2008年ベーシックコース12回目

2009.02.21

Web用サンプルa版

FPGAの中にARMを作る

奈良高専 土井滋貴

FPGA (Field Programmable Gate Array) と組み込みシステム

I/O I/O I/O

FPGA

CPU DSP Memory

FPGA採用による利点：

必要な性能のプロセッサとその周辺回路(ペリフェラル)を柔軟且つ最適に構築できる

具体的には

- ①手元でプログラミングするだけでよく製造工程なし
- ②システムのカスタマイズ及び機能向上が迅速且つ容易に実現

複線的なアプローチ

記述言語

チップメーカー

アルテラ

ザイリンクス

FPGA用CPUコア Cortex-M1

The screenshot shows a Microsoft Internet Explorer browser window displaying the Actel website. The address bar shows the URL: <http://www.actel.com/intl/japan/products/mpu/cortexm1/>. The page features the Actel logo with the tagline "POWER MATTERS" and navigation menus for "ライセンス & 登録 | Customer Portal | 資料請求" and "Home", "アクテルについて", "製品とサービス", "テクニカル・サポート資料", "アクテルのソリューション", "お問い合わせ". A secondary menu includes "デバイス", "プロセッサ", "ソフトウェア・ツール", "IP(知的設計財産)コア", "プログラミング". The main content area is titled "Cortex-M1" and "Intelligent Processors by ARM". Below this, the text reads: "FPGA向けに設計されたARM®のプロセッサ". The main body text describes the Cortex-M1 as a 32-bit processor developed by Actel and ARM, designed for FPGA implementation. It highlights features such as being cost-free, having a balanced size and speed, and supporting up to 72MHz. It also mentions the 4,300-tile implementation and the use of the ARMv7-M architecture. The text concludes by stating that the processor is easy to integrate with various tools and support resources provided by Actel and ARM.

Actel
POWER MATTERS

ライセンス & 登録 | Customer Portal | 資料請求

Home | アクテルについて | 製品とサービス | テクニカル・サポート資料 | アクテルのソリューション | お問い合わせ

デバイス | プロセッサ | ソフトウェア・ツール | IP(知的設計財産)コア | プログラミング

ARM

- » Cortex-M1
 - » M1 デバイス
 - » 知的設計財産(IP)
 - » 設計ソフトウェア
 - » 設計ハードウェア
 - » ドキュメンテーション(英語)
 - » In the News(英語)
 - » アップデート登録
 - » CoreMP7
- 8051 コア (英語)
 - » Core8051 (英語)
 - » Core8051s (英語)
- CoreABC (英語)
- LEON3 (英語)

Home » 製品とサービス » プロセッサ » ARM » Cortex-M1

Cortex[®]

Intelligent Processors by ARM[®]

FPGA向けに設計されたARM[®]のプロセッサ

ARMがアクテルと共同で開発した32ビットのCortex-M1プロセッサは、FPGAの実装用に設計された初のARMプロセッサです。Cortex-M1プロセッサは無償で、サイズとスピードのバランスが取れており、最大72MHzで使用でき、最小の4,300タイル数で実装できます。スリム化した3段階のパイプラインソリューションであるCortex-M1は古典的な命令セットのサブセットを駆動するため、既存のThumbコードを変更せずに使用できます。コンフィギュラブルなCortex-M1プロセッサはアドバンスド・ハイパフォーマンス・ハブ(AHB)に接続するため、設計者がサブシステムを作って周辺機能を追加することが簡単にできます。アクテルのSmartDesign、CoreConsoleおよびSoftConsole、ARMのRealViewの各ツールに加えて、サードパーティのベンダーが、コンパイラやデバッガーからRTOSソリューションまで、各種のサポート用ツールを提供します。

Cortex-M1

- ・FPGA実装に向けて開発
- ・ARMv6-M命令セット・アーキテクチャ
- ・すべてのThumbコードを実行 - ARM7、ARM9 Thumbサブルーティンを実行
- ・3段階、32ビットのパイプライン
- ・メモリーとAHB-Liteインターフェースを分離
- ・コンフィギュラブルで、ネストされ、ベクターされたインターラプト・コントローラ
- ・オプションの高速もしくは小型のマルチプライヤー
- ・Cortex-M3 との上位互換性
- ・ユーザーによる FPGAへのプログラム書き込み
- ・Cortex-M1の全I/Oやシグナルにアクセス可能
- ・アクテルのフラッシュベースM1デバイスに最適化
- ・ライセンス料やロイヤルティ不要で使用可能
- ・組み込み、リアルタイムのデバッグ、JTAGインターフェース
- ・すべての開発ツールがサポート

スタータキットの1つ、IGLOO開発キット

IGLOO開発キットに付属するIGLOO開発ボード

M1AGL Development Board

IGLOO開発ボードのブロック図

主要な4つのツール(開発キットに付属するDVDで提供される)

SmartDesign

グラフィックなブロック・システム設計作成ツール

プロセッサ、DirectCores、標準ライブラリ・コア、ユーザIP、およびカスタムHDLを含むFPGAおよびSoC設計の作成

シンセシスが用意されたHDLの自動作成

[Liberio IDE](#)内での運用

CoreConsole

アクテルのIP設置プラットフォーム(IDP)

FPGAへの組み込みアプリケーション用にIPコアの組立を簡単にするグラフィカル・インターフェースとブロック・ステッチャーを含む

Liberio IDEとのシームレス、もしくは独立した運用

SoftConsole

アクテルのプロセッサ・プログラム開発環境

GNU C/C++ コンパイラ、GDBデバッガ、シミュレータを含む

統合設計環境Liberio®

アクテルの包括的なFPGA設計開発ソフトウェア

代表的なEDAベンダーの最新の設計作成、物理的実装、検証ツールを統合

統合環境 Libero IDE

The screenshot displays the Libero IDE Project Manager interface. The main window shows a project flow diagram for a project named 'kand'. The flow starts with 'Design Entry Tools' (HDL Editor, SmartDesign, CoreConsole, ViewDraw) leading to 'Source Files'. This is followed by 'Synthesis' (Synplify) using 'kand.edn', which produces 'Post-Synthesis Files'. These files lead to 'Place&Route' (Designer) using 'kand.adb', which produces 'Post-Layout Files'. The 'Post-Layout Files' are then used for 'Simulation' (ModelSim, Stimulus Editor, WaveFormer) and 'Programming' (FlashPro, Identity Debugger, Silicon Sculptor). The 'Post-Layout Files' also lead to a 'PDB File'. The 'Simulation' step is currently selected, showing 'Pre-Synthesis', 'Post-Synthesis', and '> Post-Layout' options. A 'Configure Project Flow...' button is also visible.

The left sidebar contains the 'Design Explorer' showing the project hierarchy, including 'Components', 'User Files', 'Block Symbol Files', 'Schematic Files', 'HDL Source Files', 'Stimulus Files', 'Simulation Files', 'Constraint Files', 'Synthesis Files', 'Physical Synthesis Files', and 'Designer Views'. The 'Catalog' window shows a list of components, including 'Basic Blocks', 'Bus Interfaces', 'Clock & Management', 'Memory & Controllers', 'Peripherals', 'Power Management', and 'Processors'. The 'Information Window' shows the 'Properties' of the selected component.

The bottom status bar displays the current design flow: 'Project Flow' with 'kand', 'kand.pdc', and 'kand.edn' selected. The 'Log Window' shows the following messages:

```
Reading file 'kand.v'.  
Warning: The project was not using the same profiles.  
The CoreConsole v1.3 or later profile was different.  
The k_and_2 project was opened.
```

The bottom right corner of the status bar shows the current design parameters: 'VERILOG', 'FAM: IGLOO', 'DIE: M1AGL600V2', and 'PKG: 484 FBGA'.

デモ1 回路図入力でAND回路を作る(対象ハードはIGLOO開発ボード)

The screenshot shows the Verilog IDE interface for a project named 'kand'. The main workspace is divided into several panes:

- Design Explorer:** Shows the project hierarchy, including components, user files, HDL source files, and designer views.
- Canvas:** Displays the 'Top Level' design with an AND2_0 component.
- Schematic:** Shows the internal logic of the AND2_0 component, including inputs inx0 and inx1, and output outx0.
- Grid 1:** Shows the Instance-Instance View table.

Instance	Port Name	Slice	Attribute	kand	AND2_0
kand	inx0				A
	inx1				B
	outx0				Y
AND2_0	A			inx0	
	B			inx1	
	Y			outx0	

The bottom status bar shows the project path and hardware target: Page 1 of 1 | VERILOG | FAM: IGLOO | DIE: M1 AGL600V2 | PKG: 484 FBGA

ピンのアサイン(割り振り)はpdcファイルに記述する

The screenshot displays the Project Manager software interface for a project named 'kand.pdc'. The main window shows a Verilog code editor with the following code:

```
1 set_io inx0 -pinname D18
2 set_io inx1 -pinname D17
3
4 set_io outx0 -pinname E10
5
```

The interface includes several panels:

- Design Explorer:** Shows a tree view with 'Components' (kand) and 'User Files' (Block Symbol Files, Schematic Files, HDL Source Files).
- Catalog:** Shows a list of components under 'Function, Name:' with 'Basic Blocks' and 'Bus Interfaces' visible.
- Information Window:** Shows 'Properties'.
- Log Window:** Shows a warning message: 'Warning: The project was not using the same CoreConsole v1.3 or later project. The k_and_2 project was opened.'

The status bar at the bottom indicates 'Ready' and provides details about the project: 'Ln 1, Col 1 | VERILOG | FAM: IGLOO | DIE: M1 AGL600V2 | PKG: 484 FBGA'.

デモ2 回路図入力でカウンタを作る(対象ハードはIGLOO開発ボード)

The screenshot displays the SmartDesign IDE interface. The main workspace shows a schematic diagram of a counter circuit implemented using a chain of multiplexers (MUXes). The circuit is designed to count from 0 to 9, with the output of the final MUX connected to a 7-segment display. The IDE window title is "Project Manager - C:\Actelprj\test04\test03.prj - [test03sd]". The menu bar includes "Project", "File", "Edit", "View", "Tools", "SmartDesign", "Schematic", "Window", and "Help". The toolbar contains various icons for file operations, design manipulation, and simulation. The status bar at the bottom indicates "Ready" and provides details about the project: "Page 1 of 1 | VERILOG | FAM: IGLOO | DIE: M1AGL600V2 | PKG: 484 FBGA". The Windows taskbar at the very bottom shows the system tray with the time 0:29 and various utility icons.

デモ3 VerilogでAND回路を作る(対象ハードはIGLOO開発ボード)

デモ4 Verilogでカウンタを作る(対象ハードはIGLOO開発ボード) 回路図入力に比べて繰り返し等の記述が格段にしやすくなる

The screenshot displays the Xilinx ISE Project Navigator environment. The main window shows the Verilog code for a counter module named 'not1'. The code is as follows:

```
01 // test00v.v
02 module not1 ( CLK, RES, Q);
03     input CLK, RES;
04     output [23:0]Q;
05
06     wire CLK, RES;
07     reg [23:0]Q;
08
09
10
11     always @(posedge CLK or negedge RES)
12     begin
13         if( RES == 1'b0 )
14             Q <= 24'd0;
15         else if(Q == 24'hffffff )
16             Q <= 24'd0;
17         else
18             Q <= Q+ 24'd1;
19     end
20
21 endmodule
22
23
```

The interface includes several panels: 'Design Explorer' showing the project hierarchy with 'not1 (test00v.v)' under 'work'; 'Catalog' with filters and categories like 'Basic Blocks', 'Bus Interfaces', 'Clock & Management', 'Memory & Controllers', and 'Peripherals'; 'Information Window' showing 'Properties' for the file path 'C:\Actelprj\test08v\hdl\test00v.v' and library 'work'; and a 'Log Window' at the bottom showing the message 'Reading file 'test00v.v'. The test00 project was opened.'

The status bar at the bottom indicates 'Ln 1, Col 1', 'VERILOG', 'FAM: IGLOO', 'DIE: M1 AGL600V2', and 'PKG: 484 FBGA'.

デモ5 M1CPUを作る

CoreConsoleでCPU周りを設計する。図中のブロック1つ1つがCorexxxと呼ばれるIP。これらを組み合わせて構築する。

デモ5 M1CPUを作る(対象ハードはIGLOO開発ボード)

CoreConsoleで作成したCPU部分を取り込むTopのHDLを記述する

The screenshot displays the Project Manager software interface. The main window shows the HDL code for a module named `Example_M1AGL_UJTAG`. The code is written in Verilog and includes various input and output signals, as well as external memory interfaces.

```
001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
002 // Created by Actel CoreConsole - Mon Feb 18 13:30:20 2008
003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
004
005 `timescale 1ns/100ps
006
007 module Example_M1AGL_UJTAG(
008     // Inputs
009     CoreUARTapBRX,
010     HCLK,
011     NSYSRESET,
012     Remap,
013     UJTAG_UJTAG_TCK,
014     UJTAG_UJTAG_TDI,
015     UJTAG_UJTAG_TMS,
016     UJTAG_UJTAG_TRSTB,
017     gpioDataIn,
018     // Outputs
019     CoreUARTapBTX,
020     ExternalMemoryInterface_FlashCSN,
021     ExternalMemoryInterface_FlashOEN,
022     ExternalMemoryInterface_FlashWEN,
023     ExternalMemoryInterface_MemAddr,
024     ExternalMemoryInterface_MemReadN,
025     ExternalMemoryInterface_MemWriteN,
026     ExternalMemoryInterface_SramByteN,
027     ExternalMemoryInterface_SramCSN,
028     ExternalMemoryInterface_SramClk,
029     ExternalMemoryInterface_SramDEN,
030     ExternalMemoryInterface_SramWEN,
031     HRESETn,
032     UJTAG_UJTAG_TDO,
033     gpioDataOut,
034     // Inouts
035     ExternalMemoryInterface_MemData
036 );
037
```

The interface also shows the Design Explorer on the left, listing the project files and the current designer view (Imp1). The Catalog window shows the Basic Blocks, Bus Interfaces, Clock & Management, Memory & Controllers, Peripherals, Power Management, and Processors. The Information Window shows the Properties of the selected file (socTop.v), including the file path, library, and creation/modification dates. The Log Window shows the project loading process and a warning message: "Warning: The project was not using the same profiles. The CoreConsole v1.3 or later profile was different from the Example_M1AGL_UJTAG project was opened."

デモ6 M1を使ったカウンタ 構築したCPUで動かすプログラムはSoftConsoleでコーディングする

```

//
// test00src.c
// M1テストプログラム Actel IGL00 スターターキット用
//
typedef unsigned   uint32;
#define PUT_UINT32(addr, data)  (*((volatile uint32 *)
(addr)) = (data))
#define GPIO_BASE_ADDR          0xA0000000

// 空の割り込みハンドラ HAL のsys_boot.cで定義されている
void IRQ_Handler (void)
{
}

void gpio_out(unsigned baseAddr, unsigned dataOut)
{
    PUT_UINT32(baseAddr , dataOut);
}

int main( void )
{
    unsigned m, n;

    m = 0;

    while(1)
    {
        n = m >> 16;

        // Output to the LEDS
        gpio_out(GPIO_BASE_ADDR, n );

        m++;
    }
}

```

通常の
C言語プログラムと
変わらない

記述はいわゆる
32ビット版記述

JTAGを利用したデバッグ実行の様子

The screenshot displays the Actel SoftConsole IDE interface during a JTAG debugging session. The main window shows the source code for `test00src.c` with the following content:

```
int main( void )
{
    unsigned m, n;

    m = 0;

    while (1)
    {
        n = m >> 16;

        // Output to the LEDS
        gpio_out(GPIO_BASE_ADDR, n );
    }
}
```

The line `n = m >> 16;` is highlighted in green, indicating the current execution point. The Disassembly window on the right shows the corresponding assembly instructions:

```
while (1)
{
    n = m >> 16;
    0x0000008c <main+12>: adds r2, r7, #4
    0x0000008e <main+14>: adds r3, r7, #0
    0x00000090 <main+16>: ldr r3, [r3, #0
    0x00000092 <main+18>: lsrs r3, r3, #16
    0x00000094 <main+20>: str r3, [r2, #0

    // Output to the LEDS
    gpio_out(GPIO_BASE_ADDR, n );
    0x00000096 <main+22>: ldr r2, [nc, #2
```

The Variables window at the top right shows the current values of variables `m` and `n`:

Name	Value
(x)= m	2
(x)= n	0

The Debug console at the bottom shows the current thread and its state:

```
test00 [Cortex-M1 debug (Actel)] C:\Documents and Settings\doi\workspace\test00\Debug\test00 (09/02/16 22:52)
```