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

2009-SLDM-138 (29)
2009 / 1 / 30

大規模回路エミュレーション用 90nm CMOS マルチコンテクスト FPGA の遅延評価

宮本 直人† 大見 忠弘†

†東北大学 未来科学技術共同研究センター 〒980-8579 宮城県仙台市青葉区荒巻字青葉 6-6-10
E-mail: †miyamoto@fff.niche.tohoku.ac.jp, ohmi@fff.niche.tohoku.ac.jp

あらまし 大規模な回路のハードウェアエミュレーションを目的として，90nm CMOS マルチコンテクスト FPGA『フレキシブルプロセッサ』を設計・試作した．今までは，マルチコンテクスト FPGA を用いた回路エミュレーションは，時分割多重をする分，性能面では不利になると考えられてきた．本論文では，コンテクスト間のデータ交換を高速に行う時通信モジュールと，クリティカルパスを全コンテクストに最短かつ均等に分割する時分割アルゴリズムを提案する．これらの技術により，フレキシブルプロセッサは，マルチコンテクスト実行した場合でも，シングルコンテクストとほぼ同じ速度で動作することを確認した．

キーワード マルチコンテクスト，FPGA，コンテクスト間データ通信，時分割アルゴリズム

Delay Evaluation of a 90nm CMOS Multi-Context FPGA for Large-Scale Circuit Emulation

Naoto MIYAMOTO† and Tadahiro OHMI†

†New Industry Creation Hatchery Center, Tohoku University 6-6-10 Aza-Aoba, Aramaki, Sendai, 980-8579 Japan
E-mail: †miyamoto@fff.niche.tohoku.ac.jp, ohmi@fff.niche.tohoku.ac.jp

Abstract For large-scale circuit emulation with using a multi-context FPGA (MC-FPGA), a circuit is divided into multiple sub-circuits, each sub-circuit is assigned to a context, and the MC-FPGA sequentially executes all the contexts one by one. So, the total execution delay is the sum of the delays of all the contexts. It is, therefore, said that the total execution delay of the MC-FPGA increases proportional to the number of contexts used. However, in this paper, we show that the total execution delay remains constant if a shift-register-type temporal communication module (SR-TCM) is used instead of D-FlipFlop (D-FF) to implement sequential circuits. The SR-TCM is used not only for sequential circuit like D-FF, but also for signal communication from preceding context to succeeding contexts. In order to quantify the execution delay, a MC-FPGA named Flexible Processor (FP), which equips the SR-TCM, have been designed and fabricated in 90nm CMOS process technology. From the measurement results, the total execution delay of the FP was kept constant regardless of the number of contexts used.

Keyword Multi-Context, FPGA, Temporal Signal Communication, Temporal Circuit Partitioning

1. はじめに

現在のシステム LSI 設計は，RTL 設計・検証の短期間化が最重要課題の一つである．図 1 に示すように，RTL 設計・検証に要する期間や工数がシステム LSI 開発全体の 6〜7 割を占めている[1].

システム LSI の回路検証を効率化する一手段としてハードウェアエミュレータがある．ハードウェアエミュレータは，ハードウェアの機能検証をより高速化するために用いたり，ソフトウェアの開発効率を上げるため，実チップが製造される前のプロトタイピングとして用いられる．

ハードウェアエミュレータは汎用の FPGA か，それに類似した LSI を使用する．FPGA は同じデザインルールのもとで集積度は 5〜12 分の 1 になる[2]ため，大規模回路をエミュレーションするためには大型の FPGA を複数個用い，さらにその間を配線するためのスイッチング LSI も必要となる．必然的に，装置単価が高くなってしまう．

図 1 システム LSI 設計全体の工数の割合

そこで，我々はワンチップで大規模回路をエミュレーションするため，マルチコンテクスト FPGA『フレキシブルプロセッサ』を開発してきた[3]-[6]．従来のフレキシブルプロセッサの問題は，マルチコンテクスト実行すると，時分割多重をする分，エミュレーション速度が遅くなることだった．

本論文では，コンテクスト間の高速なデータ通信を行う時通信モジュール（TCM）と，クリティカルパスを全コンテクストに最短かつ均等に分割する時分割アルゴリズムを提案する．

TCM を搭載するフレキシブルプロセッサを設計し，90nm CMOS プロセスを用いて試作した．LSI テスタによる測定の結果，フレキシブルプロセッサはマルチコンテクスト実行した場合でも，シングルコンテクストとほぼ同じ速度で動作することを確認した．

2. フレキシブルプロセッサ

2.1. 動作メカニズム

フレキシブルプロセッサは，大規模回路のエミュレーション用に開発されたマルチコンテクスト FPGA である．その動作概要を図 1 に示す．エミュレーション回路（Original Circuit）のネットリストは，後述する時分割アルゴリズムにより複数の部分回路ネットリストに分割される．ここで，部分回路の規模はフレキシブルプロセッサのロジックキャパシティ以下となる．次に，部分回路のネットリストはそれぞれコンテクストに配置配線される．その後，コンテクストはフレキシブルプロセッサ上で 1 つずつ順番に実行される．

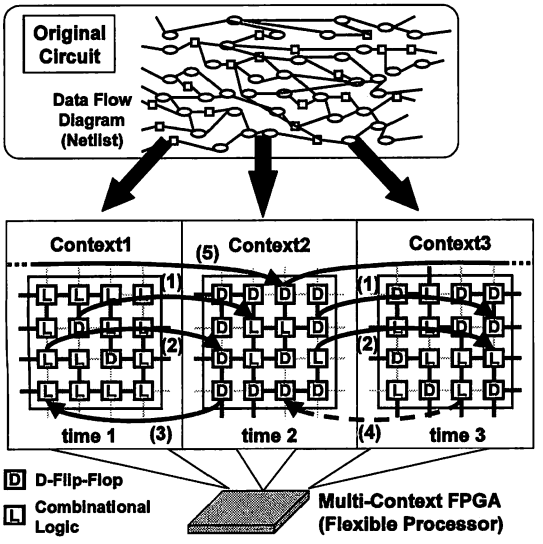

図 2　フレキシブルプロセッサを用いたハードウェアエミュレーションの概念図

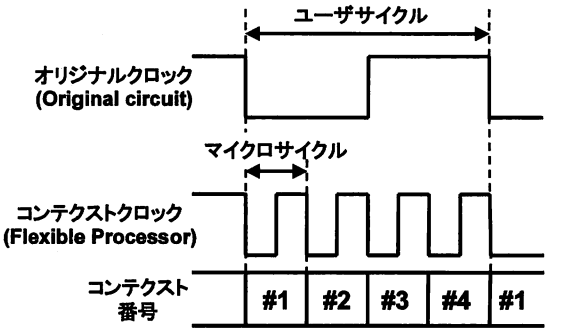

図 3　エミュレーション回路の実行サイクルとマルチコンテクスト FPGA の実行サイクルの関係（エミュレーション回路を 4 コンテクストに時分割した場合）

エミュレーション回路の 1 クロックサイクルをユーザサイクルと呼び，1 コンテクストのクロックサイクルをマイクロサイクルと呼ぶ（図 2 参照）．全てのコンテクストを順に 1 クロックずつ実行した結果は，エミュレーション回路を 1 クロック実行した結果に等しい．したがって，1 ユーザサイクルの実行時間は全コンテクストのマイクロサイクルの実行時間の総和となる．

2.2. 時通信モジュール（TCM）

回路を複数の部分回路に分割した場合，部分回路間にわたる配線が生じる．これらの配線をフレキシブルプロセッサに実装するためには，コンテクスト間のデータ通信を論理的矛盾なく行う記憶要素が必要となる．ここで，コンテクスト間のデータ通信を時通信と呼ぶ．時通信をサポートする記憶要素が時通信モジュール（TCM）である．

時通信には以下の 5 種類がある．

(1) Forward Sequential：D-FlipFlop から出力され，後ろのコンテクストへ向かう通信

(2) Forward Combinational：組合せ論理回路から出力され，後ろのコンテクストへ向かう通信

(3) Backward Sequential：D-FlipFlop から出力され，前のコンテクストへ向かう通信

(4) Backward Combinational：組合せ論理回路から出力され，前のコンテクストへ向かう通信（注：この通信は論理的矛盾を起こすため，使用してはならない）

(5) State Communication：D-FlipFlop から出力され，次のユーザサイクルの同一マイクロサイクルへ向かう通信

ここで，(1)～(5)までの番号は図 1 に示した時通信の矢印と対応している．

現在までさまざまな TCM が提案されている．その中で最も典型的なものが Time-Multiplexed FPGA に採用された micro-register である[7]．図 4 にそのブロック図を示す．Micro-register は N ビットのランダムアクセス可能なレジスタファイルを持つ．ここで N はオンチップの最大コンテクスト数である．

"Input"信号は全てのレジスタの入力ポートに接続するため，オンチップコンテクスト数が増加すると比例的にファンアウトの数が増える．その結果，TCM のセットアップ時間が増加してしまう．さらに，micro-register はどのレジスタに書き込むかを決定する"Enable"信号が必要である．そのため，付加的な制御回路やコンフィギュレーションメモリが必要となる．

本研究で開発した新 TCM を図 5 に示す．この TCM はレジスタファイルのかわりにシフトレジスタを持つ．シフトレジスタの R_1 から R_N までの N 個の D FlipFlop で構成される．先程と同様に，N はオンチップの最大コンテクスト数である．

"Input"信号は一番先頭のレジスタにのみ接続されるため，最大コンテクスト数が増えてもファンアウトは 1 のまま変わらない．また，縦列接続のシフトレジスタは"Enable"信号を必要としないため，付加的な制御回路やメモリが必要ない．

シフトレジスタが保持する時通信データがコンテクスト実行毎に変化する様子を図 6 に示す．図中，四角で囲われた#N (N=1～4) は，最後にコンテクスト#N を実行した時の LUT 出力をそのレジスタが保持していることを意味する．シフトレジスタ型 TCM は，最

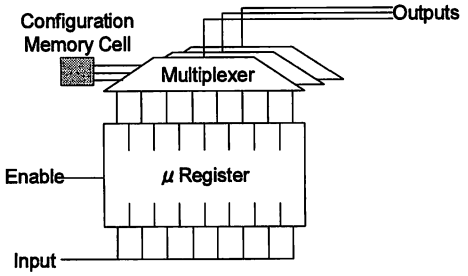

図4　Time-Multiplexed FPGA[7]のmicro register

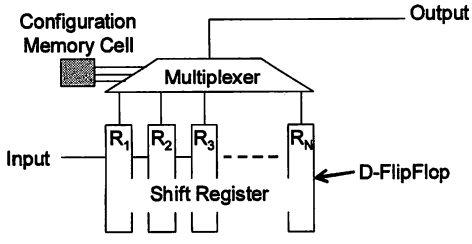

図5　提案するシフトレジスタ型TCM

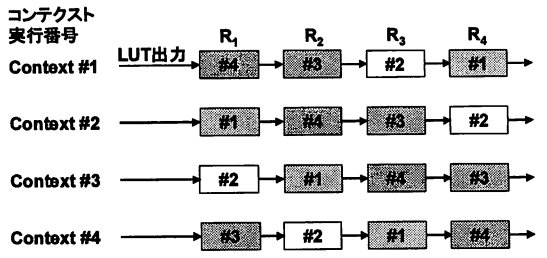

図 6　シフトレジスタが保持する時通信データの変遷（4 コンテクスト実行の場合）

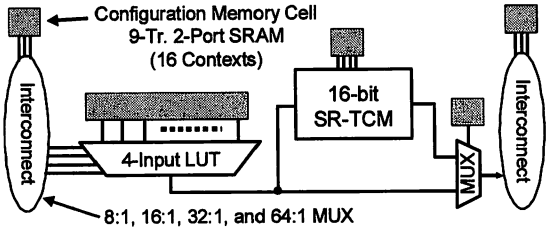

図 7　論理素子のブロック図

大 1 ユーザサイクル前に遡り，エミュレーションに必要な時通信データを全て保持していることがわかる．

このように，エミュレーション用途に特化した場合，実行順序が規則的であるため，TCM の構造を単純化することができる．

2.3. 論理素子 (LE)

図 7 にフレキシブルプロセッサに搭載した論理素子のブロック図を示す．この論理素子は 4 入力 LUT，シフトレジスタ型 TCM（SR-TCM），出力 MUX，コンフィギュレーション用メモリセルで構成される．通常の FPGA が持つ論理素子との違いは，D-FlipFlop のかわりに TCM が搭載されていることである．

2.4. プログラマブル配線アーキテクチャ

フレキシブルプロセッサのプログラマブル配線アーキテクチャとして，8:1MUX，16:1MUX，32:1MUX，および 64:1MUX から成る Nearest Neighbor 型配線アーキテクチャを採用した．一般的に，Nearest Neighbor 型配線アーキテクチャはルータビリティが低い．しかし，エミュレーション用途に特化した場合，実行順序が規則的になるため，時間方向の配線を活用することにより，空間的な配線ルータビリティの不足を補うことができると考えている．

時間方向の配線は，シフトレジスタ型 TCM の場合，シフトレジスタ内でデータが移動するだけで済むため，空間配線を一切必要としない．

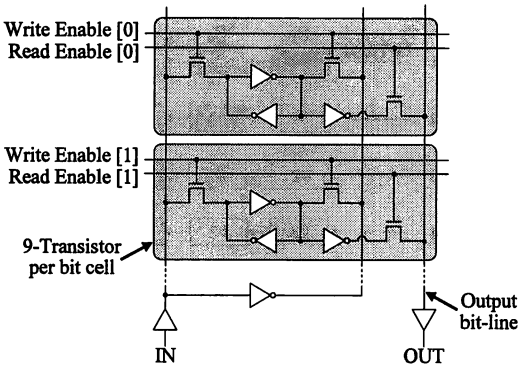

図8　コンフィギュレーションSRAMの回路図

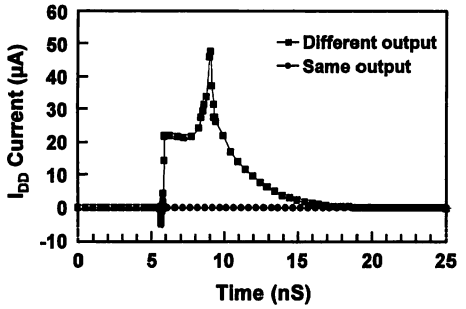

図9　コンフィギュレーションSRAMの消費電流（SPICEシミュレーション結果）

2.5. コンフィギュレーション SRAM

コンテクスト切り替え時，3%以下のコンフィギュレーションデータビットしか変化しないという報告がある[8]．そこでフレキシブルプロセッサのコンフィギュレーション SRAM として，図 8 に示すようなデュアルポート SRAM を開発した．この SRAM は 1 セル 9 ビット構成で，デュアルポートのうち 1 つは書き込み専用，もう 1 つは読み出し専用となっている．前後のコンテクストで同一のデータビットの場合，出力ビットラインでの電荷の充放電が生じない．

この SRAM の消費電力を SPICE シミュレーションにより求めた．図 9 から，切り替え前後のコンフィギュレーションビットが異なる場合と同じ場合とで流れる電流は大きく異なることがわかる．消費電力を算出したところ，ビットが異なる場合は 68.7fJ/bit でかかり，ビットが同じ場合 0.52fJ/bit で済むことがわかった．

2.6. 実装結果

フレキシブルプロセッサを 90nm CMOS テクノロジ 1P6M プロセスを用いて設計・試作した．チップ写真を図 10 に示す．オンチップ 16 コンテクストで，1 コンテクストあたり 16x16 個の論理素子と 16x4 個のユーザ I/O が利用できる．ダイサイズは 2.5x2.5mm^2，コアサイズは 1.36mmx1.15mm^2 である．コアと I/O の電源電圧はそれぞれ 1.0V と 2.5V である．

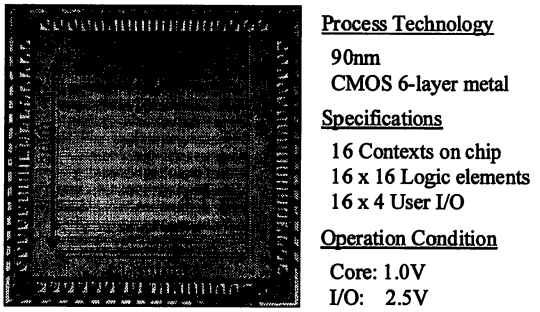

図10　フレキシブルプロセッサのチップ写真

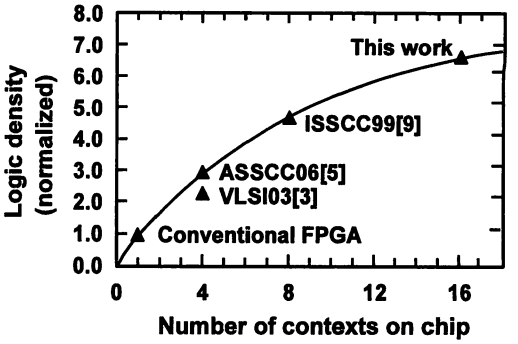

図11　論理密度（単位面積あたりのLUT数）の比較

図 11 は過去に IEEE Solid-State Circuit Society で発表されたマルチコンテクスト FPGA の論理密度を比較したものである．ここで論理密度は単位面積あたりの LUT 数で，通常の FPGA を 1 として規格化した．プロセステクノロジの違いも反映している．図から，オンチップコンテクスト数が増加すると論理密度が増加することがわかる．一方，論理密度の増加はコンテクスト数に比例しない．これは，コンテクストメモリの面積的なオーバーヘッドによるものである．

3. 時分割アルゴリズム

3.1. 自動配置配線ツール

マルチコンテクスト FPGA 用自動配置配線ツール PELOC（Processor Element Locator）を開発した．これは，データフローグラフを入力とし，マルチコンテクストのコンフィギュレーションデータセットを出力する．PELOC はエミュレーション回路を時間的に分割するだけでなく，分割された部分回路間の時通信を TCM に割り付けることができる．

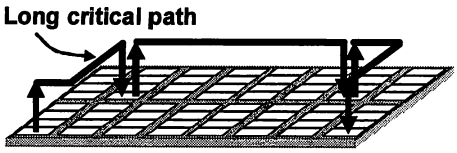

図12(a)　FPGA上のクリティカルパス

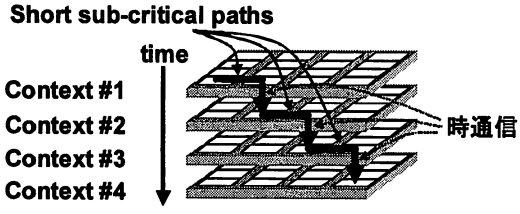

図12(b)　マルチコンテクストFPGA上のクリティカルパス（4コンテクストに分割した場合）

3.2. マルチコンテクスト実行時の遅延最小化

通常のFPGAに対してマルチコンテクストFPGAはクリティカルパスを短くできる可能性があると我々は考えている．図12に示すように，マルチコンテクストFPGAはエミュレーション回路を一時刻に部分的に実装できるからである．全コンテクストを通して，クリティカルパスは一部の領域に集中して配置される．その結果，不要な配線スイッチや長距離配線の挿入を抑止し，クリティカルパスを最短で配置配線できると考えている．

このような動機から，新たに時分割アルゴリズムを開発した．このアルゴリズムはエミュレーション回路中のクリティカルパスを検出し，それを等間隔にコンテクスト数だけ分割する．分割されたクリティカルパス同士は時通信で結合される．

3.3. 時分割アルゴリズムの詳細

まず最初にForce Directed Scheduling (FDS)[10]を用いてエミュレーション回路の初期分割を行う．FDSにより回路中のクリティカルパスの位置と長さが判明する．分割されたクリティカルパスが全コンテクストに均一に分散するよう回路分割を行う．

FDSはネットリスト中のセルのIN-OUTの繋がりを探索するに過ぎないため，しばしばカットコストの増大を招いてしまう．TCMの物理的な数に制限があるため，カットコストの増大は配置配線の失敗につながる可能性がある．そこでFDS後のカットコストを減少させるために，Modified Fiduccia-Matheyses（FM）法を開発した[11]．FM法[12]はミンカット分割法の一種で，分割線と交差する配線数を最小化するための発見的アルゴリズムである．

提案するModified FM法がFM法と異なる点を以下に示す．

（1）　Baseセルの移動により Backward Combinational時通信（図2および2.2節参照）を発生させない

（2）　Baseセルの移動によりクリティカルパスが許容される長さを超えない

特に第(2)項はmodified FM法により性能が劣化しないことを保証する．許容されるクリティカルパス長として ceil (M/N) を用いた．ここで，Mはクリティカルパスの全長，Nは分割コンテクスト数である．

3.4. その他の機能

提案する時分割アルゴリズムにより，PELOCは複数の部分回路を生成する．それぞれの部分回路は個々に分割・配置・配線される．また，時通信もこの段階でTCMに割り当てられる．最後にPELOCは全てのコンテクストのコンフィギュレーションデータストリームを生成する．

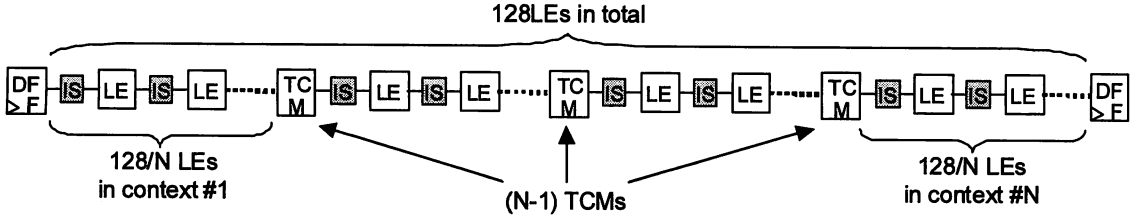

図13　遅延測定に用いたクリティカルパス

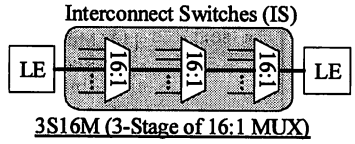

図14　プログラマブル配線スイッチのパラメータ命名規則

4. 実験

図 13 にマルチコンテクスト実行による遅延測定で用いたテスト回路のクリティカルパスを示す．全 128 個の 4 入力 LUT が縦列接続されている．N コンテクストに分割した場合，各コンテクストに配置される論理素子数は 128/N 個である．また，コンテクスト間の時通信用に N－1 個の TCM を使用する．隣り合う論理素子間にはプログラマブル配線スイッチが存在する．この配線スイッチの命名規則を図 14 に示す．

実験条件は以下の通りである．

- 分割コンテクスト数：1, 2, 4, 8, 16
- 配線スイッチの MUX のサイズ:8:1, 16:1, 32:1, 64:1
- 配線スイッチの MUX の接続段数=1, 2, ･･･, 8

図 15 に 8:1MUX を用いたときのマイクロサイクル遅延とユーザサイクル遅延をそれぞれ示す．ユーザサイクル遅延はマイクロサイクル遅延×コンテクスト分割数で表される．図 15(b)より，ユーザサイクル遅延は全コンテクストでほぼ 1 点に収束することがわかる．この結果から，フレキシブルプロセッサは，マルチコンテクスト実行した場合でも，シングルコンテクストとほぼ同じ速度で動作することを確認できた．

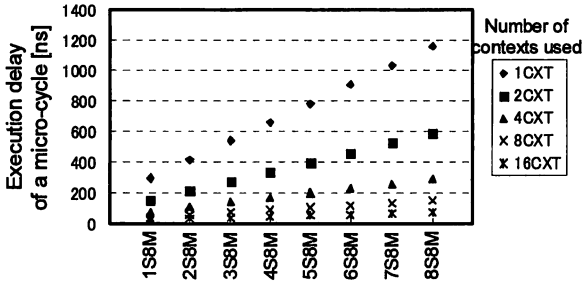

図15(a)　8:1MUXを用いた時のマイクロサイクル遅延

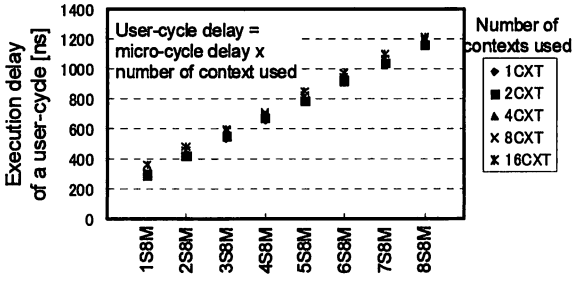

図15(b)　8:1MUXを用いた時のユーザサイクル遅延

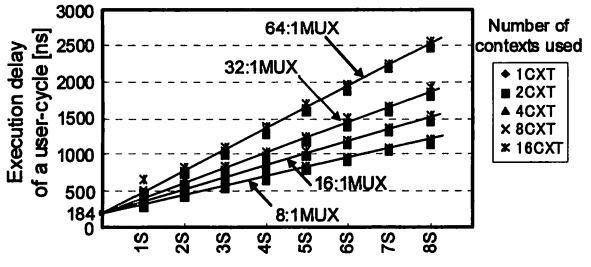

図16　全てのMUXを用いた時のユーザサイクル遅延

図 16 はすべての配線パラメータでユーザサイクル遅延を測定した結果である．図 15(b)同様，全コンテクストでほぼ 1 点に収束した．

また，この図から，論理素子のみの遅延（184nsec）に対するプログラマブル配線スイッチの遅延の割合が大きいことがわかる．マルチコンテクスト FPGA の時通信を積極的に利用することで，配線スイッチのサイズと段数を削減することが今後の課題である．

5. 結論

大規模な回路のハードウェアエミュレーションを目的として，90nm CMOS マルチコンテクスト FPGA『フレキシブルプロセッサ』を設計・試作した．測定の結果，フレキシブルプロセッサは，マルチコンテクスト実行した場合でも，シングルコンテクストとほぼ同じ速度で動作することを確認した．

謝　辞

本チップ試作は東京大学大規模集積システム設計教育研究センターを通し STARC, 富士通, Panasonic, NEC エレクトロニクス, Renesas テクノロジおよび東芝の協力で行われたものである．

文　献

[1] 西尾誠一他，“システム LSI 設計を支える検証技術，”デザインウェーブマガジン，設計ソリューション・ガイド 2003.

[2] 今井正治編著，“ASIC 技術の基礎と応用，”電子情報通信学会，p.191，1997.

[3] T. Ohkawa, et al., Digest of Technical Papers, Symposium on VLSI Circuits, pp.279-282, 2003.

[4] K. Kotani et al., in proc. of the IEEE A-SSCC 2005, pp. 329-332, 2005.

[5] Md. A. Khan, et al., in proc. of the IEEE A-SSCC 2006, pp. 275-278, 2006.

[6] N. Miyamoto et al., in proc. of the IEEE A-SSCC 2008, pp. 89-92, 2008.

[7] S. Trimberger, et al., in proc. of the IEEE FCCM 1997, pp.22-28, 1997.

[8] I. Kennedy, in Proc. of FPL, pp. 262-271, 2003.

[9] T. Fujii, et al., Digest of Technical Papers, ISSCC 1999, pp. 364-365, 1999.

[10] D. Chang, et al., IEEE Trans. on Computer, Vol.48, Issue 6, pp.565-578, June 1999.

[11] Md. A. Khan, “A Study on Time-multiplexing of Reconfigu-rable LSI”, Master’s Thesis, Tohoku University, Japan, 2006.

[12] C.M. Fudiccia, et al., in Proc. of the DAC, pp 175-181, 1982.