Vol.2015-SLDM-170 No.48
Vol.2015-EMB-36 No.48
2015/3/7

製造ばらつきと配線遅延を同時に考慮した低レイテンシ指向のマルチシナリオ高位合成の評価

井川 昂輝[†1,a)]　阿部 晋矢[†2,b)]　柳澤 政生[†3]　戸川 望[†2,c)]

概要：増大を続ける製造ばらつきや配線遅延への解決策として，HDR アーキテクチャを対象としたマルチシナリオ高位合成手法を提案している．チップ全体をハドルと呼ばれる配線遅延の影響のない範囲に分割することで高位合成段階における適切な配線遅延の予測が可能となる．加えて製造ばらつきによる演算器の遅延ばらつきをシナリオとして扱う．演算器の遅延が Typical ケースの場合の Typical シナリオ，Worst ケースの場合の Worst シナリオを同時に 1 つのチップ上に高位合成し，製造されたチップの特性に応じてシナリオを切り替えることで高い歩留りと高い性能の両立が可能となる．提案手法は各シナリオの動作コントロールステップ数を最小化し，ハドル間データ通信やモジュール間結線をシナリオ間で揃える共通化と呼ばれる処理により全体の面積を削減する．本稿では，計算機実験により各動作条件におけるレイテンシを従来手法と比較し評価する．また，演算器の遅延分布から Typical シナリオで動作可能な確率を算出し，レイテンシの期待値も評価する．提案手法は従来手法と比較し，レイテンシの期待値を最大 35%削減できることを確認した．

1. はじめに

製造ばらつきや配線遅延が LSI 設計へ与える影響は増大を続けているため，これらを設計段階において適切に扱う必要がある．また，LSI 設計の生産性を向上させる設計技術として，高位合成の重要性も増加している．したがって，本稿では製造ばらつきと配線遅延を同時に扱う高位合成手法に注目する．

製造時のさまざまな要因により，製造された LSI はばらつきを持ち，遅延や消費電力はチップごとに異なる．したがって，設計時に製造ばらつきの影響を考慮しない場合，歩留りの低下が問題となる．従来はこの問題に対し，性能を悲観的に見積もる Worst ケース設計により対処してきた．しかし，製造されたチップの動作速度の低下や電力消費の増大は避けられず，Worst ケース設計に代わる高位設計手法が提案されている．[1, 2] では，製造ばらつきと配線遅延を同時に考慮しながら Performance Yield の最大化を行う手法が提案されている．しかし，合成された回路の性能は Typical ケース設計よりも劣る可能性がある．[3] では Stallable-FSM Architecture に基づいた，タイミングエラー耐性を持ちレイテンシの期待値が最大となるような回路を得る ILP ベースのスケジューリング手法が提案されている．Stallable-FSM Architecture の下では，各演算でタイミングエラーが起きた場合，FSM をストールさせることでタイミングエラーを回復できる．しかし，[3] は配線遅延が考慮されている．

製造ばらつきへの有力な解決策の 1 つとして，Post-Silicon Tuning（製造後の回路機能/性能の調整）がある．Post-Silicon Tuning は，製造ばらつき以外にも製造後の機能変更 [4] や Multimode 機能 [5] にも適用され得る概念である．[6] は，ばらつきに対する適応的基板バイアス制御を用いた性能補償を設計段階において考慮した高位合成手法である．しかし，配線遅延を考慮していない．[7] は配線遅延のばらつきがある場合，配線遅延のばらつきがない場合の動作を制御信号により切り替え可能な高位合成手法である．しかし，配線遅延の製造ばらつきのみを扱っており，演算器の遅延の製造ばらつきは考慮されていない．

一方，高位合成段階で配線遅延を適切に扱うため，レジスタ分散型アーキテクチャが提案されている．レジスタ分散型アーキテクチャはレジスタを各演算器に分散させることで，それらの間の配線遅延の影響を軽減する．その 1 つとして，HDR アーキテクチャ [8] が提案されている．HDR アーキテクチャはチップをハドルと呼ばれる区画に分割する．ハドルの大きさを高位合成段階で予測することにより，配線遅延の見積もりが可能となる．

以上の議論より，[9] で製造ばらつきと配線遅延を同時に考慮した HDR アーキテクチャを対象とするマルチシナリオ高位合成を提案した．製造ばらつきによる演算器の遅延ばらつきをシナリオとして扱うことで，それらに応じた複数の動作が可能な LSI を合成できる．これにより，製造ばらつき耐性を持ちながら高い性能を実現できる．

本稿では，最初に HDR アーキテクチャを対象としたマルチシナリオ高位合成手法を紹介する．提案された手法は各シナリオの動作コントロールステップ数（以下動作 CS 数）を最小化し，ハドル間データ通信やモジュール間結線をシナリオ間で揃える共通化と呼ばれる処理により全体の面積を削減する．本稿では，計算機実験により各動作条件におけるレイテンシを従来手法と比較し評価する．また，

†1 現在，早稲田大学基幹理工学部情報理工学科
Presently with Dept. of Computer Science and Engineering, Waseda University
†2 現在，早稲田大学大学院基幹理工学研究科情報理工学専攻
Presently with Dept. of Computer Science and Engineering, Waseda University
†3 現在，早稲田大学大学院基幹理工学研究科電子光システム学専攻
Presently with Dept. of Electronic and Photonic Systems, Waseda University
a) koki.igawa@togawa.cs.waseda.ac.jp
b) shinya.abe@togawa.cs.waseda.ac.jp
c) togawa@togawa.cs.waseda.ac.jp

Vol.2015-SLDM-170 No.48
Vol.2015-EMB-36 No.48
2015/3/7

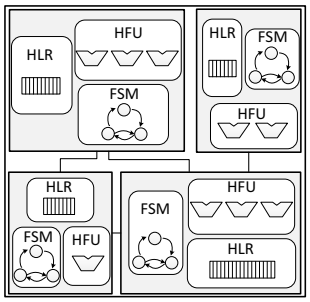


図 1: HDR アーキテクチャ [8].

演算器の遅延分布から Typical シナリオで動作可能な確率を算出し，レイテンシの期待値も評価する．提案手法は従来手法と比較し，レイテンシの期待値を最大 35%削減できることを確認した．

2. 問題の定式化

2.1 HDR アーキテクチャ

レジスタ分散型アーキテクチャは，レジスタを各演算器に分散させ，それらの間の配線遅延の影響を軽減する．レジスタ分散型アーキテクチャの 1 つである HDR（huddle-based distributed-register）アーキテクチャでは，チップをクロック周期によって決定されるハドルと呼ばれる区画に分割し，ハドル内に演算器やレジスタ，コントローラなどのモジュールを詰め込む．また，ハドルは任意の矩形をとるため，小さな面積オーバーヘッドで回路を合成できる．ハドルの大きさを高位合成の段階で予測することにより，配線遅延の見積もりを可能とする．HDR アーキテクチャの概要を図 1 に示す．ハドルは以下の要素で構成される．

Huddled Functional Unit (HFU) ハドルに集められた演算器の集合．

Huddled Local Register (HLR) ハドル専用のローカルレジスタとマルチプレクサの集合．

Finite State Machine (FSM) 同一ハドル内の HFU と HLR を制御するコントローラ．

2.2 シナリオとマルチシナリオ高位合成

製造ばらつきによる演算器の遅延ばらつきを高位合成段階で扱うため，シナリオと呼ばれる概念が提案されており，次のように定義される．

定義 1. CDFG と，演算器の遅延などその動作条件の組をシナリオと呼ぶ．

複数のシナリオのスケジューリング，バインディング結果を同時に LSI 上に高位合成することをマルチシナリオ高位合成と呼ぶ．製造ばらつきによる演算器の遅延ばらつきをシナリオとして想定し，マルチシナリオ高位合成を行う．製造されたチップの特性に応じた複数の動作が可能な LSI を合成できるため，製造ばらつき耐性を持ちながら高い性能を実現できる．本稿では，演算器の遅延が Typical ケースの場合の Typical シナリオ，演算器の遅延が Worst ケースの場合の Worst シナリオを想定する．両シナリオの CDFG は同一とする．

マルチシナリオ高位合成の入力として，シナリオ集合 $Sc = \{typ, wst\}$，演算器集合 $F = \{f_1, ..., f_p\}$，クロック周期 T_{clk} を与える．シナリオ $x_s \in Sc$ は CDFG $G = (V, E)$ とその動作条件の組で構成される．p 個の演算器を q 個（$q \leq p$）のハドル集合 $H = \{h_1, ...h_q\}$ に割り当てる．演算器 f_i を割り当てるハドルを $Hud(f_i)$ と表す．シナリオ $x_s \in Sc$ における演算器 f_i の実行時間を $D_f^{x_s}(f_i)$ として，実行サイクル数を $S_f^{x_s}(f_i) = \lceil (D_f^{x_s}(f_i) + D_{reg})/T_{clk} \rceil$ と計算する．D_{reg} はレジスタの遅延を表す．シナリオ x_s における演算器 f_i のスラック時間を以下の式で計算する．

$$Slack^{x_s}(f_i) = T_{clk} \times S_f^{x_s}(f_i) - D_f^{x_s}(f_i). \quad (1)$$

ハドル $h_j = Hud(f_i)$ の幅 $W(h_j)$，高さ $H(h_j)$ は，以下の式で表されるハドルサイズ制約を満たすように決定する．

$$2 \cdot D_w(W(h_j) + H(h_j)) + D_{reg} \leq \min_{x_s \in Sc} \left\{ \min_{f_i \in F(h_j)} \{Slack^{x_s}(f_i)\} \right\}. \quad (2)$$

$D_w(x)$ は距離 x における配線遅延を表す．シナリオ x_s において演算器 f_i からハドル h_k の HLR へデータを転送する際必要なクロック数を $DT^{x_s}(f_i, h_k)$ と表す．

以上の定義より，製造ばらつきと配線遅延を同時に考慮した低レイテンシ指向のマルチシナリオ高位合成問題を次のように定義する．

定義 2. 製造ばらつきと配線遅延を同時に考慮した低レイテンシ指向のマルチシナリオ高位合成問題とは，シナリオ集合，演算器の集合，クロック周期制約が与えられたとき，各シナリオの動作 CS 数を最小化し，動作 CS 数が同じ場合には面積を最小化するようにスケジューリングおよびバインディングを行い，各演算器をハドルに割り当てて RTL 記述およびハドルフロアプランを出力することである．

3. HDR アーキテクチャを対象としたマルチシナリオ高位合成手法

フロアプランを考慮した高位合成の主要な工程には，スケジューリング，バインディング，コントローラ合成，フロアプランがある．スケジューリング，バインディング，コントローラ合成結果はフロアプランに影響を及ぼし，フロアプラン結果はスケジューリング，バインディング，コントローラ合成に影響を及ぼすため，これらの工程の最適な順序関係は合成前に確定できない．したがって，[8,10,11] と同様に，スケジューリング，バインディング，コントローラ合成，フロアプランについて反復フローを採用する．

マルチシナリオ高位合成では，各シナリオごとにスケジューリング結果，バインディング結果が必要となる．最も簡単にこれを実現するには，各シナリオのスケジューリング，バインディングをそれぞれ独立に実行し，データパスとシナリオを制御する FSM を構成することである．しかしながら，生成される結線や制御の関連性は考慮されないため，マルチプレクサやコントローラの面積の増大が問題となる．そこで本節では，シナリオ間の共通化に注目し，マルチシナリオスケジューリング/FU バインディング，マルチシナリオレジスタバインディングの工程に共通化と呼ばれる処理を導入することで全体の面積を削減する．ここで共通化とは，各シナリオのスケジューリング/FU バインディング結果のハドル間データ通信，レジスタバインディング結果の演算器/レジスタ（モジュール）間結線を揃えることである．

以上の議論をもとに図 2 に合成アルゴリズムを提案する．提案アルゴリズムは初期処理，反復処理，調整処理で構成される．初期処理では，演算器と同じ数のハドルを用意し，各ハドルに 1 つの演算器を割り当て，配置情報の初期解を生成する．反復処理では仮想面積見積もりにより，効率的にタイミングを満たす解を得る．調整処理では反復処理の結果から，正確な面積見積もりを決定する．マルチシナリオコントローラ合成では，各ハドルにシナリオの制

Vol.2015-SLDM-170 No.48
Vol.2015-EMB-36 No.48
2015/3/7

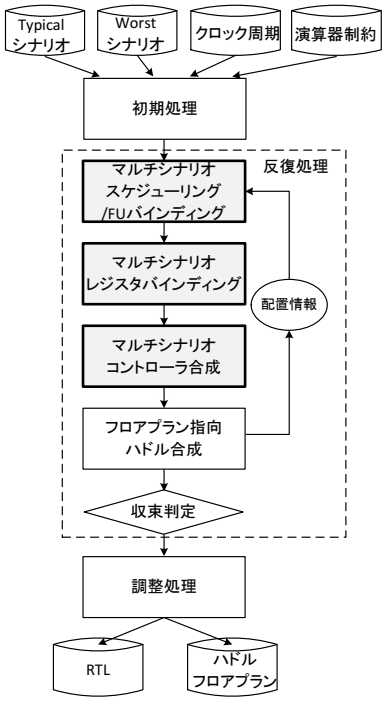


図 2: HDR アーキテクチャを対象としたマルチシナリオ高位合成アルゴリズム．

御と切り替えが可能なコントローラを合成する．初期処理，仮想面積見積もり，フロアプラン指向ハドル合成，調整処理は，[11] の手法を適用する．以下では，マルチシナリオスケジューリング/FU バインディング，マルチシナリオレジスタバインディングを示す．

3.1 マルチシナリオスケジューリング/FU バインディング

マルチシナリオスケジューリング/FU バインディングでは，シナリオ集合，クロック周期制約，DT^{typ}，DT^{wst}，ハドルの構成，配置情報を入力とし，各シナリオの動作 CS 数の最小化を目的としたスケジューリング/FU バインディングを行う．同時に，動作 CS 数が増大しない範囲内で，ハドル間データ通信共通化処理により両シナリオのハドル間データ通信を共通化する．あるシナリオのみに存在するハドル間データ通信が存在する場合，そのシナリオのみに存在するレジスタ間データ通信が生じるため，結線数やマルチプレクサ，コントローラの面積の増大を招く．両シナリオのハドル間データ通信を共通化することでこれを防ぐ．

3.1.1 共通度

ある反復においてハドルが q 個あり，Typical シナリオのスケジューリング/FU バインディングの結果が得られたとする．ハドル h_i からハドル h_j へのデータ通信回数 $HT^{typ}(h_i, h_j)$ から，これを q 行 q 列に並べることで，ハドル間データ通信回数テーブル HT^{typ} を得る．同様に Worst シナリオについて HT^{wst} を得ることができる．また，差分ハドル間データ通信回数 $HT^{typ}_{diff}(h_i, h_j)$ を，

$$HT^{typ}_{diff}(h_i, h_j) = \begin{cases} HT^{typ}(h_j, h_j) & \text{if } HT^{typ}(h_i, h_j) > 0 \quad \text{and} \\ & \quad HT^{wst}(h_i, h_j) = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

とし，これを q 行 q 列に並べることで，差分ハドル間データ通信回数テーブル HT^{typ}_{diff} を得ることができる．同様に Worst シナリオについて HT^{wst}_{diff} を得ることができる．$EQ(h_i, h_j)$ はハドル h_i からハドル h_j へのデータ通信の存在有無が Typical シナリオ，Worst シナリオで共通の場合は 1，そうでない場合は 0 の値をとるため，以下の式で表せる．

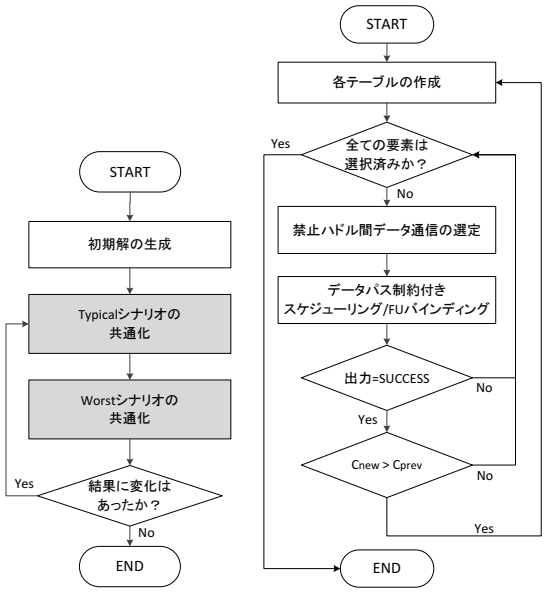
$$EQ(h_i, h_j) = \begin{cases} 1 & \text{if } HT^{typ}(h_i, h_j) = 0 \text{ and } HT^{wst}(h_i, h_j) = 0 \text{ or} \\ & \quad HT^{typ}(h_i, h_j) > 0 \text{ and } HT^{wst}(h_i, h_j) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

これを q 行 q 列に並べることで，ハドル間データ通信共通テーブルを EQ を得る．

このとき両シナリオのスケジューリング/FU バインディングの結果の共通度 C を，次式で求める．

$$C = \sum_{h_i \in H} \sum_{h_j \in H} EQ(h_i, h_j). \tag{5}$$

共通度 C が高ければ，各シナリオのスケジューリング結果/FU バインディング結果のハドル間データ通信が揃っていることになり，一方のシナリオのみに存在するレジスタ間のデータ通信が少なくなるため，マルチプレクサおよびコントローラの面積が小さくなり，全体の小面積化が期待できる．



(a) 全体フロー．　(b) 共通化処理．

図 3: ハドル間データ通信共通化アルゴリズム．

3.2 ハドル間データ通信共通化アルゴリズム

ハドル間データ通信共通化アルゴリズムを図 3 に示す．

図 3(a) の**初期解の生成**では，Typical シナリオ，Worst シナリオのスケジューリング/FU バインディング [10] をそれぞれ独立に行う．その後 Typical シナリオに対し，図 3(b) の共通化処理を実行する．この際 Worst シナリオのスケジューリング/FU バインディング結果は固定される．最初に，各テーブル HT^{typ}，HT^{wst}，HT^{typ}_{diff}，EQ を作成する．その後，HT^{typ}_{diff} の非零の未選択の要素がある場合は**禁止ハドル間データ通信の選定**へ，無い場合は共通化処理を終了する．**禁止ハドル間データ通信の選定**では，HT^{typ}_{diff} から最小の要素を，禁止するハドル間データ通信（$h_u \to h_d$）として選択し，この時点での共通度 C_{prev} を計算する．そ

の後，**データパス制約付きスケジューリング/FU バインディング**を実行する．実行結果の出力が $SUCCESS$ ならば，共通度 C_{new} を計算する．$C_{new} > C_{prev}$ ならばこの結果を採用し，**各テーブルの作成**で各テーブルを更新する．同様の操作を Worst シナリオについても行う．その後，1 度でもどちらかのシナリオのスケジューリング/FU バインディング結果に変化が生じたら，**Typical シナリオの共通化**へ．そうでないならばアルゴリズムを終了する．

3.3 マルチシナリオレジスタバインディング

マルチシナリオレジスタバインディングでは，各シナリオのスケジューリング/FU バインディング結果を入力とし，変数にレジスタを割り当てる．同時に，本工程において [5] を参考にシナリオ間のモジュール間結線の共通化を行うことで，結線数の増大によるマルチプレクサ数の増大を防ぐ．モジュール間結線共通化アルゴリズムでは，最小数のレジスタを保証した上で，生成され得るマルチプレクサ数の削減する．提案アルゴリズムは，アロケーション処理，バインディング処理で構成される．簡単のため，各レジスタはすべて同じビット幅を持つとする．以下，アロケーション処理，バインディング処理を示す

3.3.1 アロケーション処理

アロケーション処理では，各ハドルでそれぞれのシナリオが必要とする変数を抽出し，各ハドルで必要とする最小数のレジスタをアロケーションする．まず，Typical シナリオ，Worst シナリオにおいて，各ハドル $h_j \in H$ で保持する変数を抽出し，それらの集合を $HV^{typ}(h_j)$，$HV^{wst}(h_j)$ とおく．次に，各ハドル $h_j \in H$ に対し，Typical シナリオ，Worst シナリオにおいて必要とするレジスタ数をレフトエッジアルゴリズムにより算出し，それぞれ $NeedReg^{typ}(h_j)$，$NeedReg^{wst}(h_j)$ とする．$NeedReg^{typ}(h_j)$ と $NeedReg^{wst}(h_j)$ の内，大きい方の値の個数のレジスタを，ハドル h_j のレジスタとしてアロケーションする．

3.3.2 バインディング処理

バインディング処理では，Typical シナリオ，Worst シナリオにおいて，各ハドルで保持する変数にレジスタを割り当てる．この際マルチプレクサ数の増大を防ぐため，各シナリオのレジスタバインディングの結果生成される結線を可能な限り等しくする必要がある．提案手法では，一方のシナリオで生成された結線を再利用するという方針の基でこれを実現する．そのため，結線の再利用という方針に基づき，変数 v_k に対してレジスタ r_t を割り当てる場合のコスト関数 $rc(v_k, r_t)$ を導入する．$nc_{fu}(v_k, r_t)$ は変数 v_k に対してレジスタ r_t を割り当てた場合に，演算器-レジスタ間に生成される結線の個数とする．$nc_{reg}(v_k, r_t)$ は変数 v_k に対してレジスタ r_t を割り当てた場合に，レジスタ-レジスタ間に生成される結線の個数とする．このとき，$rc(v_k, r_t)$ は次式で定義される．

$$rc(v_k, r_t) = \lambda_1 \times nc_{fu}(v_k, r_t) + \lambda_2 \times nc_{reg}(vk, r_t). \quad (6)$$

λ_1，λ_2 は任意定数であり，演算器-レジスタの結線と，レジスタ-レジスタ間の結線の重みを区別するために設定している．再利用可能な生成済みの演算器-レジスタ間の結線とレジスタ-レジスタ間の結線数が増大するほど，$nc_{fu}(v_k, r_t)$ と $nc_{reg}(v_k, r_t)$ の値は小さくなるため，$rc(v_k, r_t)$ も小さくなる．

バインディング処理のアルゴリズムを図 4 に示す．

Step 1: 各ハドル $h_j \in H$ について，Step 2–Step 3 を繰り返す．
Step 2: $NeedReg(h_j)^{x_s}$ が大きいシナリオを対象に (1)，(2) を実行する．
(1) $HV^{x_s}(h_j)$ の要素を開始コントロールステップが早い順，同じ場合には終了コントロールステップが早い順にソート．
(2) $HV^{x_s}(h_j) \neq \phi$ である限り (a)–(c) を繰り返す．
(a) ソートされた順に変数 $v_k \in HV^{x_s}(h_j)$ を選択する．
(b) ハドル h_j のバインディング可能なレジスタ r_j についてコスト $rc(v_k, r_j)$ を計算する．
(c) コストが最小のレジスタに v_k をバインディングする．v_k を $HV^{x_s}(h_j)$ から消去する．
Step 3: もう一方のシナリオを対象に Step 2 を実行する．

図 4: モジュール間結線共通化アルゴリズムのバインディング処理．

表 1: 演算器情報

	面積 [μm^2]	遅延 (Typical) [ns]	遅延 (Worst) [ns]
加算器	386	1.22	2.135
減算器	417	1.27	2.222
乗算器	2161	2.7	4.725
除算器	6066	10.21	17.868
比較器	116	0.83	1.453
シフタ	294	0.89	1.558
AND	68	0.66	1.155
レジスタ	309	0.45	0.45
Razor	821	0.80	0.80
マルチプレクサ	288	0.17	0.17

4. 計算機実験結果

4.1 実験環境

提案手法を C++言語を用いて計算機上に実装した．計算機実験環境は，CPU が AMD Quad-Core Opteron 2360 SE2.5Ghz×2，メモリ容量が 16GB である．評価対象のアプリケーションとして，DCT（ノード数 48），7 次 FIR フィルタ（ノード数 75），EWF3（ノード数 102），JACOBI（ノード数 48），PARKER（ノード数 22，条件分岐あり），COPY（ノード数 378，条件分岐あり）[8,11] を使用した．入力演算器情報を表 1 に示す．表 1 の一部は [8] で使用されているものを用いた．演算器のビット幅は 16bit，電圧は 1.0V，クロック周期は 2.2ns とする．演算器 f_i の遅延分布はガウス分布を仮定し，Typical，Worst ケースの遅延を，$D_f^{typ}(f_i) = \mu_i$，$D_f^{wst}(f_i) = \mu_i + 3\sigma_i$ とする [2]．また，演算器 f_i に対し $\sigma_i/\mu_i = 0.25$ を与える．コントローラの面積は Synopsys 社の Design Compiler により論理合成して算出した．配線遅延は配線長の 2 乗に比例すると仮定し，250μm あたり 1ns と仮定する [8,11]．動作条件として，*Typical*，*Worst* の場合を考え，それぞれすべての演算器の遅延が Typical ケースである場合，Worst ケースである場合とする．それぞれにおける動作 CS 数を CS_{typ}，CS_{wst} と表す．

以下の手法を比較する．

HDR-typ 演算器の遅延に Typical ケースを想定した，HDR アーキテクチャを対象とした高位合成手法（Typical ケース設計）[8]．

HDR-typ-recover HDR-typ に [3] のようにエラー修正を加えた手法であり，タイミングエラー検出機能の実現のために Razor [12] を HDR アーキテクチャのレジスタに対して使用し合成した．この場合，HDR-typ と同様に演算器の遅延に Typical ケースを想定し，高位合成を行う．動作条件 *Worst* で動作させる場合には，必要なデータが到着するまで，他の演算の先送りにより待つことでタイミングエラーを回復する．全体の面

積のうち，各ハドルのレジスタの面積は Razor の面積として計算するが，タイミングエラー修正のためのコントローラの面積は考慮せず，通常のコントローラの面積と同一とする．また，エラー検出信号の配線遅延は考慮しないため，レイテンシはスケジューリング結果の動作 CS 数とタイミングエラーが起きた場合の挿入回復サイクル数の和となる．したがって，レイテンシ，面積に関して楽観的な結果となっている．

HDR-wst 演算器の遅延に Worst ケースを想定した，HDR アーキテクチャを対象とした高位合成手法（Worst ケース設計）[8].

MSHDR-normal HDR アーキテクチャを対象とした共通化処理を行わないマルチシナリオ高位合成手法．この場合，Typical シナリオ，Worst シナリオのスケジューリング，バインディングを独立に行い，それらの結果を 1 つの LSI 上に実現する（提案手法から共通化処理を除いた手法）.

MSHDR HDR アーキテクチャを対象としたマルチシナリオ高位合成手法（提案手法）.

4.2 動作 CS 数の期待値の算出

演算器 f_i の遅延分布は平均値 μ_i，標準偏差 σ_i のガウス分布 $N(\mu_i, \sigma_i^2)$ であると仮定するため，確率密度関数を $g_i(t)$ は，

$$g_i(t) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{(t-\mu_i)^2}{2\sigma_i^2}\Big), \tag{7}$$

となる．したがって，演算器 f_i の遅延が t 以下である確率 $G_i(t)$ は，

$$G_i(t) = \int_0^t \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{(\tau-\mu_i)^2}{2\sigma_i^2}\Big) d\tau, \tag{8}$$

のように計算できる．シナリオ境界遅延値 $B_i(T_{clk})$ を，クロック周期 T_{clk} において，演算器 f_i が Typical シナリオでタイミングエラーを起こさない上限の遅延値とすると，これは

$$B_i(T_{clk}) = \left\lceil \frac{\mu_i + D_{reg}}{T_{clk}} \right\rceil \cdot T_{clk} - D_{reg}, \tag{9}$$

と定義できる．したがって，$G_i(B_i(T_{clk}))$ は，Typical シナリオにおいて，クロック周期 T_{clk} の下で，演算器 f_i でタイミングエラーが起こらない確率を表す．

したがって，アプリケーション Z が Typical シナリオで動作する確率 P_Z は，クロック周期 T_{clk} の関数として，

$$\begin{aligned} &P_Z(T_{clk}) \\ &= \prod_{f_i \in F} G_i(B_i(T_{clk})) \\ &= \prod_{f_i \in F} \left\{ \int_0^{B_i(T_{clk})} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{(\tau-\mu_i)^2}{2\sigma_i^2}\Big) d\tau \right\}, \end{aligned} \tag{10}$$

と求められる．

アプリケーション Z に対し，MSHDR，MSHDR-normal の動作 CS 数の期待値 CS_{ex} は，以下の式で算出する．

$$CS_{ex} = P_Z(T_{clk}) \cdot CS_{typ} + (1 - P_Z(T_{clk})) \cdot CS_{wst}. \tag{11}$$

アプリケーション Z に対し，HDR-typ-recover の動作 CS 数の期待値 CS_{ex} は，以下の式で算出する．

$$\begin{aligned} &CS_{ex} \\ &= \sum_{WF_k \in \mathfrak{P}(F)} \left\{ \left(\prod_{f_i \in F} X_i(WF_k) \right) \cdot CS_r(WF_k) \right\}. \\ &X_i(WF_k) = \begin{cases} 1 - G_i(B_i(T_{clk})) & \text{if } f_i \in WF_k, \\ G_i(B_i(T_{clk})) & \text{otherwise.} \end{cases} \end{aligned} \tag{12}$$

WF_k は，F のべき集合 $\mathfrak{P}(F)$ の k 番目の要素であり，Worst ケースの遅延の演算器の集合を表す．$CS_r(WF_k)$ は，WF_k の時の回復サイクルを含めた動作 CS 数を表す．

4.3 実験結果とその評価

実験結果を表 2 に示す．以下では，各動作条件における動作 CS 数，動作 CS 数の期待値，面積を評価する．

4.3.1 動作条件が *Typical* の場合の動作 CS 数の評価

MSHDR を Typical シナリオで動作させる．MSHDR は，DCT，FIR，EWF3，JACOBI，PARKER で，HDR-typ と同等の CS 数で動作させることが可能である．COPY はわずかに HDR-typ よりも動作 CS 数が増加しているが，面積増大による配線遅延の増大が原因であると考えられる．HDR-wst と比較すると，動作 CS 数を最大 39%，平均 29%削減できる．

4.3.2 動作条件が *Worst* の場合の動作 CS 数の評価

MSHDR を Worst シナリオで動作させる．MSHDR は，DCT，FIR，EWF3，JACOBI，PARKER で，HDR-wst と同等の CS 数で動作させることが可能である．COPY はわずかに HDR-wst よりも動作 CS 数が増加しているが，前述と同様に面積増大による配線遅延の増大が原因であると考えられる．タイミングエラー時に回復サイクル挿入によりエラー修正できる HDR-typ-recover と比較すると，動作 CS 数を最大 54%，平均 28%削減できる．

4.3.3 動作 CS 数の期待値の評価

式 (11)，(12) より，動作 CS 数の期待値を算出した．MSHDR は，すべてのアプリケーションで，HDR-wst 以下のコントロールステップで動作可能であり，動作 CS 数を最大 35%，平均 20%削減できる．HDR-typ-recover と比較すると，動作 CS 数を平均 4%削減している．HDR-typ-recover は配線遅延考慮下における楽観的な結果であることに注意する．楽観的な設計に対しても，提案手法の平均的な動作 CS 数は優れているということが注目すべき点である．

4.3.4 面積の評価

MSHDR は，MSHDR-normal と比較すると，すべてのアプリケーションにおいて，2 つの動作条件および期待値で同等以下の CS 数，同等以下のレジスタ数で，全体の面積を最大 18%，平均 10%，マルチプレクサ数を最大 27%，平均 13%削減できる．したがって，共通化処理による面積削減効果を確認できる．しかし，提案手法 MSHDR は，HDR-wst と比較し平均 19%の面積オーバーヘッドがある．MSHDR-normal の HDR-wst と比較したときの面積オーバーヘッドは平均 34%である．したがって，面積オーバーヘッドは共通化処理により大きく改善できているが，改善の余地はあり得る．今後の課題として，より面積を削減できるような共通化処理の改良が挙げられる．

5. おわりに

本稿では，HDR アーキテクチャを対象としたマルチシナリオ高位合成手法のレイテンシおよび面積の評価を行った．計算機実験により，動作条件が *Typical*, *Worst* の場

表 2: 実験結果

App	演算器制約	アルゴリズム	CS_{typ}	CS_{wst}	CS_{ex}	面積 [μm^2]	MUX [個]	REG [個]	Controller [μm^2]	反復 [回]	CPU 時間 [sec]
DCT	Add×4	HDR-typ	12	-	-	36938	53	24	2305	2	572.81
	Mul ×4	HDR-typ-recover	13	26	14.7	58300	60	25	2458	5	1143.92
		HDR-wst	19	19	19	43758	66	30	2952	2	575.86
		MSHDR-normal	12	19	13.8	60480	113	30	4199	4	986.34
		MSHDR	12	19	13.8	51940	87	30	3680	3	789.27
FIR	Add×3	HDR-typ	33	-	-	29988	44	18	2630	2	461.55
	Mul×3	HDR-typ-recover	35	57	37.1	39038	41	17	2738	3	626.82
		HDR-wst	44	44	44	30016	41	20	3415	2	510.56
		MSHDR-normal	33	44	35.2	37558	57	20	4430	2	529.38
		MSHDR	33	44	35.2	32856	50	20	4352	2	529.03
EWF3	Add×3	HDR-typ	64	-	-	30744	57	14	3480	2	369.75
	Mul×2	HDR-typ-recover	70	126	73.9	58695	68	19	3741	2	368.69
		HDR-wst	90	90	90	38076	76	15	4150	2	379.01
		MSHDR-normal	64	90	68.4	49815	109	15	5953	2	401.24
		MSHDR	64	90	68.4	41013	80	15	5756	3	528.26
JACOBI	Add×2	HDR-typ	32	-	-	31668	23	13	1924	2	408.44
	Sub×1	HDR-typ-recover	34	64	39.9	42237	23	12	1932	2	407.94
	Mul×2	HDR-wst	51	51	51	30856	23	12	2070	2	413.95
	Div×2	MSHDR-normal	32	51	46.3	38850	39	15	3252	2	497.84
		MSHDR	32	51	46.3	37440	35	14	2901	2	474.67
PARKER	Add×1	HDR-typ	11	-	-	8094	11	10	543	2	211.6
	Sub×1	HDR-typ-recover	11	21	11.9	13674	10	10	523	2	212.72
	Comp×1	HDR-wst	18	18	18	8448	10	10	601	2	259.88
		MSHDR-normal	11	18	11.8	11025	19	10	934	2	215.49
		MSHDR	11	18	11.8	11025	19	10	934	2	214.17
COPY	Add×1	HDR-typ	145	-	-	252144	432	137	15357	8	2892.01
	Sub×1	HDR-typ-recover	180	352	186.5	345072	423	132	14265	5	1914.56
	Mul×5	HDR-wst	158	158	158	234207	447	143	14849	9	3679.36
	Comp×1	MSHDR-normal	164	179	168.5	353238	679	156	22968	10	4900.73
	RShift×2	MSHDR	150	162	153.6	300736	629	148	21053	8	5028.12
	AND×1										

合，Typical ケース設計，Worst ケース設計の場合とほぼ同一の CS 数で動作できることを確認した．また，Worst ケース設計と比較し，動作条件が *Typical* の場合の動作 CS 数を最大 39%，平均 29%，動作 CS 数の期待値を最大 35%，平均 20%削減できることを確認した．今後の課題として，より面積を削減できるような共通化処理の改良や，レジスタやマルチプレクサ，配線遅延のばらつきへの対応が考えられる．

謝辞

本研究は独立行政法人新エネルギー・産業技術総合開発機構（NEDO）の先導的産業技術創出事業の支援を受けて行われた．

参考文献

[1] Lucas, G., Cromar, S. and Chen, D.: FastYield: variation-aware, layout-driven simultaneous binding and module selection for performance yield optimization, *Proc. of ASP-DAC 2009*, pp. 61–66 (2009).

[2] Lucas, G. and Chen, D.: Variation-aware layout-driven scheduling for performance yield optimization, *Proc. of ICCAD 2010*, pp. 17–24 (2010).

[3] Cong, J., Liu, A. and Liu, B.: A variation-tolerant scheduler for better than worst-case behavioral synthesis, *Proc. of CODES+ISSS 2009*, pp. 221–228 (2009).

[4] Yoshida, H. and Fujita, M.: An energy-efficient patchable accelerator for post-silicon engineering changes, *Proc. of CODES+ISSS 2011*, pp. 13–20 (2011).

[5] Andriamisaina, C., Coussy, P., Casseau, E. and Chavet, C.: High-level synthesis for designing multimode architectures, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 29, No. 11, pp. 1736–1749 (2010).

[6] Wang, F., Wu, X. and Xie, Y.: Variability-driven module selection with joint design time optimization and post-silicon tuning, *Proc. of ASP-DAC 2008*, IEEE, pp. 2–9 (2008).

[7] Hagio, Y., Yanagisawa, M. and Togawa, N.: A delay-variation-aware high-level synthesis algorithm for RDR architectures, *IPSJ Trans. on System LSI Design Methodology*, Vol. 7, No. 0, pp. 81–90 (2014).

[8] Abe, S., Yanagisawa, M. and Togawa, N.: Energy-efficient High-level Synthesis for HDR Architectures, *IPSJ Trans. on System LSI Design Methodology*, Vol. 5, pp. 106–117 (2012).

[9] 井川昂輝, 阿部晋矢, 柳澤政生, 戸川望：HDR アーキテクチャを対象とした製造ばらつき耐性と低レイテンシを両立可能なマルチシナリオ高位合成手法，信学技報, VLD2014-86， Vol. 114, No. 328, pp. 105–110 (2014).

[10] Ohchi, A., Kohara, S., Togawa, N., Yanagisawa, M. and Ohtsuki, T.: Floorplan-driven high-level synthesis for distributed/shared-register architectures, *Information and Media Technologies*, Vol. 3, No. 4, pp. 691–703 (2008).

[11] Abe, S., Shi, Y., Yanagisawa, M. and Togawa, N.: MH^4 : multiple-supply-voltages aware high-level synthesis for high-integrated and high-frequency circuits for HDR architectures, *IEICE Electronics Express*, Vol. 9, No. 17, pp. 1414–1422 (2012).

[12] Ernst, D., Kim, N. S., Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K. et al.: Razor: a low-power pipeline based on circuit-level timing speculation, *Proc. of 2003 IEEE/ACM International Symposium on Microarchitecture*, pp. 7–18 (2003).