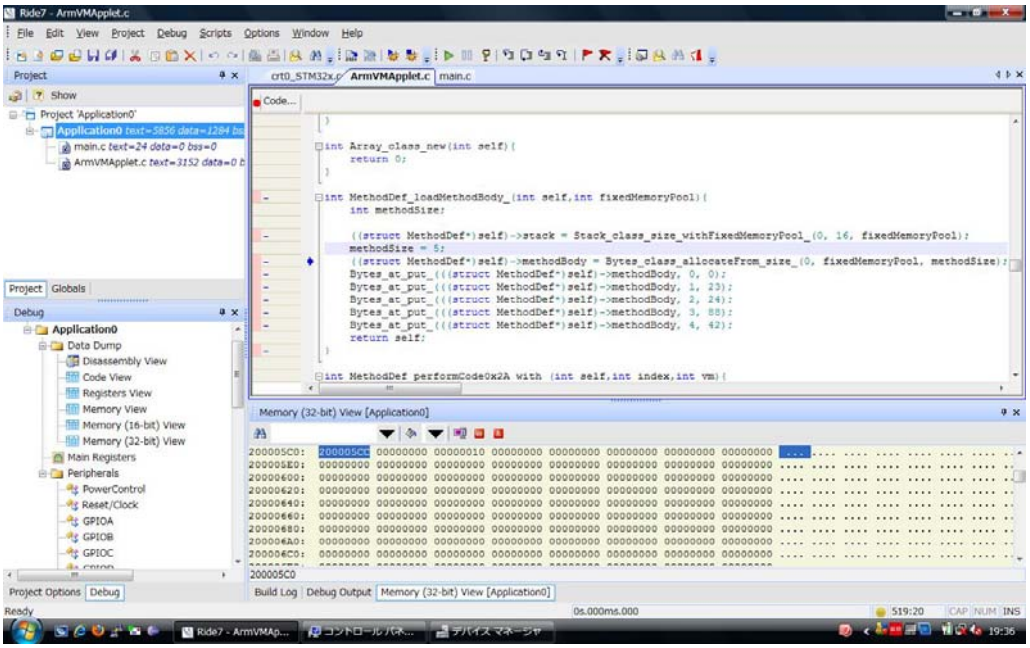
２-２-２-①-３-１-２試作ボード②機器動作環境
（１）ユーティリティ
電源： DC5V、Mini-USB ポートからの給電。
（２）温度
＋５℃～＋５０℃。(ただし、無結露であること)

２−２−２−①−３−２試作ボード②インターフェイス構成

- 電源用ボタン　１個
- リセット用ボタン　１個
- ＬＡＮインターフェイス　１個（１００ＢＡＳＥ―T）
- Mini-USB　Slave ポート x1　(PC 接続用、および本体への電源供給)
- SPI ポート(通信用)
- GPIO インターフェイス(10 ポート)
- UART（デバッグ・通信用）
- ARM は、32bit ベース Cortex-M1 プロセッサを使用。
- 環境に配慮し、ＲｏＨＳ適合部品を使用しております。

２−２−２−①−３−３試作ボード②の ARM に搭載時の CLI プロセッサ動作

試作ボード②で CLI プロセッサを動作させるため、 C#のソースコードを CLI バイトコードはコンパイルし実行します。



ARM で動作する CLI プロセッサに CLI バイトコードをわたし、VM 上でアプリケーションを実行します。

以下は、「b+a」の演算を実行した際の、CLI プロセッサの Stack(ARM の SRAM 上)での変更結果を測定している状態です。

上記のコードの実行前と実行後の CLI プロセッサの Stack(ARM の SRAM 上)は、以下のように変更されます。

２－２－（２）－②スタックマシン型プロセッサのチューニング

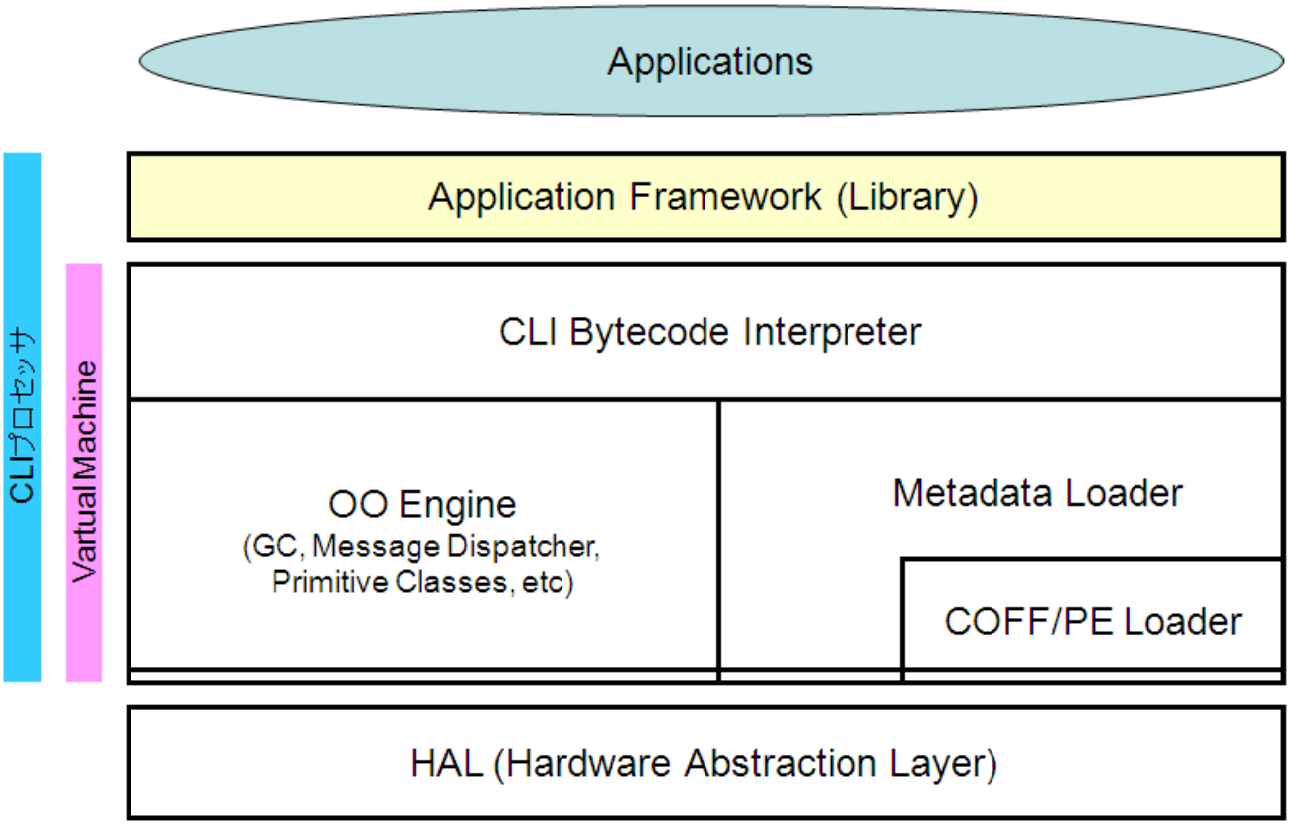
２－２－（２）－②－[１]実施方針

PC 上のシミュレータに実装した CLI プロセッサ上に各種ライブラリを実装し、その実行速度を検証しながら、調整を行う。

２－２－（２）－②－[２]ライブラリの開発

CLI プロセッサ用ライブラリは、以下の黄色の部分に相当し、.NET アプリケーションから VM を呼び出すことを可能にするためのインターフェイスおよび補助クラス郡です。ビルド時に使用される VM Interface for VS と実行時に使用される VM Interface by C の２つにわかれています。

VM Interface for VS には VM API のライブラリが含まれています。



作成したPC上のシミュレータで動作するライブラリは下記のものですが、ARMプロセッサで動作するVMとの結合は、作業途上となっています。

２-２-（２）-③C#でハード/ソフトのミックスシステムの開発環境

２-２-２-③-１-１統合開発環境プラグインシステム構成

ハードウェア、FPGA開発環境であるAltiumDesignerのPlug-inとしてVisualStudioを起動し、C#のコードが動作できるように設計検討を行った。

２-２-２-③-１-２開発環境統合機能
開発環境統合機能は、開発用 PC 上で、携帯電話向け.NET アプリを開発・デバッグを行うためのVisual Studio 2008用Add-inである。

開発環境統合機能は、以下の機能を提供する。

1)ARM用C#テンプレート機能

2)VisualStudio2008用C#Add-in機能

3)ARM ターゲットデバイス通信機能
4)Altium Designer の Add-on と VisualStudio2008 上の Add-in が機能連携し、FPGA エディタや、必要な Script エディタ機能と連動

２-２-（３）まとめ

高級言語（C#）によって記述されたネットワークアプリケーションのソフトウェア・ハードウェアミクスド動作・開発環境の実現のまとめとしては、CLI プロセッサのハードウェア論理での実装において、PC ベースと ARM コアを搭載した試作ボード②での比較が行えたこと、また、ARM に搭載可能な CLI プロセッサのコアの開発が行えたことから、また、.NET ソフトウェアエンジニアだけでは、方針を立てることが難しい組み込み型システムにおいての最小限のメモリ消費、必要最小限の機能セットからサポートするという VM の基本構想段階で目標にしていたことが、スタックマシン型プロセッサのチューニング段階での測定において、洗い出され、問題を解消できたことで、最小限のメモリ消費を維持したままで、速く、機能性に優れた CLI プロセッサの開発が順調に進んでいると考える。

C#でハード/ソフトのミックスシステムの開発環境では、PC ベースと ARM コアと、ARM に搭載した CLI プロセッサのコアの両方で、実装したライブラリの範囲ではアプリケーションの動作が確認できたことで、基本構想段階で目標にしていた部分は達成できたと考えている。今後は VM とその API の多機能化によって、開発環境のライブラリも充実させていく方針である。

一方で、ECMA-335 の MetadataTable と Bytecode への対応、および、.NET がサポートする C#の API 群は、数が多いため、組み込み型の FPGA や CPU での市場調査を行ない、どの機能を、どこまで実現するべきかは、引き続き検討が必要である。更に、CLI プロセッサのコアが後々アップデートできるような機能も盛り込むと、大変便利になることがスタックマシン型プロセッサのチューニング段階で理解できた。また、これまでは、ARM コアへの実装を考えて来たが、近年の FPGA や CPU が標準でサポートする各種デバイスや、現在市場で活性の見られる半導体への対応も検討にいれた開発が必要であると考える。

第３章全体総括

３-１平成 21 年度の成果総括

本研究期間には、TCP/IP プロトコルをハードウェアで実装した IP コアの設計は、CPU と TCP/IP ハードウェア IP コアを FPGA 上にインテグレートした場合でも、CPU を外付け可能な IP コアを設計し、外部 CPU からの制御を可能にした場合でも、順調に機能し、動作検証することが出来た。また、CLI プロセッサは組み込み型 CPU 上で動作できることが確認できた。実装したライブラリはシュミレータ上の範囲ではアプリケーションの動作が確認できたが、組み込み型 CPU 上で動作する CLI プロセッサとの結合作業は本期間で、完成することは出来なかったが、方針と実現可能であることは確認できた。基本構想段階で目標にしていた部分は達成できたと考えている。

３-２本研究開発の成果をふまえた今後の開発

情報家電や携帯電話の低消費電力、高速化のためのネットワークプロトコルのハードウェア論理ブロックの実現では、当初予定の、開発の第三段階の低消費電力の FPGA プラットフォームへのポーティングを行ないつつ、平行して、ARM コアを搭載できる可能性のある FPGA 自体、または対応開発環境の調査も行なう。また、調査の結果、より良い FPGA が見つかった際には、積極的に仕様調査や環境にふれることを行い、更に ToE スループットの高速化、商品化可能な実用的なプロトコルの実装、低消費電力化を目指すこととする。

同時に今回の研究成果を、弊社顧客にも紹介し、実際の市場開拓や、顧客の要望が無いか等もデモを行なうなどして、調査してゆくこととする。

情報家電や携帯電話の低消費電力、高速化のためのネットワークプロトコルのハードウェア論理ブロックでは、ECMA-335 の MetadataTable と Bytecode への対応、および、.NET がサポートする C#の API 群は、数が多いため、組み込み型の FPGA や CPU での市場調査を行ない、どの機能を、どこまで実現するべきかは、引き続き検討を行ないながら、更に最小限のメモリ消費を維持したままで、速く、機能性に優れた CLI プロセッサの開発を行なってゆく。

中でも組込み開発の経験がない高級言語 C#の開発者が組み込み型システム開発を行えるようになることが、組込み市場の活性化になると考えているため、該当開発者には、より多くの要望に耳を傾け、実装検討に盛り込んでいくこととする。それらを出来うる限り取り込んでいき、CLI プロセッサと統合開発環境が今後どうなっていけば、使い勝手がよく、普及するために優れるかの検討を最優先事項として進めていくこととする。

また、今回のスタックマシン型プロセッサのチューニングの手法を生かし、更なるチューニングによる CLI プロセッサのみに限らず、C#でハード/ソフトのミックスシステムの開発環境に置けるライブラリやコンパイル環境も含めたソフトウェア・ハードウェアのチューニングを行なっていくこととする。