C-022

Android アプリからの FPGA の利用による処理の高速化

High Performance Computing by Using FPGA from Android App

塩谷 丈史† 成見 哲†
Takeshi Shiotani Tetsu Narumi

1. はじめに

近年では，携帯端末の普及と利用方法の多様化により，アプリケーションから要求されるスペックや機能が増加している．これは専用の周辺回路やモジュール，チップを搭載することで前述の 2 つの問題は解消されるが，携帯端末のプロセッサには電力や放熱の問題によりそれらの要求に応えるには限度がある．FPGA を用いてアプリケーション毎に回路を書き換えられれば，個別の専用回路が無くても専用回路並みの性能が期待できる．

PC を利用する場合には，PC-FPGA Hybrid Cluster [1] に見られるように CPU と FPGA を搭載したシステムを使用することで，アプリケーションに最適化された回路を使用した例がある．近年，ARM コアと FPGA を組み合わせた Programmable SoC と呼ばれる製品が出現しており，本研究ではこの方法を携帯端末に適用することを考えた．

Android OS 上の Android アプリから，パーシャルリコンフィギュレーションにより動作中に処理回路を書き込むことで，ビットパターンの検索回路を実装した．CPU による実行に対し, FPGA での処理速度は約 60 倍高速であるという結果が得られた．以下では，Android とパーシャルリコンフィギュレーションについて重点を置いて説明する．

2. ハードウェア

(1) ZedBoard, Rev. D

Xilinx が販売している Programmable SoC を搭載した ZedBoard を本稿では使用した．また，OS を動作させるための周辺回路を FPGA 部分に構築する必要があり，Xilinx から提供されている ZedBoard_OOB_Design を使用した．

(2) パーシャルリコンフィギュレーション

パーシャルリコンフィギュレーション(以下 PR)と呼ばれる機能を使用することで，FPGA 内部の回路を一部分だけ動作中に書き換えることができる．これを利用してアプリケーションごとに処理用の回路を切り替える．PR で書き込むための回路データは Xilinx Wiki[2] を参考に作成した．

(3) 専用処理回路

処理回路は DMA Engine と接続され，DMA Engine は AXI バスを通して ARM コア及びメモリコントローラと接続される（図１）．処理回路へのデータ転送は DMA によって行われ，データは AXI Stream によって 1 クロックあたり 32bit が伝送される．処理回路自体はアプリによって異なり，使用時に書き換えられる．

3. ソフトウェア

(1) 使用したソフトウェア

本稿では，Xilinx – ISE Design Suite 14.4 を使用して回路を作成した．また，ZedBoard 上で動作させるための OS として，Zynq Android 4.1 [3] を使用した．なお，Zynq Android 4.1 は起動及び動作が不安定だったため，若干の修正を行った．

(2) ドライバ

PR ドライバ，DMA ドライバ，メモリアクセスドライバの 3 つを使用する．このうち PR ドライバについては Digilent が配布している Linux Kernel バージョン 3.6 にドライバ xdevcfg（xilinx_devcfg.c）のコードがあるのでこれを使用した．DMA ドライバは主に AXI によるメモリマ

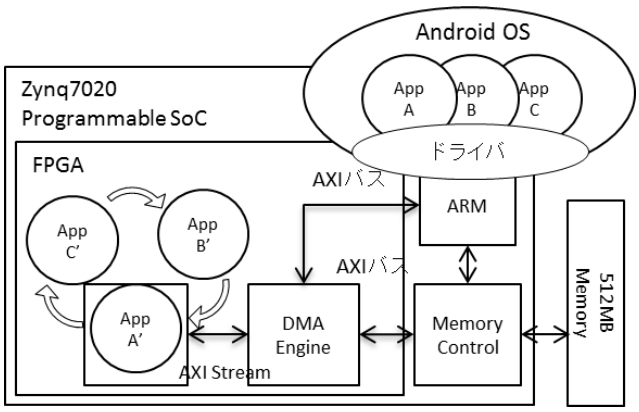

図１ ZedBoard の構成

†電気通信大学 情報・通信工学専攻
Mail: narumi@cs.uec.ac.jp

```
module bitsearch(
  input clk,input resetn,input stall,
  input [31:0] pat,input [63:0] in,
  output [5:0] out);
  wire [31:0] p;
  reg  [31:0] pr = 0;
 assign p[0] =  ((in[63-:32]) == pat);
                :
 assign p[31] = ((in[32-:32]) == pat);
 assign out = sum(pr); // 一致した数を数える
 always @(posedge clk) begin
  if (!resetn) pr <= 0;
  else if (stall) pr <= pr; // DMA のストール処理
  else pr <= p;
 end
endmodule
```

図２　Verilog によるプログラム

```
 unsigned int i, j, c0, c1, pat;
 pat=tar;c0 = data[0];c1 = data[1];
 i = 0;j = 0;
 while (i++ < SIZE_BIT) {
   if (c0 == pat)  j++;
   c0 = (c0 << 1) | (c1 >> 31);
   c1 = (i%32==0)?(data[i/32+1]) : (c1 << 1);
 }
 return j;
```

図３　C によるプログラム

ップレジスタを操作し，DMA Engine（図１）に対し，読み書きアドレスやパラメータのセットを行う．Xilinx のドキュメント pg021_axi_dma [4] を参考に DMA ドライバを作成した．DMA モードはシンプルモードを使用した．また，Android OS では/dev/mem をアプリから利用することを禁止しているため，メモリアクセスのためのドライバを作成した．これは指定されたアドレスを ioremap し利用可能にするものである．

(3)　Android アプリ

前述のドライバを Android アプリから利用するために Android NDK 及び SDK を使用し，ラッパライブラリを作成した．処理回路のデータはアプリパッケージ(apk)に asset として登録した．このデータはアプリ起動時に読み出されて，ドライバを通して FPGA へ書き込まれる．処理回路へのデータの転送やデータの回収は前述の DMA ドライバを通して行う．

4.　速度比較実験

速度比較実験の対象として，ビットパターンの検索を行った．32M ビットのランダムなビット列の中から指定した 32 ビットのビット列が何回現れるかをカウントするものである．Verilog によるプログラムと C によるプログラムをそれぞれ図２と図３に示す．処理回路は 100MHz，CPU は 667MHz による動作である．また，実行はそれぞれ 100 回行い，平均値を求めた．実行した結果，FPGA を用いた方が約 61 倍高速であった(表１)．なお，処理時間には検索対象をメモリに読み込む時間は含めていない．

表 1 実行結果の比較

	CPU	FPGA
処理時間(ms)	650.03	10.606
相対時間	61.289	1

5.　まとめ

携帯端末での利用を想定し，Programmable SoC を用いた場合の CPU-FPGA ハイブリッドデバイスの実現のために，アプリからのパーシャルリコンフィギュレーションの方法と処理回路の利用方法を示した．ここでは，AXI バスへの処理回路の接続と，それを制御するためのドライバが必要であること，アプリからのメモリアクセスには別の手段を用意する必要があること，さらに，それらドライバを Android アプリから利用するにはラッパライブラリの作成に NDK を使用する必要があることを述べた．また，実際に FPGA を利用した処理回路による高速化の例をビットパターンの検索を挙げて実装した. 今回の実験では CPU と比べて FPGA のほうが約 60 倍早く処理できることがわかった．なお，今回使用したテストプログラムは研究室のウェブサイト[5] で公開する．

今後は実際にアプリへ搭載する場合に有効になるシーンや機能を考察する必要がある．また，AXI DMA がプリミティブハードとして実装されていないために，AXI DMA IP コアの占有するリソースが多くなるという問題があり，今後登場する Programmable SoC への搭載が待たれる．

参考文献

[1] 尾崎亮，上嶋明，小畑正貴，”PC-FPGA 複合クラスタにおける部分再構成とその応用”，CPSY，コンピュータシステム 111(398)，2012-01-18
[2] Zynq 7000 Partial Reconfiguration Reference Design, http://www.wiki.xilinx.com/Zynq+7000+Partial+Reconfiguration+Reference+Design
[3] Zynq Android 4.1 (cambridgehackers), https://github.com/cambridgehackers/zynq-android4/wiki
[4] pg021_axi_dma, http://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v6_03_a/pg021_axi_dma.pdf
[5] 成見研究室 wiki ビットパターン検索デモ，http://narumi.cs.uec.ac.jp/wiki720/pub/zedboard/bitbench