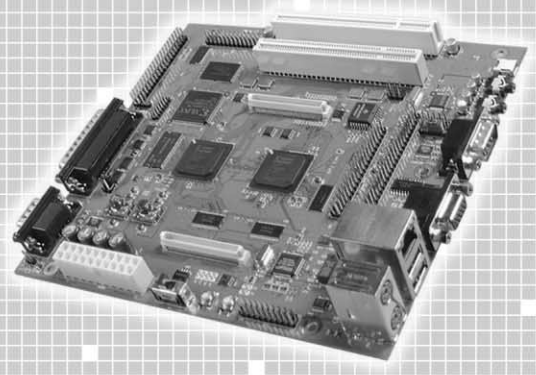
組み込みシステム開発評価キット活用通信



井倉 将実

第2回 リソース情報の取得と割り込みシステム

前回(2006年11月号，pp.161-171)は，BLANCAシステムのアーキテクチャについて解説しました．今回は，ソフトウェアから各種コントローラに割り当てられたベース・アドレスを取得する方法や，各種コントローラからの割り込み出力を束ねるBLANCAシステム割り込みコントローラの仕様について解説します．

1．システム・リソースの取得方法

● リソース定義ファイルの内容をソフトウェアも取得

前回解説したように，BLANCAシステムではアドレス・デコーダ回路のHDLソースに直接アドレス値を埋め込むのではなく，**リスト1**に示すリソース定義ファイル(再掲載)で設定したパラメータを読み込み，`if generate`文で必要なデコーダだけを生成しています．

実はこのリソース定義ファイルは，BLANCAシステム・バスのアドレス・デコーダだけで使うわけではありません．この情報は，**リスト2**に示すリソース・テーブル・コントローラを経由してCPUでも読み込むことができます．

● リソース・テーブルのイメージ

リソース定義ファイルの内容をソフトウェアが取得できるといっても，**リスト2**ではイメージがわかないと思います．そこで**図1**に，リソース・テーブル領域をメモリ・ダンプしたようすを，**表1**にメモリ・マップを示します．A/Vプロセッサのリ

```
 address     +0       +4         +8       +C     | ---- ASCII -----
80000000  424C414E 43412041 - 56502020 00000000 | BLANCA AVP  ....
80000010  00000000 00000000 - 00000000 00000000 | ................
80000020  41565053 80000000 - 01FFFFFF 00000000 | AVPS............
80000030  52474243 82000000 - 01FFFFFF 00000000 | RGBC............
80000040  41433937 84000000 - 01FFFFFF 00000000 | AC97............
80000050  4D4D4343 86000000 - 01FFFFFF 00000000 | MMCC............
80000060  434F4D32 88000000 - 01FFFFFF 00000000 | COM2............
80000070  4C454449 8A000000 - 01FFFFFF 00000000 | LEDI............
80000080  54494D45 8C000000 - 01FFFFFF 00000000 | TIME............
80000090  53445241 20000000 - 00000000 00000000 | SDRA ...........
800000A0  46524F4D A0000000 - 0FFFFFFF 00000000 | FROM............
800000B0  4C425553 B0000000 - 0FFFFFFF 00000000 | LBUS............
800000C0  50434948 C0000000 - 3FFFFFFF 00000000 | PCIH....?.......
800000D0  00000000 00000000 - 00000000 00000000 | ................
800000E0  00000000 00000000 - 00000000 00000000 | ................
800000F0  00000000 00000000 - 00000000 00000000 | ................
```

(**a**) A/Vプロセッサ

```
 address     +0       +4         +8       +C     | ---- ASCII -----
B0000000  424C414E 43412049 - 4F502020 00000000 | BLANCA IOP  ....
B0000010  00000000 00000000 - 00000000 00000000 | ................
B0000020  494F5053 B0000000 - 007FFFFF 00000000 | IOPS............
B0000030  50434954 B1000000 - 007FFFFF 00000000 | PCIT............
B0000040  4C414E43 B2000000 - 00FFFFFF 00000000 | LANC............
B0000050  49444543 B4000000 - 00FFFFFF 00000000 | IDEC............
B0000060  43464343 B6000000 - 00FFFFFF 00000000 | CFCC............
B0000070  55534248 B8000000 - 00FFFFFF 00000000 | USBH............
B0000080  55534254 BA000000 - 00FFFFFF 00000000 | USBT............
B0000090  50533248 BC000000 - 00FFFFFF 00000000 | PS2H............
B00000A0  434F4D31 BE000000 - 00FFFFFF 00000000 | COM1............
B00000B0  00000000 00000000 - 00000000 00000000 | ................
B00000C0  00000000 00000000 - 00000000 00000000 | ................
B00000D0  00000000 00000000 - 00000000 00000000 | ................
B00000E0  00000000 00000000 - 00000000 00000000 | ................
B00000F0  00000000 00000000 - 00000000 00000000 | ................
```

(**b**) I/Oプロセッサ

図1 リソース・テーブルをメモリ・ダンプしたようす(MicroBlazeの例)

表1 サンプル設計プロジェクトのメモリ・マップ(MicroBlazeの例)

アドレス	バス幅	デバイスID	用 途
0000_0000h～	32ビット	—	ブロックRAM
2000_0000h～		SDRA	SDRAM空間
4060_0000h～		—	MicroBlaze UART Lite(COM2)
8000_0000h～		AVPS	A/Vプロセッサ・リソース・テーブル
8200_0000h～		RGBC	アナログRGBコントローラ
8400_0000h～		AC97	AC97コントローラ
8600_0000h～		MMCC	MMCカード・コントローラ
8800_0000h～		COM2	UARTコントローラ(COM2)
8A00_0000h～		LEDI	LED＆DIPスイッチ・コントローラ
8C00_0000h～		TIME	タイマ・コントローラ
A000_0000h～		FROM	フラッシュ・メモリ空間
B000_0000h～		IOPS	I/Oプロセッサ・リソース・テーブル
B100_0000h～		PCIT	PCIターゲット・コントローラ
B200_0000h～		LANC	LANコントローラ
B400_0000h～		IDEC	IDEコントローラ
B600_0000h～		CFCC	CompactFlashコントローラ
B800_0000h～		USBH	USBホスト・コントローラ
BA00_0000h～		USBT	USBターゲット・コントローラ
BC00_0000h～		PS2H	PS/2ホスト・コントローラ
BE00_0000h～		COM1	UARTコントローラ(COM1)
C000_0000h～FFFF_FFFFh		PCIH	PCIホスト・コントローラ

KEYWORD ―― システム・リソース，A/Vプロセッサ，I/Oプロセッサ，ローカル・バス，割り込みルーティング，割り込みコントローラ，ベクタ割り込み

リスト1　A/Vプロセッサ・リソース定義ファイル(AVP_DEF.VHD)

```
-- ************************************************************************
-- ***** A/Vプロセッサ リソース・マッピング定義ファイル
-- ***** 8000_0000h ～ FFFF_FFFFh  2Gバイト空間  ビッグ・エンディアン用
-- ************************************************************************

library ieee;
use ieee.std_logic_1164.all;

package AVP_DEF is

    -- ***** AVPシグネチャ定義 ***** --
    constant AVP_ID0         : std_logic_vector(31 downto 0) := X"424C414E";        -- BLAN
    constant AVP_ID1         : std_logic_vector(31 downto 0) := X"43412041";        -- CA A
    constant AVP_ID2         : std_logic_vector(31 downto 0) := X"56502020";        -- VP

    constant LITTLE_ENDIAN   : integer := 0;          -- リトル・エンディアンのとき1，ビッグ・エンディアンのとき0
    -- ↑接続するCPUに合わせて設定の必要あり

    constant BUSERR_ALTRDY   : integer := 1;          -- バス・エラー発生時，代替レディを返す
    -- ↑1の場合は，デバイスが存在しないエリアにアクセスしてもハングアップしない
    --   0の場合は，デバイスが存在しないエリアにアクセスするとハングアップする

    constant SYSCLK_DIV      : integer := 13;         -- シリアル・コントローラ用分周カウント値
    -- ↑システム・クロック 66.666MHz時:18分周  48.000MHz時:13分周  33.333MHz時:9分周  25MHz時:7分周


    constant CHIP_SELECT     : integer := 11;         -- チップ・セレクト本数

    -- ***** チップ・セレクト0 リソース定義 ***** --
    constant CHIPSEL0        : integer := 0;          -- チップ・セレクト番号0
    constant CHIPSEL0_ID     : std_logic_vector(31 downto 0) := X"41565053";        -- "AVPS"  AVP
    constant CHIPSEL0_ADRS   : std_logic_vector(31 downto 0) := X"80000000";
    constant CHIPSEL0_SIZE   : integer := 25;         -- 32Mバイト

    -- ***** チップ・セレクト1 リソース定義 ***** --
    constant CHIPSEL1        : integer := 1;          -- チップ・セレクト番号1
    constant CHIPSEL1_ID     : std_logic_vector(31 downto 0) := X"52474243";        -- "RGBC"  RGB
    constant CHIPSEL1_ADRS   : std_logic_vector(31 downto 0) := X"82000000";
    constant CHIPSEL1_SIZE   : integer := 25;         -- 32Mバイト

    -- ***** チップ・セレクト2 リソース定義 ***** --
    constant CHIPSEL2        : integer := 2;          -- チップ・セレクト番号2
    constant CHIPSEL2_ID     : std_logic_vector(31 downto 0) := X"41433937";        -- "AC97"  AC97
    constant CHIPSEL2_ADRS   : std_logic_vector(31 downto 0) := X"84000000";
```

A/Vプロセッサ用シグネチャID

各種パラメータ定義

ASCII文字4文字でデバイスIDを表す

ベース・アドレス

割り当てる空間サイズ

リスト2　A/Vプロセッサ・リソース・テーブル・コントローラ(AVP_SYS.VHD)

```
-- ********** ACCESSステート時の動作 ********** --
when REG_ACC =>
    if (SYS_ReadWrite = '1') then     -- リード・アクセス
         case SYS_Address(7 downto 2) is

         when "000000" =>    -- +00h シグネチャ表示
              SYS_ReadData <= AVP_ID0;
         when "000001" =>    -- +04h
              SYS_ReadData <= AVP_ID1;
         when "000010" =>    -- +08h
              SYS_ReadData <= AVP_ID2;
         when "000011" =>    -- +0Ch
              if (LITTLE_ENDIAN = 1) then
                   SYS_ReadData(31) <= '1';-- リトル・エンディアン時はMSBが'1'
              else
                   SYS_ReadData(31) <= '0';-- ビッグ・エンディアン時はMSBが'0'
              end if;
              SYS_ReadData(30 downto  0) <= (others => '0');

         when "000100" =>    -- +10h 割り込みステータス
              SYS_ReadData(31 downto 16) <= IOP_Interrupt;
              SYS_ReadData(15 downto CHIP_SELECT) <= (others => '0');
              SYS_ReadData(CHIP_SELECT-1 downto  0) <= AVP_Interrupt;
         when "000101" =>    -- +14h 割り込みイネーブル
              SYS_ReadData(31 downto 16) <= IOP_IntEnable;
              SYS_ReadData(15 downto CHIP_SELECT) <= (others => '0');
              SYS_ReadData(CHIP_SELECT-1 downto  0) <= AVP_IntEnable;
         when "000110" =>    -- +18h 割り込みレベル・レジスタ
              SYS_ReadData(31 downto  8) <= (others => '0');
              SYS_ReadData( 7 downto  0) <= SYS_IntLevel;

～中略～
         when "001000" =>    -- +20h チップ・セレクト0 デバイスID
              SYS_ReadData <= CHIPSEL0_ID;
```

リソース定義ファイルで定義した定数が入る