Vol.2014-SLDM-165 No.21
Vol.2014-EMB-32 No.21
2014/3/15

高位合成におけるマルチプレクサの遅延の削減手法

祖父江 亮哉† 原 祐子‡ 谷口 一徹† 冨山 宏之†

† 立命館大学 〒525-8577 滋賀県草津市野路東 1 丁目 1-1
‡ 東京工業大学大学院 / 独立行政法人科学技術振興機構 さきがけ

〒152-8552 東京都目黒区大岡山 2 丁目 12-1

E-mail: † ryoya.sobue@tomiyama-lab.org, {i-tanigu, ht}@fc.ritsumei.ac.jp
‡ hara@cad.ce.titech.ac.jp

現在まで，マルチプレクサの遅延を削減することを目的とした高位合成技術の研究が多々行われている．これらの研究は大きく 2 つに大別される．一方はデータパスのアロケーションとバインディングに関する研究であり，マルチプレクサの入力数を削減する，あるいは，マルチプレクサを削減することを狙っている．他方は制御回路に関する研究であり，マルチプレクサへの制御信号を生成する回路の遅延を削減することを狙っている。本研究では，両技術を併用することでマルチプレクサの遅延を更に削減し，より高いクロック周波数の実現を目指す．

Multiplexer Delay Minimization in High-level Synthesis

Ryoya SOBUE† Yuko HARA-AZUMI‡ Ittetsu TANIGUCHI† and Hiroyuki TOMIYAMA†

† Ritsumeikan University 1-1-1 Nojihigashi, Kusatsu-shi, Shiga, 525-8577 Japan
‡ Tokyo Institute of Technology, JST, PRESTO
2-12-1 Ookayama, Meguro, Tokyo, 152-8552 Japan

E-mail: †ryoya.sobue@tomiyama-lab.org, {i-tanigu, ht}@fc.ritsumei.ac.jp
‡ hara@cad.ce.titech.ac.jp

A number of research efforts have been made to minimize delays of multiplexers in high-level synthesis. The past studies are classified into two approaches. One approach tries to minimize input ports of the multiplexers during datapath synthesis, while the other one tries to minimize the delay to generate control signals for the multiplexers during controller synthesis. This paper proposes an integrated methodology which combines both of the two approaches in a single framework for further reduction of the multiplexer delay.

1. はじめに

近年のハードウェアの大規模化・複雑化に伴い，Register-Transfer (RT) レベルでのハードウェア設計が困難になっている．そのため，動作レベルの言語を RT レベルのハードウェア設計言語 (Hardware description language: HDL) に自動変換する技術である，高位合成[1]が注目されている．より高い抽象度で回路設計ができるため，設計コストの削減や，生産性の向上が期待されている．しかしながら，人手で設計したものと比較して未だ性能が劣る場合が多いのが現状である．

これまで，高位合成を用いてより高性能なハードウェアを設計するための研究が広く行われてきた[2][3][4][5]．特にクロック周波数の向上は，最重要な課題の 1 つである．従来の高位合成により設計されたハードウェアのクリティカルパスは，状態レジスタからデータレジスタまでのパス (図 1 参照) となることが多い．このパスを改善するために重要なのは，機能ユニット (Function Unit : FU) の前に挿入されたマルチプレクサに関わる遅延である．これを改善するため

Vol.2014-SLDM-165 No.21
Vol.2014-EMB-32 No.21
2014/3/15

に，大きく分けて 2 種類の研究が行われている．1 つは，[2][3]のようにデータパスのアロケーションやバインディングを変更することにより，マルチプレクサを削除，または入力数を削減する研究である．もう 1 つは，マルチプレクサの制御信号を計算する制御回路に着目した研究で，[4][5]などがある．これらの研究のほとんどは，どちらか一方しか行っておらず，回路全体を改善できていない．

本論文では，制御回路とデータパスの最適化の両面に着目し，マルチプレクサの遅延を改善する手法を提案する．データパスの合成では，FU の選択的共有[6]とマルチプレクサの置き換えを適用する．FU の選択的共有は，データパス内の FU の一部を共有，残りを非共有することにより，マルチプレクサを最小限に抑えることを狙いとしている．すべての FU を非共有することによる面積オーバーヘッドは非常に大きいため，規模の大きい FU のみを共有することで，マルチプレクサの削除と面積コストのバランスをとる[7]．マルチプレクサの置き換えでは，制御信号をワンホットエンコーディングしたマルチプレクサを使用することで，計算遅延を改善する．制御回路の最適化には，制御回路の RT レベルリタイミングを用いる．FU の前に挿入されたマルチプレクサの制御信号の計算を 1 つ前のサイクルで行い，これをレジスタに格納する．これにより，計算が完了した制御信号を必要なときに直接レジスタから読み出せるため，制御関数の遅延が削減される．評価実験では，クロック周波数と面積コストの評価を行い，制御回路とデータパスの両面に着目して最適化することの有効性を示す．

以下，本論文では，2 章で本研究で対象とするハードウェアについて述べ，3 章で提案手法で組み合わせる 3 手法について説明する．4 章で提案手法の評価を行い，最後に 5 章でまとめを述べる．

2. 対象とするハードウェアアーキテクチャ

本章では，本研究で対象とするハードウェアアーキテクチャについて述べる．

図 1 と図 2 に，従来の集中型の制御回路を用いた回路の簡略図を示す．長方形は，メモリやレジスタ等の記憶ユニット，楕円形が組み合わせ回路をそれぞれ表している．このアーキテクチャでは，制御回路内の制御関数で状態レジスタの値に応じた制御信号が計算される．これらの制御信号がデータパスの各ユニットに送信され，各状態におけるデータパスでの計算が制御される．また，制御回路では状態遷移も管理される．状態遷移には，状態レジスタのデータのみ使用する場合（無条件分岐）と，状態レジスタの値とデータパスでの演算結果が影響する場合(条件分岐)が存在する．

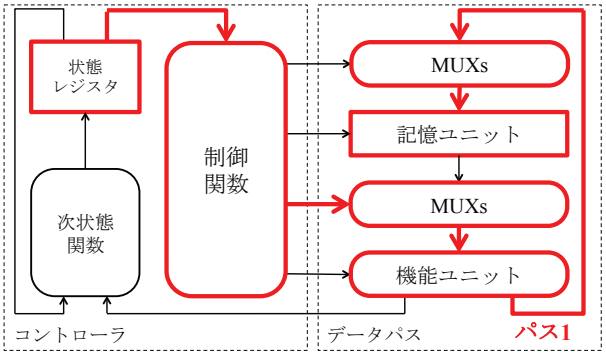


図 1：状態レジスタからデータレジスタまでのパス

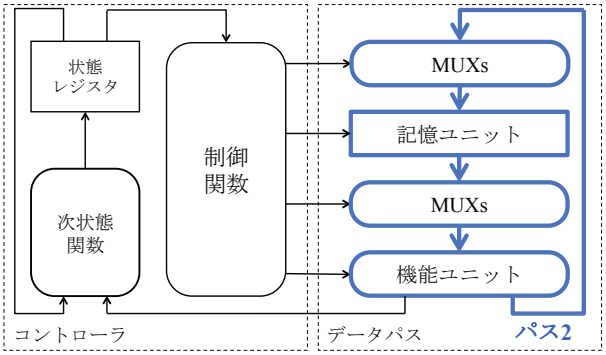


図 2：データレジスタからデータレジスタまでのパス

この回路におけるクリティカルパスは，一般的に 2 種類に大別される．状態レジスタからデータレジスタまでのパス（図 1：パス 1）と，データレジスタからデータレジスタまでのパス（図 2：パス 2）である．実際，制御関数の遅延は回路規模が大きくなるほど長くなるため，パス 1 がクリティカルパスになることが多い[4]．そこで本研究では，パス 1 を改善することに重点を置いた手法を提案する．

3. マルチプレクサの遅延の削減手法

本章では，開発した高位合成フレームワークを示し，データパスと制御回路のそれぞれに着目したマルチプレクサの遅延の改善手法について説明する．

3.1. 提案手法の合成フレームワーク

本研究の高位合成フレームワークを用いたフローを図 3 に示す．このフレームワークでは，高位合成エンジンとして Soliton 社の商業用高位合成ツール eXCite を使用している．灰色の長方形が高位合成の処理を示しており，3 つの設計手法を組み込んだ．まず，データパスの生成時にアロケーションで FU の選択的共有（3.2 節参照）を行なった後，データパス内のマルチプレクサを置き換える（3.3 節参照）．さらに，生成されたデータパスを入力として，制御回路の RT レベル

Vol.2014-SLDM-165 No.21
Vol.2014-EMB-32 No.21
2014/3/15

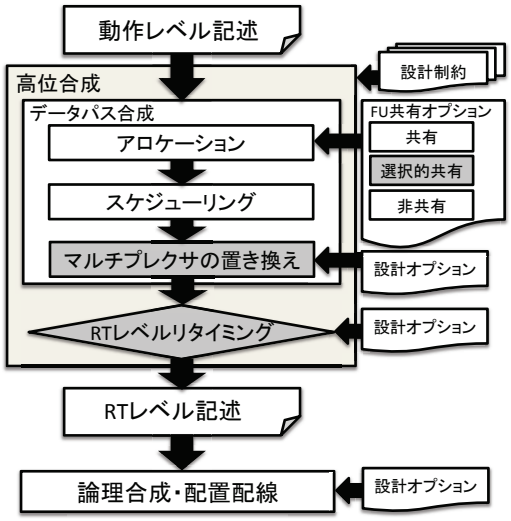


図 3：提案する合成フレームワークのフロー

リタイミング[5]を行う（3.4 節参照）．高位合成で生成した RTL 回路を入力として論理合成と配置配線を行い，回路のクロック周波数と面積コストについての評価する．また，論理合成時にはゲートレベルでのレジスタリタイミングを適用できる．

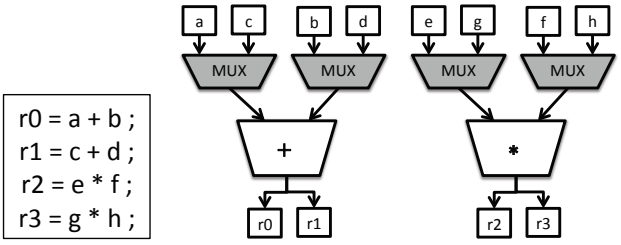
次節から，FU の選択的共有とマルチプレクサの構成，制御回路の RT レベルリタイミングについて解説する．

3.2. FU の選択的共有

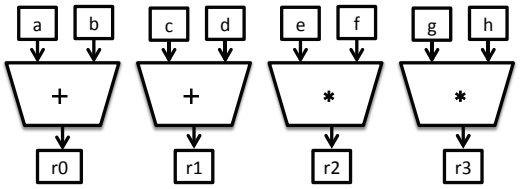
本節では，データパス内のマルチプレクサ数を削減する，FU の選択的共有について説明する．

最初に，データパス設計の例を図 4 に示す．図 4(a)で示した演算命令を，最小限の FU を用いたデータパスで実現する場合，図 4(b)で示す回路となる．2 つの加算演算と 2 つの乗算演算がそれぞれ 1 つの加算器，1 つの乗算器に割り当てられる．どちらも入力の前にマルチプレクサが挿入されており，演算に使用するデータが選択される．FU を共有することで面積コストを抑えられるが，マルチプレクサが挿入される分クロック周波数が低下する．対して，図 4(a)で示す各演算命令に 1 つずつ FU を割り当てる場合，図 4(c)で示す回路となる．FU を非共有することでマルチプレクサがなくなるため，クロック周波数は向上する．しかし，FU を複数使用するため面積コストは非常に大きくなる．

図 4 で示した例を踏まえて，データパスの最適化手法として FU の選択的共有を用いる．FU の非共有によるマルチプレクサの削除と，これによる面積オーバーヘッドのバランスをとる[6]．FU を非共有する場合，乗算器や除算器のように規模の大きな FU が面積コストに与える影響は特に大きい．そこで，FU の選択的共有では，乗算器と除算器のみを共有し，残りの FU を非共有する．図 4(a)を FU の選択的共有を用いて実現



(a) 演算命令の例　　(b) すべての FU を共有



(c) すべての FU を非共有

図 4：データパスの設計例

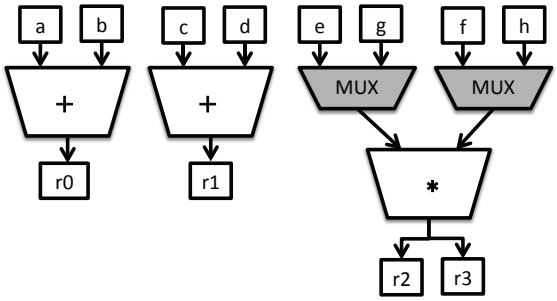


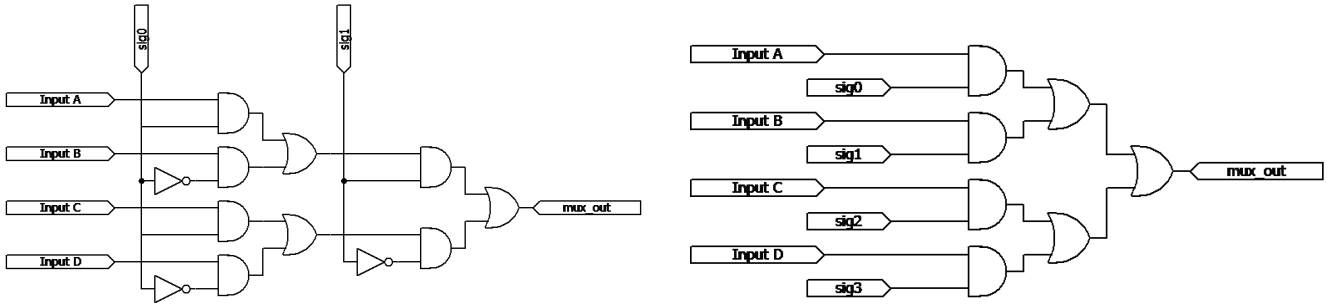
図 5： FU の選択的共有

する場合，図 5 のようになる．コストの大きい乗算器は共有，小さい加算器は非共有している．これにより，図 4(b)よりもマルチプレクサ数を減らし，かつ，図 4(c)よりも面積コストを小さく抑えている．

しかしながら，FU の選択的共有のみでは，クリティカルパスにマルチプレクサの遅延が残ってしまう場合がある．そこで次節では，マルチプレクサが残ったパスの遅延の改善手法について述べる．

3.3. マルチプレクサの設計手法

現在，マルチプレクサには様々な設計手法が用いられている．図 6 に，4 入力マルチプレクサの設計例を示す．図 6(a)は，2 入力のマルチプレクサがツリー状に配置されている設計で，多くの場合この手法が用いられる．制御信号のビット数が $\log_2 N$（N はマルチプレクサの入力数）で，AND 素子と OR 素子が $\log_2 N$ 段ずつ配置される．図 6(b)に，制御信号をワンホットエンコーディングしたマルチプレクサを示す．入力それぞれに制御信号が用意されているため，制御信号のビット数は N となり図 6(a)よりも多くなるが，AND 素子の段数が 1 段となり，計算遅延の改善が見込める．



(a) ツリー状に配置したマルチプレクサ　　(b) 制御信号をワンホットで設計したマルチプレクサ

図 6：マルチプレクサの設計手法の例

本研究では，図 3 のマルチプレクサの置き換えで，図 6(a)の構成から図 6(b)に変更する．

3.4. 制御回路の RT レベルリタイミング

本節で説明する制御回路の RT レベルリタイミング[5]は，状態レジスタからデータレジスタまでのパス（図 1：パス 1）を改善する手法である．図 7 に，RT レベルリタイミングを適用した回路の簡略図を示す．本手法では，FU の前に挿入されているマルチプレクサへの制御信号を，これらが必要となる 1 つ前のサイクルで計算し，新たに挿入したレジスタ（図 7：影付きレジスタ）に格納する．これをゲートレベルリタイミング[8]より高い抽象度（RT レベル）で，制御関数を前のサイクルに移動させることで，より積極的に各サイクルの遅延のバランスをとることができる．図 7 の影付き制御関数では，FU の前に挿入されたマルチプレクサの制御信号が計算され，白色で示す制御関数ではそれ以外のユニットの制御信号が計算される．RT レベルリタイミングを適用することにより，制御信号を直接レジスタから読み出せるため，図 1 で示したパス 1 は図 7 で示すパス 3 となり，クリティカルパスを改善できる．

これまで提案された RT レベルリタイミングでは，リタイミングに必要なレジスタ等による面積コストの増加を最小限に抑えるため，部分的に適用する手法が用いられている[5]．この手法は， FU の前に挿入されたマルチプレクサの中から，クリティカルパスに関わっているマルチプレクサのみを選択し，選択されたマルチプレクサの制御信号を計算する制御関数のみをリタイミングの対象とするアルゴリズムである．しかしこのアルゴリズムは，データパスの設計次第では，多くの制御回路をリタイミングの対象としなければならない場合がある．本論文で用いたフレームワークでは，FU の選択的共有を用いることで，FU の前に挿入されているマルチプレクサは，乗算器と除算器の前のみに限定される．これにより，RT レベルリタイミングでは，FU の前に挿入されたマルチプレクサ用の制御関数をすべて対象としても，リタイミングの対象は少なく抑えられる．さらに RT レベルリタイミングを適用する制御関数を選択するアルゴリズムも簡単化できる．

4. 評価実験

本章では，提案するマルチプレクサの遅延の削減手法の性能ついて，クロック周波数と面積コストに関する定量的評価を行う．

4.1. 実験環境

前章で紹介した提案手法を評価するために，ベンチマークとして，CHStone[9]から *ADPCM*, *AES_enc* (AES の暗号化), *DFDIV*, *DFMUL*, *GSM* を，PowerStone[10]から，*JPEG* を使用した．すべてのベンチマークで，乗算器を 1 つ以上使用している．高位合成ツールとして eXCite，論理合成・配置配線のために Xilinx 社の ISE14.7，ターゲットデバイスとして Virtex6 (XC6VLX75T, speed grade 6) を使用した．比較対象には 6 手法を用いた：最適化を行わない手法（**Baseline**），論理合成時にゲートレベルリタイミングのみを適用（**Gate**），制御回路の最適化（RT レベルリタイミング）のみを適用（**Cnt.opt**），データパスの最適化（FU の選択的共有と MUX の置き換え）のみを適用（**Data.opt**），Cnt.opt と Data.opt を併用（**Cnt&Data.opt**：提案手法），FU をすべて非共有（**Unshare**[1]）．Gate だけでなく，Cnt.opt，Data.opt，Cnt&Data.opt，Unshare においても同様にゲートレベルリタイミングを適用した．また，演算チェイニングは行わず，乗算器は 4 段のパイプライン乗算器を使用した．

[1] FU をすべて非共有する場合は FU の前にマルチプレクサが入らないため，これらのマルチプレクサを対象とした最適化である Cnt.opt および Data.opt は併用しない．

Vol.2014-SLDM-165 No.21
Vol.2014-EMB-32 No.21
2014/3/15

表 1：各 FU の共有手法における必要演算器数

	DFMUL (28)			DFDIV (58)			JPEG (129)			ADPCM (234)			AES_enc (244)			GSM (267)		
	共有	選択	非共有	共有	選択	非共有	共有	選択	非共有	共有	選択	非共有	共有	選択	非共有	共有	選択	非共有
Add / Sub	6	26	26	7	44	44	16	151	151	11	156	156	10	204	204	18	210	210
Mul	1	1	4	4	4	8	8	8	33	4	4	65	1	1	7	1	1	53
Div	0	0	0	1	1	1	0	0	0	1	1	2	1	1	11	0	0	0
Cmp	9	25	25	11	36	36	3	40	40	9	113	113	5	23	23	4	93	93
Shift	2	4	4	2	4	4	1	4	4	1	4	4	0	0	0	2	3	3
2入力MUX	94	8	0	98	20	0	127	29	0	138	33	0	54	13	0	141	9	0

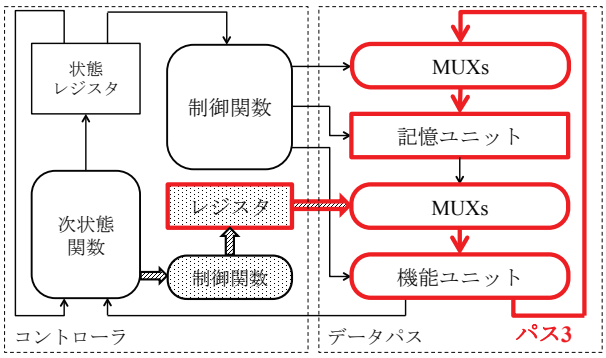


図 7：RT レベルリタイミング適用後のアーキテクチャ

表 1 に，FU の共有手法を変更した場合の各ベンチマークの演算器数を示す．ベンチマーク名の右の括弧内の数値は，状態数を表している．2 入力マルチプレクサは，FU の前に挿入された 2 入力マルチプレクサ数を示している．選択的共有や非共有では必要な FU 数が大幅に増加する．対してマルチプレクサは，FU を非共有することで減少し，すべて非共有する場合は 0 となる．特に演算数が多いベンチマークでは，この増減は顕著に現れる．

4.2 節でクロック周波数，4.3 節では面積コストについて提案手法が与える影響を評価する．

4.2. 提案手法がクロック周波数に与える影響の評価

本節では，提案手法がクロック周波数に与える影響について評価する．本実験で示すグラフでは，各手法の結果は Baseline を 1 として正規化し，ベンチマークを状態数順に左から右に昇順に並べている．

図 8 に本実験のクロック周波数の変化を示す．各手法がそれぞれクロック周波数に与える影響について議論する．Gate では，*AES_enc* と *GSM* で高い改善率を示している．これらのベンチマークでは，パス 1（図 1 参照）やパス 2（図 2 参照）以外のパスが長くなっている．ゲートレベルリタイミングを併用することで，Cnt.opt や Data.opt では改善できないパスを改善でき，更に高いクロック周波数を実現できる．Cnt.opt による改善率が高いのは，大規模ベンチマーク(*AES_enc*, *GSM*)である．回路が大規模化・複雑化するほど制御関数の遅延は長くなるため，制御関数のリタイミングによる効果は大きくなる． Data.opt は，パス 1 とパス 2 の両方を改善できるため，すべてのベンチマークで改善効果を示した．特に小規模ベンチマークでは，制御関数の遅延が短く，クリティカルパスにおけるデータパスの遅延が占める割合が高いため，Data.opt がより有効である．これらを併用した Cnt&Data.opt では，ほとんどのベンチマークにおいて最も高いクロック周波数を示しており，平均 52%，最大 92%の改善に成功した．全体平均では，Unshare の 65%に近い改善率を実現した．この結果から，制御回路とデータパス両面を考慮した最適化を行うことは有効であると言える．

4.3. 提案手法が面積コストに与える影響の評価

図 9 に，提案手法による面積コスト（Slice 数[2]）の結果を示す．最も高いクロック周波数を示した Unshare では，乗算器や除算器の必要数が大幅に増加しているベンチマーク（表 1 参照）において，面積コストの増加が顕著に現れている．最大で 4.6 倍となっており，面積コストを考慮する場合，FU をすべて非共有することは現実的ではない．提案手法である Cnt&Data.opt では，平均 18%増加した．Gate や Cnt.opt のみではほとんど増加は見られないが，FU の選択的共有を行った Data.opt では一部で大きな増加が見られる．特に *AES_enc* の Data.opt と Cnt&Data.opt では増加が顕著である．表 1 を見ると，選択的共有することで，加算器と減算器の合計値が従来の共有と比較して 20 倍と大きく増加している．しかし，選択的共有で一部の FU を非共有した際のマルチプレクサの減少は，1/4 程度となっている．このため，*AES_enc* の選択的共有において面積コストが大きく増加したと考えられる．対して FU の増加よりもマルチプレクサの減少が大きい *DFMUL* では，提案手法の面積コストが減少した．つまり，FU の選択的共有で共有する FU 量を調整するアルゴリズムを用いることにより，提案手法を適用した場合の面積コストを更に抑制できる．

[2] DSP を Slice 数に換算したコストも含む．

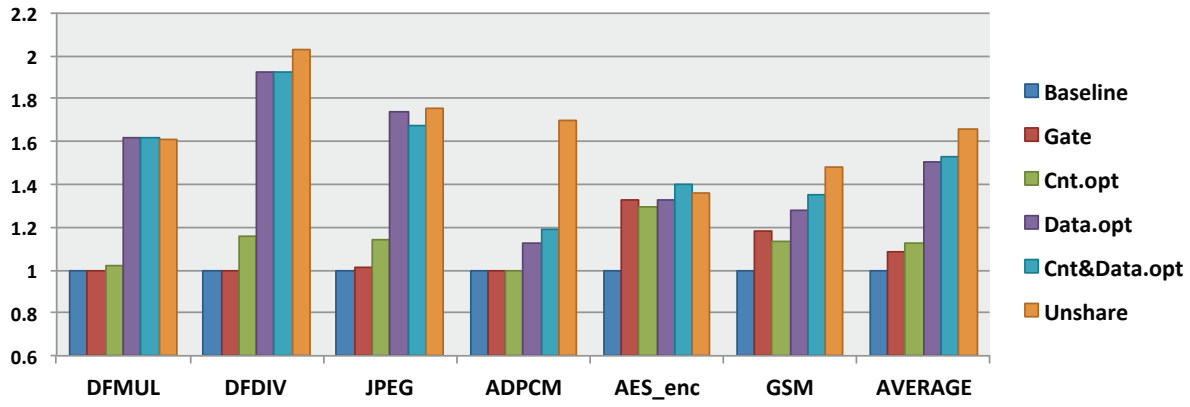


図 8：クロック周波数の比較

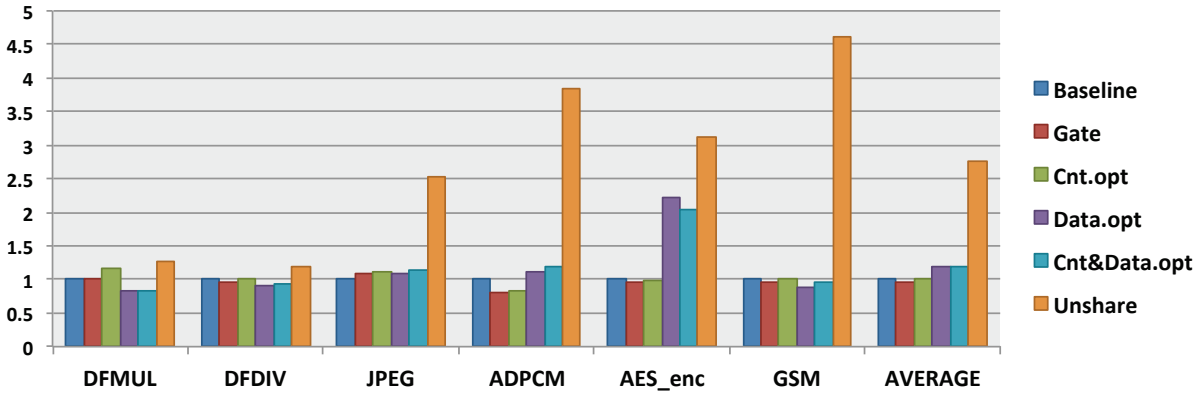


図 9：面積コストの比較

5. おわりに

これまで，マルチプレクサに関わる遅延を改善する高位合成の研究が多く行われてきた．これらは，データパス合成中に，マルチプレクサの入力数の削減や削除をする手法と，制御回路合成中にマルチプレクサの制御信号を計算する遅延を改善する手法の 2 種類ある．本研究では，前者の研究に分類される，FU の選択的共有とマルチプレクサの構成変更に加えて，後者の研究となる，制御回路の RT レベルリタイミングを併用することで，マルチプレクサの遅延を削減する手法を提案した．評価実験では，クロック周波数を最大 92%，平均 52%の改善を実現した．

謝　辞

本研究の一部は，科研費(課題番号 23300019)の助成による．

文　献

[1] D. D. Gajski, N. D. Dutt, A. C-H Wu and S. Y-L Lin, *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publisher, 1992.

[2] Y. Hara-Azumi and H. Tomiyama, “Clock -Constrained Simultaneous Allocation and Binding for Multiplexer Optimization in High-Level Synthesis,” *ASP-DAC,* 2012.

[3] T. Kim, X. Liu, “Compatibility Path Based Binding Algorithm for Interconnect Reduction in High Level Synthesis,” *ICCAD,* Nov.2007.

[4] S. Lee and K. Choi, “High-Level Synthesis with Distributed Controller for Fast Timing Closure”, *ICCAD*, 2011.

[5] R. Sobue, Y. Hara-Azumi and H. Tomiyama, “Partial Controller Retiming in High–level Synthesis,” *ESLsyn*, 2013.

[6] Y. Hara-Azumi, T. Matsuba, H. Tomiyama, S. Honda, H. Takada, “Selective Resource Sharing with RT-Level Retiming for Clock Enhancement in High-Level Synthesis,” *HPCC-ICESS*, 2012.

[7] Y. Hara-Azumi, T. Matsuba, H. Tomiyama, S. Honda and H. Takada, "Impact of Resource Sharing and Register Retiming on Area and Performance of FPGA-based Designs," *IPSJ Transactions on System LSI Design Methodology*, vol.7, 2014.

[8] C. E. Leiserson and J. B.Saxe, “Retiming Synchronous Circuitry,” *Algorithmica*, vol.6, 1991.

[9] Y. Hara, H. Tomiyama, S. Honda and H. Takada, “Proposal and Quantitative Analysis of the CHStone Benchmark Program Suite for Practical C-based High-level Synthesis,” *Journal of Information Processing*, vol.17, 2009.

[10] J. Scott, L. Lee, J. Arends, and B. Moyer, “Designing the low-power MCORE architecture,” *Power Driven Microarchitecture Workshop*, 1998.