社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－ＳＬＤＭ－123（６）
2006／1／17

Ｃレベル設計によるストリームアプリケーションのプログラマブルデバイスへの実装

桂 直弘† 長谷川揚平† ヴマン トウアン† 金森 飛匡† 天野 英晴†

† 慶應義塾大学大学院理工学研究科 〒223-8522 横浜市港北区日吉 3-14-1
E-mail: †drp@am.ics.keio.ac.jp

あらまし 本稿では，複数のプログラマブルデバイス（FPGA と動的リコンフィギャラブルプロセッサ）を対象として，いくつかのアプリケーションを C レベル設計技術を用いて実装し，評価を行った．対象とする FPGA のデバイスとして，Xilinx 社の Virtex-II Pro（XC2VP50）を選択し，動的リコンフィギャラブルプロセッサは，NEC エレクトロニクス社の DRP-1 を選択した．FPGA は Look-Up-Table (LUT) によりロジックを実現する細粒度構成で，DRP-1 は粗粒度の演算器のアレイから構成されるプログラマブルデバイスである．実装したアプリケーションの評価結果より，性能や消費電力や設計手法などの違いを比較した．

キーワード プログラマブルデバイス，DRP，FPGA，C レベル設計

Implementation of Stream Application on Programmable Devices by C Level Design

Naohiro KATSURA†, Yohei HASEGAWA†, Vu MANH TUAN†,
Takamasa KANAMORI†, and Hideharu AMANO†

† Graduate School of Science and Technology, Keio University
3-14-1, Hiyoshi, Kohokuku, Yokohama, 223-8522, Japan
E-mail: †drp@am.ics.keio.ac.jp

Abstract While FPGA is a fine grain composition, the Dynamically Reconfigurable Processor (DRP) developed by NEC Electronics is a coarse grain reconfigurable processor; therefore, it would be useful to be able to find out the differences between these two devices in terms of the performance, the power consumption and the design technique through practical applications. This paper quantitatively evaluates these aspects by implementing several applications in C level designs.

Key words Programmable Device, DRP, FPGA, C Level Design

1. はじめに

近年，VLSI の集積率向上にともなって，ハードウェアが大規模化・複雑化している．そのため，開発期間の長期化などの課題が顕在化しつつある．このような課題に対応するために SystemC [1]，SpecC [2] などのシステムレベルの設計技術が発達すると共に，ハードウェアの設計に関しても，Handel-C [3]，Behavioral Description Language (BDL) [4] などの C レベルの高位言語記述などの設計技術が実用化されている．C レベル記述は，Verilog-HDL，VHDL などの RT レベルのハードウェア記述言語と比べて，ユーザがメモリや演算器に対するアクセスタイミング，共有化を明示的に制御する必要がないため，特にマルチメディア処理などを短期間で設計する場合に有効である．特にその設計の容易さから，プログラマブルデバイスに対する利用も進んでいる [5]，[6]．

C レベル設計法は，LUT を用いた汎用の FPGA に対して利用が可能である一方，基本構成要素に演算器やレジスタからなる PE（Processing Element）を用い，それらの構成を動作中に切り替えることができる動的リコンフィギャラブルデバイスに対して特に有効であると言われている [7]．これは，動的リコンフィギャラブルデバイスが，多数の PE を用いてデータパスを形成する点，データパスを動的に変化できる点で，通常のプロセッサに近く，本来通常のプロセッサ上での実行を念頭に置いた C レベル設計法に適しているのではないか，という期待

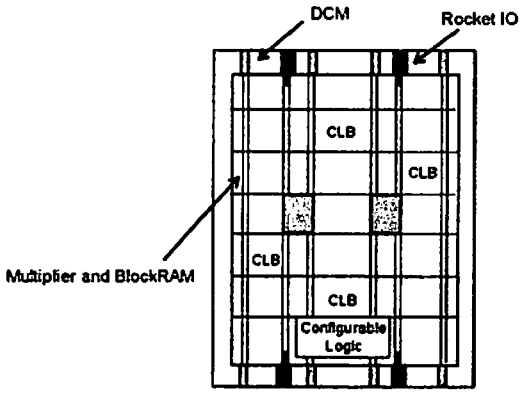

図 1 Vitex-II Pro の構造

があるためである．

動的リコンフィギャラブルプロセッサと FPGA の性能比較はいくつか行われているが，C 言語ベースで設計した同一のアプリケーションに対する比較はほとんど報告されておらず，この点で C レベル設計法とプログラマブルデバイスの適合性についての定量的な検討は行われていない．

そこで，本報告では，Xilinx 社の Virtex-II Pro [8] と，NEC エレクトロニクス社の Dynamically Reconfigurable Processor (DRP) [9] を対象として，C レベル設計によりアプリケーションを実装し，性能や消費電力や設計手法などの違いを比較する．この時，Virtex-II Pro への実装は，Handel-C 言語を用いて設計し，DRP-1 への実装は BDL 言語を用いて設計を行うが，シンタックス上の違いはわずかであり，ほぼ同一の記述を用いることができる．

本稿の構成は以下の通りである．2 節では，対象とするプログラマブルデバイスについて述べる．3 節では，C レベル設計について述べる．4 節では，Virtex-II Pro FPGA と DRP-1 上に実装したアプリケーションの評価結果について述べる．最後に 5 節で結論を述べる．

2. 対象とするプログラマブルデバイス

2.1 Virtex-II Pro

最近の FPGA の発展はめざましく，単に LUT ベースの構成要素数を増やすだけでなく，RAM や演算器，CPU コアを内蔵するものも登場している．FPGA は，汎用の論理回路を搭載できる上，これらの演算器や RAM を用いることによりメディア処理などでも高い効率で動作可能となっている．ここでは，現在ハイエンドのチップとして頻繁に利用される Virtex-II Pro を用いる．Virtex-II Pro は，細粒度構成のデバイスで，2002 年に Xilinx 社から発表され，130nm，1.5V の 9 層の銅配線プロセス技術を使用して製造された．図 1 に Virtex-II Pro の構造を示す．

Virtex-II Pro は，4 つの Slice で構成された CLB によって，論理ブロックを構成している．1 つの CLB 内に搭載されている各 Slice は 2 つの LUT，2 つのフリップフロップを搭載している．各 LUT はロジックとして使用する以外に，16 ビットの RAM，ROM としても使用できる．

表 1 XC2VP50 の構成要素

ロジックセル	53136
CLB 数	88 × 70
SLICE 数	23616
FF 数	47232
Block RAM ビット	4176K
18 ビット× 18 ビット 乗算器	232
PowerPC 405 CPU	2
Rocket IO 数	16
DCM 数	8

データやパリティビットなどを収容する 18K ビットの Block-RAM，高速な信号処理を実現可能にする 18 × 18 ビットの乗算器などの付加機能も多数搭載している．また，FPGA 内部の位相調整や位相シフトなどを行う DCM (Digital Clock Manager) などによって高速な回路を実現することができる．

さらに，Virtex-II Pro には様々なシリアル I/O 規格に対応可能な高速シリアル「RocketIO」がハードウェアに組み込まれており，600Mbps から 3.125Gbps までのデータ転送が可能である．

Virtex-II Pro には，デバイスの大きさとして XC2VP2 から XC2VP100 まであるが，DCT で多くの乗算器を用いるので，Virtex-II Pro XC2VP50 を選択した．今回用いる Virtex-II Pro XC2VP50 の構成要素を表 1 に示す．

2.2 DRP-1

DRP-1 は，2002 年に NEC エレクトロニクス社から発表された動的リコンフィギャラブルデバイスであり，Tile と呼ばれるリコンフィギャラブルユニットを構成単位とする．

各 Tile は図 2 に示す通り，8 × 8 の Processing Element (PE) アレイ，状態制御を行なうシーケンサである State Transition Controller (STC) から構成される．また，8 ビット× 256 エントリのメモリ 8 セットとこれらを制御するメモリコントローラ 2 セットを両側にもち，8 ビット× 8192 エントリのメモリ 4 セットとメモリコントローラを Tile の上部もしくは下部にもつ．DRP-1 全体では 8 ビット× 256 エントリのメモリを合計 80 セット，8 ビット× 8192 エントリのメモリを合計 32 セットもつ．

各 PE は図 3 に示す通り，8 ビットの ALU，シフトやデータ制御，単純な論理演算を行う Data Manipulation Unit (DMU)，8 ビットの Flip Flop，レジスタファイルから構成される．また，コンフィギュレーション時には命令データが命令メモリに書き込まれ，実行中に STC が発行した命令ポインタを命令メモリが受け，利用する命令データをロードすることでハードウェア構成を変更する．コンテキスト切り替えは 1 クロックで可能であり，Tile 単位の部分再構成も可能である．

PE アレイの周辺部には 32 ビットの乗算器を 8 セット，メモリモジュール，PCI バスインタフェース，SDRAM/SRAM/CAM インタフェースを搭載している．また，PCI バスインタフェー

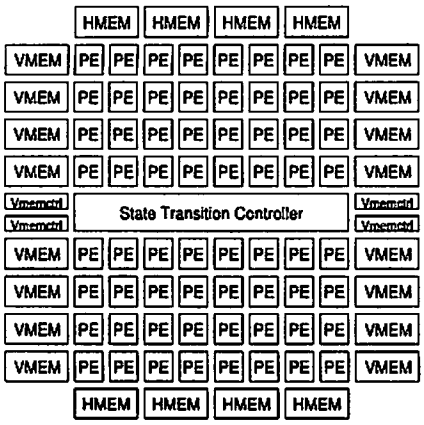

図 2 Tile の構造

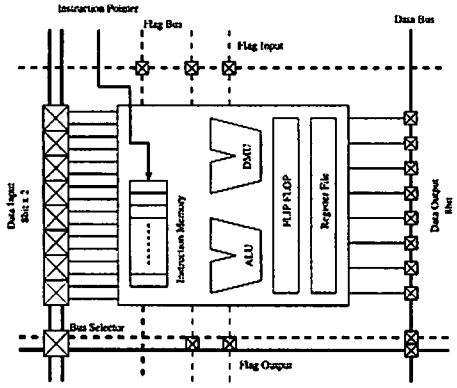

図 3 Processing Element (PE) の構造

ス，SDRAM/SRAM/CAM インタフェースにより単独で PCI バスへの接続や外部メモリの制御が可能である．

3. C レベル設計

3.1 FPGA への実装

図 4 に Handel-C を用いた FPGA への実装の流れ図を示す．Handel-C 言語は，C 言語をハードウェア記述用に拡張した言語である [3]．記述レベルは，RTL と動作レベルの中間に位置し，遅延のない同期回路設計に抽象化されている．

Celoxica 社の DK デザインスイート (DK4) は，設計エントリからシミュレーションおよびアーキテクチャの探求から合成までを通して，C 言語ベースの設計のための完全な統合開発環境である．Handel-C による記述は，DK4 により機能合成が行われ，ハードウェアの構造と動作を示す Verilog ソースコードに変換される．この Verilog ソース・コードは，Xilinx 社の ISE 7.1i の Project Navigator で Virtex-II Pro XC2VP50 FPGA をデバイスの対象として，論理合成と検証を行い，FPGA 上の性能を検証する．

ISE 7.1i の Project Navigator では次の順番で論理合成と検証を行う．

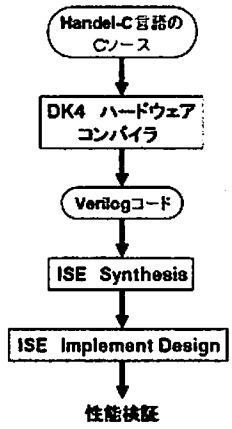

図 4 Handel-C を用いた FPGA への実装フロー

(1) DK4 が生成した全 Verilog ソースファイルの読み込む．ここで，デバイス Virtex-II Pro (XC2VP50) を指定する．

(2) Synthesize．論理合成を行う．

(3) Implement Design (Map)．FPGA デバイスへのモジュールへのマッピング．

(4) Implement Design (Place & Route)．FPGA デバイス上での配置配線．

(5) Timing Analyzer．動作周波数の検証

(6) XPower Simulation Model を用いて VCD (Value Change Dump) ファイルを作成し，それを用いて XPower で消費電力を測定する

本報告では，上記の Timing Analyzer により動作周波数を求め，XPower により消費電力を測定している．

3.2 DRP-1 への実装

BDL 言語は，C 言語をハードウェア記述用に拡張した言語で，アルゴリズムレベルから機能レベルのハードウェアを記述することができる [4]．DRP-1 の開発環境として，NEC 社の統合開発環境 Musketeer が提供されている．Musketeer は，BDL ソースコードの記述から実行可能オブジェクトコードの合成，シミュレータによる検証，実機でのデバックまでの全ての開発・検証作業をサポートする．

図 5 に Musketeer のコンパイルフローを示す．コンパイラは大きく分けるとフロントエンド合成部とバックエンド合成部に分けられる．フロントエンド合成部である動作合成部が，C 記述から RTL への変換をする．動作合成部の出力する RTL は制御回路である FSM とデータパス回路からなる．制御 FSM は STC に，データパス回路が PE アレイに割り当てられる．DRP アーキテクチャは，状態遷移マシンを STC へ，状態毎のデータパスを PE アレイへそれぞれ自然にコンパイルすることができる．動作合成部の出力は Verilog 形式で，各状態のデータパスをモジュール化した構造となっている．動作合成の結果として，使用される状態数，各状態の予測 PE 数，クリティカルパスの予測遅延などのレポートが参照可能である．

BDL ソースコードは，gcc などの標準的な C コンパイラに

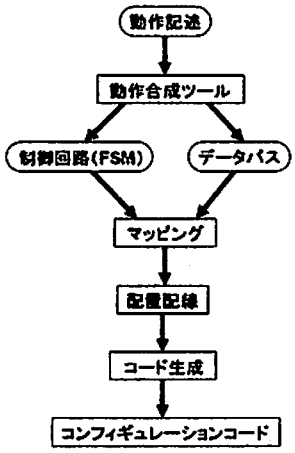

図 5　DRP-1 におけるコンパイルフロー

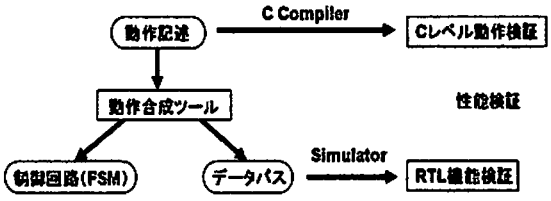

図 6　DRP-1 における機能検証

よってコンパイルすることもでき，デバッガなど C 言語の開発環境を利用してアルゴリズムの動作検証を行うことができる（C レベル動作検証）．また，動作合成部の出力（Verilog 形式）を標準的な Verilog シミュレータによって機能検証を行うことも可能である（RTL 機能検証）．この RTL 機能検証および実機を用いたオンチップエミュレーションを行うと，DRP-1 上でのサイクル数や消費電力などの性能を検証できる．ただし，現状では消費電力については，一部をモデル化したに過ぎず，参考値である．図 6 に C レベル動作検証と RTL 機能検証の流れ図を示す．

3.3　Handel-C と BDL

Handel-C と BDL は，ANSI-C に以下のような制限を設けている．

- ポインタ利用の制限
- 再帰呼び出しの禁止
- 動的メモリ確保の禁止
- ライブラリ関数コールの禁止

一方，以下の拡張を行っている．

- ビット幅の指示
- ビット幅の変換指示
- 入出力の指定

基本的な制限と拡張は，ほとんど変らない．今回の評価では，共通の C レベル記述から，ほとんど直接，両者の記述に変換した．両者共に，最適化のための記述を持っているが，これらは今回の評価では利用せず，後に解説するようにもっとも基本的な記述を用いてハードウェアを生成した．

表 2　XC2V50 と DRP-1 の性能（DCT）

プロセッサ	クロック数	動作周波数	実行時間	消費電力
XC2V50	653	70.4 MHz	9.27μs	757.6 mW
DRP-1	89	27.8 MHz	3.20μs	808.9 mW

XC2V50 への実装に用いた Handel-C 言語は，par 構文を使用して，複数行の動作を 1 サイクルの動作として合成する最適化手法がある．これに対し，DRP-1 への実装に用いた BDL 言語のコーディングスタイルとして，手動スケジューリングモードと自動スケジューリングモードの 2 つがある．

本稿では公平を期すために，両者共に最適化手法を利用せず，自動的な実装法を用いた方法を採用した．すなわち，Handel-C では，人手での最適化手法は用いず，FPGA 実装時に人手で割り当てる必要がある BlockRAM などは使用しないこととした．一方，BDL では自動スケジューリングモードを選択した．したがって，今回の評価結果は，人手による最適化によりさらに高速化あるいは低コスト化を測ることが可能である点に注意されたい．

4.　評　　価

4.1　評価条件

本報告では，以下のアプリケーションを用いた．

- JPEG エンコーダにおける離散コサイン変換（Discrete Cosine Transform : DCT）
- 一方向ハッシュ関数 Secure Hash Algorithm 1（Secure Hash Algorithm 1 : SHA-1）
- JPEG2000 デコード時の離散ウェーブレット変換（Discrete Wavelet Transform : DWT）

上記のアプリケーションを XC2V50 と DRP-1 それぞれに C レベル設計により実装し，実行クロック数，動作周波数，実行時間，消費電力を測定した．サイクル数に関しては，XC2V50 と DRP-1 に関して，入出力のサイクル数は含めないこととし，アプリケーションを処理している部分のみカウントした．

XC2V50 の消費電力は XPower を用いて見積もった．XPower は ISE と連携させることで，論理合成後の XC2V50 の消費電力を測定することができる．消費電力推定のためには，各信号のスイッチング確率が必要となり，VCD ファイルを用いて計算される．

DRP-1 の消費電力には，統合開発環境 Musketeer における電力プロファイラの算出した値を用いる．この消費電力は，DRP-1 の PE アレイ部のみの電力で，I/O などの 3.3V 系の電力は含まれていない．また，プロファイラの算出する推定値は 8Tile での消費電力である．以上のように消費電力は DRP-1 に対しては参考値であると言って良い．

4.2　評価結果とその検討

4.2.1　DCT

DCT を XC2V50 および DRP-1 に実装した場合の測定結果を表 2 に示す．

表 3 DCT の使用リソース (DRP-1)

context	ALU	DMU	FFU	RFU
0	0	2	0	2
1	15	14	0	11
2	231	228	160	8
3	107	117	279	4
4	181	166	165	53
5	75	76	32	74
6	36	36	30	42
7	23	23	12	4
合計	668	662	678	198

表 4 XC2V50 と DRP-1 の性能比較 (SHA-1)

プロセッサ	クロック数	動作周波数	実行時間	消費電力
XC2V50	3750	77.9 MHz	48.1 μs	955.0 mW
DRP-1	639	27.1 MHz	23.6 μs	612.7 mW

表 2 より，動作周波数については XC2V50 が DRP-1 の 2.5 倍であるが，DRP-1 の実行クロック数が大幅に少なく，全体の実行時間については DRP-1 の方が XC2V50 より約 2.9 倍高速になった．これは，DRP コンパイラが DRP-1 上の演算器を並列に動作させているのに対し，XC2V50 上のリソースを Handel-C による記述と合成が十分使いきっていないことに起因するものと考えられる．

リソースについて調べてみると，XC2V50 は，全体のスライス数の 4784 (20%) を費やしたに過ぎない．一方で DRP-1 では表 3 のようになり，コンテキスト上の資源を用いてうまく並列処理を行っていることがわかる．ただし，このため消費電力については，やや XC2V50 の方が有利となっている．

4.2.2 SHA-1

SHA-1 の実装結果を表 4 に示す．

同様に XC2V50 は周波数は高いが実行クロック数が大きい，このアプリケーションは全体として DRP-1 の方が約 2.0 倍性能が良いという結果となった．

コストについて調べると XC2V50 は，スライスを 10862 (46%) を費やしたのに対して DRP-1 の利用リソースは表 5 に示す結果となった．

SHA-1 は全体的に並列性が小さいため，DRP-1 上でもさほど並列処理ができず，一方で，XC2V50 は DCT に比べて資源をある程度は有効に利用することができ，このため，両者の速度の差は少なくなっている．一方で，消費電力については，PE 利用数が減った DRP-1 の方が XC2V50 よりも小さくなり，このアプリケーションでは DRP-1 が全面的に有利となった．

4.2.3 DWT

DWT に関する測定結果を表 6 に示す．

DWT についても同様に XC2V50 が周波数の点で有利だがクロック数を要するために，DRP-1 の方が約 2.7 倍性能が良いという結果になった．利用リソースについては，XC2V50 は，スライスを 8974 (38%) を費やしたのに対して DRP-1 上の利用リソースを表 7 に示すようになった．

表 5 SHA-1 の使用リソース (DRP-1)

context	ALU	DMU	FFU	RFU
0	0	0	2	2
1	56	0	25	56
2	28	3	52	52
3	3	0	4	4
4	3	0	4	4
5	3	0	4	4
6	3	0	4	4
7	3	0	4	4
8	61	20	68	68
9	11	6	6	11
10	116	28	80	119
合計	287	57	253	328

表 6 XC2V50 と DRP-1 の性能比較 (DWT)

プロセッサ	クロック数	動作周波数	実行時間	消費電力
XC2V50	5641	69.6 MHz	81.8 μs	1002 mW
DRP-1	903	29.5 MHz	30.6 μs	596.3 mW

表 7 DWT の使用リソース (DRP-1)

context	ALU	DMU	FFU	RFU
0	0	2	0	0
1	57	60	12	28
2	52	52	5	25
3	17	17	0	6
合計	126	131	17	59

SHA-1 程ではないが，DWT もさほど多くの並列性を持たないため，利用 PE 数はさほど多くない．このため，消費電力においても DRP-1 の方が XC2V50 よりも消費電力が低い結果になった．

5. むすび

全てのアプリケーションに対して DRP-1 は XC2V50 に比べて周波数の点で低いが，実行クロック数が少なくて済むことから性能的に優位に立っている．結果として DRP-1 は XC2V50 に比べて DCT は約 2.9 倍，SHA-1 は約 2.0 倍，DWT は 2.7 倍の速度で実行することができた．これは，DRP コンパイラが，通常の C 言語記述から積極的に並列性を検出して多数の PE を活用するのに対して，Handel-C の合成系の DK4 が基本的に C 言語の 1 ステップを 1 クロックで実行するように合成するため，周波数を上げることができても，並列性を生かすことが難しいためであると考えられる．また，消費電力に関しては DRP-1 は XC2V50 と比べて，DCT 以外のアプリケーションにおいて低いことがわかった．

デバイスプロセスの点で考えると DRP-1 は 0.15μm であるのに対して XC2V50 は 0.13μm であり，むしろ XC2V50 の方が進んでいる．したがって，今回の評価結果から，C 言語による設計で最適化を人手で施さない場合は，動的リコンフィギャ

ラブルプロセッサは FPGA よりも有利であると言える．今回の評価はアーキテクチャが優れているのかコンパイラが優れているのかその貢献度を区別することができないが，C レベル設計を行う場合についてはコンパイラはアーキテクチャを効率良く利用することが可能であり，総合的に考えて動的リコンフィギャラブルプロセッサが有利であると結論できる．

FPGA の場合，DK4 レベルで人手で最適化を行った上、ISE レベルでも最適化を行うことができる．一方，DRP-1 でもマニュアルモードを用いればより高い性能が得ることが可能であり，限界までチューニングした場合についても性能比較を行う必要がある．ただし，この場合はチューニングを行う設計者の腕と時間が関係するため、公平な評価を行うのは難しい．

今回の評価は，必ずしも FPGA が動的リコンフィギャラブルシステムに比べて性能面で劣るという主張をしているのではない．FPGA は現在では HDL での設計がほとんどであるので，HDL を用いて設計し最適化を行えば，DRP-1 に匹敵あるいは凌駕することも考えられる．しかし，この場合アルゴリズムレベルからの変換が必要であり，設計工数も相当異なる点で比較するのは公平な評価であるとは言えない．

ただし，組み込みシステムでは，場合によっては相当な時間をかけて対象アプリケーションに対してチューニングすることが許されるため，これらの評価も将来は必要であると考えられる．

謝　　辞

本研究を進めるにあたり DRP-1 とその開発環境は NEC エレクトロニクス社および NEC システムデバイス研究所に提供していただきました．デバイスおよび開発環境について，日頃から有益な御助言，御指導を頂きました事に著者一同深く感謝します．

文　　献

[1] Open SystemC Initiative. http://www.systemc.org/.
[2] SpecC Technology Open Consortium. http://www.specc.org/.
[3] Celoxica Inc.: "Handel-C Software-Compiled System Design". http://www.celoxica.com.
[4] 粟島, 戸井, 中村, 紙, 加藤, 若林, 宮淵, 李："動的再構成可能チップ DRP の C コンパイラ", 電子情報通信学会技術研究報告 VLD2003-118, 103, 578, pp. 23–28 (2004).
[5] 久重路, 樋口, 福山, 柴村, 久我, 末吉："System C による JPEG2000 エンコーダの設計と FPGA 実装", 第 2 回リコンフィギャラブルシステム研究会予稿集, pp. 10–15 (2003).
[6] 金森, 福田, 味岡, 新井, 橋本, 天野："Reconfigurable System におけるナンバープレート認識アルゴリズムの実装", 電子情報通信学会技術研究報告 RECONF2005-16, 105, 43, pp. 7–12 (2005).
[7] 末吉, 天野（編）："リコンフィギャラブルシステム", オーム社 (2005).
[8] Xilinx Inc.: "Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet".
[9] M. Motomura: "A Dynamically Reconfigurable Processor Architecture", Microprocessor Forum (2002).