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

2005－ＳＬＤＭ－121（29）
2005／10／21

Ｃ言語からの高位合成を用いたハードウェア最適化に関する一検討

井上 諭[†]　近藤 毅[‡]　泉 知論[*]　福井 正博[*]
[†]立命館大学大学院理工学研究科情報システム学専攻
[‡]立命館大学理工学部電気電子工学科
[*]立命館大学理工学部電子情報デザイン学科
〒525-8577 滋賀県草津市野路東 1-1-1
E-mail: {re000019, re004016, t-izumi, mfukui}@se.ritsumei.ac.jp

あらまし　本来，人知をより抽象度の高いレベルでの設計空間探索におくべきであるとの考え方に基づき，ハードウェアモデルをＣ言語により記述し，高位合成を用いてハードウェアを設計した場合に，SWの考え方で設計した場合とHWの考え方で設計した場合を比較して効率の点，および，設計最適化の点でどのような違いがでるかを実験により検討を行ったので報告する．

キーワード　Ｃベース設計，高位合成

A study for hardware optimization using a high level synthesis from C

Satoru Inoue,[†]　Tsuyoshi Kondo,[‡]　Tomonori Izumi,[*] and　Masahiro Fukui[*]
[†]Department of Information Science and Systems Engineering, Ritsumeikan University
[‡]Department of Electrical and Electronic Engineering, Ritsumeikan University
[*]Department of VLSI System Design, Ritsumeikan University
Noji-higashi 1-1-1, Kusatsu, Shiga 527-8577, Japan
E-mail: {re000019, re004016, t-izumi, mfukui}@se.ritsumei.ac.jp

Abstract　Higher abstraction design has been highly required so that a human can focus on inspired part of design by making computers do boring but time consuming jobs. This paper discusses the usefulness and problems of C base design with high-level synthesis of hardware description comparing software description.

Keyword　C base design, high-level synthesis

１．はじめに

現在の半導体設計の主流であるRTLの設計に対して，より抽象度を高めた動作レベル設計の一例として，Ｃ言語からの高位合成の活用が挙げられる．高位合成を用いると，RTL による設計に比べてより高い抽象度，より少ないコード数で機能記述できるため大幅な設計生産性の向上が期待される．そのため，設計段階で様々な仕様の変更も容易に対応できるようになる．

Ｃ言語からの高位合成では，状態遷移やデータパスの合成，リソース割り当てなどが自動的に行われ，設計者はクロックタイミングや回路の詳細構成を意識することなく，処理の流れのみを記述すればよい．一方で，従来のＣ言語は空間的な概念，並列動作の概念が含まれておらず，面積，性能，消費電力の面で，自動合成で最適なハードウェアが合成されるとは限らない．そこで，記述の容易なソフトウェア的な記述から開始し，要求仕様を満たせない部分について，より高性能なハードウェアが合成されるような記述に置き換えていくような設計フローが考えられる．

本稿では，設計事例を提案していくことで動作合成の設計手法の確立を目指す．Ｃ言語を用いた記述方法の違いにより，合成される回路の違いを推定・解析し，それらの結果から，HW 化に適した記述方法を検討する．

また，プログラムの記述の工夫として，剰余算の代わりに，シフト演算を使う．定数演算を予めおこなうことで，演算を減少させる．配列は，メモリにマッピングされることが多いので，配列を削減することで，メモリ最適化をおこなう．といったものが挙げられる[2]が，これらは自明なので，ＨW設計をおこなうときには，以上の処理はおこなわれているという前提で考える

著者らは文献[6]において，本稿とは別のもう少し簡略化したリフティング回路によって実験を行い，高位合成を行いた最適化手法について論じる．性能面では，本稿と同様の傾向を示しており，あわせて参照されたい．

2．Cベースによる設計フロー[4,5]

C ベース設計は，設計工程の上流で仕様を明確に定義し検証をおこなうことで効率の良い設計をおこなうことを目的としている．従来，ハードウェア設計者が経験によっておこなっていた作業をツールで自動的におこなうため，従来の方法では実現できなかった新しい構造を持つ回路が得られる可能性がある．

C ベース設計の一般的なフローを図１に示す．システムの仕様設計を行い，機能・性能・消費電力の目標仕様を決定する．次に機能仕様を満たすアルゴリズムを ANSI-C 等で記述し，C 検証をおこなった後，HW/SW トレードオフ解析をおこなう．HW/SW トレードオフをおこなう際に，HW/SW 間の同期・通信を，チャネルを使用しておこなう．チャネルは、共有メモリあるいはＦＩＦＯなどによって実現される． [7]

HW/SW トレードオフ解析後，クロックサイクル数の制約，加算器や乗算器などの資源制約を決め，高位合成をおこなった後，システム検証として RTL でのタイミング検証，C レベルとの等価検証をおこなう．この際，資源制約を変更することで，複数の RTL の中から最適なものを選ぶことができる．

C ベースによる設計をおこなうことで，アーキテクチャ設計時の C 検証と HW/SW のトレードオフ解析の C 検証との等価検証が容易にできるようになり，仕様変更にも柔軟な対応ができるようになる．

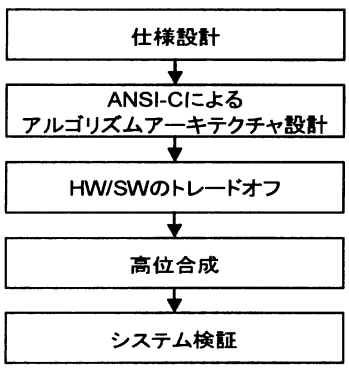

図１　Cベース設計フロー

3．Cベース合成による最適化設計

C ベース設計は，まだ一部で実用化されている段階で，手法の確立にはいたっていない．目的の機能仕様に対して，さまざまなアーキテクチャが考えられ，それぞれ回路規模，スループット，レイテンシ，動作周波数，消費電力が異なるが，それらの中から，設計者の要求に合った最適な回路を設計者の意図どおりにいかに合成できるかが，実用化のためには重要なポイントである．

今回の検討では，JPEG2000 等で用いられる離散ウェーブレット変換（以下，DWT）のリフティング回路に対して，記述のちがいによる，合成される HW の違いを解析することで，高位合成の活用方法に関する最適化方法を議論する．

また，合成ツールは最適化オプションがいくつも用意されている．チャネルの種類の変更やコンポーネントの使用数の設定，ビット幅の削減，パイプライン設定などが挙げられる．今回はコンポーネント数設定とパイプライン設定をおこなったことでの回路性能のちがいについても議論する．

3.1　実験環境

今回は動作合成ツールとして eXCite(米 YeXplorations Inc.)を，論理合成ツールとして Synplify Pro(Synplicity Inc.)を使用する．eXCite は ANSI-C で記述したプログラムを入力とし，合成結果を Verilog HDL または VHDL で出力する．回路合成をおこなうときに，オプションとして要求動作周波数の変更，外部接続をするためのチャネル設定，コンポーネントの最大使用数の設定，パイプライン設定などをおこなうことができる．合成の戦略として，速度優先(共有なし)(best performance / maximum components)，面積優先(maximum component sharing)，速度優先(共有あり)(best performance with maximum component sharing)の 3 パターンから選択することができる．

今回は，要求動作周波数を 50MHz とした．

3.2　離散ウェーブレット変換[1,3]

JPEG2000 の DWT では，入力信号に低域通過フィルタと高域通過フィルタを適用し２：１ダウンサンプリングされた各サブバンド係数を出力し，これを低域成分に対して分解レベル数だけ再帰的に適用する．画質と圧縮率を考慮して，変換係数が整数で構成される整数型（可逆変換用）と実数型（非可逆変換用）が規定されている．整数型は分析側が５タップ低域通過フィルタと３タップ高域通過フィルタとで構成される 5/3 可逆フィルタからなる．実数型は分析側が９タップ低域通過フィルタと７タップ高域通過フィルタとで構成される 9/7 非可逆フィルタからなる．本文では 9/7 非可逆フィルタを対象とする．9/7 非可逆 DWT の処理を式 (1)から(6)に挙げる．式(1)から(6)を用いてこれを実現するリフティング回路を図２に示す．

$$Y(2n+1)=X_{ext}(2n+1)+\alpha[X_{ext}(2n)+X_{ext}(2n+2)] \quad (1)$$

$$Y(2n)=X_{ext}(2n)+\beta[Y(2n-1)+Y(2n+1)] \quad (2)$$

$$Y(2n+1)=Y(2n+1)+\gamma[Y(2n)+Y(2n+2)] \quad (3)$$

$$Y(2n)=Y(2n)+\delta[Y(2n-1)+Y(2n+1)] \quad (4)$$

$$Y(2n+1)=KY(2n+1) \quad (5)$$

$$Y(2n)=Y(2n)/K \quad (6)$$

注　α、β、γ、σ、Kはパラメータ

これは1次元のフィルタ処理でありこの演算を水平方向，垂直方向の順に行って2次元変換係数を算出する．

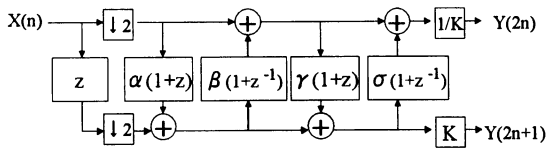

図2　9/7非可逆DWTのリフティング構成

3.3　設計空間の探索

C言語ベース設計手法は単に処理手順の短縮による設計効率の向上だけでなく，HW設計者にとっても使いやすく，必要に応じてハードウェアの最適化が行えるものでなければならない．高位合成ツールを用いたとしても，イメージどおりの回路を生成するために，記述の仕方やツールのオプションなどによって上手く制御できることが重要である．

本文では，まず解析を容易にするために，図2に示したリフティング回路の一段だけ抜き出し，図3のよう簡略化し，その記述例を提案する．SW設計よりおよびHW設計よりの2種類の記述を行い，実際のリフティング演算に対して両記述方法で回路合成を行ったときにどうなるかを検討する．

SW設計よりの記述では配列を使用し，まず全データを配列に読み込み，配列から値を読み出し演算しては配列に書き込むことを繰り返す．演算をおこなうたびに配列から読み書きするため，常にメモリとのアクセスが生じる．簡単な例として図3のリフティング演算を考える．ここでは，入力Xに対してパラメータαを乗算し，現在のXの値と，一つ前のXの値と，入力Bを加算する．この演算に対するSW設計よりの記述例を図4に示す．図4の記述により合成される回路のイメージ図を図5に示す．

HW設計よりの記述では，データはレジスタからレジスタへの流れ（ストリーム）として扱い，FIFOを用いた入出力を想定する．そのため，遅延素子（レジスタ）に相当する変数を用意する．同じく図3のリフティング演算に対するHWよりの記述例を図6に示す．d2は遅延素子を実現するための変数である．maにパラメータαの計算値をいれ，p1に加算の結果を格納する．加算の結果が確定した後に変数d2にmaの入力をおこなうことで，遅延素子となる．図6の記述によりレジスタによってSWとHWのデータのやり取りをおこなっている回路のイメージ図を図7に示す．

なお，今回の実験では，固定小数点演算で実装するため，C言語の記述においてすべての値は左シフトした整数として扱っている．

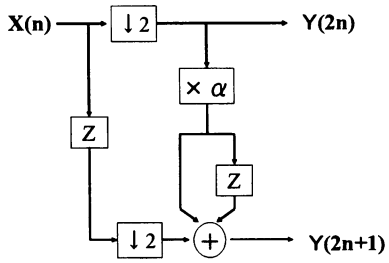

図3　リフティング回路のモデル図

```
⋮
for(n = 0; n < 128; n++){
  Y[2*n+1] = X[2*n+1] + a * (X[2*n] + X[2*n-2]);
}
⋮
```

図4　SW設計よりの記述例

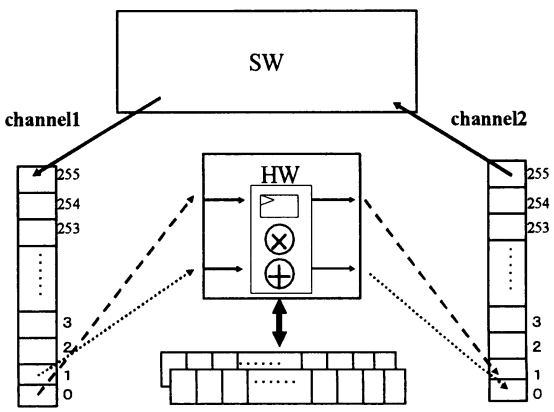

図5　SW設計回路のイメージ図

```
      ⋮
d1 = X0;
ma = a * X0;
p1 = X1 + ma + d2;
d2 = ma;
      ⋮
```

図 6　HW 設計よりの記述

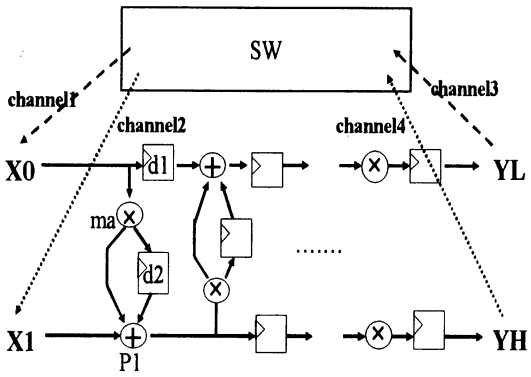

図 7　HW 設計回路のイメージ図

また，今回はパイプライン回路をオプションで実現させているが，上記の HW 設計記述のように式をいくつかのブロックにわけて記述をおこなっている場合に，暗黙のレジスタを想定し，処理手順を逆順に記述することで，意図的にパイプライン回路を合成することもできる．

4. 比較評価

リファティング処理について上記に示した記述方法をもとに，合成をおこない SW 設計と HW 設計についてまとめた結果を以下に示す．図 8 に性能優先(共有なし)で合成した回路を，図 9 にパイプライン設定で合成した回路を示す．図 9 の ST は動作合成ツールが，自動で割り当てたステートマシンの状態番号．ＳＴ0～10まで順に遷移していき，ＲＳＴが入力されるとＳＴ2に戻る．図 10,11,12 にスループット，レイテンシ，動作周波数をまとめた．表 1,2 について，それぞれ SW 設計と HW 設計に関する使用セル数，使用レジスタ数，使用 ROM/RAM 数をまとめた．このセル数は、セル単体は，どんなロジックを実現しても同じ面積になるようになっているので、セル数を比較することで面積の比較がおこなえる．

図 8 について，性能優先(共有なし)でおこなったので，加乗算器が全ての計算に割り当てられ一つ一つのレジスタに状態が割り当てられる回路が合成された．

図 9 について，パイプライン設定でおこなったので，ステートマシンも生成されず，期待どおりのパイプライン回路が合成された．

図 10 について，動作周波数は，パイプライン設定をおこなった回路が期待どおり最大周波数で動作し次に，性能優先(共有なし)，性能優先(共有あり)，面積優先の順に出力された．やはり，全体的にＳＷ設計に比べてHW 設計のほうが速度は高い値となった．

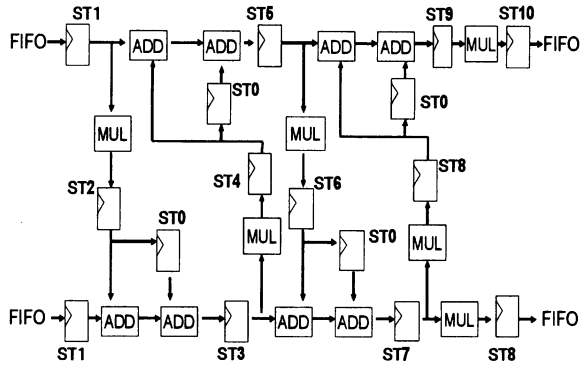

図 8　HW 設計より記述で性能優先(共有なし)の合成回路

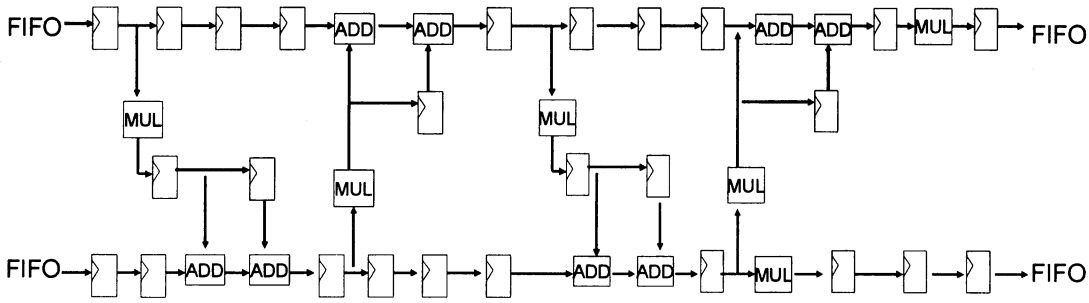

図 9　HW 設計より記述でパイプライン化をおこなった合成回路

図11について，レイテンシは，ＳＷ設計はＨＷ設計と比べ，メモリアクセスが発生し，大きな差が生じた．その差が大きいため，評価の段階で，詳しい数値は求めず，大まかな数値を表示している．HW 設計回路では，性能優先(共有なし)と性能優先(共有あり)は共有化の有無に関係なく同じ性能が出た．

図12について，スループットは，パイプラインで合成を行ったものは 1clock cycle となっており，きちんとしたものが合成された．今回の回路は，性能優先(共有なし)と性能優先(共有あり)は共有化の有無に関係なく同じ性能が出た．尚，SW に関しては，論文を書いた時点で，シミュレーションがおこなえていないため NA としている．

表 1 について使用セル数が性能優先(共有あり)で最小となったが，これは最大速度がでるようにした後に，共有化した場合に最適化される時にこうなる場合があるとかんがえられる

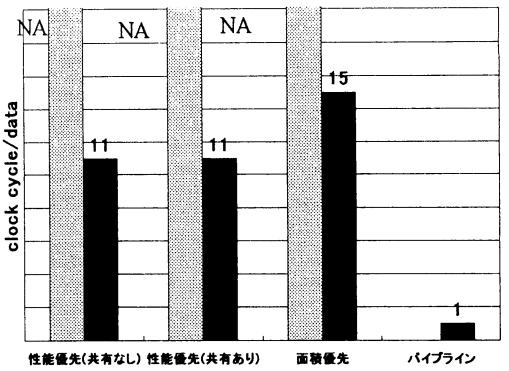

図 12　合成回路のスループット

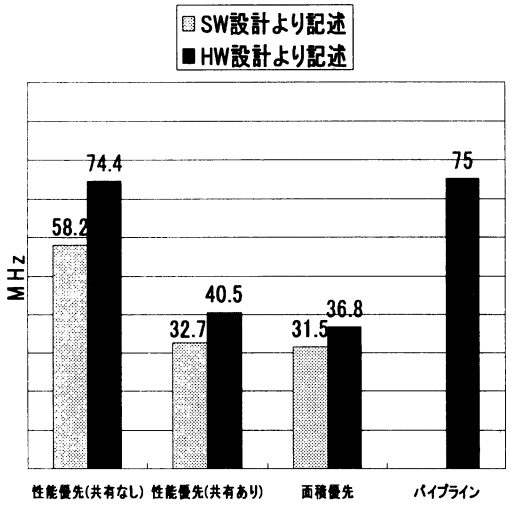

図 10　合成回路の動作周波数

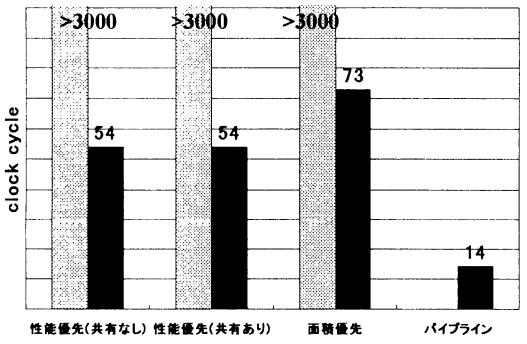

図 11　合成回路のレイテンシ

表 1　ＳＷ設計より記述による回路の回路規模

	性能優先(共有なし)	性能優先(共有あり)	面積優先
TOTAL Cell usage	2818	2118	2199
Register bits not including I/Os (bit)	342	296	331
RAM/ROM usage summary (block)	512	512	512

表 2　HW設計より記述による回路の回路規模

	性能優先(共有なし)	性能優先(共有あり)	面積優先	パイプライン
TOTAL Cell usage	2233	2148	2092	2598
Register bits not including I/Os (bit)	521	531	671	886
RAM/ROM usage summary (block)	0	0	0	0

5. まとめ

SW 設計よりの記述の回路では，メモリアクセスを頻繁に繰り返してしまい，結果，動作速度も低下し，メモリアクセスの繰り返しによる消費電力の増加を招いた．回路規模はHW設計と比べて，差がほとんど生じなかった．各性能評価のグラフを総合的に比較すると，HW 設計を考え記述しなければ，最適な回路を合成することができない．

以上のことから，今回のものが最適な記述とは断定できないが，記述を工夫すれば，RTL の設計に比べて簡単な記述でイメージどおりの回路が得られることが実証でき，HW 設計者が目的とするものに近い回路を得られた．尚，今回は，我々の実験環境の制約により特定のツールを用いた場合についての評価を実施したが，他の高位合成ツールを用いた場合においても同様のオプションや設計手法をとることができると推測できる．本文が今後のCからの合成の活用を実用レベルに発展させるための一助になることを期待するものである．

6. 今後の課題

今後，C 言語ベースによる高位合成の手法の探索として，DWT 全体や算術符号化などに取り組み，JPEG2000 全体のモジュールのハード化や，現在研究室で取り組んでいるシステムレベルからの電力解析の手法を取り入れ性能だけでなく，低消費電力化も考えた最適な高位合成手法の確立を目指す．

謝辞

最後に，ソリトンシステムズの千星健一様，橋口和幸様，Y eXplorations Inc.の Daniel D. Gajski 教授，Ted Hadley 博士に関連の資料提供および有益な議論をいただき感謝します．立命館大学電子情報デザイン学科山内寛紀教授に有益な議論や助言をいただき感謝します．この研究の一部は，文部科学省ハイテクリサーチセンター整備事業プロジェクト“インテリジェント・シリコン・ソサエティにおける研究”による．

参考文献

[1] 小野定康，鈴木純司，“わかりやすい JEPG2000 の技術，” オーム社，2003 年

[2] 森江善之，富山宏之，村上和彰，“動作合成の効率化を指向した動作レベル記述・トランスフォーメーション，” 電子情報通信学会，信学技法，VLD2003-81，2003 年 11 月

[3] 出口勝昭，安部昌平，安生健一郎，栗島亨，天野英晴，“DPR-1 上への JPEG2000 の離散ウェーブレット変換器ウェーブレット変換器と算術符号の実装，” 電子情報通信学会，第 4 回 リコンフィギャラブルシステム研究会 予稿集，2004 年 9 月

[4] 白崎義雄，松野浩之，後藤守孝，伊藤仁貴，折井謙，“C 言語を用いた LSI 設計の一手法，” 電子情報通信学会総合大会，A-3-13，p.92，2002 年 3 月

[5] MRI，http://safety.mri.co.jp/colum/vol009.html

[6] 近藤毅，井上諭，泉知論，福井正博，“C 言語ベース設計によるハードウェア最適化に関する一検討，” 情報処理学会関西支部大会，C-3，2005 年 10 月(予定)

[7] J.Staunstrup, W.Wolf, “Hardware/Software Co-design,” Kluwer Academic Publishers, 1997