組み込みシステムの開発環境整備に関する研究

平川　寛之・清水　章良・宮本　博永・阿部　正人

Research on Preparation of Development Environment for Embeded Systems

Hiroyuki HIRAKAWA, Akio SHIMIZU, Hironaga MIYAMOTO and Masahito ABE

要　約

近年の組み込みシステムは，高性能化，高機能化の一途にあり，その開発サイクルも短くなる一方である．これらに対応するため，組み込みシステムにおいてもOS（オペレーティングシステム）を導入し，既存のソフトウエア部品の再利用を図る事例が増える一方，組み込みシステムは，対象となるハードウエアが様々であるため，既存のソフトウエア資産がそのまま移行できない事例も数多い．

このような状況を改善するため，オープンソースの組み込みOSとして実績のあるTOPPERS/JSPを対象に，異なるハードウエア間での移植を支援するツールと，ハードウエア・ソフトウエアを同一レベルで扱うことで，効率的なシステム開発が可能な環境の整備を行った．移植支援ツールについてはH8Sを対象に開発環境の整備を行った．ハードウエア・ソフトウエア混在開発環境については，言語仕様の策定と予備的な実装を行った．

1．緒　　言

携帯電話や情報家電等に代表される組み込みシステムは，近年，高性能化，高機能化が進められており，その開発サイクルも短くなる一方である．これらに対応するため，組み込みシステムにおいてもOS（オペレーティングシステム）を導入し，既存のソフトウエア部品の再利用を図る事例が増えている．

一方，組み込みシステムは，対象となるハードウエアや用途が様々であるため，OSを導入する際もハードウエアに合わせた移植作業が必要となる．これまで，移植作業は人手で行われており，OSは最も基本となるソフトウエアであるが故に，開発ツール類の支援も受けにくく，OSそのものの知識も要求されるため，経験のある技術者に依存するところが非常に大きかった．

このような状況を改善するため，オープンソースの組み込みOSとして実績のあるTOPPERS/JSP[1]を対象に，特別な知識を必要とせずに異なるハードウエア間での移植を支援するツールの開発を行う．

また，近年，FPGA（書き換え可能な論理素子）が広く普及してきており，ハードウエア設計の敷居がこれまでになく低くなってきている．これによって，従来，ソフトウエアでしか実現できなかった処理を容易にハードウエア化したり，ソフトウエア処理を補完する目的でハードウエア処理を行う事例が増えてきた．このような環境変化によって，ハードウエア・ソフトウエアの境界をどのように設定するか，システム構成をどのように設定するかが新たな課題となってきている．また，おのおのの性格の異なるシステムをどのように共存させるか，設計者にとって見通しの良い開発環境をどのように提示するかも課題となってきている．

以上の現状から，C言語で書かれたプログラムにハードウエア記述言語の業界標準として広く普及しているVHDL（VHSIC　Hardware Description Language）で記述された処理を埋め込むことで組み込みシステム全体の見通しを良くし，効率よく開発可能な環境の構築を行うこととした．

2．開発した環境について

2－1　移植支援環境

TOPPERS/JSPを新しいハードウェアに移植する際に必要な手順を把握するために，現在TOPPERS/JSPが対応していない市販のボードにTOPPERS/JSPを移植する作業を試みた．

2－1－1　開発に用いたマイコンボード

移植に用いるマイコンボードは㈱北斗電子のHSB8SX1653Fを選択した．TOPPERS/JSP1.4.2以降では，㈱ルネサステクノロジ社製のCPUであるH8SXと互換性のあるH8Sが参考実装扱いとなっており，H8Sのコードをベースに移植を行った．作業はCygwin[2]上で行い，具体的な仕様は以下の通りである．

- OS　　TOPPERS/JSP1.4.3
- Cコンパイラ　　GCC-2.9.15

・バイナリユーティリティ　Binutils-2.14
・標準Cライブラリ　newlib-1.11

2－1－2　外部メモリ部の作成

H8SX1653Fに内蔵されているROMは384KB，RAMの容量は40KBである．TOPPERS/JSPを実装するにはRAMの容量が足りないため，外部メモリの増設を行った．

H8SX1653Fには16ビットのデータバスがあるので，512K×8ビットのSRAMを2つ用いて上位8ビット用と下位8ビット用に分けて接続して外部メモリ部分を作成する．

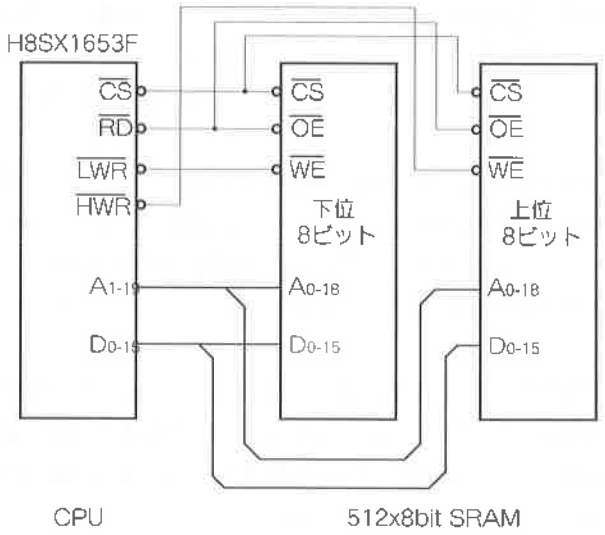

図1　マイコンと外部メモリの接続

HSB8SX1653FのJ1，J2端子からアドレスバス，データバス，その他図1示す信号線が取り出せるので，これらを用いてメモリとの接続を行うための基板を作成した（図2）．

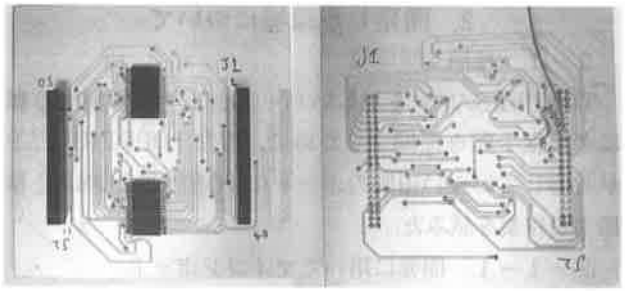

図2　外部メモリ部分（表面，裏面）

2－1－3　ソフトウェアの変更

TOPPERS/JSPのソースコードは多くの部分がC言語で記述されており，移植作業が容易になるようシステムやCPUに依存する部分が明確に分離されている．また，TOPPRES/JSPのソースコードには移植作業の手引きを示したドキュメントが含まれており，各依存部での関数についての解説が記載されているので，それに従い必要箇所の変更を行った．

2－2　ハードウエア・ソフトウエア混在開発環境

図3の例に示すように，C言語のソースプログラムとVHDLのソースを一括して，同一のソースファイル上で記述可能なシステム（トランスレータ）の開発を行った．

```
/* ハードウエアによる関数の記述 */
hdl_inline void led_display( std_logic_vector(3 downto 0)  data)
{
library IEEE;
use ieee.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity Latched_Decoder is
   port (data : in std_logic_vector(3 downto 0);
         cs_n : in std_logic;
         l_led_A : out std_logic;

         .......
)

Int
main()
{
   int i, j, c = 0;
   while(1) {
      led_display(c);  /* ハードウエア関数の呼び出し */
      c++;
   }
}
```

図3　C言語，VHDLの混在ソース

2－2－1　開発環境の構成について

今回の開発環境では，C言語によるソフトウエアルーチンとVHDLによって記述されたハードウエアモジュールが共有のCPUバスを通じて相互に協調して動作するシステムを効率よく開発する状況を想定している（図4）．このようなシステムではCPUによるソフトウエア処理と，ハードウエアモジュール（HDL_FUNC）が並列して動作し，CPUバスは両者のデータ共有や，タイミングの調停などを行う．

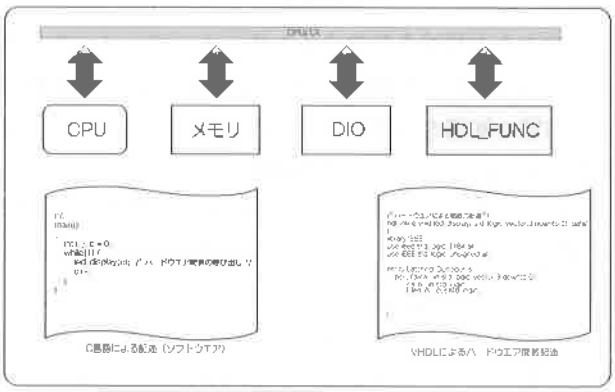

図4　開発対象となるシステムの構成

図4の対象システムを効率よく開発するための開

発環境の全体構成を図５に示す．トランスレータはVHDL-C言語混在ソースファイルから，(1) ハードウエア・モジュールの記述，(2) アドレスデコーダ，(3) 割り込みコントローラ，(4) C言語ソースの4つのファイルを生成する．ハードウエア記述に関する (1) ～ (3) のファイルは既存の論理合成・配置配線ツールを介してpofファイルに変換される．また，C言語ソースファイルはCコンパイラを介してHEX[3] ファイルに変換される．最後に，２つのファイルはFPGAに対するデバイスプログラマを介してデバイスに書き込まれる．

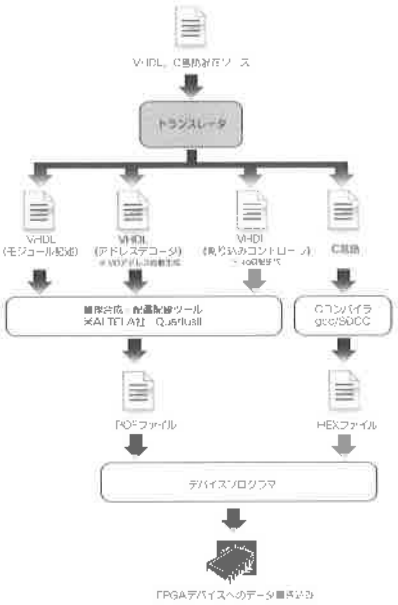

図５　全体構成

２－２－２　言語仕様

VHDLおよびC言語は，本来，混在した環境で扱える言語ではないため，両者を無理なく扱うために，言語仕様に一部拡張を施す必要がある．拡張は，VHDLによるハードウエアモジュールの記述の宣言部分（図６）に対して行う．BNF[4] で記述した拡張仕様を図７に示す．

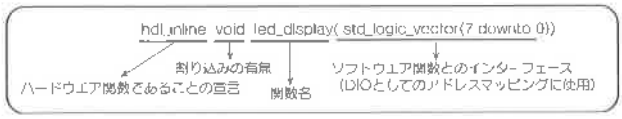

図６　ハードウエアモジュールの宣言部分と機能

２－２－３　トランスレータの動作

トランスレータは２－２－２で定めた仕様に従ってハードウエア記述とソフトウエア記述を分離する．同時に，以下の動作を行う．

(1) ハードウエアモジュールの記述をハードウエアモジュール呼び出しハンドラに置き換え

ハードウエア・モジュールをC言語の関数として呼び出すために，モジュール全体を図８に示すような呼び出しハンドラに置き換える．呼び出しハンドラはC言語で記述されているので，Cコンパイラでコンパイルすることが可能である．

```
関数を置き換えるlist := hdl_inline type func_name ( param )

param :=  /* nothing */
        | type value
        | param, type value

type :=  std_logic
       | std_logic_vector ( NUMBER dir NUMBER )
       | integer

dir  :=  downto
       | to

NUMBER := [0-9]+

value := [a-zA-Z0-9_¥-]+
```

図７　ハードウエアモジュール宣言部の言語仕様拡張

```
/*
 *
 * HDL FUNCTION HANDLER
 *
 */

 sfr _at (_HDL_AUTO_ASSIGN_DIO0, ADDR_) _HDL_AUTO_ASSIGN_DIO0_

void
_hd_handler(char _v0_)
{
   _HDL_AUTO_ASSIGN_DIO0  _ = _v0_;
}
```

図８　ハードウエア・モジュール呼び出しハンドラ

(2) ハードウエアモジュールをCPUバスに接続するためのアドレスデコーダの生成

ハードウエアモジュールをCPUバスに接続するために，モジュールにアドレスを付与する．モジュールに対してCPUからのアクセスが生じた場合，図９に示すようなモジュールを動作させるためのアドレスデコーダ回路を設置する必要がある．

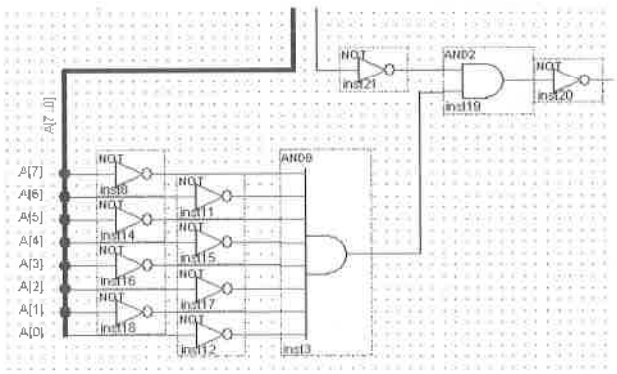

図９　アドレスデコーダ

(3) ハードウエアモジュールが使用する割り込みを調停するための割り込みコントローラの生成

ハードウエアモジュールが割り込みを使用する場合

は，必要に応じて優先順位などを調停する割り込みコントローラを設置する必要がある．

トランスレータが生成するファイルのハードウエア部分をまとめると，図10の例のようになる．これを論理合成・配置配線ツール及びCコンパイラを通じてFPGAに書き込めば，所望のシステムを構築できる．

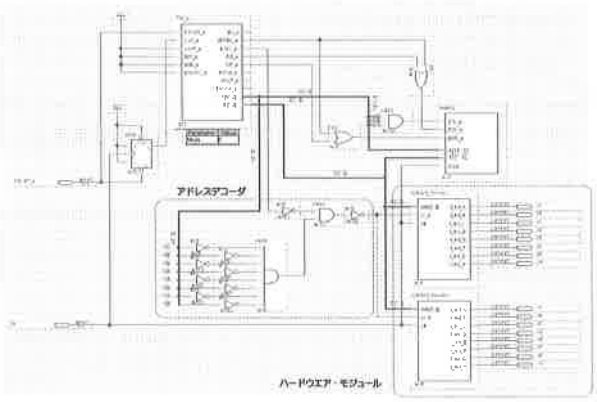

図10　生成されるハードウエアの例

3．考　　察

移植支援ツールについては，現在，外部メモリの動作確認中であり，この作業が終了次第ROM上にTOPPES/JSPを書き込み動作確認を行う．

HSB8SX1653F付属のモニタプログラムは，外部RAMには対応しておらず，デバッグ作業が困難である．このため，現状では簡易なテストプログラムを用いてデバッグ作業を行っている．今後は，GDB[5]などのデバッガの導入などを検討する必要がある．

ソフト＆ハード混在開発環境については，現状でトランスレータの開発を行っており，H20年度に実際の開発に適用し問題点等の洗い出し，改良等を行う予定である．

4．結　　言

本研究はH19，H20年度の2カ年で実施する計画となっており，H19年度は仕様の検討と試作開発を行ってきた．H20年度は，組み込み技術研究会などを通じて現場での使用に耐える環境の構築を行っていく予定である．

参考文献

1）http://www.toppers.jp
2）http://cygwin.com
3）http://www.tamasoft.co.jp/lc/hlp/F063.html
4）Niklaus Wirth，アルゴリズム＋データ構造=PASCALプログラム，p.321（1979）
5）http://directory.fsf.org/project/gdb/