2011年度 修士論文

HDRアーキテクチャを対象とした複数電源電圧指向の低電力化高位合成に関する研究

指導教授 戸川 望 教授

早稲田大学大学院 基幹理工学研究科
情報理工学専攻

5111B004–2

阿部 晋矢

2012年2月6日

目 次

第1章

序論

1.1 本論文の背景と意義

消費電力の考慮は携帯機器，据え置き器のどちらのシステム LSI の設計においても重要な要素となっている．携帯機器に搭載される LSI では駆動時間を伸ばすことが問題となり，据え置き器に搭載される LSI では動作時の発熱が問題となっている．双方を解決するため低電力化を意識した LSI 設計が求められる．その他の LSI 設計時の問題として，配線遅延がある．配線遅延はモジュール間の通信に必要な遅延である．LSI 全体の遅延を考えた場合，半導体の微細化技術により演算処理の遅延であるゲート遅延が減少する一方，配線遅延の占める割合は増加傾向にある．今後もこの傾向が継続すると予想されるため，配線遅延を考慮した LSI 設計が必要となる．

LSI 設計の高速化手法として，高位合成があげられる．高位合成では C や C++といった抽象度の高い動作記述から，レジスタやクロックによる同期などというハードウェア特有の概念を意識した RTL(Register Transfer Level) 記述を自動合成する．高位合成により設計者の記述量や，人手によるミスは減少し，設計生産性を向上できる．また，高位合成を用いることで設計規模が増大し，人手では設計不可能だった回路を合成できる．低電力化と配線遅延は高位合成においても意識すべきである．本論文では，低電力化および配線遅延を考慮した高位合成手法に着目する．

低電力化を意識した高位合成手法として，複数電源電圧を考慮する手法 [4, 10, 16, 17, 25, 26, 29, 39, 40] がある．電圧を低くできれば消費電力を抑えられるが，回路の遅延が増加する．複数電源電圧では回路のクリティカルパスに高電圧を用い，速度を求められない部分に低電圧を割り当てることで低電力化を図る．複数電源電圧を考慮した高位合成には多くの手法があり，さまざまな制約のもとでアルゴリズムが考案されている．[25] は時間制約，[26] は資源制約，[10, 16, 17, 29, 39, 40] は時間と資源の両方を制約とする．最適解の求め方にも違いがあり，[4, 10, 16] は電力の最適化問題を ILP(integer linear programming) に帰着させ，これを解くことで何らかの最適解を得る手法で，[16, 17, 29, 39, 40] は発見的手法を用いる．[16] によると，ILP を用いた方が電力は削減できるが，発見的手法の方が短時間で解を得られる．しかし，これらの手法は電力の最適化のみを目的とし，配線遅延は考慮されておらず配線遅延の割合が増加する現状に則さない．

配線遅延を考慮した高位合成手法として，フロアプランを意識する高位合成手法がある [5, 9, 11, 21, 22]．これらは従来のレジスタ集中型アーキテクチャではなく，レジスタを演算器に分散させレジスタ-レジスタ間の通信にクロックサイクルを割り当てる，レジスタ分散型アーキテクチャ[9, 11] を基にしている．[22] では Generalized Distributed-Register アーキテクチャ(以下 GDR) を提案している．GDR ではローカルレジスタ，共有レジスタ群という 2 種類のレジスタを用意し，レジスタ-レジスタ間通信を実現する．[22] ではフロアプラン結

果をフィードバックする合成フローを取り，反復改良することで合成結果を得る．GDR の利点として，演算器やレジスタといった各モジュールを個別にフロアプランすることで，大きな解空間を考察できることがあげられる．結果として高速かつ小面積な解を得ることが可能である．しかし，低電力化を視野に入れた場合，各モジュールを個別に扱うことは欠点となる．複数電源電圧を含めた多くの低電力化手法は，レベルコンバータ等の新たなモジュール追加を前提とする．GDR の合成に新たなモジュールを追加することは難しく，低電力化に適さない．[5] では Regular Distributed-Register アーキテクチャ(以下 RDR) を提案している．RDR ではチップを均一の区画に分割し，区画内には演算器群，ローカルレジスタ，コントローラを配置する．区画内の演算はローカルレジスタを利用することで配線遅延を無視でき，区画外にデータを送る際レジスタ-レジスタ間通信を実現している．均一の区画にすることで，高位合成段階での配線遅延の予測を容易にしている．また，各モジュールを抽象化する区画により，モジュールの変化にも対応しやすい．新たなモジュールを追加し接続数減少を図った RDR-Pipe [5] や，FPGA 利用のためローカルレジスタをレジスタファイルに変更した DRFM [7]，モジュールを追加することでフォールトセキュアを考慮した RDR [30] といった派生アーキテクチャも存在する．しかし，RDR はチップを一定の大きさに分割するため，面積のオーバヘッドを伴う欠点がある．面積のオーバヘッドは無駄なモジュールや，配線遅延の増加を導き，低電力化の障害となる．

以上の背景から，本論文では低電力化と配線遅延を同時に考慮する高位合成手法を提案する．まず，低電力化することを視野に入れた配線遅延を考慮するレジスタ分散型のアーキテクチャ，Huddle-based Distributed-Register アーキテクチャ(以下 HDR) を提案する．HDR は GDR に区画を導入し，演算器，レジスタ，コントローラを抽象化したアーキテクチャである．ハドルは HDR における区画で，演算器群，ローカルレジスタ，コントローラ，レベルコンバータのモジュールを含む．続いて，HDR を対象とした複数電源電圧による低電力化高位合成手法を提案する．提案手法では配置情報をフィードバックし，反復改良する合成フローを取る．その際，近くに配置されたモジュールをハドルとして扱い，レジスタ，コントローラ，レベルコンバータを共有する．ハドル内の演算はモジュール間が十分近いため配線遅延を無視でき，ハドル間データ通信をする場合レジスタ-レジスタ間通信を行う．電源電圧の操作はハドルを単位として行い，資源制約と時間制約から，クリティカルパスでないハドルを低電圧に割り当てることで低電力化する．本手法は既存のレジスタ分散型アーキテクチャと比較し、低電力な高位合成結果を得ることを目的とする．

1.2 本論文の概要

本論文では，初めに複数電源電圧指向の高位合成手法の研究動向を紹介する．次にフロアプラン指向の高位合成手法の研究動向を紹介する．その中で低電力化することを視野に入れた配線遅延を考慮するレジスタ分散型のアーキテクチャとして HDR を提案する．HDR は GDR に区画を導入し，演算器，レジスタ，コントローラを抽象化したアーキテクチャである．ハドルは HDR における区画で，演算器群，ローカルレジスタ，コントローラ，レベルコンバータのモジュールを含む．続いて，HDR を対象とした複数電源電圧による低電力化高位合成手法を提案する．以下に本論文の構成を示す．

第 2 章「**複数電源電圧指向の高位合成手法の研究動向**」では，低電力化を目的とし複数電源電圧の適用を考慮した高位合成手法に関する調査を報告する．まず，システム LSI の低電力化技術である複数電源電圧について紹介する．次に，複数電源電圧を考慮した高位合成について紹介する．

第 3 章「**Huddle-based Distributed-Register アーキテクチャを対象とする低電力化高位合成手法**」では，まずフロアプラン指向の高位合成手法に関する調査報告として，GDR を対象とした高位合成手法，RDR および RDR の派生アーキテクチャを対象とした高位合成手法を紹介する．続いて，低電力化することを視野に入れた配線遅延を考慮するレジスタ分散型のアーキテクチャである HDR を提案する．そして，HDR を対象とした複数電源電圧による低電力化高位合成手法を提案する．提案手法では配置情報をフィードバックし，反復改良する合成フローを取る．その際，近くに配置されたモジュールをハドルとして扱い，レジスタ，コントローラ，レベルコンバータを共有する．ハドル内の演算はモジュール間が十分近いため配線遅延を無視でき，ハドル間データ通信をする場合レジスタ-レジスタ間通信を行う．電源電圧の操作はハドルを単位として行い，資源制約と時間制約から，クリティカルパスでないハドルを低電圧に割り当てることで低電力化する．最後に実験により提案手法を評価する．

第 4 章「**計算機実験結果**」では，第 3 章で提案したアルゴリズムを計算機上に実装し，提案手法を評価する．

第 5 章「**結論**」では，本論文の内容について総括し，今後の研究課題について検討する．

第2章

複数電源電圧指向の高位合成手法の研究動向

2.1 本章の概要

消費電力の考慮は携帯機器，据え置き器のどちらのシステム LSI の設計においても重要な要素となっている．携帯機器に搭載される LSI では駆動時間を伸ばすことが問題となり，据え置き器に搭載される LSI では動作時の発熱が問題となっている．双方を解決するため低電力化を意識した LSI 設計が求められる．本論文で対象とする高位合成においても電力消費の意識は重要である．

CMOS の LSI における消費電力は，ダイナミック電力，リーク電力，ショート・サーキット電力の大きく 3 つに分けられる．3 つのうち現在最も大きな割合を持つのはダイナミック電力である．本章ではダイナミック電力を削減するシステム LSI の設計手法として複数電源電圧を紹介する．

まず，複数電源電圧の消費電力を削減する原理を紹介し，既存の研究動向を報告する．続いて複数電源電圧を高位合成において考慮する既存研究を報告し，課題を考察する．

2.2 複数電源電圧

システム LSI を構成する CMOS 回路の電力消費は以下の式で表される [3].

$$P_{total} = p_t(C_L V V_{dd} f_{clk}) + I_{SC} V_{dd} + I_{leakage} V_{dd} \tag{2.1}$$

式の 3 つの項はそれぞれ，ダイナミック電力，ショートサーキット電力，リーク電力を表す．現在の設計プロセスにおいて大部分を占めるのがダイナミック電力である．ダイナミック電力を削減するシステム LSI の設計手法として複数電源電圧がある．複数電源電圧は式 (2.1) 中の V_{dd} を下げる手法である．2 乗で効果の表れるダイナミック電力はもちろん，ショートサーキット電力とリーク電力についても削減が期待できる．本節では，複数電源電圧の動作原理と適用事例を報告する．

複数電源電圧とは，回路中で動作速度が必要な部分にのみ従来の電源電圧を使い，動作速度が必要されない部分には低い電源電圧を使う低電力化手法である [2,8,24,32,33]．ダイナミック電力は電源電圧の 2 乗に比例するため，電源電圧を下げることはダイナミック電力の削減に最も効果が大きい．しかし，電源電圧を下げると MOS トランジスタの性能が低下し回路速度が遅くなるため，回路の性能を維持したまま電力消費量を下げるには工夫が必要である．複数電源電圧を実装する方法として，“回路ブロック単位 (粗粒度) で電源電圧を変える手法” と “ゲート単位 (細粒度) で電源電圧を変える手法” がある．

粗粒度の複数電源電圧の例を図 2.1 [2] に示す．高速動作が要求される回路ブロックは高い電源電圧 V_{DDH}(1.2V) で動作させ，ほかの論理ブロックおよびメモリは低い電源電圧 V_{DDL}(1V) で動作させる．

一方，細粒度の複数電源電圧では，高速動作が必要なクリティカル・パス上にあるゲートは従来の高い電圧で動作させ，クリティカル・パス上にないゲートは低電圧で動作させる．論理合成を使って生成された回路では，クリティカル・パス上のゲートは全体の中でも 10〜30% 程度と言われており，その他のゲートを低電圧にできる．

レベルコンバータ

電源電圧の異なるブロックに信号を送出する場合には，信号の電圧振幅を変換するレベルコンバータ[1]と呼ばれる回路を挿入する必要がある [8,32,33]．レベルコンバータ自体に遅延，消費エネルギーが存在するため，複数電源電圧適用時にはレベルコンバータのオーバヘッドを考慮することが必要である．

[1] レベルシフタと呼ばれる場合もある．

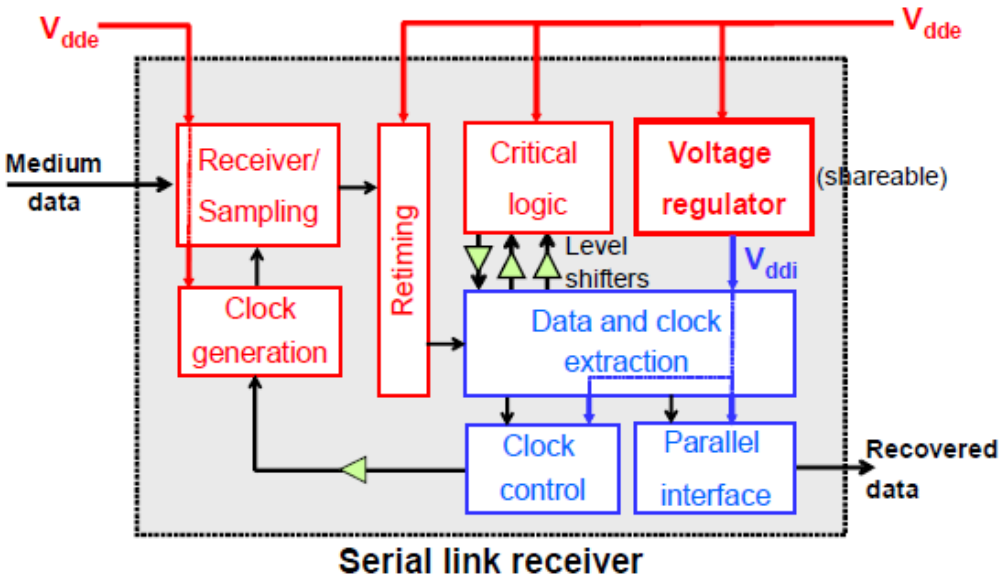


図 2.1: 粗粒度複数電源電圧 [2].

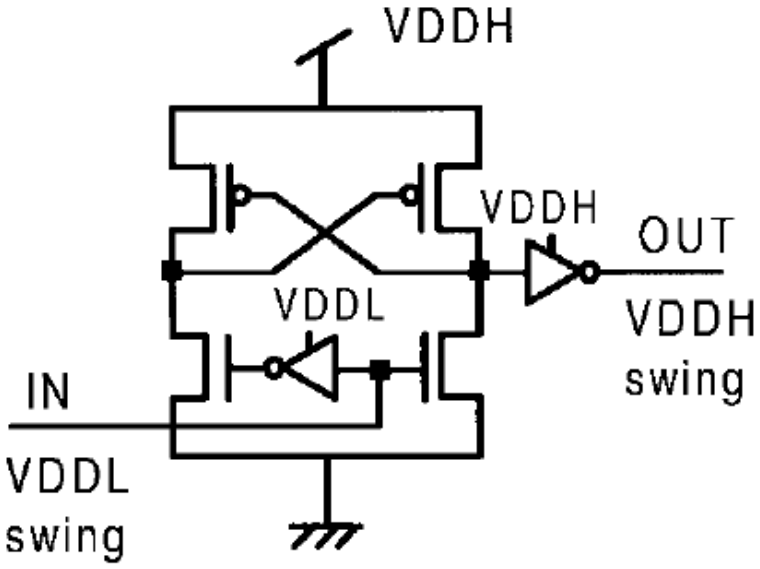


図 2.2: レベルコンバータ回路 [33].

図 2.2 に基本的なレベルコンバータ回路を示す [33]．図 2.2 の回路は遅延や消費エネルギーのオーバヘッドが大きいため，レベルコンバータ回路の改良に関しても研究されている [1,31]．[31] では遅延，消費エネルギーを削減した Contention mitigated level shifter (CMLS)，消費エネルギーは増加するが遅延が減少する Bypassing enabled level shifter (BELS) が報告されている．[1] では 1 つの電源で実現可能なレベルコンバータ回路が報告されている．

複数電源電圧の適用

複数電源電圧の適用に関しては多くの報告がある [4,8,10,13,14,16–18,24–29,32,33,38–41]．[32, 33] では 2 つの電源電圧を用い，メディアプロセッサの消費エネルギー削減を実現している．

設計自動化において複数電源電圧を適用する場合，設計におけるどの段階で複数電源電圧を適用するのかが問題となる．高位合成段階で複数電源電圧を適用した手法 [4, 10, 13, 16–18, 25, 26, 28, 29, 38–40] は後述する．高位合成以外の段階で複数電源電圧を適用する手法として，回路ブロックをレイアウトするフロアプラン時に複数電源電圧を適用する手法がある [14, 27, 41]．フロアプラン時に複数電源電圧を適用する場合，回路ブロックに割り当てる電圧，もしくは電圧を割り当てられた回路ブロック自体を voltage island と呼ぶ．voltage island の実現には，図 2.3 に示す電源網 (Power Network Resource) [14] が必要となる．複数電源電圧を用いる場合のフロアプラン結果は，資源となる電源網の量で評価できる．図 2.3 の (a)，(b) では，(b) の方が電源網が少ないため，良いフロアプラン結果と評価できる．フロアプラン時に複数電源電圧を適用する場合の目的は，消費エネルギーの削減，電源網の削

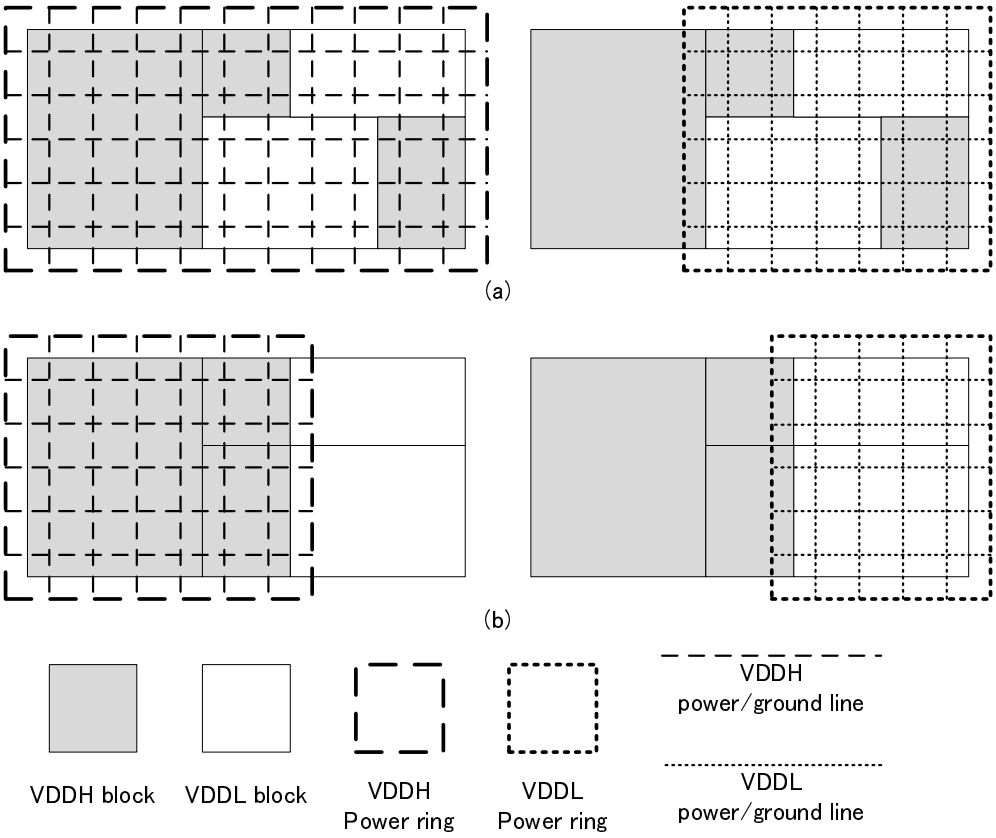


図 2.3: 複数電圧における電源網.

減となる.

フロアプラン時に複数電源電圧を適用する場合の問題点として，消費エネルギーの削減量が少ないことがあげられる．複数電源電圧において消費エネルギーを削減するには，回路ブロックに低い電圧を割り当てる必要がある．低い電圧を割り当てられた回路は処理速度が落ち，クリティカルパスとならない回路に低い電圧を割り当てることとなる．フロアプラン時には回路が同期するタイミングが決定しており，同期するタイミングを超えた電圧の変更は適用できない．結果としてクリティカルパスとなる回路が増え，また延長可能な時間が短くなり，低い電圧を割り当てることのできる回路ブロックが減少する．そのため，フロアプラン時に複数電源電圧を適用しても消費エネルギーの削減量は少なくなる．更なる消費エネルギーの削減へ向け，タイミングが決定する前の複数電源電圧の適用が必要となる．

2.3 複数電源電圧を考慮した高位合成

本章では高位合成段階で複数電源電圧を適用した研究動向を報告する．複数電源電圧を考慮した高位合成には多くの手法があり，さまざまな制約のもとでアルゴリズムが考案されている [4,10,13,16–18,25,26,28,29,38–40]．[25] は時間制約，[26] は資源制約，[10,16,17,29,39,40] は時間と資源の両方を制約とする．最適解の求め方にも違いがあり，[4,10,16] は電力の最適化問題を ILP(integer linear programming) に帰着させ，これを解くことで何らかの最適解を得る手法で，[16,17,29,39,40] は発見的手法を用いる．[16] によると，ILP を用いた方が電力は削減できるが，発見的手法の方が短時間で解を得られる．

これらの手法の問題点としてレベルコンバータの考慮が少ないことがあげられる．レベルコンバータについて考慮してる高位合成として [39,40] があるが，レベルコンバータのエネルギーのみ考慮している．レベルコンバータの遅延について詳細に考慮している手法は存在せず，レベルコンバータの遅延は無視する，もしくはレベルコンバータは常に使用すると考えた手法しか存在しない．

違う問題点として，配線遅延を考慮していないことがあげられる．設計プロセスの微細化を考慮した場合，配線遅延の考慮は必須となる．配線遅延を考慮しないこれらの手法を適用した場合，高位合成以降の論理合成やレイアウト時にタイミングを違反することは明確である．タイミング違反を起こさないように遅延を多めに見積もったり，時間制約を厳しくした場合，消費エネルギーの削減量が減少してしまう．配線遅延を考慮しないということはフロアプランを考慮しないこととなる．フロアプランまで考慮しない場合，電源網の考慮ができない．これらの手法により決定したタイミングを満たすようにレイアウトした場合，電源網のオーバヘッドにより期待した効果を得られない場合がある．

2.4 本章のまとめ

本章ではシステム LSI の低電力化手法として複数電源電圧を紹介した．複数電源電圧の適用事例としてフロアプラン時に複数電源電圧を適用する手法と，高位合成段階で複数電源電圧を適用する手法を紹介した．フロアプラン時に複数電源電圧を適用する手法には，タイミングが変更できないため消費エネルギーの削減量が少なくなる問題点がある．高位合成段階で複数電源電圧を適用する手法には，レベルコンバータの考慮が足りない点，配線遅延を考慮していない点という問題点がある．これらの問題点を受け，レベルコンバータの考慮と配線遅延の考慮が可能な HDR アーキテクチャを対象とした複数電源電圧指向の高位合成手法を提案する．

第3章

Huddle-based Distributed-Register アーキテクチャを対象とする低電力化高位合成手法

3.1 本章の概要

高位合成とはLSI設計において，抽象度の高い動作記述からハードウェアの概念を意識したRTL記述を自動合成する技術である．高位合成により設計者の記述量は減少し，設計生産性を向上できる．従来の高位合成手法では，下位工程に当たる論理合成やデバイスレベルとは独立なものとして高位合成を扱ってきた．しかしプロセス微細化の進む現在，演算実行にかかるゲート遅延に対し，演算器間のデータ転送による配線遅延の占める割合が増え，演算器の配置などといった従来の下位工程に当たる要素を意識した高位合成が必要となっている．高位合成段階で配線遅延を考慮する方法として，従来下位工程で意識する各演算器やレジスタなどの配置,「フロアプラン」を考慮する高位合成手法があげられる [5,9,11,21,22]．これらは従来のレジスタ集中型アーキテクチャではなく，レジスタを演算器に分散させレジスタ-レジスタ間の通信にクロックサイクルを割り当てる，レジスタ分散型アーキテクチャ[9,11]を基にしている．

本章では，フロアプラン指向の高位合成手法として，まずGeneralized Distributed-Registerアーキテクチャ(以下GDR)，Regular Distributed-Registerアーキテクチャ(以下RDR)およびその派生アーキテクチャを紹介する．既存のレジスタ分散型アーキテクチャの欠点を考慮し，低電力化することを視野に入れた配線遅延を考慮するレジスタ分散型アーキテクチャ，Huddle-based Distributed-Registerアーキテクチャ(以下HDR)を提案する．提案したHDRを対象に高位合成問題を定式化し，消費エネルギー削減を目的としたアルゴリズムを提案する．

3.2 GDR アーキテクチャと RDR アーキテクチャ

微細化技術の発達に伴い，回路遅延全体に占める配線遅延の割合の増加は避けられない．低電力化を意識した高位合成においても，配線遅延を考慮し複数サイクルレジスタ間通信を実現した，レジスタ分散型アーキテクチャを対象とすべきである．既存のレジスタ分散型アーキテクチャとして GDR [22] および RDR [5,30] がある．

GDR アーキテクチャ

GDR [22] は，レジスタ分散型アーキテクチャを基に提案された，ローカルレジスタおよび共有レジスタ群をそれぞれ必要数用意するアーキテクチャである．

GDR における 2 種類のレジスタを以下に示す．

- ローカルレジスタ
  ある演算器に隣接して配置されたレジスタ．隣接した演算器の入力・出力データしか保持しない．演算器がローカルレジスタを持つ場合，演算器の入力ポート数の入力側ローカルレジスタと 1 個以上の出力側ローカルレジスタを持つ．
- 共有レジスタ
  回路内にあるどの演算器からも利用可能なレジスタ．共有レジスタ群は共有レジスタが集合したものを指す．

制御信号を考えない場合における，入力レジスタから出力レジスタまでの演算時間 t^d_{toal} は以下の式で表される．

$$t^d_{total} = t_{m_i} + t_{r_if} + t_f + t_{m_o} + t_{fr_o} + t_{reg} \tag{3.1}$$

ここで t_{m_i}，t_{r_if}，t_f，t_{m_o}，t_{fr_o}，t_{reg} はそれぞれ入力側マルチプレクサ遅延，入力側レジスタ r_i から演算器 f までの配線遅延，出力側マルチプレクサ遅延，演算器 f から出力側レジスタ r_o までの配線遅延である．

ローカルレジスタを付加した場合を考えると，ローカルレジスタが付加された演算器の入力はローカルレジスタからと決まっているので，入力側マルチプレクサは必要なくなる．また，入力側ローカルレジスタから演算器までの配線遅延を無視でき，$t_{m_i} = 0, t_{r_if} = 0$ となる．もし出力先が対象演算器のローカルレジスタである場合，$t_{m_o} = 0, t_{fr_o} = 0$ となる．

GDR はボトルネックとなる演算器にローカルレジスタを付加し，配線遅延による影響を軽減する．ボトルネックでない演算器同士はレジスタを共有する．共有レジスタ群は共有レジスタの集合であり，共有レジスタ群内のレジスタの入出力マルチプレクサを含むモジュールのこととする．

次に制御信号を考慮した場合の演算時間を考える．まずローカルレジスタが付加された演算器 (図 3.1(a)) では演算時間 t^l_{total} は以下の式で表される．

$$t^l_{total} = t_{cf} + t_f + t_{fr_o} + t_{m_o} + t_{reg} \tag{3.2}$$

ここで，t_{cf} は制御回路から演算器までの配線遅延である．一方，共有レジスタを使用する演算器 (図 3.1(b)) の場合の演算時間 t^s_{total} は以下の式で表される．

$$\begin{aligned} t^s_{total} \quad = \max \quad & \{t_{cm_i} + t_{m_i} + t_{r_if} + t_f + t_{fr_o} + t_{m_o} + t_{reg}, \\ & t_{cf} + t_f + t_{fr_o} + t_{m_o} + t_{reg}\} \end{aligned} \tag{3.3}$$

ここで t_{cm_i} は制御回路から入力側マルチプレクサへの配線遅延時間である．さらに全ての制御回路は以下の式を満たす必要がある．

$$T_{clk} \geq t_{cm_o} + t_{m_o} + t_{reg} \tag{3.4}$$

ここで T_{clk} はクロック周期時間であり，t_{cm_o} は制御回路から出力側マルチプレクサへの配線遅延時間である．GDR は制御回路にローカルレジスタを付加するため，制御回路の実行時間 t_c は上記の式では無視できる．式 (3.2) および式 (3.3) では，t_{cm_i}，t_{cf}，t_{cm_o} がボトルネックとなり得る．これを解決するためには，ある演算器専用の制御回路をその演算器の近くに配置し，配線遅延を軽減する必要がある．そこで特定の演算器専用コントローラとして，ローカルコントローラを導入する．すべての演算器にローカルコントローラを付加するのは面積オーバヘッドが増加するので，GDR では以下の 2 種類の制御回路を使用する．

- ローカルコントローラ
  ある演算器に隣接して配置された制御回路．隣接した演算器と付随するローカルレジスタのマルチプレクサを制御する．
- グローバルコントローラ
  回路内にある複数の演算器，ローカルレジスタ付きの演算器，および共有レジスタに付随しているマルチプレクサに対する制御回路．

図 3.1(b) より共有レジスタを利用する演算器では共有レジスタから演算器の配線遅延，制御回路から共有レジスタへの配線遅延と演算時間が増加する．制御回路から共有レジスタへの配線遅延を削減するために，GDR ではグローバルコントローラは共有レジスタの近くに配置する．

以上のことから，GDR には以下の 4 種類のモジュールが存在する．

- 演算器のみ
- 共有レジスタ群+グローバルコントローラ

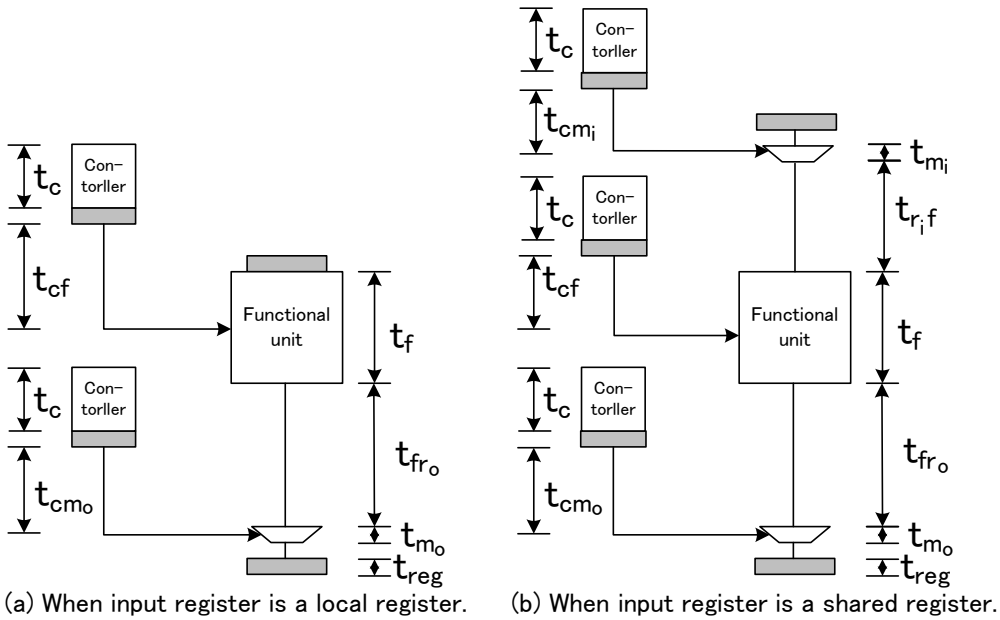

Con-
torller
Functional
unit
t_c
t_{cf}
t_f
t_{fr_o}
t_{cm_o}
t_{m_o}
t_{reg}
t_{cm_i}
t_{m_i}
$t_{r_i f}$
(a) When input register is a local register.
(b) When input register is a shared register.

図 3.1: 演算時間 [22].

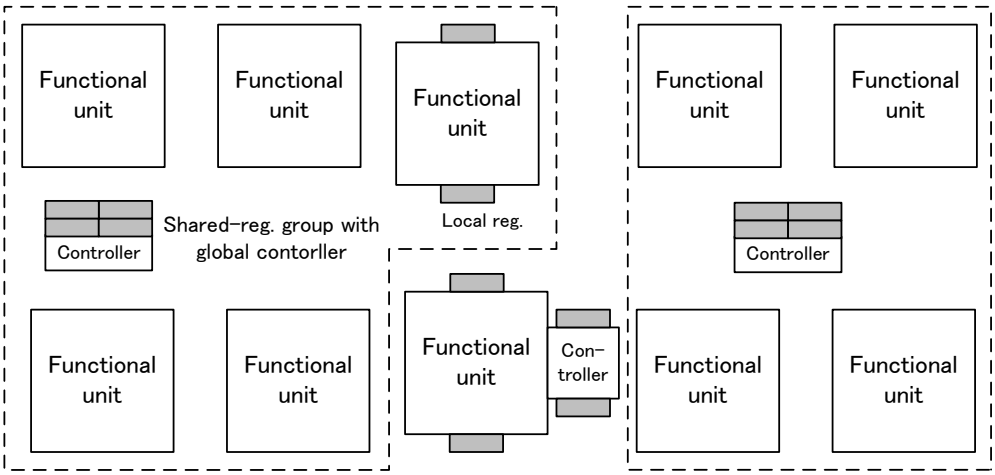


図 3.2: GDR アーキテクチャ[22].

- 演算器+ローカルレジスタ
- 演算器+ローカルレジスタ+ローカルコントローラ

GDR を図 3.2 に示す．GDR には複数の共有レジスタ群がある．各共有レジスタ群はグローバルコントローラを持ち図中の破線内の演算器，マルチプレクサを制御する．レジスタ演算器間の配線遅延がボトルネックとなる演算器はローカルレジスタを持ち，制御回路演算器間の配線遅延がボトルネックとなる演算器はローカルコントローラを持つ構成となる．GDR は共有レジスタ群を用いることで，完全なレジスタ分散アーキテクチャと比較しレジスタ数を削減できる．またローカルコントローラにより制御回路による遅延が減り，ローカルコントローラを必要な演算器のみ付加するため，面積の増加を抑えることができる．

GDR はレジスタ・レジスタ間データ転送を用いる．レジスタ間データ転送を扱うには，スケジューリング段階で各演算器間の配線遅延情報が必要となる．またレジスタの種類の決定，制御回路構成の決定には配置情報が必要である．以上を満足するため，配置情報，配線遅延情報をフィードバックする合成フロー [図 3.3] を用いる．

合成フローは初期フェーズ (Initial phase)，改良フェーズ (Improvement phase) の 2 段階に分けられる．初期フェーズ (Initial phase) では，まず入力 DFG をリストスケジューリングでスケジューリングする．次に共有レジスタが演算器数分あると仮定し，モジュール配置する．モジュール配置はデータ構造として Sequence-pair [20] を用い，SA(Simulated Annealing) により最適化する．SA のコスト関数は，A をデッドスペースを含む回路の総面積，W を各モジュールを結ぶ総配線長，V を各演算器間のデータ転送においてクロック周期制約条件を違反したディレイの総合計とした時

$$cost = \frac{A_{BB}}{A_{total}} + \alpha\frac{V}{T_{clock}} + \beta\frac{W}{W_{MAX}} \tag{3.5}$$

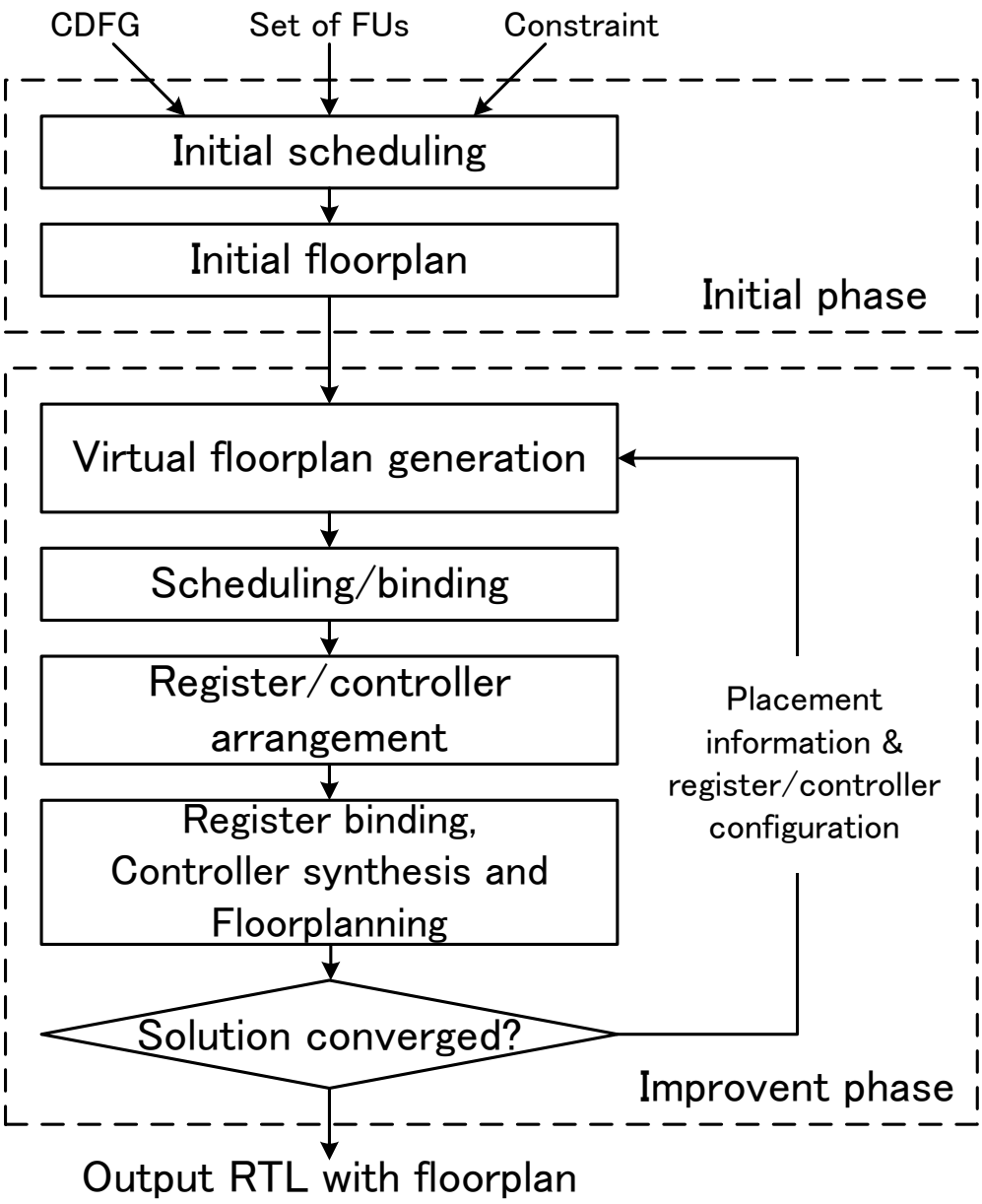


図 3.3: GDR 向け合成フロー [22].

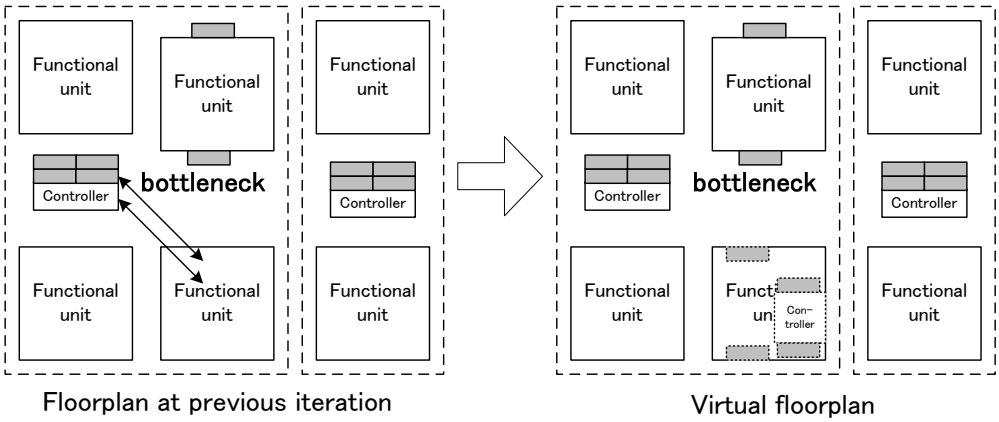


図 3.4: 仮想フロアプラン生成 [22].

と計算する．ただし，α，β は，任意のパラメータである．A_{BB} は最小矩形の面積，A_{total} はモジュール面積の総計，T_{clock} はクロック周期，V はクロック周期制約違反の総合計，W は配線長，W_{MAX} は配線長の最大値であり (最小矩形の縦+横)× 配線数である．これらの工程により，クロック周期制約違反がある可能性はあるが，アプリケーションの実行時間が高速となるフロアプラン解を生成する．

改良フェーズ (Improvement phase) は「仮想フロアプラン生成」,「スケジューリング/FU バインディング」,「レジスタ/コントローラ構成決定」,「レジスタバインディング」,「コントローラ合成」,「モジュール配置」から構成される．そのうち初期フェーズと異なるのは「仮想フロアプラン生成」,「スケジューリング/FU バインディング」,「レジスタ/コントローラ構成決定」である．

仮想フロアプラン生成では 1 つ前のイタレーション時のレジスタ/制御回路構成が，1 つ前のイタレーション時の配置結果で実現可能かチェックを行う．このチェックにより違反が発見されたならば，その演算器に仮想的にローカルレジスタ，ローカルコントローラを付加する (図 3.4)．この状態で各モジュール間のデータ転送制約を計算し，次工程のスケジューリング/FU バインディング工程に進む．

仮想フロアプラン後は，仮想フロアプラン生成で得られたデータ転送制約をもとにスケジューリング/FU バインディングを行う．初期フェーズと違い，データ転送制約をスケジューリングで考慮できる．

レジスタ/コントローラ構成決定では，スケジューリング/FU バインディングを満たすよう，ローカルレジスタ数，ローカルコントローラ数が最小化する．最初に各演算器が使用する共有レジスタ群を割り当て，次にクロック周期制約を違反する演算器にローカルレジスタ・ローカルコントローラを付加する．最後に隣接して配置された共有レジスタ群の併合をローカルレジスタが増加しない場合に行う．

モジュール配置の方法は初期フェーズと変わらないが，i 回目のイタレーションにおける最終的な配置結果を，$i+1$ 回目のイタレーションのモジュール配置工程における SA の初期配置として用いる．前回の配置を活かすため，イタレーションごとに SA の初期温度を下げていく．合成フローの i 回目のイタレーションの SA の初期温度を T_i としたとき $i+1$ 回目のイタレーションの初期温度を

$$T_{i+1} = T_i/K \tag{3.6}$$

とする．ただし $K > 1$ とする．

解が収束もしくは規定回数まで改良フェーズを繰り返し最良の解を出力する．

GDR は面積の増加を共有レジスタ群の導入により，レジスタ分散型アーキテクチャの問題点である面積の増加を解決している．また，4 種類のモジュールを扱うことで大きな解空間を対象とできる利点がある．

しかし，4 種類のモジュールを扱う利点は，一方で粒度の細かさという問題点がある．この問題点は GDR に対し，演算器，レジスタ，制御回路以外の新たなリソースを加えようとした際重要となる．新たなリソースについてローカルなもの，共有するものを考えると，さらにモジュールの種類は増加し最適解を求めるのに多くの実行時間を必要とする．粒度の細かさという問題点は，低電力化する上で新たなリソースを追加しようと考えた場合無視できない．

GDR は共有レジスタ群，ローカルレジスタ，グローバルコントローラ，ローカルコントローラ導入による小面積かつ高性能なレジスタ分散型アーキテクチャである．各モジュールを個別にフロアプランするため，大きな解空間を考察できる点が GDR の利点である．しかし，大きな解空間を考察できる利点は，高位合成問題が複雑になる欠点でもある [22]．レジスタ，コントローラの 2 つのモジュールについて，共有と専用を考慮ことが複雑さの原因である．例えば複数電源電圧を用いて低電力化を実現する場合，電圧を変更するレベルコンバータ等の新たなモジュールを追加する必要がある．追加されるこれらのモジュールについても共有と専用を考慮する必要があり，更に合成の複雑さが増す．そのため，小面積かつ高性能な GDR だが，低電力化には適さない．

RDR アーキテクチャ

RDR [5] は，チップ全体をアレイ状に均等な大きさの island に分割するアーキテクチャである．各 island は次の要素によって構成される [図 3.5].

- Local Computational Cluster (LCC)
  クラスタ化された機能ユニット群である．ここで，機能ユニット群とは，(1)FU(Functional Unit) および (2)MUX を指すものとする．
- Register file
  各 island 専用のローカルレジスタファイルである．LCC に含まれる機能ユニット群は同一の island 内のレジスタファイルのみにアクセスすることができる．
- Finite State Machine (FSM)
  各 island 専用のコントローラである．同一の island に含まれる LCC とレジスタを制御する．

island は図 3.6 のように配置される．異なる island に配置された FU 同士でデータをやり取りしたい場合には，ローカルレジスタファイル間データ転送を用いる．RDR は，各 island にローカルなレジスタを分散させて配置する点，およびマルチサイクルのレジスタ間データ転送を用いる点で，レジスタ分散型アーキテクチャに似ている．しかし，規則的に分割されているため，各 island 間の配線遅延の見積もりが容易になり，さらにモジュールを island により抽象化したことにより配置の粒度が粗くなる (island 内の配置は関係なる) という利点がある．フォールトセキュア計算に向け，余裕のあるアイランドに演算器を追加する高位合成手法 [30] は，アイランドによるモジュールの抽象化を活用した例である．

RDR のもう一つの利点は，ターゲットとなるクロック周期の変化に，island の大きさの変化で対応できる点である．ターゲットとなるクロック周期が与えられたとき，式 (3.7) が満たされるように island の大きさが決定される．

$$D_{intra-island} \leq D_{logic} + 2 \times D_{opt_int}(W_i + H_i) \leq T_{clk} \tag{3.7}$$

ただし，式 (3.7) 中で用いられているパラメータは以下の数値を表すものである．

- T_{clk}:クロック周期
- D_{logic}:LCC に含まれるロジックの遅延の最大値
- $D_{intra-island}$:island 内の遅延の平均値
- W_i:island の幅
- H_i:island の高さ

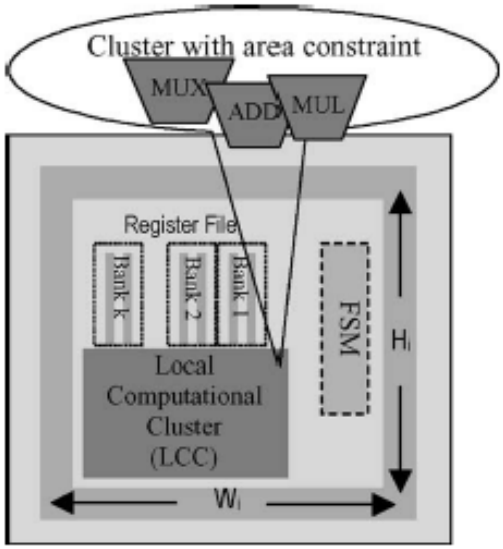


図 3.5: RDR における island の構成 [5].

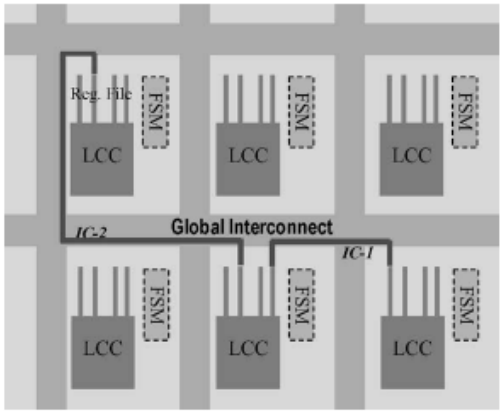


図 3.6: 2×3 の island 構成の RDR [5].

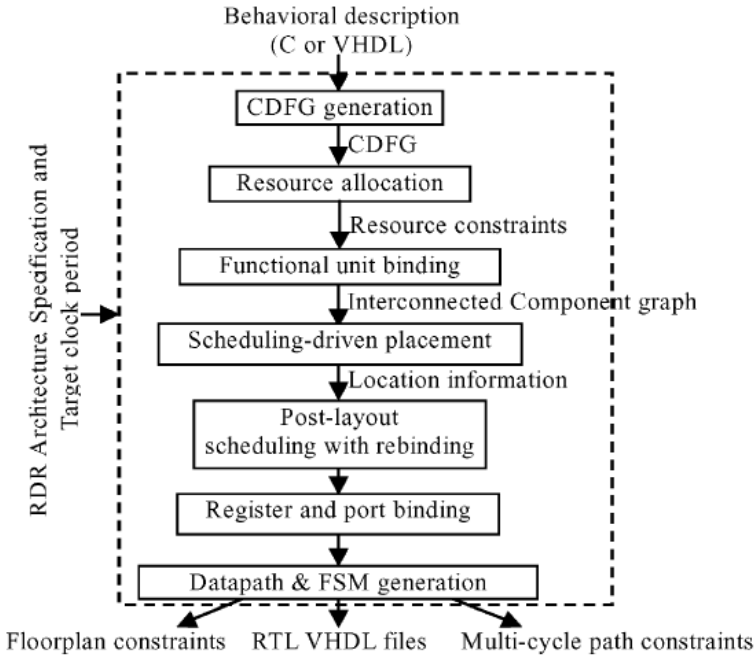


図 3.7: MCAS [5].

また $D_{opt_int}(x)$ は，距離 x に対して配線遅延の見積もり値を計算する関数である．上式が満たされるかぎり，island 内の演算器やレジスタの配置関係に関わらず，クロック周期制約が必ず満たされる．

RDR に対応した高位合成システム MCAS [5] のフローを図 3.7 に示す．クロック周期制約と CDFG を入力として，MCAS では (1) 演算器 (リソース) 数と (2)FU(Functional Unit) のバインディングを先に決定する．リソースアロケーションは FDS(Force Directed Scheduling) [23] により決定する．FU バインディング (CDFG の演算をどの演算器に割り当てるか) を決定するには，クリークパーティショニングを応用する手法を用いる．

MCAS システムのコアである Scheduling-driven placement の工程では，演算器へのバインディング済みの CDFG を入力として，各演算器がどの island に配置されるかという情報と，スケジューリング結果を出力する．Post-laytou rebinding & scheduing では，Scheduling-driven placement で得られた配置情報結果を基にスケジューリングとリバインディングを実行する．

バックエンドの Register and port binding，Datapath & FSM generation では，レジスタ及びポートのバインディングと，データパスと分散されたコントローラが合成が実行される．Datapath & FSM generation からは，論理合成用の RTL の VHDL とフロアプラン制約，及びマルチサイクルパス制約が出力される．最終的に出力された回路がパフォーマンス制約を満足できない場合，クロック周期制約や island の大きさを変えて，合成フローを再実行する．

高位合成の工程とフロアプランの工程を統合化するアプローチとして，MCAS は配置とスケジューリングを同時に実行する手法を取る．Scheduling–driven placement では SA(Simulated Annealing) によって，どの演算器がどの island に属するかを決定する．SA のムーブの種類は以下の 2 種類である．

- コンポーネントの移動
  ランダムに 1 つの演算器を選んで異なる island に移動する．
- コンポーネントの交換
  異なる island に配置されている 2 つの演算器を交換する．

コスト関数の計算のためにスケジューリング結果が必要となるが，スケジューリングをやり直すタイミングはムーブと同時ではなく，温度パラメータが更新されるときであるので，何万回もスケジューリングが繰り返されるわけではない．

RDR は配線遅延の見積もりが容易になり，配置問題の粒度が粗くなるという特徴を持つ．しかし，

- チップを均等に分割してしまうことで，面積の利用効率が悪くなる．
- 配置段階でコントローラやレジスタファイルの面積を考慮していない．

という問題点があると考えられる．特にチップを均等に分割し面積の利用効率が悪くなる点は消費電力の増加につながり，RDR は低電力化に向かないアーキテクチャであることが指摘されている [5].

RDR の派生アーキテクチャとして，Distributed Register–File Microarchitecture(以下 DRFM) [6,7] がある．DRFM は，FPGA 上での実装を想定したアーキテクチャである．FPGA ではマルチプレクサの面積が大きく換算されるため，実装する際なるべくマルチプレクサの量を減らしたい．DRFM では従来のレジスタではなく，レジスタファイルを用いることでマルチプレクサを削減している．レジスタファイルにはポート数の制限があるため，1 つのレジスタファイルで全ての変数を扱う場合，同時に実行できる演算数が制限される．同時に複数の演算を実行するため，DRFM ではレジスタファイルと演算器を island という括りで分散させる．DRFM における island は以下の要素で構成される [図 3.8].

- Local Register File(LRF)
  アイランド内のデータを保存するローカルレジスタファイル．
- Data-Routing Logic(DRL)
  外部アイランドから得られるデータを制御する要素．読み込みポート数に制限がある場合，Input Buffer でデータを保存する．
- Functional Unit Pool(FUP)
  アイランド内で処理する演算ユニットの集まり．

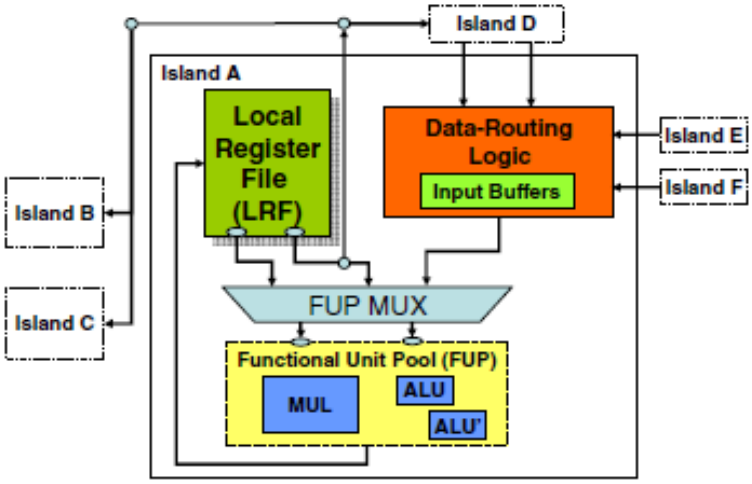


図 3.8: DRFM における island の構成 [6].

- FUP MUX
  FUP への入力を LRF，DRL から選択する．

DRFM ではスケジューリングされた DFG を入力とし，各ノードを island にバインディングする．その際，レジスタファイルの書き込みポートは 1 つと考え，同じコントロールステップの演算が違う island に割り当てられるよう，図 3.9 の様にバインディングする．目的関数は，DRL の複雑さが最終的な FPGA での面積に影響するため，island 間接続数 (Inter Island Connection：以下 IIC) の最小化とする．

DRFM バインディングは，ISB バインディング，VR バインディングの 2 つの工程に分けられる．ISB バインディングでは，ノードに対し最小重みのマッチングを行い，ノードを island に割り当てる．VR バインディングは ISB バインディングの後に行う．VR バインディングは min–cut 法を応用し，island に割り当てるノードを移動することで，IIC が最小となる island 割り当てを探す．

DRFM を対象とした研究として，[15] があげられる．この研究では，IIC を削減するためのスケジューリング，バインディングを提案している．

DRFM は island の大きさについて言及がなく，RDR と違い可変アイランドと考えられる．可変アイランドのため，island 単位でフロアプランを考慮する際，面積の無駄は軽減される．また GDR と比較して，island 内の配置を考慮せずに済み，フロアプランの粒度が粗くなる利点がある．しかし，

- 演算の種類を考慮しないため FUP が大きくなりすぎる可能性がある．
- FUP が全てのアイランドで似たような構成となり，可変アイランドを活かせない．
- コントローラを考慮していない．

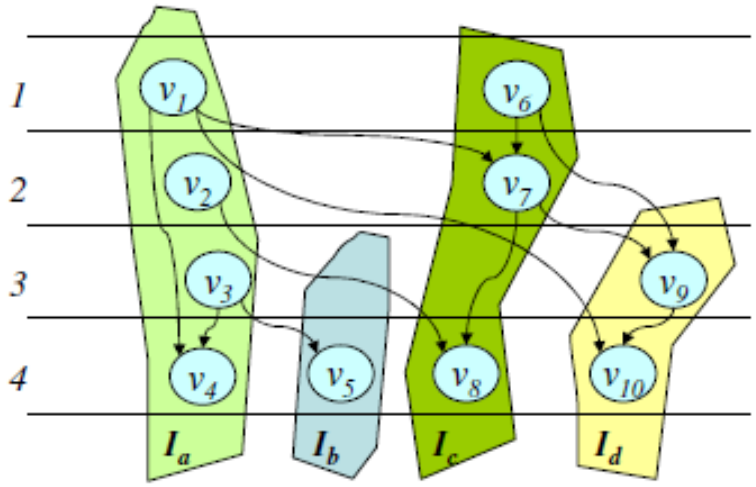


図 3.9: DRFM バインディング [6].

という問題点があると考えられる．特に FUP が大きくなりすぎる場合，リソース数増大に伴う消費電力の増加を無視できない．また，DRFM は FPGA での実装を前提としているため，FPGA 以外でのレジスタファイル利用によるマルチプレクサ削減の効果は分からない．

3.3 HDR アーキテクチャ

GDR は共有レジスタ群，ローカルレジスタ，グローバルコントローラ，ローカルコントローラ導入による小面積かつ高性能なレジスタ分散型アーキテクチャである．各モジュールを個別にフロアプランするため，大きな解空間を考察できる点が GDR の利点である．しかし，大きな解空間を考察できる利点は，高位合成問題が複雑になる欠点でもある [22]．レジスタ，コントローラの 2 つのモジュールについて，共有と専用を考慮ことが複雑さの原因である．例えば複数電源電圧を用いて低電力化を実現する場合，電圧を変更するレベルコンバータ等の新たなモジュールを追加する必要がある．追加されるこれらのモジュールについても共有と専用を考慮する必要があり，更に合成の複雑さが増す．そのため，小面積かつ高性能な GDR だが，低電力化には適さない．

RDR はアイランド (island) と呼ばれる一定の区画にチップを分割することで，配線遅延の予測を容易にしたレジスタ分散型アーキテクチャである．アイランドは専用の演算器，レジスタ，コントローラを持ち，他のアイランドと通信する際複数サイクルレジスタ間通信を行う．RDR の利点として，各モジュールを抽象化するアイランドにより，容易にモジュールを追加できる点があげられる．フォールトセキュア計算に向け，余裕のあるアイランドに演算器を追加する高位合成手法 [30] は，アイランドによるモジュールの抽象化を活用した例である．一方 RDR の欠点として，チップを一定に分割することによる，面積のオーバヘッドや無駄なモジュールの増加があげられる．面積のオーバヘッドや無駄なモジュールの増加は消費電力の増加につながり，RDR は低電力化に向かないアーキテクチャであることが指摘されている [5]．

低電力化に適したアーキテクチャとは，モジュールを追加しやすく，かつ小面積で消費電力の少ないアーキテクチャである．本節では，GDR と RDR の利点を併せ持つ低電力化に適したレジスタ分散型アーキテクチャ，Huddle-based Distributed Register アーキテクチャ(以下 HDR) を提案する．

HDR は GDR にハドル (Huddle) という区画を導入し，各モジュールを抽象化したアーキテクチャである．ハドル導入によるモジュールの抽象化で，RDR と同様にモジュールを容易に追加できる．ハドルはクロック周期により決定される範囲内において任意の矩形を取り，演算器やレジスタ，コントローラ，レベルコンバータを共有する．任意の矩形を取るハドルに対しフロアプランするため，GDR と同様に小面積で消費電力の少ないアーキテクチャとなる．

図 3.10 にハドルの構成を示す．ハドルは以下の要素からなる．

- Huddled Local Register (HLR)
  各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．

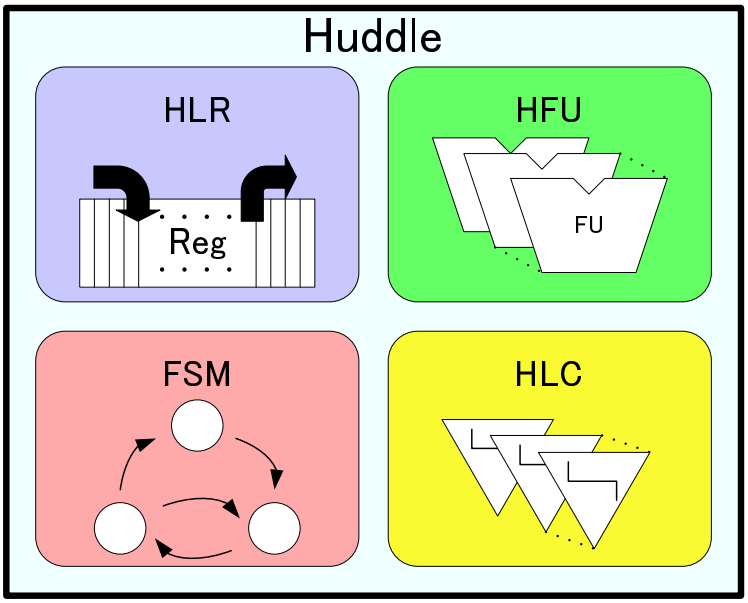


図 3.10: ハドルの構成.

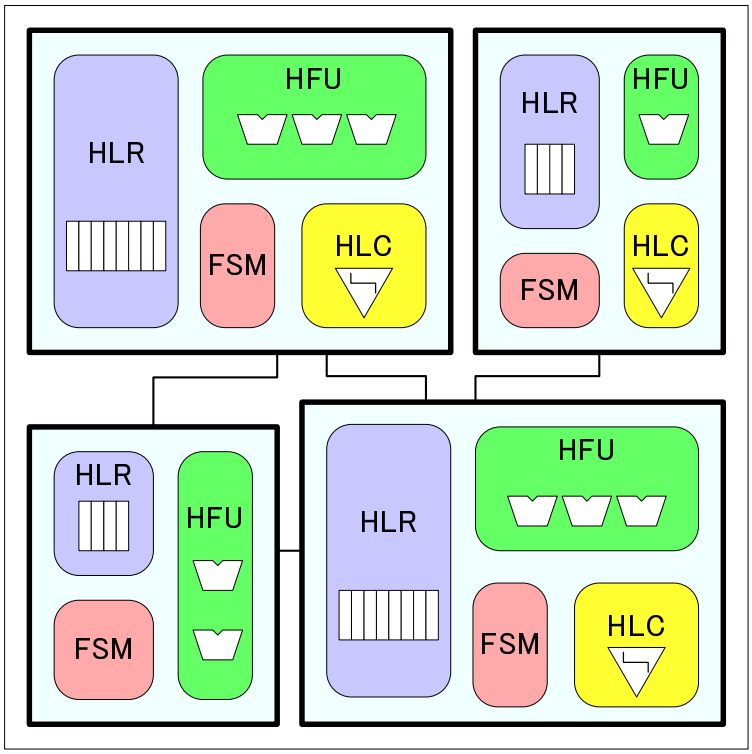
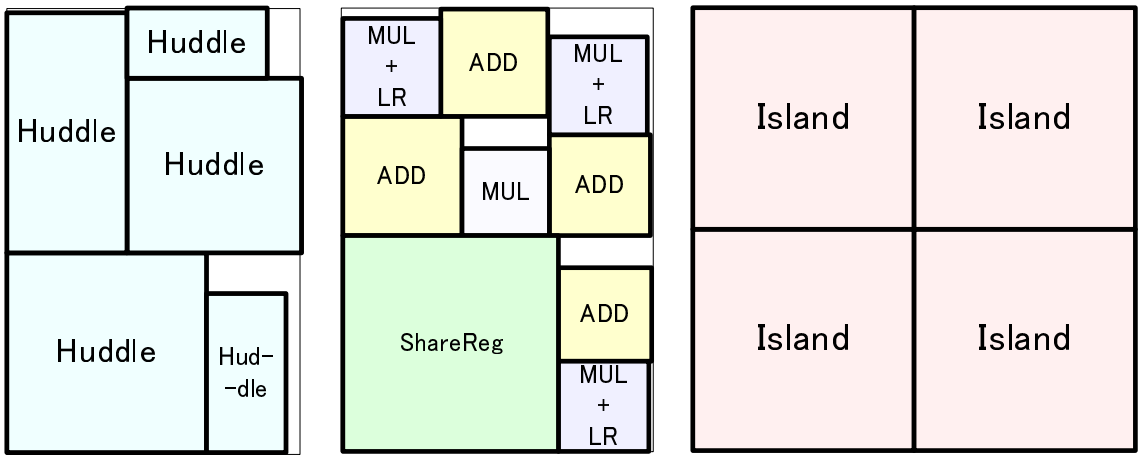


図 3.11: HDR アーキテクチャの構成.

- Huddled Functional Unit (HFU)
  ハドルに集められた演算器の集合．ハドル内で処理する演算に必要な演算器を必要数持ち，同一のハドル内の HLR のみにアクセスできる．



(a) HDR. (b) GDR. (c) RDR.

図 3.12: DCT を HDR，GDR [22]，RDR [5] で高位合成した場合の構成.

- Finite State Machine (FSM)
  各ハドル専用のコントローラ．同一ハドルの HFU と HLR を制御する．
- Huddled Level Converter (HLC)
  ハドルに集められたレベルコンバータの集合．電圧の異なるハドルとデータ転送を行う際，HLC を用いる．

ハドルは図 3.11 の様に配置する．同一ハドル内の HFU でデータを処理する場合，ハドル内の HLR を使うことでデータ転送時間は無視できる．異なるハドルの HFU 同士でデータ通信する場合，HLR 間データ転送を行う．HLR 間データ転送をする際，各ハドルの電圧が異なる場合 HLC を用いる．

例 1. ベンチマークアプリケーションの DCT を，提案アーキテクチャの HDR，従来型アーキテクチャの GDR [22]，RDR [5] それぞれで高位合成した結果を図 3.12 に示す．モジュールをハドルにより抽象化した HDR は，モジュールを個別に扱う GDR と比較し，ハドル 1 種類のみをフロアプランすればよいため，複雑さが減少している．一方 RDR と比較した場合，ハドルが任意の矩形を取る HDR は，一定の区画に分割する RDR よりも面積を削減できている．

□

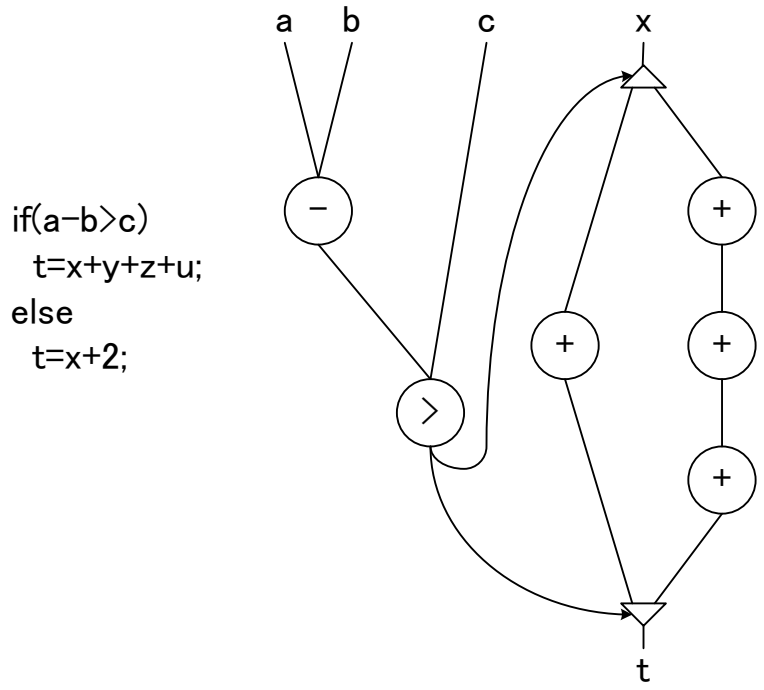


図 3.13: CDFG.

3.4 問題の定式化

高位合成において動作記述の中間形式のグラフ表現としてDFG（Data-Flow Graph），CFG（Control-Flow Grah），CDFG（Control-Data Flow Graph）が使われてきた．DFG はデータの流れを表したグラフであり，制御構造を表現できない．CFG では，並列処理を表現することができない．その問題を解決するために，DFG と CFG の概念を両方取り入れた CDFG を本論文では使用し，分岐処理の表現が容易な [37] をベースとした CDFG モデルを使用する．CDFG $G(N, E)$ は有向グラフで表現される．N は演算ノード N_o と制御ノード N_c からなる．演算ノード集合 N_o には ADD，MUL などの演算とメモリリード，メモリライト，ローカル配列を含む．メモリリードノードは 1 入力 1 出力であり，メモリライトノードは 2 入力（アドレス値と書き込む値）で出力はない．ローカル配列ノードはデータの参照時は 1 入力（index 値）1 出力であり，データの格納時は 2 入力（index 値と書き込む値）で出力はない．ローカル配列は回路上ではレジスタファイルとして実現される．制御ノード集合 N_c はフォークノード △ とジョイン ▽ ノードがある．フォークノードは分岐の始まりを表し，ジョインノードは分岐の終わりを示している．E はデータフローエッジ E_d とコントロールフローエッジ E_c からなる．データフローエッジはデータの流れを現しており，コントロールフローエッジはどちらに分岐するなどの依存関係を表しており，分岐の条件判定に用いられる結果を演算するコントロールノードからフォークノードとジョインノードに接続される．

入力としてクロック周期制約 T_{clk}，ステップ制約 S_{max} を与える．入力として p 個の演算器の集合 $F = \{f_1, \cdots, f_p\}$ を与え，演算器 f_i の遅延を $D_f(f_i)$ で表す．$S_f(f_i)$ は f_i の必要ステップ数を表し，クロック周期 T_{clk} より $S_f(f_i) = \lceil D_f(f_i)/T_{clk} \rceil$ と計算される．演算器 f_i が1回の処理につき消費するエネルギーを $E(f_i)$ で表す．

q 個 $(q \leq p)$ のハドル $H = \{h_1, \cdots, h_q\}$ に各演算器を割り当てる．演算器 f_i を割り当てるハドルを $Hud(f_i)$ で表す．一方，ハドル h_j に割り当てられた演算器の集合を $F(h_j)$ で表す．ハドル h_j における HLR の遅延を $D_{reg}(h_j)$ で表す．

各ハドルに割り当てる電圧は $v_l, v_m, v_h (v_l < v_m < v_h)$ の3種類とする．ハドル h_j の電圧を $V(h_j)$ で表す．v_l から v_m へ電圧を変更する際のレベルコンバータの遅延を $D_{lc}(v_l, v_m)$ で表す．v_l から v_h など他の電圧についても同様である．

演算器 f_i における，クロック周期から演算処理に必要な時間を除いた時間を以下の式で表す．

$$Slack(f_j) = T_{clk} \cdot S_f(f_i) - D_f(f_i) \tag{3.8}$$

RDR のアイランド [5] を参考に $Slack(f_i)$ から矩形であるハドル $h_j = Hud(f_i)$ の幅，高さを以下の式で求める．

$$\begin{aligned} &2 \cdot D_w(W(h_j) + H(h_j)) + D_{reg}(h_j) \\ &\qquad \leq \min_{f_i \in F(h_j)} \{Slack(f_i)\} \end{aligned} \tag{3.9}$$

式 (3.9) において $W(h_j)$ はハドル h_j の幅，$H(h_j)$ はハドル h_j の高さを表す．$D_w(x)$ は距離 x における配線遅延を表す．配線遅延は距離の2乗に比例するとし，配線遅延係数 C_d を用いて $D_w(x) = C_d x^2$ で計算される．式 (2) をハドルサイズ制約と呼ぶ．

演算器 f_i からハドル h_k へデータ転送する場合を考える．$Hud(f_i) = h_j$ とする時，h_j と h_k の中心間のマンハッタン距離を $Dist(h_j, h_k)$ で表すと，h_j，h_k 間の配線遅延は $D_w(Dist(h_j, h_k))$ となる．f_i からデータを転送し，h_k の HLR へ書き込むまでの時間は以下の式で表される．

$$\begin{aligned} Tr(f_i, h_k) &= D_w(Dist(h_j, h_k)) \\ &+ D_{lc}(V(h_j), V(h_k)) \\ &+ D_{reg}(h_k) \end{aligned} \tag{3.10}$$

$Slack(f_i)$ および $Tr(f_i, h_k)$ より f_i から h_k へデータ転送する際必要なクロック数 $DT(f_i, h_k)$ は以下の式で表される．

表 3.1: 演算器の遅延/エネルギー.

	加算器	乗算器
v_h	1 ns/144 fJ	2 ns/1440 fJ
v_m	2 ns/100 fJ	4 ns/1000 fJ
v_l	4 ns/64 fJ	8 ns/640 fJ

表 3.2: レベルコンバータの遅延.

	v_h	v_m	v_l
v_h	-	0.5 ns	1.0 ns
v_m	0.5 ns	-	2.0 ns
v_l	1.0 ns	2.0 ns	-

$$DT(f_i, h_k) = \begin{cases} 0 & (Slack(f_i) \geq Tr(f_i, h_k)) \\ \lceil Tr(f_i, h_k)/T_{clk} \rceil & (Slack(f_i) < Tr(f_i, h_k)) \end{cases} \tag{3.11}$$

このとき，DT とは p 行 q 列のテーブルで，i 行 k 列の要素が $DT(f_i, h_k)$ となるものとする．

例 2. 図 3.14，図 3.15 に $T_{clk} = 3\,\mathrm{ns}, S_{max} = 5$ の例を示す．演算器の各電圧における遅延と消費エネルギーを表 3.1 に示し，レベルコンバータの遅延を表 3.2 に示す．

入力演算器は f_1 : MUL1，f_2 : MUL2，f_3 : MUL3，f_4 : ADD1 の 4 つで，図 3.14 の通り $F(h_1) = \{f_1\}, F(h_2) = \{f_2\}, F(h_3) = \{f_3, f_4\}$ となる．電圧は図 3.14 の通り $V(h_1) = V(h_2) = v_h$，$V(h_3) = v_m$ である．

n_4 から n_7 における，演算器 f_1 からハドル h_3 への HLR 間データ転送に注目する．$D_w(Dist(h_1, h_3)) = 1\,\mathrm{ns}$，$D_{reg}(h_3) = 0.5\,\mathrm{ns}$ とする．このとき $Slack(f_1)$ は

$$\begin{aligned} Slack(f_1) &= 3\mathrm{ns} \times 1 - 2\mathrm{ns} \\ &= 1\mathrm{ns} \end{aligned}$$

となり，$Tr(f_1, h_3)$ は

$$\begin{aligned} Tr(f_1, h_3) &= 1\mathrm{ns} + 0.5\mathrm{ns} + 0.5\mathrm{ns} \\ &= 2\mathrm{ns} \end{aligned}$$

となる．$Slack(f_1) < Tr(f_1, h_3)$ のため，$DT(f_1, h_3)$ は

$$\begin{aligned} DT(f_1, h_3) &= \lceil 2/3 \rceil \\ &= 1 \end{aligned}$$

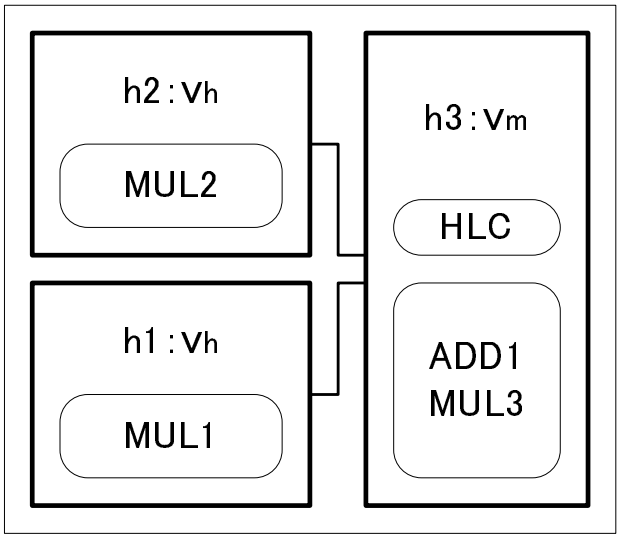


図 3.14: HDR アーキテクチャの構成例.

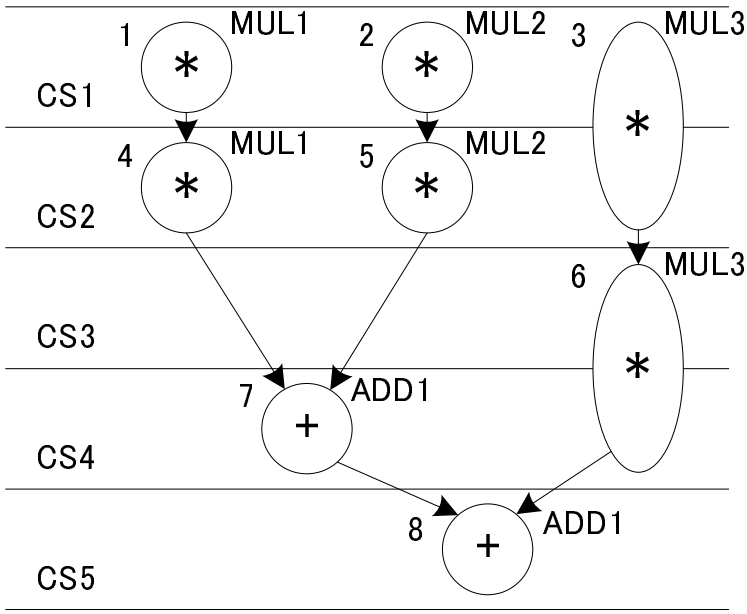


図 3.15: HDR アーキテクチャへのスケジューリング・バインディング例.

となる．よって n_4 から n_7 における，f_1 から h_3 への HLR 間データ転送には 1 クロック必要であり，図 3.15 ではコントロールステップ 3(CS3) をデータ転送に費やしている．

□

以上より，HDR アーキテクチャを対象とした複数電源電圧高位合成問題を次のように定義する．

定義 1. HDR アーキテクチャを対象とした複数電源電圧高位合成問題とは，CDFG，クロック周期制約，ステップ制約，演算器の集合が与えられた時，消費エネルギーを最小化するよ

うにCDFGをスケジューリングおよびバインディングし，各演算器をハドルに割り当て，ハドルに電圧を割り当てることである．

□

3.5 アルゴリズム

HDR アーキテクチャは複数電源電圧と複数サイクルレジスタ間通信を扱う，新しいレジスタ分散型のアーキテクチャである．そのため HDR アーキテクチャに適した，複数電源電圧と複数サイクルレジスタ間通信を同時に考慮する高位合成アルゴリズムが必要となる．

スケジューリングやバインディング，フロアプランなど複数の工程からなる一般的な高位合成手法として

手法 1 各工程を決まった順番に決まった数のみ順番に行う方法.

手法 2 反復改良する方法.

が考えられる．手法 1 は決まった数だけ処理を行うため，合成に必要な時間が決定的となる利点がある．各工程の必要数が事前に分かっている場合，手法 1 が適している．例えば，RDR を対象とした高位合成手法 [5] は手法 1 のアルゴリズムとなっている．RDR はアイランドの大きさや配置があらかじめ決まっているため，スケジューリングやバインディング結果が変わり，アイランド内の構成が変わったとしても，アイランド間の配線遅延が変化しない．そのため必要な工程数が決定的で，手法 1 が適している．

一方手法 2 では，必要な回数だけ処理を反復することで，各工程に情報をフィードバックできる利点がある．例えば，GDR を対象とした高位合成手法 [22] では反復改良するアルゴリズムとなっている．GDR のスケジューリング，バインディングは事前に各モジュールをフロアプランし，配置情報から予測された配線遅延を結果に反映している．一方フロアプランは事前にスケジューリング，バインディングし，バインディング結果から計算された面積によりモジュールの形を決定し，フロアプラン結果に反映している．このように GDR のスケジューリング，バインディング，フロアプランはそれぞれの結果が密接に影響し合っているため，あらかじめ回数を決定できない．そのため GDR では反復改良する手法 2 が適している．

加えて，複数電源電圧を考慮した低電力化手法に関しても手法 2 が有効である．手法 1 をベースとした従来手法では，レベルコンバータの遅延は無視する，もしくは常時レベルコンバータを使用すると考えるものが多い [4, 10, 16, 17, 25, 26, 29, 39, 40]．手法 2 をベースとする場合，配線遅延と同様にスケジューリング，バインディング結果に必要分のみレベルコンバータの遅延を考慮できる．

HDR はスケジューリング，バインディング結果でハドルの構成が決まるため，あらかじめ回数を決定することは難しい．また，HDR は複数電源電圧を扱うため，反復改良によりレベルコンバータの遅延を考慮すべきである．以上より，本稿では反復改良するアルゴリズムを採用する．

HDR を対象とする高位合成は以下の要素から構成される．

- スケジューリング/FU バインディング
- レジスタ/コントローラ合成/フロアプラン
- ハドル合成
- ハドル分割

スケジューリング/FU バインディングでは，フロアプラン結果から配線遅延を計算し，複数電源電圧，複数サイクルレジスタ間通信を考慮したスケジューリングおよび FU バインディングを行う．レジスタ/コントローラ合成/フロアプランでは，スケジューリング/FU バインディング結果よりハドルの構成を決定し，ハドルを対象にフロアプランする．ハドル合成は HDR 特有のハドルに関わる要素で，隣接するハドルを併合する．ハドル分割もハドルに関わる要素で，ハドル合成と反対にハドルを分割する．

高位合成アルゴリズムの構築に向けて要素の実行順序が問題となる．HDR の合成を考えた際，スケジューリング/FU バインディング，ハドル合成，ハドル分割はフロアプラン結果が必要である．従来のレジスタ集中型アーキテクチャではフロアプランを考慮しないため，バインディング問題をグラフのクリーク分割問題に帰着させるなど複雑な処理が必要であった．しかし，レジスタ分散型アーキテクチャを対象としたスケジューリング/FU バインディング [21] では，フロアプラン結果を利用することでスケジューリングとバインディングを同時に単純な処理で実行できる．同様に HDR は，複数電源電圧や複数サイクルレジスタ間通信，HDR 特有のハドルの構成問題についてフロアプラン結果を利用する．

フロアプラン結果を利用する際，HDR 特有の問題がハドルの構成問題である．ハドルの構成問題の解決法として，

解決法 1 1 つのハドルに全ての演算器が割り当てられた状態から分割する方法.

解決法 2 演算器数と同じ数ハドルがある状態から合成する方法.

が考えられる．解決法 1 は 1 つのハドルに全ての演算器が割り当てられる，最良の状態から高位合成をはじめ，大きさが上限を超えるなどの理由によりスケジューリング結果を満たさないハドルに対しハドル分割する．解決法 1 の利点は，最良の状態から合成を始めることができるため，制約内におけるハドル数を最も少なくする解を得やすい点である．一方で分割の条件の設定にフロアプラン結果を利用できない欠点がある．HDR ではハドルによるモジュールの抽象化により，ハドル内の配置については考慮しない．そのため，フロアプランしたとしてもスケジューリング結果を違反する原因が分からず，ハドルから分割すべき演算器が分からない．また，複数電源電圧を考慮する際も，全ての演算器が 1 つのハドルに割り当てられた場合，ハドル単位で電圧を操作できない点や，ハドル内に複数の電圧が混在してしまう点など問題がある．

解決法2は初期状態として各ハドルに1つの演算器が割り当てられる状態から合成が始まり，フロアプラン結果を基にハドルを合成していく．フロアプランにより2つのハドルが近くに配置た場合，その2つのハドルは併合すべきハドルとなる．ハドル合成によりハドルを併合すると，レジスタやコントローラが変化するため，クロック周期違反を起こす可能性がある．ハドル合成によるクロック周期違反は，ハドル合成後にもう一度レジスタやコントローラを合成しフロアプランすることで検証できる．フロアプランの結果クロック周期違反を起こすハドルに関してはハドル分割をすればよい．解決法2では，従来複数サイクルレジスタ間通信の考慮のために行うフロアプランをHDR特有のハドル構成に利用できる．また，複数電源電圧を考慮する際も，全ての演算器が個別のハドルに割り当てられている場合，ハドル単位で電圧操作できる．

以上より，提案手法では解決法2を採用する．よって最終的な高位合成アルゴリズムは図3.16となる．なお，説明を簡単にするため本章では入力はDFGのみを考えるが，入力をCDFGとしても同様である．また，演算器はビット幅を固定の加算器と乗算器のみを考える．

初期ハドル合成

初期ハドル合成は入力の演算器制約から，初期のハドルの構成，配置を決定する．演算器数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る．全てのハドルの電圧はv_hとし，配線長が0となるよう重なって配置する．よって，$F(h_i), V(h_i), Dist(h_i, h_j)$は以下の通りとなる．

$$
\begin{aligned}
F(h_i) &= \{f_i\}, && (1 \leq i \leq p) \\
V(h_i) &= v_h, && (1 \leq i \leq p) \\
Dist(h_i, h_j) &= 0, && (1 \leq i, j \leq p)
\end{aligned}
$$

以降は初期ハドル合成結果を受け，第1回目の反復を開始する．

スケジューリング/FUバインディング

演算に割り当てる電圧を変更した際，演算の実行時間，演算実行による消費エネルギーが変化する．ある演算に低い電圧を割り当てた場合，実行時間は増加し，消費エネルギーは減少する．実行時間が変化すると演算の実行を開始するステップや，実行にかかるステップ数，実行を終了するステップが変化する可能性がある．スケジューリング/FUバインディング，レジスタ/コントローラ合成/フロアプラン，ハドル合成，ハドル分割の要素のうち，演算の実行タイミングを決定するのはスケジューリング/FUバインディングである．そこで，提案

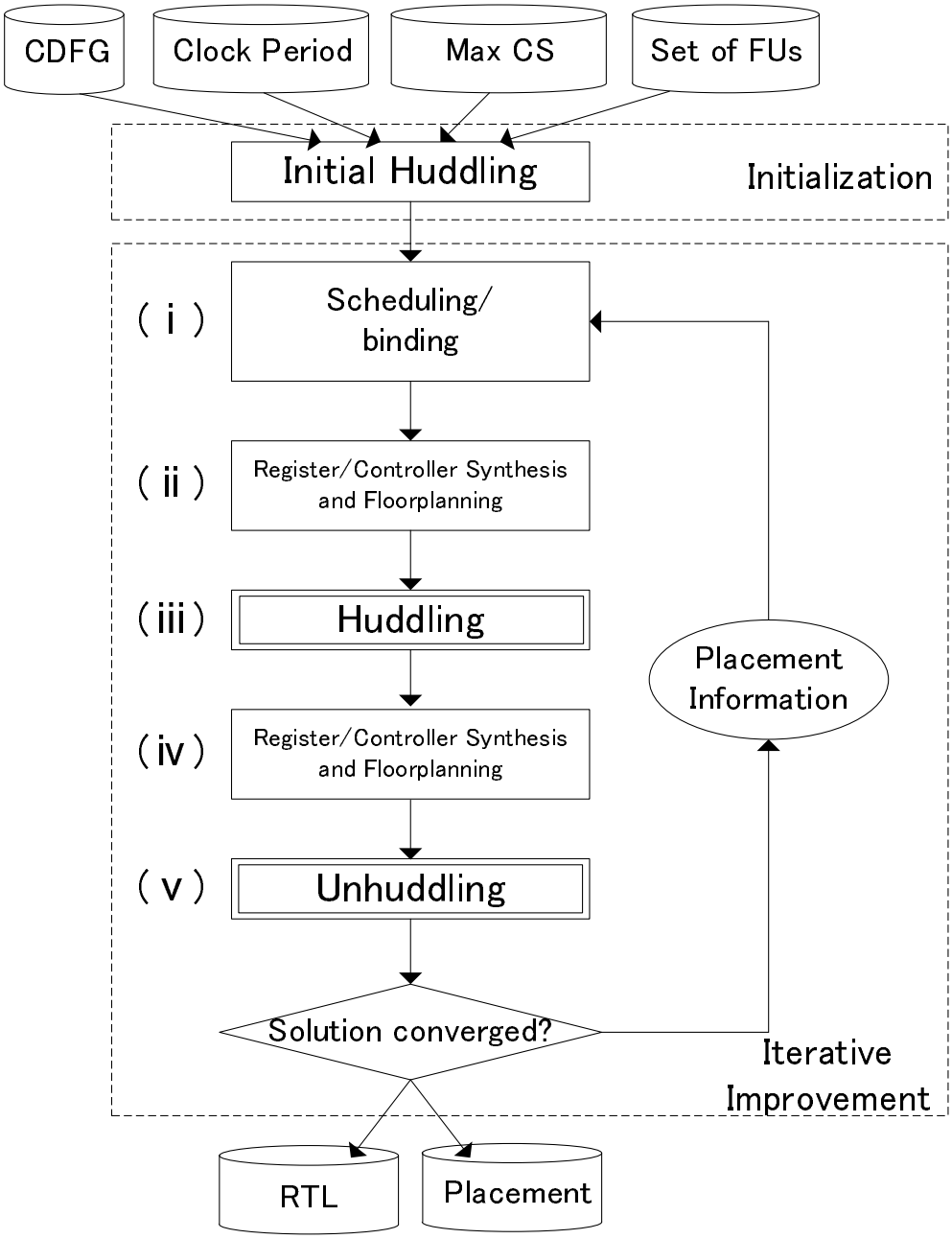


図 3.16: HDR 合成アルゴリズム.

手法ではスケジューリング/FU バインディングで演算に電圧を割り当て，ここで決定した電圧をもとに以降の要素を実行する．

提案するスケジューリング/FU バインディングでは，各ハドルへ電圧を割り当てることで，ハドルに含まれる演算器に電圧を割り当てる．電圧を割り当てられた演算器に演算をバインディングすることで，演算に電圧を割り当てる．スケジューリング/FU バインディングの入力は，クロック周期制約 T_{clk}，ステップ制約 S_{max}，CDFG $G(N, E)$，演算器数，ハドルの構成，ハドルの電圧，配置情報である．出力は演算ノード集合 N_o を実行するコントロールス

テップ，実行する演算器，ハドルの電圧である．ただし，第1回目のループはフロアプランの実行前のため，全てのハドル間の配線遅延を0とする．

電圧を割り当てるハドルは優先度により選択する．優先度はハドルに所属する演算器が1回の処理に消費するエネルギーの合計とし，ハドル h_j の優先度 $P_s(h_j)$ を以下の式で表す．

$$P_s(h_j) = \sum_{f_i \in F(h_j)} E(f_i) \tag{3.12}$$

スケジューリング/FUバインディングは初期フェーズ，電圧上昇フェーズ，電圧下降フェーズの3つのフェーズで構成される．初期フェーズは前回の反復時の配置，電圧を変更せずにスケジューリング/FUバインディングを行う．電圧上昇フェーズは初期フェーズの結果がステップ制約を満たさない場合実行され，ステップ制約を満たすようハドルの電圧を上げる．電圧下降フェーズはステップ制約を満たす範囲で消費エネルギーが最小となるようハドルの電圧を下げる．各フェーズを通し，電圧が最小となる結果を採用する．以下に各フェーズを示す．

初期フェーズ

初期フェーズはスケジューリング/FUバインディングで最初に実行され，前回の反復時の配置，電圧におけるスケジューリング/FUバインディングを行う．初期フェーズ，電圧上昇フェーズ，電圧下降フェーズ全てのフェーズで共通する処理を図3.17に示す．初期フェーズでは共通処理のみを実行し，初期フェーズの結果がステップ制約を満たさない場合電圧上昇フェーズへ移行し，ステップ制約を満たす場合は電圧下降フェーズへ移行する．

電圧上昇フェーズ

電圧上昇フェーズは初期フェーズでステップ制約を満たさない場合に実行される．電圧上昇フェーズの目的は初期フェーズ結果がステップ制約を満たさない場合，電圧を上昇することでステップ制約を満たすハドルの電圧を発見することである．電圧上昇フェーズの処理を図3.18に示す．

電圧下降フェーズ

電圧下降フェーズはステップ制約を満たす中で，エネルギーを最小とする電圧を探すフェーズである．電圧下降フェーズの処理を図3.19に示す．

共通処理

1. DT 計算.
2. DT ベースのスケジューリング/FU バインディング [21].
3. ステップ制約を満たす場合，以下の操作で動的エネルギーを最小化.
   (a) 電圧 v_l，v_m の乗算器がある場合以下を実行.
      i. v_m で実行する乗算を 1 つ選択.
      ii. 選択した乗算を v_l で実行することとし，他の演算の電圧は変更せず，DT ベースのスケジューリング/FU バインディング [21] を実行.
      iii. ステップ制約を満たす場合採用し，ステップ制約を満たさない場合電圧を戻す.
      iv. v_m で実行する乗算をすべて試行したら (a) を終了し，そうでない場合 i へ戻る.
   (b) 電圧 v_l，v_h の乗算器がある場合，v_h の乗算に対し (a) と同じ処理を実行.
   (c) 電圧 v_m，v_h の乗算器がある場合，v_h の乗算に対し (a) と同じ処理を実行.
   (d) 電圧 v_l，v_m の加算器がある場合，v_m の加算に対し (a) と同じ処理を実行.
   (e) 電圧 v_l，v_h の加算器がある場合，v_h の加算に対し (a) と同じ処理を実行.
   (f) 電圧 v_m，v_h の加算器がある場合，v_h の加算に対し (a) と同じ処理を実行.

図 3.17: 共通処理.

電圧上昇フェーズ

1. v_l のハドルのうち，最小の優先度を持つハドルを選択，電圧を v_m へ変更
2. 共通処理
3. ステップ制約を満たす場合，終了
   ステップ制約を満たさない場合，v_l のハドルがあれば 1 へ，なければ 4 へ
4. v_m のハドルのうち，最小の優先度を持つハドルを選択，電圧を v_h へ変更
5. 共通処理
6. ステップ制約を満たす場合，もしくは v_m のハドルがない場合，終了
   ステップ制約を満たさない場合，v_m のハドルがあれば 4 へ

図 3.18: 電圧上昇フェーズの処理.

例 3. クロック周期制約 $T_{clk} = 3\text{ns}$，ステップ制約 $S_{max} = 8$，図 3.20 を入力とする．各演算器，レベルコンバータの情報は表 3.1，表 3.2 の値を用いる．図 3.20(b) の配置において，$D_w(Dist(h_1, h_3)) = D_w(Dist(h_1, h_2)) = D_w(Dist(h_2, h_3)) = 1\,\text{ns}$ とし，$D_{reg}(h_1) = D_{reg}(h_2) = D_{reg}(h_3) = 0.5\,\text{ns}$ とする．

まず初期フェーズとなり，DT を作成し DT ベースのスケジューリング/FU バインディングを実行する．図 3.20(c) の通りに DT が計算され，DT を基に図 3.20(a) とスケジューリングされる．図 3.20(a) はステップ制約を満たさないため，電圧上昇フェーズへ移行する．

電圧上昇フェーズでは，まず v_l のハドルのうち最小の優先度となるハドルを選択する．図 3.20 において，v_l のハドルは h_3 のみのため h_3 を選択する．h_3 を v_m へ変更した場合が図 3.21 である．続いて DT を作成し DT ベースのスケジューリング/FU バインディングを実行する．図 3.21(c) の通り

電圧下降フェーズ

1. v_h のハドルのうち，最大の優先度を持つハドルを選択 (ただし，一度選んだハドルは選ばない)，電圧を v_m へ変更
2. 共通処理
3. ステップ制約を満たさない場合，電圧を v_h へ戻す
4. 選択したことのない v_h のハドルがある場合，1へ
5. v_m のハドルのうち，最大の優先度を持つハドルを選択 (ただし，一度選んだハドルは選ばない)，電圧を v_l へ変更
6. 共通処理
7. ステップ制約を満たさない場合，電圧を v_m へ戻す
8. 選択したことのない v_m のハドルがある場合，5へ
   全ての v_m のハドルを選択した場合，終了

図 3.19: 電圧下降フェーズの処理.

DT が計算され，DT を基に図 3.21(a) とスケジューリングされる．全てのハドルが v_m のため，動的エネルギーは図 3.21 が最小である．ステップ制約を満たすため，電圧下降フェーズへ移行する．

電圧下降フェーズでは，まず v_h のハドルのうち最大の優先度となるハドルを選択する．しかし v_h のハドルは存在しないため，v_m のハドルのうち最大の優先度となるハドルを選択する．図 3.21 において，v_m のハドルのうち最大の優先度となるハドルは h_3 である．そのため h_3 を選択し，電圧を v_l へ変更する．DT を作成し DT ベースのスケジューリング/FU バインディングを実行した結果は図 3.20 と同じである．ステップ制約を満たさないため，h_3 の電圧を v_m に戻す．

以降も電圧下降フェーズを行うが，ステップ制約を満たす結果は得られない．ステップ制約を満たす結果のうち，動的エネルギーが最小である図 3.21 が出力となる．

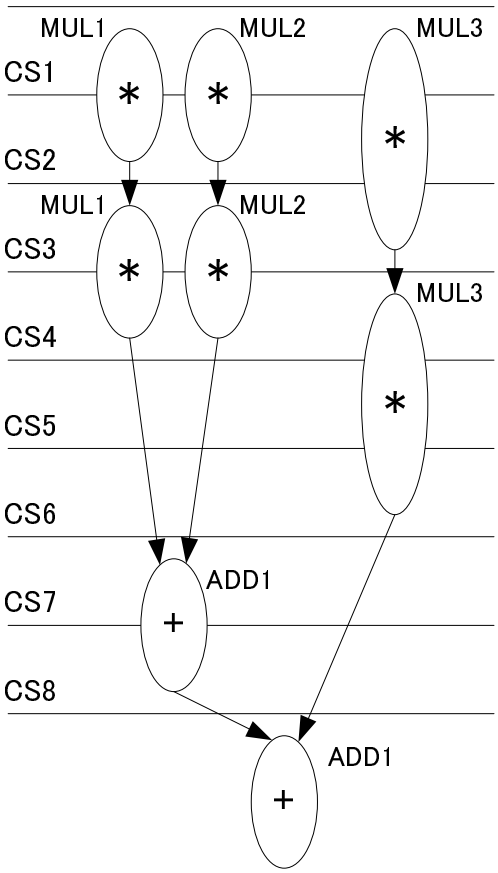
□

レジスタ/コントローラ合成/フロアプラン

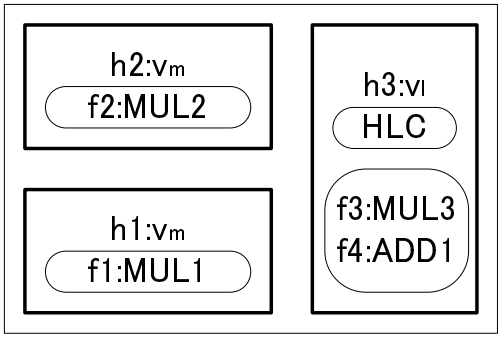
レジスタ/コントローラ合成/フロアプランは，スケジューリング，バインディング結果からハドルのレジスタ，コントローラ構成を決定し，フロアプランする．レジスタ，コントローラの構成の決定は GDR におけるレジスタ/コントローラ合成 [22] を HDR に適用する．フロアプランではデータ構造に Sequence-pair [20] を用い SA(Simulated Annealing) による最適化を行う．その際，電源網 [14] を考慮したフロアプランを行う．図 3.16 のレジスタ/コントローラ合成/フロアプラン (1)，レジスタ/コントローラ合成/フロアプラン (2) は同じ処理を実行する．

Sequence-pair を用い SA によりフロアプランする場合 [20]，通常以下の 3 つの操作を行う．

move 1　2 要素を選択し，Γ_+ で入れ替える．



(a) CDFG.



(b) 配置情報.

	h1	h2	h3
f1	0	1	2
f2	1	0	2
f3	2	2	0
f4	2	2	0

(c) データ転送テーブル.

図 3.20: スケジューリングの入力.

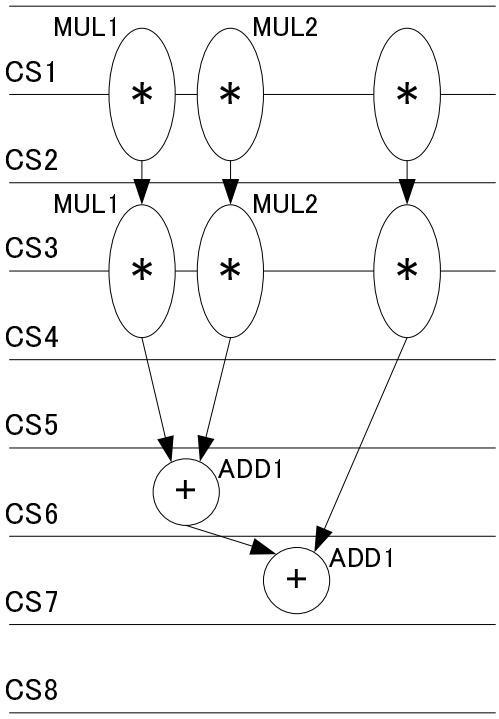
move 2　2 要素を選択し，Γ_+, Γ_- で入れ替える．

move 3　1 要素を選択し，90° 回転させる．

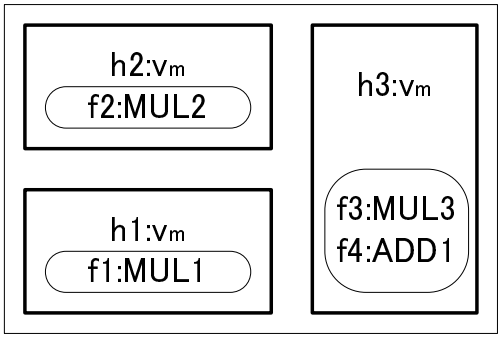
提案手法で HDR のハドルをフロアプランの対象とする．ハドルはクロック周期制約の範囲内で任意の矩形を取るため，move 3 をハドルの変形操作に変更する．

ハドルの変形操作を考えたとき，以下の 2 つの手法がある．

1. 自由に変形する．



(a) CDFG.



(b) 配置情報.

	h1	h2	h3
f1	0	1	1
f2	1	0	1
f3	1	1	0
f4	1	1	0

(c) データ転送テーブル.

図 3.21: スケジューリングの出力.

2. 事前に設定した形から選択する.

手法1は自由に変形するため，面積の利用効率を最大とできる利点がある．一方，自由に変形する場合，変形の度合いを決める方法が問題となる．ハドルの周囲の長さはクロック周期制約により制限され，制限を超えるハドルは作ることができない．そのため，変形の度合いはクロック周期や，内部の構成など多くの要素で決められる．変形操作には多くの考慮すべき要素があり，SA と相性が良くない．

手法2は事前に形を決めるため，面積の利用効率を最大とできない欠点がある．一方，事前に形を決めるため，クロック周期違反を起こすことはない．そのため形の選択は操作が容易で，SA と相性が良い．以上より，提案手法では手法2を採用し，move 3 を以下の move 3'

に変更する．

move 3' 1要素を選択し，アスペクト比を変更する．

アスペクト比はクロック周期制約などから事前に設定する．

SA のコスト関数 $cost$ を以下に示す．

$$cost = \frac{A_{BB}}{A_{total}} + \alpha\frac{V}{T_{clock}} + \beta\frac{W}{W_{MAX}} + \gamma\frac{A_{PNR}}{A_{total}} \tag{3.13}$$

ただし，α，β，γ は，任意のパラメータである．A_{BB} は最小矩形の面積，A_{total} はモジュール面積の総計，T_{clock} はクロック周期，V はクロック周期制約違反の総合計，W は配線長，W_{MAX} は (最小矩形の縦+横)× 配線数で計算される配線長の最大値，A_{PNR} は各電圧における電源網の面積の総和である．

フロアプランでは，i 回目のイタレーションにおける最終的な配置結果を，$i+1$ 回目のイタレーションのフロアプランにおける SA の初期配置として用いる．イタレーションごとに SA の初期温度を下げていく．合成フローの i 回目のイタレーションの SA の初期温度を T_i としたとき $i+1$ 回目のイタレーションの初期温度を

$$T_{i+1} = T_i/K \tag{3.14}$$

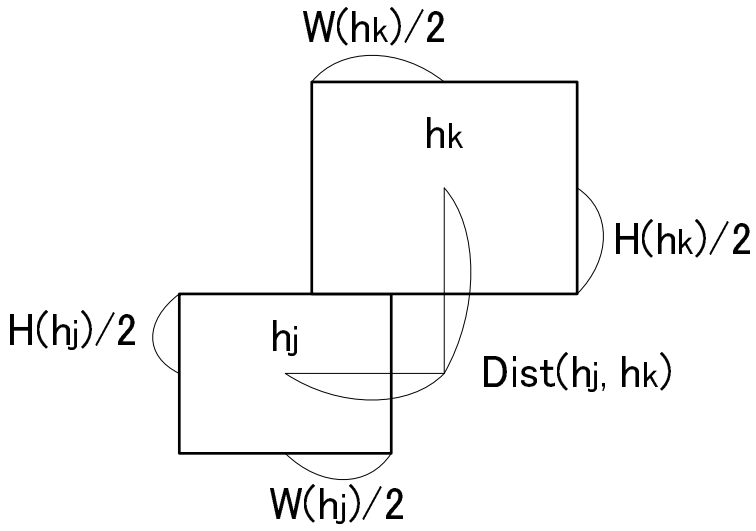
とする．ただし K>1 とする．

ハドル合成

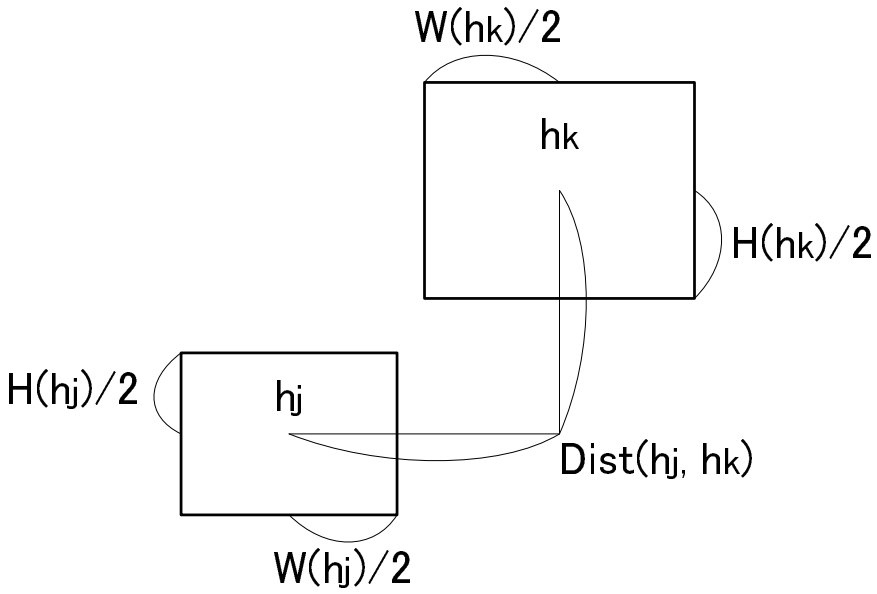
ハドル合成はレジスタ/コントローラ合成/フロアプランにより決定したハドルの配置を入力とし，ハドルの併合結果を出力する．ハドルの併合操作の際，フロアプラン結果を利用する．フロアプランでは式 (3.13) でコストを計算するため，1つのハドルに併合すべきハドルが近くに配置される．

併合すべきハドルを決定するため，ハドル h_j と $h_k(j \neq k)$ の隣接度合い $Adj(h_j, h_k)$ を以下のように設定する．

$$\begin{aligned} Adj(h_j, h_k) &= \left[\frac{H(h_j)}{2} + \frac{H(h_k)}{2}\right] \\ &+ \left[\frac{W(h_j)}{2} + \frac{W(h_k)}{2}\right] \\ &- Dist(h_j, h_k) \end{aligned} \tag{3.15}$$



(a) ハドル h_j と h_k が隣接している場合.



(b) ハドル h_j と h_k が離れている場合.

図 3.22: 隣接度 $Adj(h_j, h_k)$ の例.

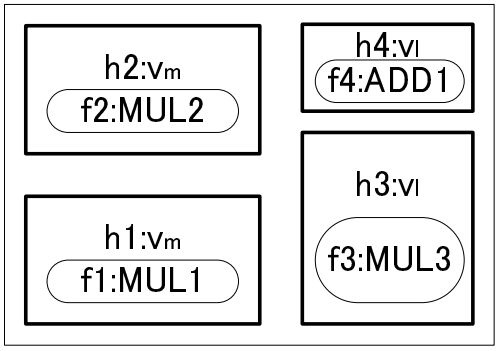
$Adj(h_j, h_k)$ はハドルが接している場合正の値を取り，接していない場合負の値を取る．図 3.22(a) はハドルが接している場合で,

$$Adj(h_j, h_k) \geq 0$$

となる．一方，図 3.22(b) はハドルが接していない場合で,

$$Adj(h_j, h_k) < 0$$

となる．続けて，ハドル h_j と h_k の接続数を表す $HC(h_j, h_k)$ を定義する．$HC(h_j, h_k)$ は常に 0 または正の値を取る.



(a) 配置情報.

	h1	h2	h3	h4
h1		15	10	−10
h2	15		5	10
h3	10	5		15
h4	−10	10	15	

(b) 優先度.

図 3.23: ハドル合成の入力.

$Adj(h_j, h_k)$ と $HC(h_j, h_k)$ から優先度 $P_h(h_j, h_k)$ を以下の式で求める.

$$P_h(h_j, h_k) = Adj(h_j, h_k) \cdot HC(h_j, h_k) \tag{3.16}$$

$P_h(h_j, h_k)$ は2つのハドルが接している場合のみ0または正の値を持ち，離れている場合は負の値を持つ．$P_h(h_j, h_k)$ の内，正の最大値を持つハドルの組について併合条件を調べる．併合条件を満たすハドルについて併合を行う．併合条件を以下に示す.

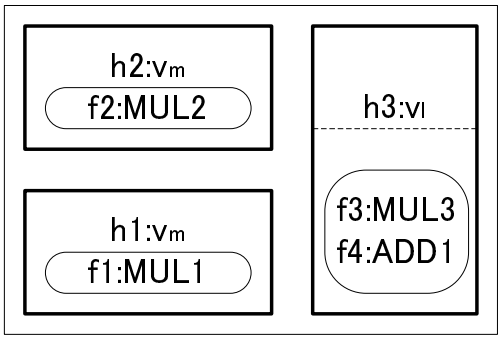
$$\begin{cases} V(h_j) = V(h_k) \\ \\ h_j \text{と} h_k \text{を併合したハドルがハドルサイズ制約を} \\ \text{満たすこと} \end{cases}$$

併合条件を満たすハドルを繰り返し探し，併合できるハドルがなくなった時点でハドル合成は終了する.

ハドル合成ではレジスタ/コントローラ合成/フロアプラン (1) の結果から，併合条件を満たすハドル全てを併合することを目指している．ハドル合成では併合したハドルの形や，併合の結果起こるハドル重なりについては考慮しない．あるハドルの組を併合後にハドルの形を考慮し，重なりをなくすと，併合条件を満たさなくなるハドルの組み合わせが出る可能性がある．そのため併合の順序により，併合できないハドルの組み合わせが出る．併合条件を満たす全ハドルを併合するため，ハドル合成では併合によるハドルの形や重なりは考慮せず，レジスタ/コントローラ合成/フロアプラン (2) でハドルの形と重なりを考慮する.

例 4. 図3.23の入力に対しハドル合成を行う場合を考える．優先度 P_h が最大の h_3, h_4 の組が選ばれる．h_3, h_4 ともに v_l でハドルサイズ制約を満たすため，ハドルを併合できる.

他のハドルについても優先度順に併合可能か調べるが，h_1 と h_2 はハドルサイズ制約を満たさず，他の組み合わせは電圧が異なるため併合できない．以上より，h_3 と h_4 を併合し，新たな h_3 が作成さ



(a) 配置情報.

	h1	h2	h3
h1		15	10
h2	15		10
h3	10	10	

(b) 優先度.

図 3.24: ハドル合成の出力.

れた図 3.24 が出力となり，ハドル合成は終了となる．

ハドル分割

ハドル合成ではハドルを併合したが，局所的な解に陥らないため，逆の手順が必要となる．ハドル合成の逆手順となるのがハドル分割である．

ハドル分割もフロアプラン結果を利用する．演算器 f_i からハドル h_k に対する $DT(f_i, h_k)$ について考える．スケジューリング/FU バインディング時の $DT(f_i, h_k)$ を $DT_s(f_i, h_k)$，フロアプラン後の $DT(f_i, h_k)$ を $DT_f(f_i, h_k)$ とする．

$$DT_s(f_i, h_k) < DT_f(f_i, h_k) \tag{3.17}$$

式 (3.17) が成り立つ場合，f_i から h_k へのデータ転送はクロック周期違反を起こす可能性がある．そのため f_i のハドル $h_j(= Hud(f_i))$ への割り当てがクロック周期違反の原因と考える．そこで f_i を h_j から取り除き，新たなハドル h_l を作成し，f_i を h_l へ割り当てる．h_l は h_k と重なる位置に配置する．新たな反復のレジスタ/コントローラ合成/フロアプラン (1) で h_l のレジスタ，コントローラが求まり，ハドルの重なりもなくなる．

ハドル分割において，式 (3.17) となる演算器とハドルの組み合わせがない場合，フロアプラン結果はスケジューリング/FU バインディング結果を満足する．この時解は収束したと考え，反復改良を終了する．

3.6 本章のまとめ

本章では，フロアプラン指向の高位合成手法として，まず GDR アーキテクチャ，RDR アーキテクチャおよびその派生アーキテクチャを紹介した．既存のレジスタ分散型アーキテクチャの欠点を考慮し，低電力化することを視野に入れた配線遅延を考慮するレジスタ分散型アーキテクチャとして HDR アーキテクチャを提案した．HDR は GDR に区画を導入し，演算器，レジスタ，コントローラを抽象化したアーキテクチャである．ハドルは HDR における区画で，演算器群，ローカルレジスタ，コントローラ，レベルコンバータのモジュールを含む．提案した HDR を対象に高位合成問題を定式化し，消費エネルギー削減を目的としたアルゴリズムを提案した．提案手法では配置情報をフィードバックし，反復改良する合成フローを取る．その際，近くに配置されたモジュールをハドルとして扱い，レジスタ，コントローラ，レベルコンバータを共有する．ハドル内の演算はモジュール間が十分近いため配線遅延を無視でき，ハドル間データ通信をする場合レジスタ-レジスタ間通信を行う．電源電圧の操作はハドルを単位として行い，資源制約と時間制約から，クリティカルパスでないハドルを低電圧に割り当てることで低電力化する．

第4章

計算機実験

4.1 本章の概要

本章では，第 3 章で提案したアルゴリズムを計算機上に実装し，実験評価する．入力アプリケーションとして EWF3，FIR，DCT を用い，制約条件を変え合計 5 種類の入力を与え実験評価した．既存のレジスタ分散型アーキテクチャ，単一電源電圧のみの HDR アーキテクチャ，既存手法と組み合わせ複数電源電圧とした HDR アーキテクチャ，提案手法を比較する．

4.2 実験方法

計算機実験に用いた計算機環境は以下の通り．

OS	CentOS 5.5
COMP	PC/AT 互換機
CPU	AMD Opteron 2360SE 2.5GHz ×2
Memory	16GB

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 SE 2.5 GHz × 2，メモリ容量が 16 GB である．対象アプリケーションとして DCT(ノード数 48)，EWF3(ノード数 102)，7 次 FIR フィルタ (ノード数 75) を用いた．使用する電圧は $v_h = 1.2\,\mathrm{V}$，$v_m = 1.0\,\mathrm{V}$，$v_l = 0.8\,\mathrm{V}$ とした．

実験で用いた演算器情報を表 4.1 に示す．各演算器は 16 bit 幅と仮定し，1.0 V の演算器の情報を事前に Synopsys 社の Design Compiler により論理合成して求めた．その際，セルライブラリとして CMOS 90nm の設計ルールを用いた．1.2V，0.8V の演算器の情報は，1.0V における情報にスケーリング則 [19] を適用し算出した．

レベルコンバータの情報を表 4.2 [31] に示す．本手法で用いるレベルコンバータの構成を図 4.1 [31] に示す．表 4.2 における面積は，図 4.1 の構成におけるトランジスタ数から算出した．遅延およびエネルギーは，[31] の情報にスケーリング則 [19] を適用し算出した．

コントローラの面積は Synopsys 社の Design Compiler により実際に論理合成して求めた．配線遅延は配線長の 2 乗に比例すると仮定し，250 $\mu\mathrm{m}^2$ あたり 1 ns とする．消費エネルギーは Synopsys 社の Design Compiler の論理合成結果をもとに導出した．

GDR を対象とした高位合成手法 [22]，RDR/MCAS [5]，1 電源の HDR，1 電源の HDR として合成した後 [41] を適用し複数電源電圧とした HDR，[40] を適用し演算に電圧を割り当てた後 1 電源の場合と同様に HDR を合成したもの (以下 [40]+HDR)，提案手法を適用した MHDR(Multiple supply voltages HDR) を比較する．提案手法と MCAS（RDR）を比較するうえで，制約を合わせるために MCAS を一部変更した．クロック周期制約は全て 2.5 ns とする．

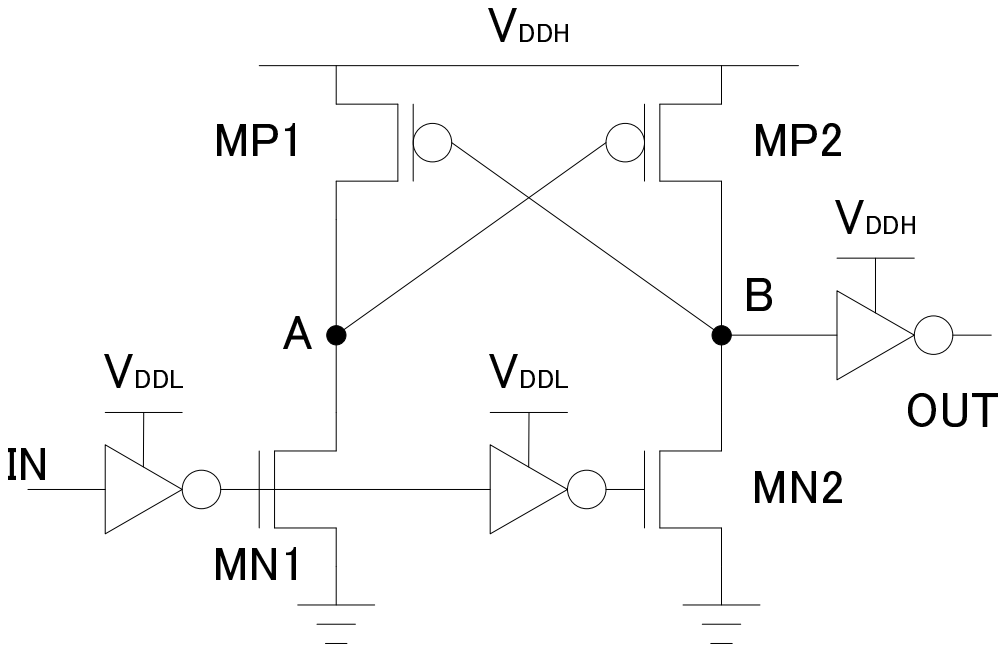


図 4.1: レベルコンバータの構成.

4.3 実験結果および考察

実験結果を表 4.3 に示す.

実面積を比較したグラフを図 4.2 に示す．提案手法は従来手法と比較し，平均 5.5 % 面積が増加している．しかし，[40] + HDR と比較すると同程度の面積である．原因は複数電源電圧適用によるレベルコンバータの増加と考えられる.

最小矩形の面積を比較したグラフを図 4.3 に示す．提案手法は従来手法と比較し，平均 1.6 % 面積が増加している．こちらは実面積の増加と比較し，影響は小さくなっている．提案手法は複数電源電圧適用により，それぞれのハドルの大きさが抑えれ数が多くなっている．それぞれのハドルが小さい方がフロアプランした際の最小矩形は小さくなると考えられる.

動的エネルギーを比較したグラフを図 4.4 に示す．提案手法は従来手法と比較し，最大 52.6 % ，平均 28.7 % 動的エネルギーを削減できている．複数電源電圧を適用したもののみを比較しても，提案手法が最も動的エネルギーを削減できている．HDR + [41] と比較し，大幅に動的エネルギーを削減できているのは，提案手法の方が低い電圧を割り当てることのできたハドルが多いためである．これにより，タイミングが決定する前の高位合成の段階で複数電源電圧を考慮することの有効性が確かめられた．[40] + HDR と比較し，動的エネルギーを削減できているのは，配線遅延を考え低い電圧を割り当てることのできた演算が増えたためである．これにより，消費エネルギー削減には配線遅延を同時に考慮することが必要であることが分かる.

静的エネルギーを比較したグラフを図 4.5 に示す．提案手法は従来手法と比較し，最大 64.7 % 静的エネルギーを削減できたが，平均 27.0 % 静的エネルギーが増加している．しかし，[40] + HDR と比較すると同程度の静的エネルギーである．原因は複数電源電圧適用によるレベルコンバータの増加と考えられる.

全エネルギーを比較したグラフを図 4.6 に示す．提案手法は従来手法と比較し，最大 45.1 % ，平均 23.1 % 全エネルギーを削減できている．複数電源電圧を適用したもののみを比較しても，提案手法が最も全エネルギーを削減できている．HDR + [41] と比較し，大幅に全エネルギーを削減できているのは，提案手法の方が低い電圧を割り当てることのできたハドルが多いためである．これにより，タイミングが決定する前の高位合成の段階で複数電源電圧を考慮することの有効性が確かめられた．[40] + HDR と比較し，全エネルギーを削減できているのは，配線遅延を考え低い電圧を割り当てることのできた演算が増えたためである．これにより，消費エネルギー削減には配線遅延を同時に考慮することが必要であることが分かる．動的エネルギーと静的エネルギーを比較すると，動的エネルギーの方が全体に占める割合が大きいため，動的エネルギーが最小な提案手法が全エネルギーに関しても最小である.

今後の課題点は静的エネルギーの削減である．今回はエネルギーに関して 90nm のプロセ

表 4.1: 演算器の情報.

演算器 (電圧)	面積 [μm^2]	遅延 [ns]	エネルギー [fJ]	リーク電力 [μW]
加算器 (1.2 V)	386	0.75	92	3.9
加算器 (1.0 V)	386	1.22	64	3.2
加算器 (0.8 V)	386	2.71	41	2.6
乗算器 (1.2 V)	2161	1.65	1135	19.8
乗算器 (1.0 V)	2161	2.7	788	16.5
乗算器 (0.8 V)	2161	6.0	504	13.2

表 4.2: レベルコンバータの情報 [31].

V_{in} - V_{out}	面積 [μm^2]	遅延 [ns]	エネルギー [fJ]	リーク電力 [μW]
1.2 V - 1.0 V	113	0.17	83	49.1
1.2 V - 0.8 V	113	0.22	71	32.3
1.0 V - 1.2 V	113	0.17	76	45.0
1.0 V - 0.8 V	113	0.30	55	18.3
0.8 V - 1.2 V	113	0.22	86	39.1
0.8 V - 1.0 V	113	0.30	55	18.3

スを適用した．更に細かいプロセスとなると静的エネルギーの増加が重大となる．動的エネルギーの削減以上に静的エネルギーが増加する場合，提案手法の有効性が問われる．更なるプロセスの微細化へ向け，静的エネルギーの削減まで考慮した高位合成手法が必要となる．

表 4.3: 計算機実験結果.

APP.	FUs	Step	Architechture	Area [μm^2]	Rectangular Area[μm^2]	Dynamic Energy [pJ]	Leak Energy [pJ]	All Energy [pJ]
ewf3	+3,*2	50	GDR	47792	55250	414.34	57.64	471.98
			RDR	69530	69530	362.35	85.06	447.41
			HDR	52506	62252	331.95	77.96	409.91
			HDR + [41]	53405	66196	267.02	99.47	366.49
			[40] + HDR	58090	74088	249.43	109.75	359.18
			MHDR	47002	49400	241.11	87.53	328.64
fir	+3,*3	35	GDR	36840	48278	223.34	38.35	261.69
			RDR	81920	81920	191.34	105.18	296.52
			HDR	30082	35910	191.65	31.61	223.26
			HDR + [41]	30082	35910	191.65	31.61	223.26
			[40] + HDR	48774	72732	131.27	71.36	202.63
			MHDR	52051	61500	122.22	60.04	182.26
fir	+4,*4	30	GDR	34540	36000	211.94	44.00	255.94
			RDR	82816	82816	188.83	123.57	312.40
			HDR	32047	35624	207.59	33.05	240.64
			HDR + [41]	32047	35624	207.59	33.05	240.64
			[40] + HDR	46274	55488	138.37	58.70	197.07
			MHDR	44554	48174	128.59	43.60	172.19
dct	+3,*3	15	GDR	48551	58843	181.96	13.56	195.52
			RDR	59544	59544	144.15	15.99	160.13
			HDR	47321	63096	127.58	17.42	145.00
			HDR + [41]	48076	55770	112.25	25.10	137.35
			[40] + HDR	50639	61915	86.17	22.70	108.88
			MHDR	51269	55221	86.17	21.17	107.34
dct	+4,*4	10	GDR	53864	58378	152.01	11.00	163.01
			RDR	81476	81476	127.87	13.69	141.57
			HDR	58184	71520	128.71	13.69	145.22
			HDR + [41]	58184	71520	128.71	13.69	145.22
			[40] + HDR	52104	53824	94.19	21.98	116.17
			MHDR	58272	68255	89.11	25.37	114.48

4.4 本章のまとめ

本章では，計算機実験を通して提案手法を評価した．提案手法は従来手法と比較し，最大 45.1 % ，平均 23.1 % 全エネルギーを削減できた．この結果より提案手法は消費エネルギーと配線遅延の双方の考慮ができていると分かる．今後の課題としては静的エネルギー削減があげられる．

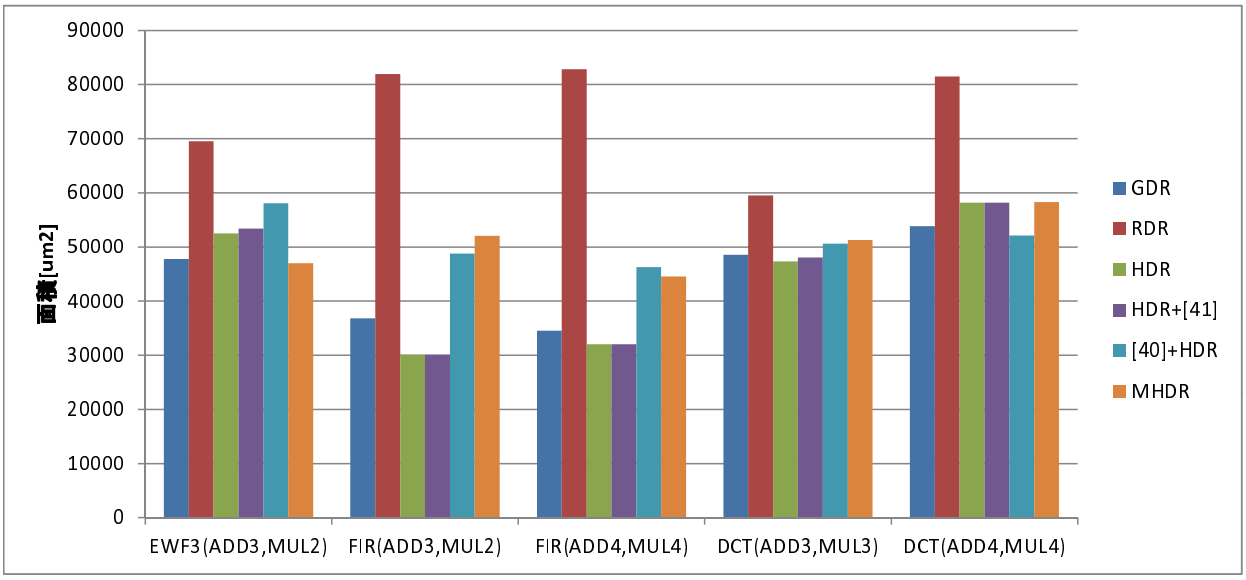


図 4.2: 実面積の比較.

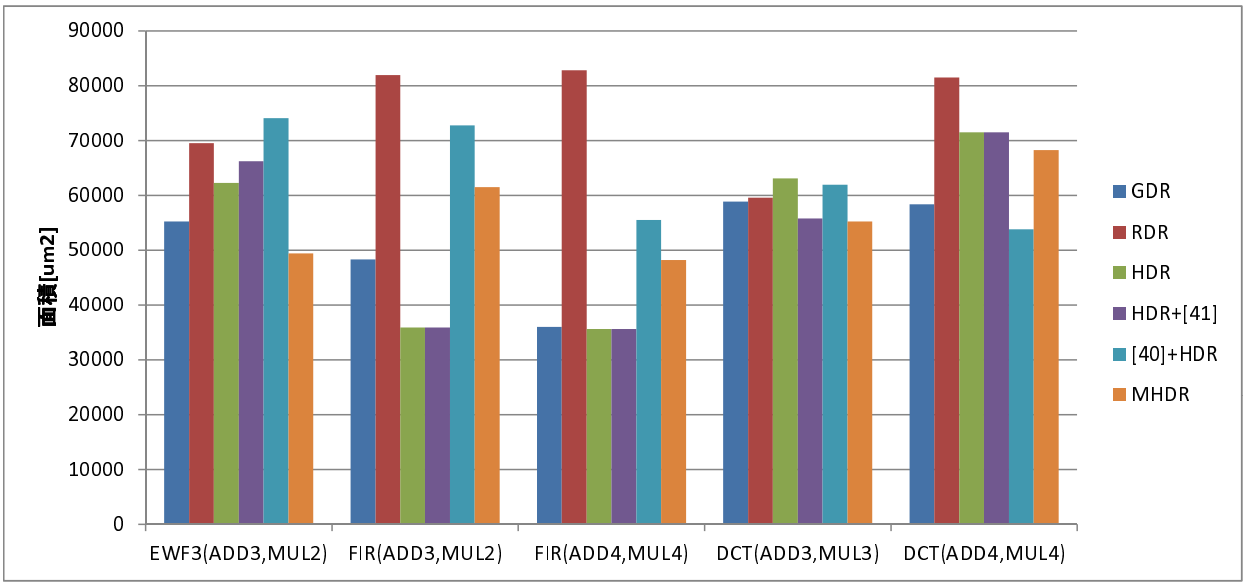


図 4.3: 最小矩形の面積の比較.

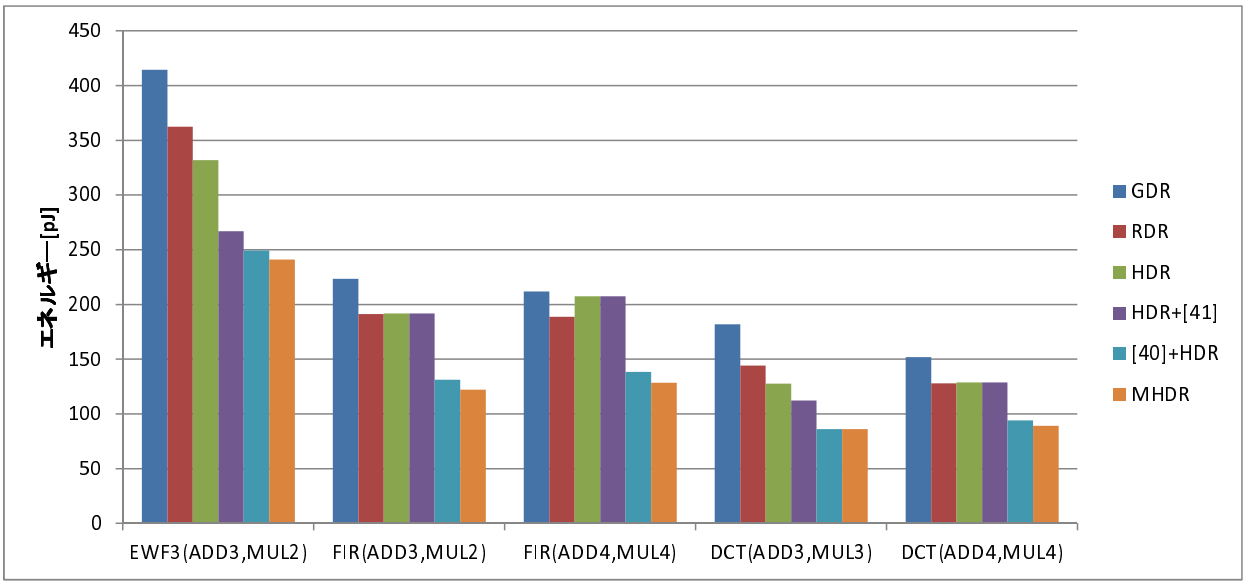


図 4.4: 動的エネルギーの比較.

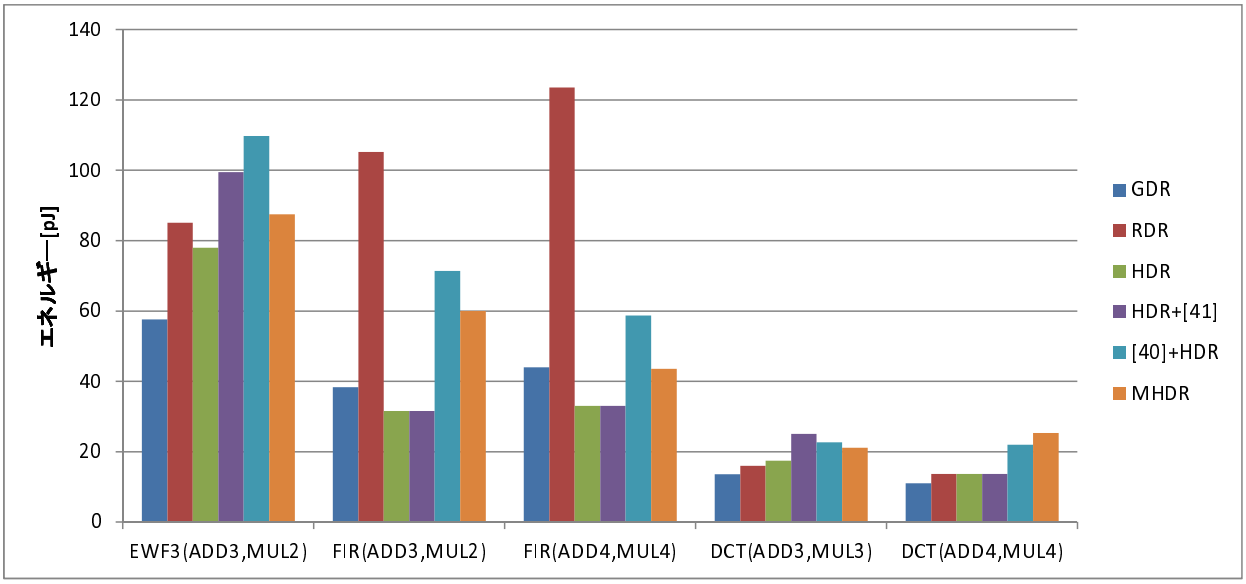


図 4.5: 静的エネルギーの比較.

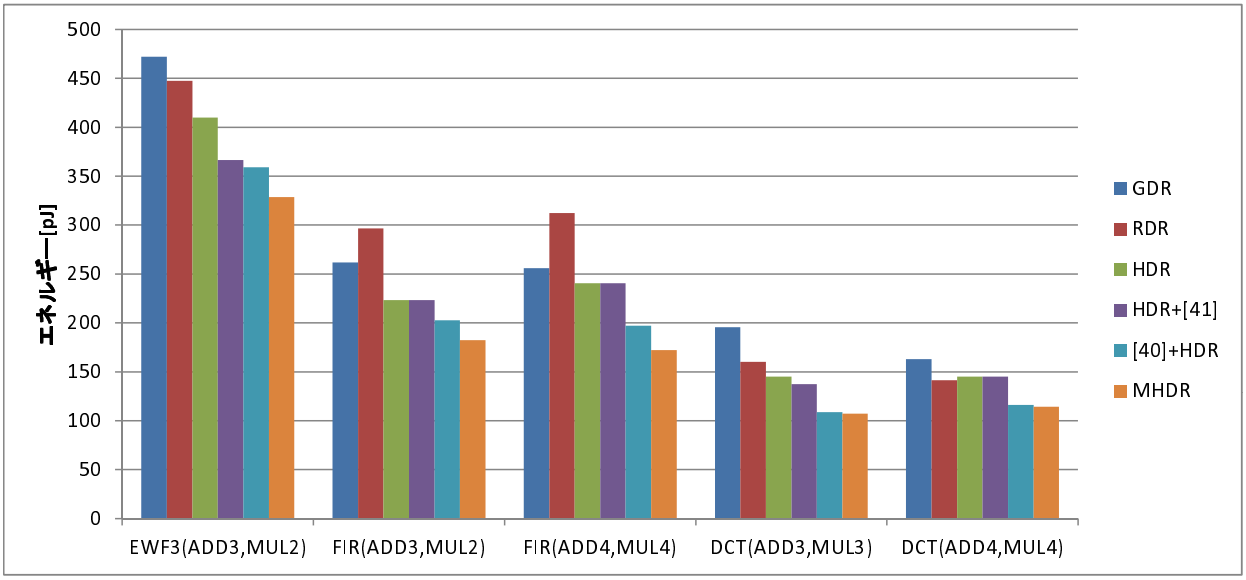


図 4.6: 全エネルギーの比較.

第5章

結論

本論文では，低電力化することを視野に入れた配線遅延を考慮するレジスタ分散型のアーキテクチャとして Huddle-based Distributed-Register アーキテクチャ(以下 HDR) を提案した．続いて，HDR を対象とした複数電源電圧による低電力化高位合成手法を提案した．提案手法は複数電源電圧と HDR による複数サイクルレジスタ間通信を扱うことで，消費エネルギーと配線遅延を同時に扱うことができる．計算機実験結果により，提案手法は従来手法と比較し全消費エネルギーを最大 45.1 % 削減できることを確認できた．

第 2 章「**複数電源電圧指向の高位合成手法の研究動向**」では，低電力化を目的とし複数電源電圧の適用を考慮した高位合成手法を報告した．複数電源電圧は高位合成段階で適用することが消費エネルギー削減において有効であることを示した．その中で既存の複数電源電圧指向の高位合成手法は配線遅延を考慮しておらず，配線遅延が支配的となる条件では消費エネルギー削減に支障があることを示した．

第 3 章「**Huddle-based Distributed-Register アーキテクチャを対象とする低電力化高位合成手法**」では，まずフロアプラン指向の高位合成手法を報告した．フロアプラン指向の高位合成手法として，GDR を対象とした高位合成手法，RDR および RDR の派生アーキテクチャを対象とした高位合成手法を紹介した．しかし，GDR はモジュールの追加が難しく，RDR は面積が大きいという欠点を示した．それぞれの欠点から，既存のレジスタ分散型アーキテクチャは低電力化に向かないことを示した．既存手法を受け，低電力化することを視野に入れた配線遅延を考慮するレジスタ分散型のアーキテクチャである HDR を提案した．HDR はハドル導入によるモジュールの抽象化で，RDR と同様にモジュールを容易に追加できる．ハドルはクロック周期により決定される範囲内において任意の矩形を取り，演算器やレジスタ，コントローラ，レベルコンバータを共有する．任意の矩形を取るハドルに対しフロアプランするため，HDR は GDR と同様に小面積で消費電力の少ないアーキテクチャとなる．続いて，HDR を対象とした複数電源電圧による低電力化高位合成手法を提案した．提案手法では配置情報をフィードバックし，反復改良する合成フローを取る．その際，近くに配置されたモジュールをハドルとして扱い，レジスタ，コントローラ，レベルコンバータを共有する．ハドル内の演算はモジュール間が十分近いため配線遅延を無視でき，ハドル間データ通信をする場合レジスタ-レジスタ間通信を行う．電源電圧の操作はハドルを単位として行い，資源制約と時間制約から，クリティカルパスでないハドルを低電圧に割り当てることで低電力化する．

第 4 章「**計算機実験結果**」では，提案手法を実装し評価した．計算機実験により提案手法は従来手法と比較し全消費エネルギーを最大 45.1 % 削減できることを確認できた．実験結果より，提案手法は消費エネルギーと配線遅延を同時に考慮できる高位合成手法であることを確認できた．

最後に HDR アーキテクチャを対象とした低電力化高位合成に関する今後の課題について

考察する．提案手法ではクロック周期制約が短くなった場合，アルゴリズムの収束性が悪化し収束に必要な反復回数が増加する．反復数が増加すると実際よりも配線遅延を多く見積もることとなり，処理における配線遅延の割合が増加する．配線遅延の増加は回路のクリティカルパスの増加につながり，低電力化の障害となる．アルゴリズムの収束性を向上し，クロック周期制約が短くなった場合も反復数が少なく収束するアルゴリズムの提案が今後の課題である．アルゴリズムを改良できれば，配線遅延を考慮しないアルゴリズムと比較し更に消費エネルギーを削減できる．また，本論文では回路の集積度が増し，回路全体の遅延において配線遅延が支配的となる場合を想定している．集積度が増す場合，クロック周期の短縮予想される．クロック周期の短縮は常にクロックの入力があるレジスタの消費エネルギー増加につながる．レジスタの消費エネルギー削減まで考慮した高位合成アルゴリズムの提案も今後の課題である．レジスタの消費エネルギー削減の手法としてはクロックゲーティング [12] がある．クロックゲーティングを考慮した高位合成アルゴリズムの提案が考えられる．集積度が増した場合，同時に問題となるのが静的電力の増加である．静的電力の削減まで考慮した高位合成アルゴリズムの提案も今後の課題である．静的電力を削減する手法としてパワーゲーティング [34–36] がある．パワーゲーティングを考慮した高位合成アルゴリズムの提案が考えられる．

謝辞

本論文全般にわたり，御指導ならびに御助言を授かった戸川望教授，柳澤政生教授，大附辰夫教授に深く感謝いたします.

また，終始，適切な御指導および御助言を頂きました本学博士の大智輝氏，および本学修士課程の砂田翔平氏，田中翔氏に深く感謝いたします.

最後に，本論文に関する研究活動全般にわたり支援していただいた戸川研究室，柳澤研究室，大附研究室の皆様に感謝いたします.

参考文献

[1] J. An, H. Park, and Y. Kim, "Level up/down converter with single power-supply voltage for multi-vdd systems," *J. of Semiconductor Technology and Science*, vol. 10, no. 1, pp. 55–60, 2010.

[2] J. Carballo, J. Burns, S. Yoo, I. Vo, and V. Norman, "A semi-custom voltage-island technique and its application to high-speed serial links," in *Proc. of ISLPED '03*, pp. 60–65, 2003.

[3] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE J. of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[4] J. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Trans. on VLSI Systems*, vol. 5, no. 4, pp. 436–443, 1997.

[5] J. Cong, Y. Fan, G. Han, X. Yang, and Z. Zhang, "Architecture and synthesis for on-chip multicycle communication," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 4, pp. 550–564, 2004.

[6] J. Cong, Y. Fan, and W. Jiang, "Platform-based resource binding using a distributed register-file microarchitecture," in *Proc. of ICCAD '06*, pp. 709–715, 2006.

[7] J. Cong, Y. Fan, and J. Xu, "Simultaneous resource binding and interconnection optimization based on a distributed register-file microarchitecture," *ACM Trans. on TODAES*, vol. 14, no. 3, pp. 1–31, 2009.

[8] M. Igarashi, K. Usami, K. Nogami, F. Minami, Y. Kawasaki, T. Aoki, M. Takano, C. Mizuno, T. Ishikawa, M. Kanazawa, *et al.*, "A low-power design method using multiple supply voltages," in *Proc. of ISLPED '97*, pp. 36–41, 1997.

[9] J. Jeon, D. Kim, D. Shin, and K. Choi, "High-level synthesis under multi-cycle interconnect delay," in *Proc. of ASP-DAC '01*, pp. 662–667, 2001.

[10] M. Johnson and K. Roy, "Datapath scheduling with multiple supply voltages and level converters," *ACM Trans. on TODAES*, vol. 2, no. 3, pp. 227–248, 1997.

[11] D. Kim, J. Jung, S. Lee, J. Jeon, and K. Choi, "Behavior-to-placed rtl synthesis with performance-driven placement," in *Proc. of ICCAD '01*, pp. 320–325, 2001.

[12] T. Kitahara, F. Minami, T. Ueda, K. Usami, S. Nishio, M. Murakata, and T. Mitsuhashi, "A clock-gating method for low-power lsi design," in *Proc. of ASP-DAC '98*, pp. 307–312, 1998.

[13] A. Kumar and M. Bayoumi, "Multiple voltage-based scheduling methodology for low power in the high level synthesis," in *Proc. of ISCAS 1999*, vol. 1, pp. 371–374, 1999.

[14] W. Lee, H. Liu, and Y. Chang, "Voltage island aware floorplanning for power and timing optimization," in *Proc. of ICCAD '06*, pp. 389–394, 2006.

[15] K. Lim, Y. Kim, and T. Kim, "Interconnect and communication synthesis for distributed register-file microarchitecture," *IET Computers & Digital Techniques*, vol. 3, no. 2, pp. 162–174, 2009.

[16] Y. Lin, C. Hwang, and A. Wu, "Scheduling techniques for variable voltage low power designs," *ACM Trans. on TODAES*, vol. 2, no. 2, pp. 81–97, 1997.

[17] A. Manzak and C. Chakrabarti, "A low power scheduling scheme with resources operating at multiple voltages," *IEEE Trans. on VLSI Systems*, vol. 10, no. 1, pp. 6–14, 2002.

[18] S. Mohanty and N. Ranganathan, "Simultaneous peak and average power minimization during datapath scheduling," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 52, no. 6, pp. 1157–1165, 2005.

[19] S. Mohanty, N. Ranganathan, and V. Krishna, "Datapath scheduling using dynamic frequency clocking," in *Proc. of ISVLSI '02*, pp. 58–63, 2002.

[20] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996.

[21] A. Ohchi, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "High-level synthesis algorithms with floorplaning for distributed/shared-register architectures," in *Proc. of VLSI-DAT 2008*, pp. 164–167, 2008.

[22] A. Ohchi, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Floorplan-aware high-level synthesis for generalized distributed-register architectures," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 92, no. 12, pp. 3169–3179, 2009.

[23] P. Paulin and J. Knight, "Force-directed scheduling for the behavioral synthesis of asics," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, pp. 661–679, 1989.

[24] S. Qureshi and K. Sanjeev, "Power and performance optimization using multi-voltage, multi-threshold and clock gating for low-end microprocessors," in *Proc. of TENCON 2009*, pp. 1–6, 2009.

[25] S. Raje and M. Sarrafzadeh, "Variable voltage scheduling," in *Proc. of ISLPED '95*, pp. 9–14, 1995.

[26] M. Sarrafzadeh and S. Raje, "Scheduling with multiple voltages under resource constraints," in *Proc. of ISCAS 1999*, vol. 1, pp. 350–353, 1999.

[27] D. Sengupta and R. Saleh, "Application-driven voltage-island partitioning for low-power system-on-chip design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 3, pp. 316–326, 2009.

[28] I. Shin, S. Paik, and Y. Shin, "Register allocation for high-level synthesis using dual supply voltages," in *Proc. of DAC '09*, pp. 937–942, 2009.

[29] W. Shiue and C. Chakrabarti, "Low-power scheduling with resources operating at multiple voltages," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 6, pp. 536–543, 2000.

[30] S. Tanaka, M. Yanagisawa, T. Ohtsuki, and N. Togawa, "A fault-secure high-level synthesis algorithm for rdr architectures," *IPSJ Trans. on System LSI Design Methodology*, vol. 4, pp. 150–165, 2011.

[31] C. Tran, H. Kawaguchi, and T. Sakurai, "Low-power high-speed level shifter design for block-level dynamic voltage scaling environment," in *Proc. of ICICDT 2005*, pp. 229–232, 2005.

[32] K. Usami, K. Nogami, M. Igarashi, F. Minami, Y. Kawasaki, T. Ishikawa, M. Kanazawa, T. Aoki, M. Takano, C. Mizuno, *et al.*, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," in *Proc. of CICC '97*, pp. 131–134, 1997.

[33] K. Usami, M. Igarashi, F. Minami, T. Ishikawa, M. Kanzawa, M. Ichida, and K. Nogami, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," *IEEE J. of Solid-State Circuits*, vol. 33, no. 3, pp. 463–472, 1998.

[34] K. Usami and N. Ohkubo, "A design approach for fine-grained run-time power gating using locally extracted sleep signals," in *Proc. of ICCD '06*, pp. 155–161, 2006.

[35] K. Usami, T. Shirai, T. Hashida, H. Masuda, S. Takeda, M. Nakata, N. Seki, H. Amano, M. Namiki, M. Imai, *et al.*, "Design and implementation of fine-grain power gating with ground bounce suppression," in *Proc. of VLSID 2009*, pp. 381–386, 2009.

[36] K. Usami, T. Hashida, S. Koyama, T. Yamamoto, D. Ikebuchi, H. Amano, M. Namiki, M. Kondo, and H. Nakamura, "Adaptive power gating for function units in a microprocessor," in *Proc. of ISQED 2010*, pp. 29–37, 2010.

[37] K. Wakabayashi and T. Yoshimura, "A resource sharing and control synthesis method for conditional branches," in *Pro. of ICCAD '89*, pp. 62–65, 1989.

[38] X. Xing and C. Jong, "Floorplan-driven multivoltage high-level synthesis," *VLSI Design*, vol. 2009, pp. 1–10, 2009.

[39] H. Yang and L. Dung, "On multiple-voltage high-level synthesis using algorithmic transformations," in *Proc. of ASP-DAC '05*, pp. 872–876, 2005.

[40] H. Yang and L. Dung, "Algorithmic transformations and peak power constraint applied to multiple-voltage low-power VLSI signal processing," *WSEAS Trans. on Signal Processing*, vol. 3, no. 12, pp. 479–486, 2007.

[41] B. Yu, S. Dong, S. Goto, and S. Chen, "Voltage-island driven floorplanning considering level-shifter positions," in *Proc. of GLSVLSI '09*, pp. 51–56, 2009.

本論文に関する発表業績

国際学会（査読つき）

1. Shin-ya Abe, Masao Yanagisawa, and Nozomu Togawa “An Energy-efficient High-level Synthesis Algorithm for Huddle-based Distributed-Register Architectures,” in *Proc. of ISCAS 2012*, 2012.(accepted)

国内学会（査読つき）

1. 阿部晋矢, 柳澤政生, 戸川望, “複数電源電圧および複数サイクルレジスタ間通信指向の低電力化高位合成手法,” 情報処理学会 DA シンポジウム 2011 論文集, 2011.

研究会

1. 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複数電源電圧指向の低電力化高位合成手法,” 情報処理学会研究報告 2011–SLDM–152(17), pp. 1–6, 2011.