C-008

OpenRISC プロセッサの FPGA への実装

Implementation of OpenRISC Processor for FPGA

福地 弘高[1]　秋本 均[1]　阿部 晋一[1]　太田 晋一[2]　小熊 博[2]
Hirotaka Fukuchi　Hitoshi Akimoto　Shinichi Abe　Shinichi Ota　Hiroshi Oguma

テセラ・テクノロジー株式会社[1]
Tessera Technology Inc.

宮城県産業技術総合センター[2]
Industrial Institute of Technology, Miyagi Prefectural Government

1. はじめに

近年, システム開発にソースコードが開示されているオープンソースを用いる事例が非常に増加している. オープンソース型の主流は, GNU や TOPPERS プロジェクトに代表されるソフトウェアであり, オペレーティングシステムやミドルウェア等が多くの組込み機器に搭載されている[1]. 一方, 我々はオープンソース型のハードウェアである OpenCores[2]に注目し, コミュニティ活動及び OpenCores のソースを活用した開発を行っている. OpenCores は, ハードウェア分野のオープンソースコミュニティとして先駆者的な存在であり, 今後も大きな発展が期待される. 本稿では, OpenCores の代表的な RISC プロセッサを含む IP (Intellectual Property)コア及び我々が独自に設計したペリフェラルを FPGA 上へ実装した結果について報告する.

2．システムアーキテクチャ及び実験方法

図 1 に実装したシステムブロック図を示す. 実装ブロックは, OpenCores の IP コアで Verilog-HDL で記述された OpenRISC コア[3], UART コントローラ及び DMA(Direct Memory Access)コントローラと我々が独自に設計した SDRAM コントローラ, VGA コントローラ, Memory コントローラから構成される. OpenRISC コアは OpenRISC プロセッサと内部キャッシュ, 割り込みコントローラ, MMU (Memory Management Unit), タイマー等から構成される. OpenRISC プロセッサは 5 ステージのパイプライン型 32bit の RSIC 型プロセッサである. OpenRISC コアのキャッシュ方式は, ダイレクトマップ構造であり, 回路は非常にシンプルである. OpenRISC プロセッサと各モジュール間はマスタースレイブ型のパラレルバス規格である WISHBONE バスにより接続される.

図 2 に本研究用に開発した FPGA ボード (TSR1000M[4]) を示す. FPGA として Altera 社製 Cyclone を使用した. ボード上には, RS232C, USB, ビデオ出力, イーサーネット等の各インターフェイスとオンボードメモリーとして, SDRAM やフラッシュメモリ等が搭載されている.

表 1 に実装したシステム全体の回路規模を示す. 論理合成及び配置配線には Altera 社製 FPGA の開発ツール Quartus II を用いた. OpenRISC コア及び各種ペリフェラルを含んだシステム全体のロジックエレメント LE 数は 11486, 内蔵メモリは 166656bit であった. FPGA デバイスとして Cyclone (EP1C12)を用いた場合のロジック領域及びメモリ領域の実装率は 95%,69%である.

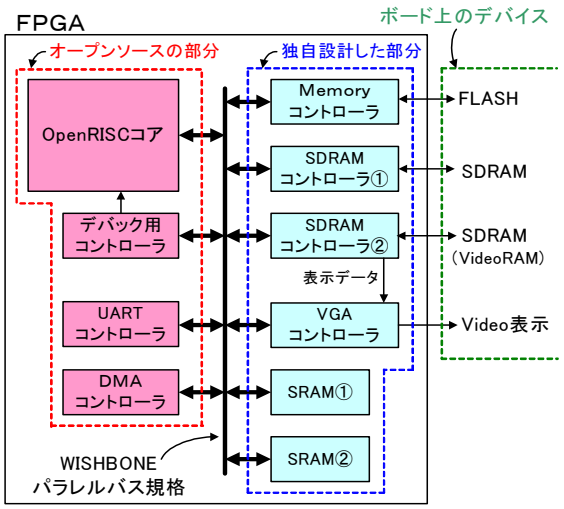

図 1　実装したシステムブロック図

図 2　本研究用に開発した FPGA ボード

表 1　回路規模

回路構成	ロジック規模	メモリー量
OpenRISC コア 各種ペリフェラル QuartusII Ver4.2	11,486(LE) 95%@EP1C12	166,656(bit) 69%@EP1C12

システム評価用に VRAM (Video RAM)への描画処理プログラムを作成した. プログラム開発には OpenRISC プロセッサ用の GCC (GNU Compiler Collection)[5] を用いた. VRAM は 1 ピクセルが 16 ビットの RGB565 の構成とし外部 SDRAM 上に実装した. 表示用原画データは外部フラッシュメモリ上に保存しサイズ 80×60 画素とした. 評価では外部フラッシュメモリ上の原画データを読み出し VRAM 上に転送する処理をし, 処理時間を計測評価した. システム評価には各 CPU, BUS, 各コントローラのクロック周波数及び OpenRISC コアの内部キャッシュの有無やバッファサイズに着目した.

表 2 にクロック周波数の構成を示す. 表 2 のように OpenRISC プロセッサの動作周波数 CPU, WISHBONE バスの動作周波数 BUS, 各コントローラのクロック周波数 FLASH, SDRAM, VGA をパラメーターに 1, 2, 3 の 3 種類の構成とした.

表 2　クロック周波数の構成

構成	クロック周波数(MHz)				
	CPU	BUS	FLASH	SDRAM	VGA
1	25	25	50	50	25
2	30	30	50	50	25
3	25	25	60	60	25

表 3 に OpenRISC コアの構成を示す. 表 3 のように CPU のローカルメモリ QMEM, ストアーバッファ SB, インストラクション用キャッシュ IC, データ用キャッシュ DC のサイズをパラメータに A, B, C, D の 4 種類の構成とした.

表 3　OpenRISC コアの構成

構成	OpenRISC コア構成			
	QMEM	SB	IC	DC
A	8KByte	4Depth	未使用	未使用
B	8KByte	4Depth	4KByte	未使用
C	8KByte	4Depth	未使用	4KByte
D	8KByte	4Depth	4KByte	4KByte

3. 試験結果

図 3 に一例としてクロック周波数を構成 1 (CPU・BUS・VGA クロック周波数を 25MHz, FLASH・SDRAM クロック周波数を 50MHz) , 構成 2 (CPU・BUS・VGA クロック周波数を 30MHz, FLASH・SDRAM クロック周波数を 50MHz) , 構成 3 (CPU・BUS・VGA クロック周波数を 25MHz, FLASH・SDRAM クロック周波数を 60MHz)及び OpenRISC コアの構成を A～D とした場合の計測結果を示す. 構成 A はインストラクション用キャッシュ及びデータ用キャッシュをともに未使用, 構成 B はインストラクション用キャッシュのみ使用, 構成 C はデータ用キャッシュのみ使用, 構成 D は両キャッシュともに使用した場合である.

構成 1 を例にすると構成 A, C のように約 1600msec 以上であった処理時間が, 構成 B, D のようにインストラクション用キャッシュを使用することにより, 400msec 以下と処理時間が 1000msec 以上, 割合にして 1/4 以下の処理時間の短縮化となった. 一方, データ用キャッシュに着目し, 構成 A と構成 C, 構成 B と構成 D とを比較すると約 200msec 程度の処理時間の短縮効果があった. この傾向は, 構成 1, 2, 3 で同じ傾向である. 構成 1 と 2 と 3 とを比較すると, メモリコントローラのクロック周波数を早くした構成 3 が構成 1 と比較して約 200msec 程度早くなった. 以上より, RISC プロセッサの性能を最大限に引き出すにはキャッシュ機能が非常に重要な技術要素であり, 特に OpenRISC プロセッサについてはインストラクション用キャッシュの効果が非常に大きいことがわかった.

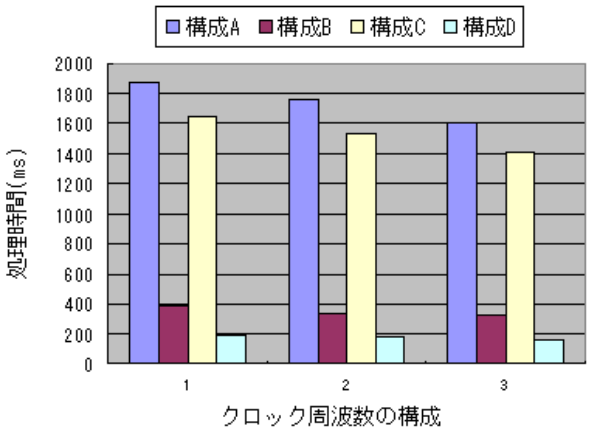

図 3　計測結果

4. まとめ

我々はオープンソース型のハードウェアである OpenCores に注目し, OpenCores の代表的な RISC プロセッサを含む IP コア及び我々が独自に設計したペリフェラルを FPGA 上へ実装し, VRAM への描画処理により評価を行った. その結果, RISC プロセッサの性能を最大限に引き出すにはキャッシュ機能が非常に重要な技術要素であり, 特に OpenRISC プロセッサについてはインストラクション用キャッシュの効果が非常に大きいことがわかった.

今後は他の FPGA 用のプロセッサとの性能比較や OpenRISC によるマルチプロセッサの開発等を行う予定である.

参照文献

[1]例えば, TOPPERS プロジェクト, http://www.toppers.jp/
[2]OpenCores, http://www.opencores.org/
[3]OpenRISC, http://www.opencores.org/ (Project : OpenRISC 1000)
[4]TSR1000M, http://www.opencores.org/ (Project : kiss-board)
[5]GCC, http://gcc.gnu.org/