Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

Mieruシステムソフトウェア

佐野正浩[†1] 高前田伸也[†1] 芝哲史[†2]
曹哲[†2] 伊藤宗平[†1] 川合秀実[†3]
笹田耕一[†2] 吉瀬謙二[†1]

本稿では，先導的 IT スペシャリスト育成推進プログラムの一つである情報理工実践プログラムの一環として 2 大学の学生がチームを結成し，開発した小型計算機システムについて述べる．実習の目的は，ハードウェアリソースの制限が大きい計算機システム開発を通してプロジェクト管理を経験し，組み込みシステム開発への理解を深めることである．FPGA ベースの教育向け小型計算機システム MieruPC をベースに，ハードウェアの拡張およびオリジナルのオペレーティングシステムの実装を行った．またこれらの拡張 · 実装を生かした新たな MieruPC 向けアプリケーションの開発を行った．開発には，開発効率を高めるために MieruPC のシミュレータの SimMieru を構築した．本稿では，開発を通してどのような経験をしたか，またそれから得られた教育的知見について述べる．

Mieru System Software

Masahiro Sano,[†1] Shinya Takamaeda,[†1]
Satoshi Shiba,[†2] Tetsu Soh,[†2] Sohei Ito,[†1]
Hidemi Kawai,[†3] Koichi Sasada[†2] and Kenji Kise [†1]

The authors developed a small FPGA-based computer system. The purpose of development is to acquire knowledge of embedded systems and to have experience of project management. The authors have customized the hardware of MieruPC and developed an original operating system. Then they also developed new applications for the MieruPC by using the extension. For efficient development, we costructed the MieruPC simulator; SimMieru. This paper describes the knowledge and the experience acquired on this project.

1. はじめに

組み込みシステムはあらゆるところで我々の生活を支えている．近年は携帯電話や携帯ゲーム機などの普及によりその重要性は高まっている．搭載されるプロセッサの高性能化に伴いシステムは複雑化の一途を辿っており，そのハードウェアおよびシステムソフトウェアの両方の知識を充分に持つ技術者が強く求められている．しかし教科書や講義で習うシステムの知識だけでは実際に組み込みシステムなどの計算機システムを開発することは困難である．なぜなら開発には理想的な環境を前提とした知識を実際の環境に適用する能力が求められる．例えばオペレーティングシステムであれば，ファイルシステム自体の知識はあったとしても，機能として提供するためには他のコンポーネントとどのように組み合わせる必要があるのかを理解した上で，実際に動作する環境の制約によって実装方法を変えなければならない．

開発に必要な技術を身に付けるためには，実際に計算機システムの開発を通して学習するのがよい．特にハードウェアとソフトウェアの両方を構築することで，両者の関連をより深く理解することが出来る．しかしハードウェアの実装はソフトウェアの実装に比べて困難である．この点においては，構成の変更が可能な **FPGA(Field Programmable Gate Array)** が有用である．FPGA 上に実装する回路は HDL(Hardware Description Language) を用いることでソフトウェアのように記述することが可能である．記述した HDL を各 FPGA ベンダーが提供する開発環境を用いて論理合成することで，実際に FPGA に書き込み可能な回路イメージを生成することが出来る．

我々は，複数人でのシステム開発スキルの習得と計算機システムへの理解を深めるために，FPGA をベースとした開発基盤 MieruPC[1)] 上に小型の計算機システムの実装を行った．具体的には，MieruPC のハードウェア面の拡張およびシンプルなオペレーティングシステムの実装と，いくつかのサンプルアプリケーションの開発を行った．また，開発したシステムと同等の機能を有するソフトウェアシミュレータの開発を行った．そしてそれらの成果物を SDK(Software Development Kit) として統合し，ドキュメントの整備を行った．本 SDK

†1 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†2 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
†3 OSASK 計画
OSASK Project

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

はアプリケーションの開発を容易にするだけではなく，SDK 自体が MieruPC を用いた計算機システム開発のサンプルとして利用できるように意識して構成されている．

MieruPC は FPGA を主体としている計算機システムのため，ハードウェア構成の変更が可能である．しかし搭載されている FPGA のゲート数は 25 万ゲート相当と小さく，ハードウェアの追加および変更を困難なものとした．そして，それはソフトウェアの拡張にも影響を及ぼした．

本稿では，開発成果と開発過程でどのような経験が得られたかについて述べる．以下に本稿の構成を述べる．まず，第 2 章で開発のモチベーションと開発のベースにした MieruPC について述べる．次に第 3 章で開発したシステムの仕様について，ハードウェア・ソフトウェアの両面から解説する．第 4 章では，開発手法について述べる．特に本システム構築の特徴であるシミュレータベースのシステム開発手法について解説する．第 5 章では，開発途中に遭遇した困難であった点とその解決方法について述べる．第 6 章では，教科書で学習した知識を実践的な能力へ変える方法について述べる．最後に第 7 章で本稿をまとめる．

なお本開発は，文部科学省先導的 IT スペシャリスト育成推進プログラムの一つである「情報理工実践プログラム」(東京大学，東京工業大学，国立情報学研究所) のソフトウェア開発プロジェクト実践という講義の一環で行った．

2. 開発にあたり

本章では，開発のモチベーションと開発方針について述べる．

2.1 モチベーション

組み込みシステムの一般への浸透に伴い，その重要性は高まっている．近年の組み込みシステムには GHz オーダーで動作する高性能プロセッサを搭載したものもあり，そのようなシステムの開発を行うには計算機システムのハードウェア・ソフトウェア両面に精通していることが求められる．組み込みシステムのみならず，情報工学に携わる者は計算機システムの仕組みを充分に理解していることが，さらなる情報工学の発展には不可欠である．しかし，開発には知識のみならず，持ち合わせの知識を実際の制約条件下に適用するスキルが求められる．例えば，ハードウェア制約に応じた機能と性能のトレードオフの選び方である．また複数人で開発を行う場合には，各レイヤー間での協調が必要となる．このようなスキルは実際に開発を通して身に付けることが効率的であると考えられる．

計算機システムの仕組みを理解するには，シンプルな開発サンプルを参考に，実際にハードウェア・ソフトウェアの実装に触れることが効率的である．しかし，回路基板からスクラッチで計算機を構築するのは困難である．その為，プロセッサなどの回路設計・実装が HDL 記述により可能な FPGA をベースとした開発環境が好ましい．またキーボードや出力装置，補助記憶などの各種デバイスが接続できることがより好ましい．そのような開発環境のひとつとして MieruPC[1)2)3)] がある．MieruPC は 2008 年 4 月に開始した **“MieruPC プロジェクト: 中身が見える計算機システムを構築する研究・教育プロジェクト”** の一環で開発された FPGA をベースとする教育用途向け小型計算機システムである．MieruPC は **「中身がみえる計算機システム」** をキーワードに，システムの性能や機能充実よりも，HDL コードやソフトウェアコードの可読性や見通しの良さを優先した実装がされている．しかし，MieruPC はシンプルな構成を採用した代わりに，一般の計算機システムにはほぼ必ず含まれている機能がいくつか省略されている．例えば，ハードディスクドライブなどの補助記憶，それらを管理するファイルシステムやオペレーティングシステムの機能全般，外部との通信機構などである．MieruPC の仕様は次節で詳しく述べるが，計算機システムの学習には欠かせない要素いくつか実装されていない．リアリティのある計算機システム構築を体験するには，これらの要素の追加が必要である．

我々は開発スキルの習得と我々自身の計算機システムへの理解を深めることを目的に，MieruPC をベースにハードウェア面の拡張およびオペレーティングシステムなどの機能追加を行った．また，いくつからのサンプルアプリケーションの作成を行った．また本開発は複数人でのシステム開発スキルの習得も目的としている．開発は **MieruPC2008 Rev.2** をベースに行い，我々が拡張した MieruPC を **MieruPC Rev.ITsp** と定義する．

2.2 ベースシステム:MieruPC2008 Rev.2

本節では，開発のベースに用いた FPGA ベースの小型計算機システム MieruPC2008 の仕様についてハードウェア・ソフトウェアの両面から述べる．

2.2.1 ハードウェア

MieruPC は FPGA をベースとする小型計算機システムである．**図 1** にその外観を示す．計算機本体に相当する部分にカードサイズの FPGA 基板を利用し，Verilog HDL で記述されたプロセッサおよび周辺回路を FPGA 上に構成する．MieruPC2008 は FPGA カードにアットマークテクノ社の SUZAKU-3S を用いた MieruPC2008 Rev.1 と，MieruPC 社 FPGA カードを用いた Rev.2 の 2 種類存在する．両者の大きな違いは，SRAM が実装されているがどうかである．後者は FPGA 基板上に $8bit \times 512entry$ の SRAM が実装されているが，前者には実装されていない．前者はその代わり DRAM が実装されているが，構成が複雑化するため MieruPC では利用されていない．その為，前者では FPGA 内のブロッ

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

図 1 MieruPC2008 Rev.2．上がコマンドインタプリタ型液晶ディスプレイ，下が FPGA カードを含む MieruPC 本体．本体には PS/2 キーボードと液晶ディスプレイが接続される．
Fig. 1 MieruPC2008 Rev.2

ク RAM をメインメモリとして利用しており，容量が 52KB と小さい．後者は SRAM をメインメモリとして利用し，512KB の容量を確保している．開発には FPGA カード版の MieruPC2008 Rev.2 を用いた．

以下に，MieruPC2008 Rev.2 のハードウェアの特徴を示す．なお MieruPC2008 Rev.1 との違いは，前述の通りメインメモリの大きさと，加えて FPGA チップの違いである．SUZAKU-3S には Xilinx Spartan-3E 1200E(120 万ゲート相当) が実装されており，一方 FPGA カードには Xilinx Spartan-3E 250E(25 万ゲート相当) が実装されている．FPGA 上に実装する回路構成はほぼ同じである．Rev.2 の動作周波数は 36MHz であり，FPGA のリソース使用率はスライス換算で約 86%である．

FPGA: Xilinx 社 Spartan-3E XC3S250E．この上にプロセッサ等の回路を構成する．

プロセッサ: MIPS32 互換ソフトプロセッサ．FPGA 上に実装されている．

- 命令フェッチ 4 サイクル，命令デコード，レジスタフェッチ，実行，(メモリアクセス 4 サイクル)，ライトバックの計 9 ステージの**マルチサイクル**プロセッサ．
- MIPS32 のほぼすべての算術 · 論理命令，メモリアクセス，分岐命令が利用可能．
- 浮動小数点には非対応，TLB および L1，L2 キャッシュなし
- 割り込みコントローラなし

メインメモリ: $8bit \times 512entry$ の 512KB SRAM

外部記憶: MMC(Multimedia Card)．MieruPC2008 Rev.2 では起動時のメモリ初期イメージを読み込むときにしかアクセスできない．ソフトウェアからアクセスすることはできない．

出力装置: コマンドインタプリタ型液晶ディスプレイ ITC-2432-035H．データ線 1 本を MieruPC 本体と接続し，描画コマンドを送信することで表示を行う．転送速度は最大 19200bps である．ソフトウェアからは描画コマンドをストア命令で 1byte ずつメモリマップド I/O 領域に書き込むことで表示が可能である．

入力装置: PS/2 キーボード．FPGA 上に実装されたコントローラによりキーボードから送信されるスキャンコードを 8bit の拡張 ASCII コードに変換し受信バッファ(FIFO)に格納する．ソフトウェア側にはメモリマップド I/O として提供されているため，ロード命令で受信バッファにアクセスできる．

電源投入時に MMC のある特定のアドレスからメモリの初期イメージを読み込むことで，MieruPC のプログラムはロードされる．MMC の制御およびプログラムのロードはすべてハードウェアで行っている．また実装されている MMC コントローラは，MMC 上のある特定のアドレスから特定のサイズを読み込むことしかできず，プログラムのロード完了後，MMC コントローラは停止するためソフトウェアから MMC へアクセスすることはできない．その為，ファイルシステムは搭載されていない．すべての I/O にはメモリマップド I/O 領域にメモリ命令を用いてアクセスすることができる．

2.2.2 ソフトウェア

MieruPC2008 にはオペレーティングシステムが搭載されていない．その為，MieruPC 上ではアプリケーションが単独で動作する．

MieruPC 向けのアプリケーションは C 言語により記述する．ソースコードに MieruPC のライブラリファイルをインクルードしてコンパイルし，リンカスクリプトを用いてライブラリとリンクする．こうして作られたバイナリからメモリイメージを作成して，MMC へと書き込む．こうして作成された MMC を MieruPC に読み込ませると，アプリケーションがメインメモリにロードされて実行される．

2.3 開 発 方 針

ベースシステムとして用いた MieruPC のコンセプトである「中身がみえる計算機システム」を我々が開発する計算機システムでも実現する為に，以下の 4 点を意識して開発を行った．

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

(1) リアリティのあるシステムを目指し，ライブラリなどには実際に Linux などのシステムと互換のあるインターフェースを採用する．
(2) 開発したプログラムには教育用サンプルとして利用できるように，拡張性を考慮する．
(3) HDL およびプログラムは可読性を重視して記述する．
(4) システムだけではなく開発行程も “みえる” ように意識する．

3. 開発成果

上記の MieruPC2008 を用いて行った，MieruPC Rev.ITsp のハードウェアとソフトウェアと新たに開発した MieruPC のための開発環境について述べる．本開発は，東京大学大学院 情報理工学系研究科 創造情報実践教育プログラムおよび東京工業大学 大学院情報理工学研究科 先導的 IT スペシャリスト人材育成推進プログラムの一環として通年の講義の中で行った．開発者は東京大学から修士 1 年の学生 2 名，東京工業大学から修士 1 年の学生 2 名の計 4 名である．また，スーパーバイザーとして東京工業大学 大学院情報理工学研究科所属の伊藤宗平先生と OSASK 計画の川合秀実さんを迎えた．

3.1 ハードウェア

ハードウェアの変更は主にオペレーティングシステムへの機能提供のために行われた．**図 2** に MieruPC Rev.ITsp のモジュール構成を示す．色がついているモジュールは MieruPC2008 Rev.2 から追加 · 変更されたものである．FPGA のリソース使用率はスライス換算でほぼ 100%である．各変更点について述べる．

CPU: MIPS アーキテクチャでは浮動小数点演算，例外処理，メモリ管理などの機能はコプロセッサとして提供される．よって，システムコールをサポートするためにコプロセッサの一部を実装．

MMC: ソフトウェアから MMC 上のデータにアクセスする方法を提供．ソフトウェアから指定された番号に対応するブロック (512byte) をメモリマップド I/O として提供し、ロードストア命令でデータにアクセスできる．

RS232C: RS232C 相当のシリアル送受信機を実装．ソフトウェアからロードストア命令で 1byte ずつメモリマップド領域に読み書きすることでデータの送受信が可能．受信バッファは 8byte．送信バッファは無し．転送速度はメモリマップド I/O により任意の値を指定可能．

Keyboard: PS/2 キーボードから送信されるスキャンコードをそのまま受信バッファに格納する．ASCII コードへ変換する機能を削った理由は第 5 章で説明する．

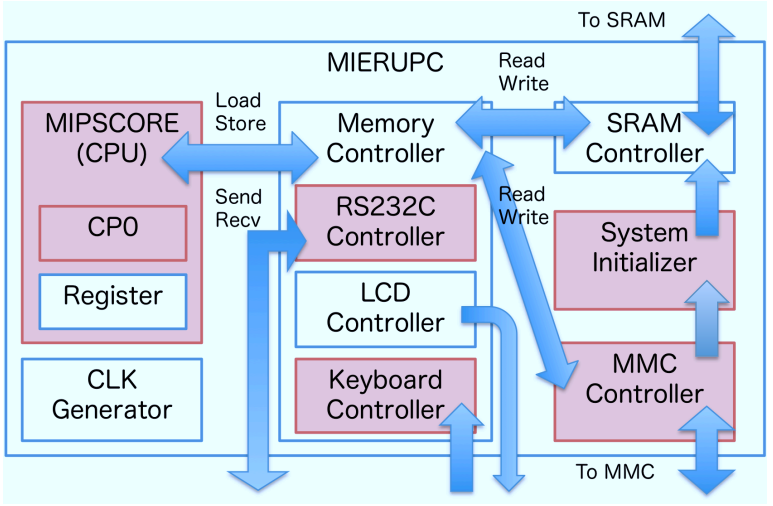

図 2 MieruPC Rev.ITsp のモジュール構成
Fig. 2 Module Structure of MieruPC Rev. ITsp

3.2 オペレーティングシステム

オペレーティングシステムはソフトウェアで必要とされた機能を最低限そなえたシンプルなものをスクラッチから作成した．OS のプログラムサイズは約 30KB である．**図 3** に OS のアーキテクチャ図を示す．プロセスはカーネルプロセスとアプリケーションプロセスの 2 つのみである．OS が起動するとプロセス管理やメモリ管理などの各種データ構造を初期化し，カーネルプロセスを起動する．カーネルプロセスではシェルが動作し，ユーザからのコマンドを受付け，アプリケーションの実行が可能である．アプリケーションを実行するとプログラムローダにより実行するプログラムイメージがメモリ上に展開され，アプリケーションプロセスとして実行される．アプリケーションを終了すると再びカーネルプロセスに戻り，新たなアプリケーションが実行可能になる．アプリケーションはシステムコールを介して OS の機能を使用することができる．OS が提供する機能は以下の通りである．

ファイルシステム: FAT32 を使用．ファイルやディレクトリの読み書きが可能．

メモリ管理: brk システムコールによるヒープ領域の増減が可能．malloc や free はライブラリにより提供される．

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

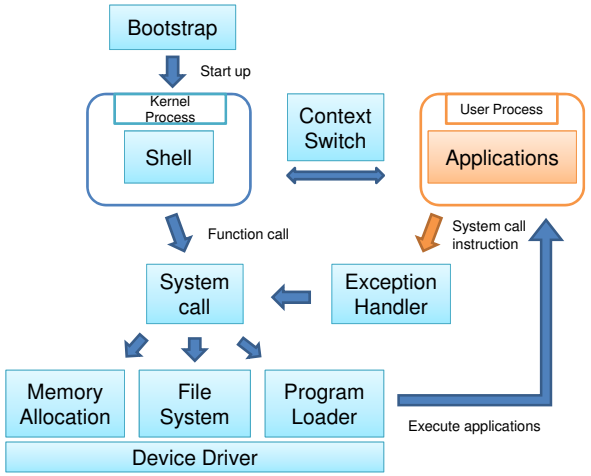

図 3 OS のアーキテクチャ
Fig. 3 Architecture of Operating System

プログラムローダ: 実行ファイルは ELF 形式をサポート.

デバイスドライバ: 各種デバイスを操作するためのドライバ.

3.3 ソフトウェアライブラリ

アプリケーションで共通して使われる機能をソフトウェアライブラリとしてまとめた．ライブラリには Linux などのシステムと互換性のあるインターフェースを持たせることで利用を容易にしている.

- ネットワークライブラリ (エラー検知 · 再送信)
- 描画ライブラリ (ビットマップ表示)
- メモリ割り当て (malloc，free)
- 圧縮 · 解凍

3.4 アプリケーション

ハードウェアの拡張とオペレーティングシステムの搭載により作成できるアプリケーションの幅が大きく広がった．そこで，新たに使用可能となったファイルシステムやネットワーク機能などを利用したアプリケーションの開発を行った.

テキストエディタ: テキストの作成 · 編集が可能なエディタ．スクラッチから実装．VIM のような操作が可能.

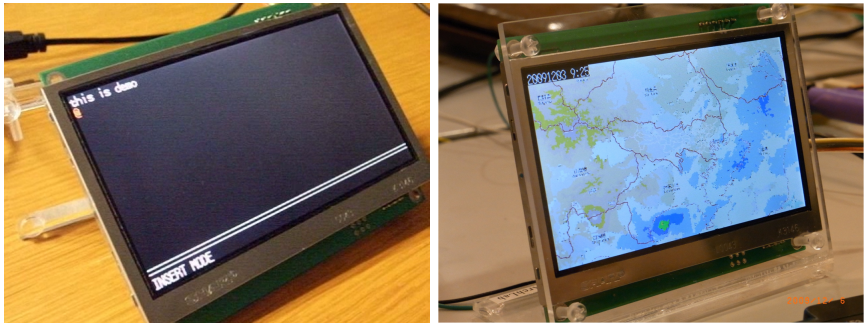

図 4 MieruPC 上で動作しているアプリケーション．左はテキストエディタ．右は Amesh Viewer.
Fig. 4 Applications running on MieruPC.

CampusLisp: Lisp 処理系．CAMPUS LIsP Lemon version[4] を移植．テキストエディタで保存したファイルを実行するか，起動後にプログラムを記述することで実行可能．ファイル入出力や文字列操作はできない.

Waba: シンプルな Java VM．Java のサブセットである Waba[5] を移植．MieruPC のメモリサイズの制限によりクラスローダや VM 内部のエラー処理を削除している．しかし，ガベージコレクションは実装されている．Java で記述された簡単なアプリケーションが動作することを確認している.

Amesh Viewer: 東京近辺の降雨量をグラフィカルに表示するアプリケーション．表示する地域のスクロールや拡大縮小表示が可能．一般的な計算機を使用したサーバがウェブサービス (東京アメッシュ[6]) から降雨量情報を取得し，MieruPC に送信している．サーバと MieruPC はシリアルケーブルで接続されている.

3.5 システムシミュレータ

MieruPC 上で動作するアプリケーション開発を支援するために 2 つのシミュレータを開発した．MieruPC のシミュレータ SimMieru と MieruPC に付属する液晶ディスプレイのシミュレータである.

SimMieru: MieruPC 本体のシミュレータ．MIPS アーキテクチャの計算機システムシミュレータ SimMips[3] を拡張．MMC や通信デバイスなどのシミュレーションを追加．液晶シミュレータと連携して MieruPC のシステム全体をシミュレート可能.

液晶シミュレータ: MieruPC 本体とシリアルケーブルで接続されているコマンドインタプリタ型液晶のシミュレータ．SimMieru と独立して使用可能.

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

3.6 MieruSDK:小型 FPGA 計算機向けソフトウェア開発キット

MieruSDK は開発したアプリケーションとそれらに用いたソフトウェアライブラリ，およびシステムシミュレータ SimMieru，汎用計算機との通信ライブラリなどを含む SDK である．これらを利用することにより，MieruPC 上で動作するアプリケーションの開発が容易になる．

MieruSDK のコンセプトは SDK 自体が計算機システム開発のサンプルとして利用できることである．よって，利用者はただアプリケーション開発を行うだけでなく，計算機システム全体を理解する学習教材としてソースコードを読んだり，拡張することが可能である．その為，コメントのあるシンプルなソースコードやハードウェア，シミュレータ，オペレーティングシステムなどの各機能毎のドキュメントが用意されており，理解しやすいという特徴がある．また，アプリケーションをテストするための様々な環境があり，それらを切り替えるための設定の変更が容易である．

4. 開 発 手 法

本章では，本開発の特徴であるシミュレータベースのシステム開発手法について述べる．また，環境の違いから発生するバグを軽減するためにバグがある可能性の低い環境でテストする方法について述べる．

4.1 スケジュール

本開発のスケジュールを**図 5** に示す．本開発は通年の講義の中で行われたが，2009 年 5 月から 2009 年 8 月までを前期期間とし 2009 年 10 月から 2010 年 2 月までを後期期間として開発期間を分けている．前期期間の目標は「MieruPC1 台で完結するシステムの構築」である．つまり，MieruPC 上で新たなアプリケーションを作成し，それを実行できるような環境を作ることである．その為のソフトウェアとして，プログラムを記述するテキストエディタ，プログラムを実行するための処理系 (Lisp，Java)，プログラムを切り替えるためのオペレーティングシステム (シェル) を作成した．オペレーティングシステムは上記のソフトウェアに必要となるファイルシステム，メモリ管理，プログラムローダを備えるシングルタスクのものとした．ハードウェアは OS の機能を実装するために必要な MMC コントローラとシステムコール命令を実装した．後期期間はグラフィカルな表示をする便利なアプリケーションの作成を目的として Amesh Viewer を作成した．その際に外部との通信が必要となったため，新たにハードウェアに通信機能を追加した．

前期期間の初めの 1ヶ月間は学習期間として，MieruPC 上でどのようなことができるのかを調べるために簡単なアプリケーション開発を行った．その後 2ヶ月をかけてハードウェアとソフトウェアの開発を行った．8 月と 9 月の 2ヶ月は開発は行わなかった．10 月からの 2ヶ月は Amesh Viewer の開発を行い，残りの期間を MieruSDK の構築に充てた．

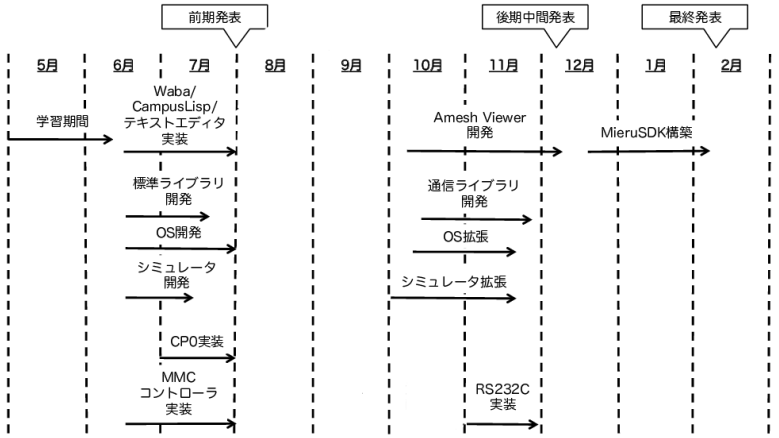

図 5 開発スケジュール
Fig. 5 Development Schedule

次に開発に充てた時間について述べる．スーパーバイザーの方々が参加した公式のミーティングだけで 29 回行われ，各ミーティングの時間は平均で約 2 時間程である．学生のみで行われたミーティングも含めれば 35 回程になる．開発時間としては個人差はあるが平均して週に 6 時間程である．また，二泊三日の開発合宿を前期後期で各 1 回ずつ行った．

4.2 シミュレータベース開発

本開発では学生 4 名を 3 つのレイヤに担当を分けた．ハードウェア開発に 1 名，オペレーティングシステム，シミュレータ開発に 1 名，アプリケーション開発に 2 名である．このように担当するレイヤを分けた場合，下位レイヤの作業が完了するまで上位レイヤが待たなければいけない期間がある．特にハードウェア開発にはソフトウェア開発に比べて時間がかかるため，ハードウェア上でテストができるまでに多くの待ち時間が発生する．

そこで，我々はシミュレータを用いることで待ち時間を減らし，それぞれのレイヤで並列開発を可能にした．具体的には，ハードウェア拡張する場合には，同時にシミュレータの拡張も行う．シミュレータの拡張が終わり次第，アプリケーション開発のための機能を提供す

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

る．アプリケーション開発者はハードウェア拡張が終わるまでシミュレータでテストし，終わり次第実機でのテストを行う．このようなシミュレータをベースとした開発は一般的なことではあるが，プロジェクトを通してこのような開発を行ったことであらためて有効性を確認した．

4.3 信頼度の高い環境でテスト

実機とシミュレータなど複数のテスト環境があることによりバグの発生する要因が増えるという問題がある．特にアプリケーションのバグなのかテスト環境のバグなのかを切り分けることは非常に難しい．

そこで，我々はより信頼度の高いテスト環境で開発を行うことによりバグが発生する可能性を軽減した．例えば，アプリケーションをテストするには実機でテストするかシミュレータでテストするかの選択がある．シミュレータは実機と比較して，実装が容易でテストがしやすい．また，ハードウェアの実装をある程度理想化しているため，シミュレータの方がバグが少ない．バグがあったときにも，シミュレータではデバッグ情報を表示しやすいため，ソフトウェアのバグなのかシミュレータのバグなのかを切り分けやすい．よって，実機よりも信頼度の高いシミュレータでテストすることによりバグが発生する可能性を軽減している．

アプリケーションのロジックを確認する場合には，MieruPC 用のアプリケーションとしてシミュレータを使用して開発するよりもより信頼度の高い Linux 環境をターゲットとしたアプリケーションを開発し，その後 MieruPC 用のアプリケーションとして移植するという方法をとった．Linux 環境を利用した場合，Linux 環境自体から発生するバグはほとんど無いと言える．よって，バグの発生しうる範囲をアプリケーションのみに限定することができる．また，Linux 環境をターゲットとしたアプリケーションを MieruPC 用のアプリケーションへ移行しやすくするために，ソフトウェアライブラリのインターフェースを Linux 互換にしている．また，計算機部分と液晶部分のシミュレータを切り離して使用可能なため，Linux 環境でのテストでも液晶シミュレータを利用できるなどの柔軟な開発ができる．

また，MieruPC 用のアプリケーションには 2 種類のターゲットが存在する．オペレーティングシステムを介さないスタンドアロンのプログラムとオペレーティングシステム上で動作するアプリケーションプログラムである．OS 上で動作するアプリケーションとして動作させる場合，バグの原因がハードウェアなのか OS なのか，それともアプリケーション自体なのかの切り分けが難しい．そこで，OS の機能が必要ない時にはスタンドアロンのプログラムとしてテストすることにより，より信頼度の高いテストができる．OS の機能の一部，例えばデバイスドライバなどは，ソフトウェアライブラリとして提供されているため，アプリケーションからでも利用することができる．しかし，メモリ管理やファイルシステムなどは利用することができない．

このような様々なテスト環境が存在するため，それぞれの環境でテストを行うための設定が複雑である．また，その設定方法を間違えるとバグの原因となってしまうため，他のバグとの切り分けが難しくなる．しかし，MieruSDK を利用することにより複雑なテスト環境を切り替えるための設定を容易に行うことができるようになった．

5. 開 発 経 験

本章では，開発中に遭遇した問題点とその解決法について述べる．そしてその結果どのような経験を得たか述べる．

5.1 想定外のバグ

本開発ではハードウェアからオペレーティングシステム，アプリケーションまでと複数のレイヤが存在し，また複数のテスト環境があることでバグの切り分けが難しいということが想定されていた．しかし，開発中には想定していた部分以外にバグがあり，それらが原因で開発が困難になった．そのような開発中に遭遇した想定外のバグについて述べる．

まず，コンパイラに関する問題について述べる．MieruPC は一般的な計算機と比べて物理メモリが少ないため，アプリケーションのプログラムサイズを少なくする必要があった．そこで，コンパイラ (GCC) の最適化オプションとして'-Os' を使用し，コードサイズを削減することにした．最適化オプションは場合によっては想定している挙動と異なるコードを出力する可能性があるということは分かっていたが，MieruPC 用の多くのアプリケーションでは問題になったことがなかったため，バグの原因となるとは考えていなかった．しかし，コンパイラの設定で浮動小数点命令を使わない設定にしてるにもかかわらず，'-Os' を使用すると浮動少数点命令が使われるということがわかった．MieruPC では浮動小数点命令が実行できないためバグの原因になる．この問題はコンパイル時にオプションとして'-msoft-float' をつけることで解決できたが，コンパイラにバグがあるかもしれないという可能性を残した．

次に MMC に関する問題について述べる．MieruPC では外部記憶として MMC を利用しており，MieruPC 起動時には MMC の特定のアドレスからプログラムをロードすることでプログラムを実行している．MMC へのプログラムの書き込みは Windows 環境を想定しているが，開発は Linux 環境で行っていたため，間違えて Linux 環境で MMC を操作することがあった．一度 Linux 環境で操作された MMC は Windows 環境でプログラムを書き

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

込んでも正しく動作しなくなる．なぜなら，Windows と Linux では MMC のパーティションの扱い方が異なるため，Linux 環境で MMC を操作すると正しい位置にプログラムを書き込めなくなるからである．この問題は Linux 環境でも Windows 環境と同様に MMC を扱うことのできるツールを開発することで解決した．

想定外の部分でバグが発生するということがわかったことで，あらゆる部分にバグがあるかもしれないという疑心暗鬼になり，バグの切り分けが更に難しくなった．バグの可能性が少数なら問題にならないが，開発当初はデバッグ環境や開発を支援するツールなどが無かったため信頼できる部分が無く，全ての部分に可能性があった．また，新たにツールやシミュレータを作ってもそれ自体にバグの可能性があるということで，なかなか開発者に使われるようにならなったため，開発効率が向上しなかった．ツールが使われるようになるには，長い時間をかけて安全性を示し，開発効率向上に繋がるということを示す必要があった．つまり，ある部分にバグが発生する可能性があるとわかると，それを解決する方法を見つけたとしても長い時間がかかるということである．本開発では想定外を含め多くの部分でバグが発生する可能性があったため，開発が困難であったと言える．

5.2 機能と性能のトレードオフ

システム開発において新規機能を追加するための仕様書を定める時に，多くの場合，上位レイヤで必要とされる機能を下位レイヤでどのように提供するか考えなければいけない．教科書では，ハードウェアには割り込みや仮想記憶があり，オペレーティングシステムはマルチタスク対応であるなどの理想的な環境を想定して，その上に機能を実装する方法を述べている．しかし，MieruPC はハードウェアの機能が一般的に使われている計算機より少なく，ハードウェアリソースも少ないため，そのような環境を再現できない．よって，教科書で習う知識をそのまま適用することができない．そこで，本来ハードウェアで提供するような機能をソフトウェアで補うことで必要とされる機能提供を達成する．このようにすることで，ハードウェアで実装していた場合よりも機能性や性能が落ちるがハードウェアリソースが足りない問題を解決することができる．キーボードコントローラと通信機能を例にどのように行ったかを説明する．

まず，ハードウェアのリソース使用率は MMC コントローラ追加の時点でほぼ 100%になっていた．よって，ハードウェアで実装されている機能のうち冗長な部分やソフトウェアでも実装可能な機能はハードウェアから削除しソフトウェアで実装することにした．例えば，MieruPC2008 Rev.2 ではソフトウェアの記述を容易にするためにキーボードから送信されるスキャンコードを ASCII コードに変換する機能をハードウェアで実装していた．しかし，この機能はハードウェアリソースを多く使っていたため，ハードウェアリソース使用量を減らすためにスキャンコードをそのまま FIFO バッファに格納することにした．そして，ソフトウェアで受け取ったスキャンコードから ASCII コードに変換する機能を実装することで以前と同じ結果を得られるようにした．また，スキャンコードは ASCII コードに比べて 1 つのキーに対応する符号のバイト数が多いため，FIFO バッファがあふれやすいという問題がある．そこで，バッファからあふれにくくするためにソフトウェアでキューを用意し，連続で得られるスキャンコードを全てキューに保存するようにした．それでもバッファからあふれてしまいスキャンコードを部分的に失った場合には，そのキーを無視するようにした．

このような工夫をした結果，ハードウェアのリソースに余裕が生じて通信機能を実装することが可能になった．通信機能として当初実装しようとしたものは，実装するのが容易でソフトウェアから扱いやすい DMA 転送による固定バイト長の通信を考えていた．これは，512byte のメモリマップド I/O 領域に転送するデータを保存することで送信を行い，受信時にも同様に 512byte のメモリマップド I/O 領域にデータが書き込まれるという仕組みである．しかしこの機能を実装するには十分なハードウェアリソースが無かったため実装することができなかった．そこでハードウェアリソースの使用量が少ない代替案としてメモリマップド I/O を用いて 1byte ずつ送受信するという方法を考えた．だが，MieruPC にはハードウェア割り込みが無いため，単純な送受信では送受信バッファからデータがあふれてしまい取りこぼす可能性がある．さらに，ハードウェアリソースが不足していたため，受信バッファは 8byte だけである．そこで，バッファからあふれないように考慮する通信プロトコルを作成することにより問題を解決した．この通信プロトコルは，データ送受信前にハンドシェイクを行い，接続を確立してから送受信を開始するプロトコルである．また，データを受信する度に受信したことを示す ACK を返すことでバッファを越えないようにしている．

しかし，完成した通信機能は非常に不安定なものでデータが正常に送れないものであった．その理由はハードウェアの通信路が不安定であり，少しの揺れなどでデータが破壊されるものだったからである．しかし，ハードウェアリソースに余裕がないため，ハードウェアによるエラー訂正や別の通信路を実装することはできない．よって，現在の通信路のままソフトウェアでエラー検知を行うことにした．1 回の送信で送る 8byte のうち 7byte をデータ，残りの 1byte をチェックサムとして，受信時にチェックサムを比較する．比較結果が正常な場合は ACK を返し，異常な場合には NACK を返す．送信者は NACK を受け取った場合には再送信する．このようなアルゴリズムを追加した結果，通信速度は遅くなったが安定して通信ができるようになり，本来の目的である通信機能を上位レイヤに提供することが

Vol.2010-ARC-189 No.16
Vol.2010-OS-114 No.16
2010/4/22

できた．

6. 知識から能力へ

教科書で習うことだけでは，ある機能が他のコンポーネントとどのように関連して成り立っているのかがわからない．また，なんらかの制約により一部の機能がない時にどのように実装すべきかわからなくなる．なぜなら，教科書や講義で習うことは理想的な環境での実装方法だからである．しかし，ハードウェアからオペレーティングシステムやライブラリなど計算機システム全体で考えたとき，ある機能を実装する方法はいくつもある．そしてその実装方法によって利点 · 欠点は異なる．状況に応じて利点 · 欠点を考慮し適切な実装方法を選択するという能力は，実際に開発する上で遭遇する様々な制約を乗り越えることで身につけることができる．このような経験をするには，ハードウェアとソフトウェアの両方を構築するシステム開発は有用である．なぜなら，ある機能が他のパーツとどう組み合わせて機能を提供しているかという知識を再確認することができ，制約に合わせて実装方法を選択するという能力が身につけられるからである．

本開発ではハードウェアの制約が非常に大きい場合に，如何に機能を取捨選択してソフトウェアで補うか主題とした．しかし，ハードウェアリソースに余裕がある場合にはソフトウェアの機能をハードウェアで実装するという選択肢もある．キーボードのスキャンコードから ASCII コードに変換する機能はまさにその例である．このような選択肢を広げるためには計算機システム全体の知識が必要であり，知識の習得のためにもシステム開発は有用である．

7. おわりに

先導的 IT スペシャリスト育成推進プログラムの一つである情報理工実践プログラムの一環として 2 大学の学生がチームを結成し，小型計算機システムを開発した．主たる目的はチームでのシステム開発スキルの習得と計算機システムの理解である．

本稿では，開発したシステムの成果と開発手法について述べた．また，ハードウェアリソースの制限が大きい場合に発生する問題点とその解決法について述べた．そしてシステム開発の教育的知見について述べた．

参考文献

1) 吉瀬謙二，佐藤真平，森谷　章，藤枝直輝，若杉祐太，渡邉伸平，植原　昂，森　洋介，高前田伸也，高橋朝英，棟岡朋也，山田裕介，権藤克彦，小林良太郎，三好健文，中條拓伯：MieruPC プロジェクト: 中身が見える計算機システムを構築する研究・教育プロジェクト，コンピュータシステム・シンポジウム (ComSys2008) (2008).
2) 藤枝直輝：MieruPC 株式会社. http://www.mierupc.com/.
3) 藤枝直輝，渡邉伸平，吉瀬謙二：教育・研究に有用な MIPS システムシミュレータ SimMips，情報処理学会論文誌, Vol.50, No.11，pp.2665–2676 (2009).
4) : CAMPUS LIsP. http://www-masu.ist.osaka-u.ac.jp/kakugawa/hacks/clisp/.
5) : Wabasoft. http://www.wabasoft.com/.
6) : 東京アメッシュ. http://tokyo-ame.jwa.or.jp/.