Prologue

今後の組み込みシステム開発で重要なツールの一つ

シミュレータ/エミュレータによるイマドキの組み込み開発

編集部

1 シミュレータとエミュレータ

近年，パソコンのCPUクロックがGHzを越え，それまでは難しかったシミュレータ/エミュレータが実用的な速度で動くようになってきました．これを使って，組み込みシステムをシミュレータ/エミュレータで開発する動きが出てきました．本特集ではシミュレータ/エミュレータを使った組み込みシステム開発について解説します．

…と，その前に，シミュレータ/エミュレータとは何なのか，あらためて考えてみましょう．

● シミュレータ？エミュレータ？

「シミュレータ」と「エミュレータ」，あなたはどのように使い分けていますか？　特集に入る前に，まずはこの二つの用語について整理しておきましょう．この二つの用語を理解するのにちょうどよい比較事例が，「Androidエミュレータ」(**図1**)と「iOSシミュレータ」(**図2**)です．比較を**表1**に示します．

Androidエミュレータで実行できるのはあくまでARM命令の実行バイナリなので，プログラムをARM用にクロス・コンパイルします．Androidエミュレータは ARM命令を実行し，さらにCPU命令の実行だけでなくAndroidが動作するハードウェアを模擬します．よってAndroidエミュレータで動作したバイナリ・イメージはそのままAndroid端末の実機でも動作します．

iOSシミュレータは，ARM命令(iPhoneに搭載されているCPUはARM系)を実行しているわけではありません．iOSシミュレータで走らせたいプログラムは，iOSシミュレータが動作するシステムのCPUのネイティブな命令(実際にはx86系命令)にコンパイルしてから実行します．よってiOSシミュレータで走らせたバイナリ・イメージは，そのままでは実機で動作しません．実機で動かすためには，ARM用に再コンパイルする必要があります．

図1　Androidエミュレータの画面例

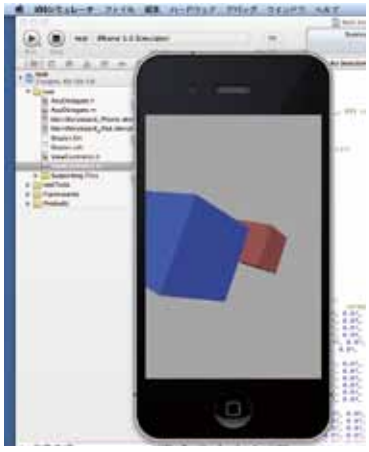

図2　iOSシミュレータの画面例

表1　「Androidエミュレータ」と「iOSシミュレータ」の比較

	Androidエミュレータ	iOSシミュレータ
実行バイナリ	ARM命令	x86命令
実機とのバイナリ互換性	互換性あり	互換性なし (再コンパイルが必要)
実行速度/重さ	重い	軽い

Androidは ARM以外のCPUでも動作するが，ここではARMに限定する

● 実機の再現性

いずれにせよ，実機とシミュレータ/エミュレータ上では，どうしても違いが発生します．どの程度実機に近いかを示す指標として，再現性という言葉が使われます．一般的にシミュレータの再現性は，エミュレータよりは劣るといわれています．

しかしiOSシミュレータはかなりの実機再現性を持っており，Androidエミュレータより実機に近いともいわれます．これはiPhoneが，ハードウェアからソフトウェアまで一貫してApple社1社で開発されているため，という理由もあります．

● シミュレータとエミュレータの使い分け

実際のところ，シミュレータとエミュレータの使われ方は少し混乱しているように見受けられます．用語の使い分けを考える上で，その源流をさかのぼってみましょう．

そもそも，コンピュータで別のコンピュータのソフトウェアを実行したいという要求は1960年代のメイン・フレームIBM System/360の時代に生まれました．参考文献(1)によると，

『それまでは，ある計算機のために書かれたプログラムをそのままの形で別の計算機で処理するのには，シミュレーション法，すなわちソフトウェアによって処理する方法が取られていたが，これでは処理速度が著しく低下する．そこで，これをハードウェア，ソフトウェアの組み合わせによって効率良く処理するようにしたのがエミュレーションであり，ハードウェアの部分は主としてマイクロプログラムを変更あるいは追加することによって実現するものである．』

とあります．マイクロプログラムとは，当時のCPUではやった方式で，乗算や除算などの複雑なアセンブリ命令を，内部的には複数の単純に置き換えて実行するものです(ただし，その「単純な命令」はソフトウェアからは見えない内部処理になってい

る）．件のSystem/360は旧機種と新機種の間でソフトウェアの互換性がなかったため，生み出されたのがエミュレーションという手法です．とはいえ，

『小型機ではマイクロプログラムだけで処理されるものもあるが，一般には語長や処理装置の構成の違いのため，マイクロプログラムだけで行うのは実際的でなく，ソフトウェアも使用されている．』

と，だんだん定義があやふやになってきています．フルICE（In-Circuit Emulator）はハードウェアを使ってCPUのふりをすることから間違いなくエミュレータですが，JTAG-ICEなどはそもそもエミュレーションしていませんし，Androidエミュレータは高速化のためにハードウェアを使っていないのが遅い原因だし…注1．

以上を踏まえて本特集では，特集タイトルこそ「シミュレータ」と題しましたが，x86系CPUを搭載したパソコン上で，x86系以外のCPUの命令バイナリを実行するということで「エミュレータ」という用語を使うことにします．

2 エミュレータの動作による違い

読者の皆さんも既にVMwareやVirturalBoxといったソフトウェアを使って，Windows上でUbuntuを動かしたりしていると思います．

このように，「パソコン上で，違うOSやハードウェアのソフトウェアを動かす」というのが漠然としたエミュレータのイメージです．しかし，これらのエミュレータは技術的に幾つかに分類できます．

● CPUをエミュレーションするかどうか

これらエミュレータを分類する上での一番大きな違いはCPUのエミュレーションをするかどうかです．例えばVMwareやXenは，エミュレータ本体がx86上のWindowsやLinuxで動作し，その上でx86のプログラムが動きます．それに対し，QEMUやBochsなどはx86上のWindowsやLinuxで動作し，その上でARMやRXなどの「x86以外の」プログラムが動作します．ここが大きな違いです．

前者はx86上でx86のプログラムを動作させることから，高速な動作が期待できます．これらの場合，エミュレータ本体が動くOS（ホストOS）の上で別のOS（ゲストOS）を動かすことが目的が大半です．

後者はCPUのエミュレーションを行うことにより，x86とは命令体系の全然違うARMやRXなどのCPUのプログラムを動かすことができます．CPUのエミュレーションとは，ARMやRXなどのCPUの機械語命令を逐一x86の機械語に翻訳しつつ，実行していくものです．そのため，前者と比較して速度が落ちますが，エミュレータさえ作ればどんなCPUのプログラムでも動かすことができます．

3 なぜ組み込み開発でエミュレータを使うのか

では，なぜ組み込み開発でエミュレータが使われるようになってきたのでしょうか？エミュレータを使った組み込みソフトウェア開発の利点を次に示します．

● ハードウェアが完成していなくてもソフトウェアの開発が進められる

たとえ試作基板であっても，ハードウェアの製作には時間がかかります．エミュレータなどを使わない開発の場合は，試作基板ができあがってこないと，その上で動作するソフトウェアの開発を始められませんでした．しかしエミュレータを使うと，ハードウェア開発と並行してソフトウェア開発を進められます（図3）．

● ハードウェアをプログラマの人数分用意しなくてもよい

複数人数で行う開発の場合，開発者の人数分だけ試作基板を用意するのが難しい場合が多々あります．そのようなとき，どうしてもハードウェアとして実機が必要なデバイス・ドライバなどの担当者にのみ試作基板を使ってもらい，上位アプリケーションの担当者にはエミュレータを使ってもらうと，分担してソフトウェア開発を進められます．

● エラー処理ルーチンの動作確認

外部との通信部分にはエラーが発生しやすい箇所です．そのため通信ドライバ部分には，エラー処理ルーチンを埋め込むのが一般的です．しかし開発の現場では実際にはなかなかエラーが発生しないものです（そもそも，そう簡単にエラーが発生する場合は，通信インターフェースの物理層に問題がある）．

エミュレータを使った開発では，普段は発生しにくいエラーをあえて発生させる処理をエミュレータ内に埋め込み，ソフト

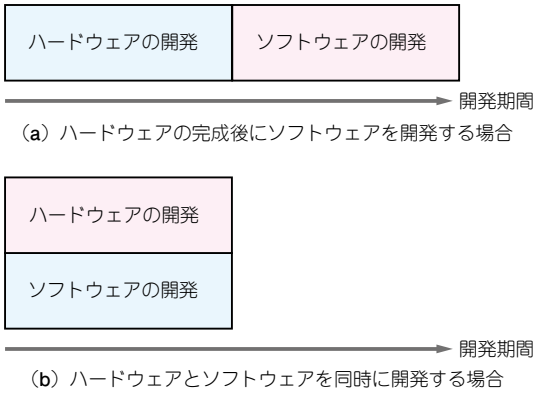

図3 ハードウェアとソフトウェア開発期間

注1：Appendix 1で解説するが，QEMUはdyngenによりアセンブリ1命令をマイクロオペレーションと呼ばれる手法により，複数の命令に変換しながら実行している．この動作が「マイクロプログラムを使ってエミュレーションする」というエミュレータの定義と似ている．

ウェアによるエラー処理が正しく行われるかどうかを確認するという使い方も可能です．

● クリティカルな処理のステップ実行

割り込み処理に代表されるようなクリティカルな部分は，バグが紛れ込みやすい処理の代表です．このような部分は，ステップ実行により一つ一つ状態を確認して実行できるとよいのですが，実機ではそうもいかないときがあります．実機上でCPUを止めることはできても，周辺機能まで停止させることはできないのが一般的です．

例えばタイマ割り込み処理の場合は，デバッガでブレークをかけた後，変数の状態などをウォッチしている間に，タイマ・コントローラのカウント・レジスタの内容はどんどんカウントアップしてしまい，ブレークしたときの状態を参照することができません．

エミュレータの場合は，周辺機能も含めて止めることができるので，ブレークしたときのタイマ・カウンタの値を参照できます．

4 組み込みで使われるエミュレータ

● x86系以外のCPUが多用される世界

パソコンのCPUがx86系一色になってしまったのに対し，組み込み機器ではx86系以外のさまざまなCPUが使われています．携帯電話や携帯ゲーム機をはじめとした多彩な機器で使われているARM，車載系で強いV850，ディジタル家電など高性能が要求される分野に使われるMIPS，携帯電話基地局など高速大容量のデータを扱う分野で使われるPowerPCなどです．もちろんx86系もAtomの登場やパソコンをそのまま流用した組み込み機器などで使われていますが，組み込み機器全体の中では一部です．

そのため，組み込み機器でエミュレータを使うには，x86以外のCPUをエミュレーションするタイプのエミュレータがほぼ必須になります．

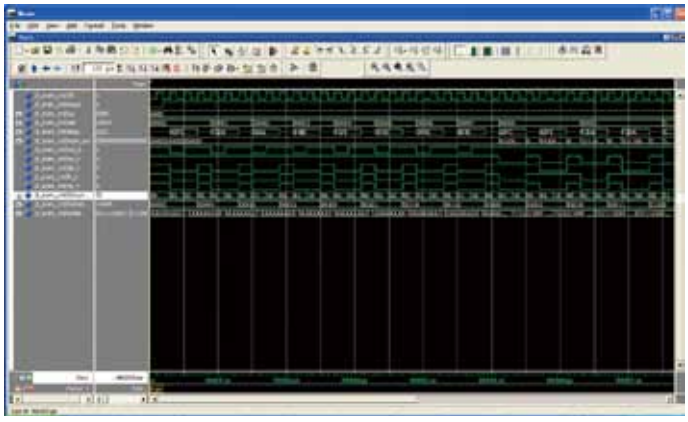

図4　ModelSimのシミュレーション画面例

● CPUのみのエミュレーションだけでは不完全

組み込みシステムの世界では，CPUのみを再現しても不完全です．組み込みシステムにはスイッチ入力やLCD表示，モータ制御などさまざまなI/O装置がつながります．これらの周辺機能も再現しなければ，組み込み開発の世界でエミュレータを使う意味はなくなります．

具体的には，I/Oポートの再現が必要です．現在のCPUはメモリマップドI/O方式なので，特定のアドレスに特定の値を書き込むことにより，端子を‘L’レベルにしてLEDを点灯させたり，逆に特定のアドレスから値を読み込むことにより，Ethernetコントローラからデータを受け取れるようになります．このようなしくみも必要になるのです．

5 シミュレータ/エミュレータが使われている組み込みシステム開発分野

ここでは，実際に組み込みシステム開発分野で使われているシミュレータやエミュレータを見てみましょう．

● ハードウェア開発の世界ではシミュレータ

• FPGAによるハードウェア開発

個人レベルですぐにでも体験できる，シミュレータが重宝する分野といえば，FPGAを使ったハードウェア開発でしょう．評価版のFPGA開発ツールとともに，ベンダ/デバイス/回路規模限定版のシミュレータを使うことができます．

• ASICなどの大規模ハードウェア開発

大規模なASIC（Application Specific Integrated Circuit）などの開発では，数千万から億の単位の開発費が必要になります．デバイスができあがり，動作させてみてから設計ミスに気が付いたのでは困ります．そこで回路設計にミスがないか，シミュレーションを重ねて入念にチェックします．

これらロジック回路のシミュレーションではModelSimなどのシミュレータがよく使われます（**図4**）．

● ソフトウェア開発でも使われる

ソフトウェア開発でもシミュレータ/エミュレータは使われます．有名どころでは冒頭で紹介したAndroidエミュレータやiOSシミュレータがあります．

本特集では，オープン・ソースのエミュレータとしてQEMUを大きく取り上げます．第1章から第3章までは，QEMUをベースとしてARMやSH-2などのCPUと限定的な周辺機能をエミュレーションできるツールを取り上げ，RTOSやLinuxなどを起動させてみます．

またQEMU以外のオープン・ソース・ソフトウェアとして，第6章ではSkyEye，第7章ではXenのシミュレータとしての応用事例について紹介しています．

さらに，マイコン向け統合開発環境にシミュレータ機能が搭載されているものもあります．第4章ではIAR Systems社の

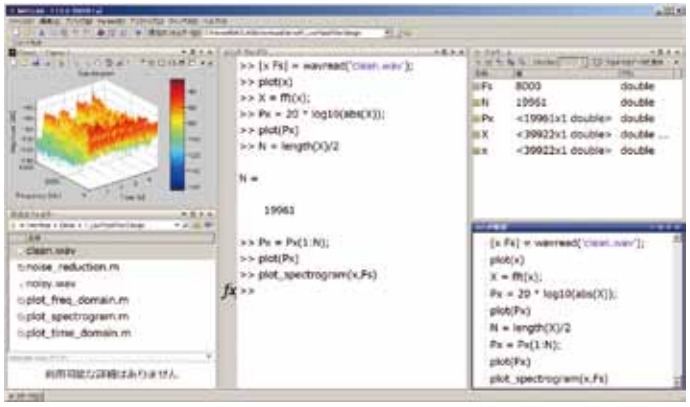

図5　MATLAB/Simulinkの画面例

IAR Embedded Workbenchを，Appendix 3ではKEIL社（ARM社）のMDK ARM（μVision）を，第5章ではルネサス エレクトロニクスのHigh-performance Embedded Workshopのシミュレータ機能について紹介します．

ちなみに，IAR Embedded WorkbenchやHigh-performance Embedded Workshopでは，JTAGデバッガなどを使って実機を接続してのデバッグをエミュレータ，実機を接続せずにソフトウェアだけで実行する場合をシミュレータと呼んで区別するようです．

● ハードウェアとソフトウェアを連携したシミュレーション

今後はさらに，ハードウェアとソフトウェアを連携したシミュレーションが重要になってくるでしょう．

・CPUを内蔵したSoCシステム開発

単に回路規模が大きなハードウェアだけでなく，昨今ではSoC（System on a Chip）と呼ばれるCPUを内蔵して1チップでシステムを構成できるデバイスの開発が増えてきています．ASIC数個分＋CPUという構成になるので，ハードウェア規模だけでも巨大なところに，さらにソフトウェアも絡んできます．

また開発したSoCにバグそのものはなかったとしても，思ったような性能が出ない…という場合もあります．SoCの内部バスが，CPUの命令アクセスとデータ転送で帯域を奪い合う形になり，CPUが待たされてしまうことがあるためです．性能が要求される場合はSoC内部に複数のバスを配置し，CPUとデータ転送とで独立してバスが使える構成にする必要があります．しかし，やみくもにバスを増やすとコストアップの要因になります．このあたりを，シミュレータを使って検討するという使い方が，参考文献(2)で紹介されています．

・アルゴリズム開発

ディジタル信号処理など，アルゴリズム・レベルから開発する場合があります．さらに処理性能も要求される場合，どこまでの処理をハードウェアで行うか，どこから先をソフトウェアに担当させるか，悩ましい問題にぶち当たります．

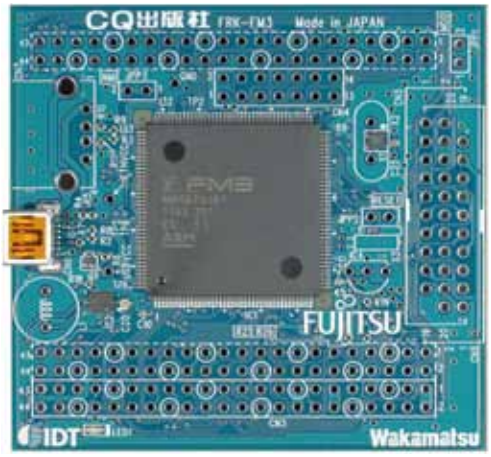

写真1
次号付属FM3マイコン基板

図6
FM3マイコン対応エミュレータのGUIアプリケーション

このような分野では，参考文献(3)で紹介しているMATLAB/Simulinkがよく使われます（**図5**）．

● 次号付属FM3マイコン基板向けプログラムが走る！

本誌で推奨したいエミュレータのメリットとして，組み込みシステム開発の入門用途があります．そこでAppendix 2では，次号本誌付属FM3マイコン基板（**写真1**）に対応したFM3マイコン基板エミュレータを用意し，最も基本的な組み込みプログラムであるLED点滅制御プログラムを実行できます（**図6**）．

参考文献

(1) 相磯 秀夫，飯塚 肇，坂村 健；ダイナミック・アーキテクチャ，bit 1980年8月号臨時増刊，共立出版，1980.
(2) 川原 常盛，竹本 正志，大塚 聡史；ハードウェア/ソフトウェア協調設計が容易なElectronic System Level設計について，Interface 2009年6月号，CQ出版社.
(3) 特集 手を動かし実践する！MATLABプログラミング，画像処理や電力制御から，行列演算＆モデルベース設計を学ぶ，Interface 2011年12月号，CQ出版社.
(4) 特集 組み込みCプログラミングを基本から攻略する！，プログラムはどう動く？シミュレータを使って体験しよう！，Interface 2009年4月号，CQ出版社.
(5) 特集 シミュレータと実機で学ぶ組み込みLinux入門，ビルド環境の構築からデバイス・ドライバの作成まで，Interface 2009年10月号，CQ出版社.