動作合成のスケジューリングアルゴリズムの評価

日大生産工(学部)　○石井 英明　　日大生産工　　細川　利典

1. はじめに

近年，半導体技術の進歩による超大規模集積回路（Very Large Scale Integrated circuit：VLSI）の大規模化に伴い，回路が複雑化してきている．従来，このようなVLSIの設計にはハードウェア記述言語（Hardware Description Language：HDL）でレジスタ転送レベル（Register Transfer Level：RTL）の記述をし，論理合成ツールを用いることでネットリストを生成してきた．しかし従来の手順では設計が困難となってきているため，さらに抽象度の高いレベルでの設計として動作合成が着目されている[1,2].

動作合成は抽象度が高いため，RTLの記述に比べ記述量が少なく設計生産性に優れる．反面どの演算操作をどの演算器に割り当てるかなどの細かい設定は動作合成ツールが行うため，スケジューリングやバインディングのアルゴリズム次第で出力される回路の面積や性能に差が生じてしまう．

本稿では三つのスケジューリングアルゴリズムを用いて動作合成を行い，スケジューリングアルゴリズムの違いによるバインディングへの影響を評価する．

2. 動作合成

動作合成は動作記述を入力とし，RTL回路記述を生成する．入力から出力までの過程は，大きく分けるとコントロール／データフローグラフ（Control/Data-Flow Graph：CDFG）生成，アロケーション，スケジューリング，バインディング，RTL回路記述生成の五つのフェーズに分けられる[3]（図1）

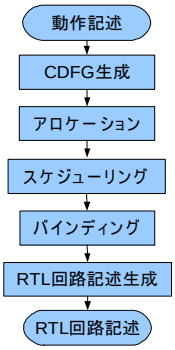

図 1. 動作合成

CDFG生成とはコントロール／データグラフ動作記述をグラフで表現するフェーズである．アロケーションとは演算器やメモリの種類や個数を選択するフェーズであり，スケジューリングやバインディングに対する制約条件を設定するフェーズとなる．スケジューリングとは各演算操作をどの時刻に実行するかを決定するフェーズである．本稿ではこのスケジューリングフェーズのアルゴリズム評価をする．バインディングとは各演算操作や変数に演算器やレジスタを割り当てるフェーズである．RTL回路記述生成とはバインディングで割り当てた演算器やレジスタ間の結線を実現するフェーズである．

3. スケジューリング

動作合成を最適化するうえでスケジューリングのアルゴリズムは重要となる．この章では代表的なスケジューリング方法について述べる．なお各スケジューリングの説明で使用するDFGの式はx=(a*b)*(c*d)-e-(f-(g*h))とする（図2）．

Evaluation of scheduling algorithm in Behavioral Synthesis

Hideaki ISHII and Toshinori HOSOKAWA

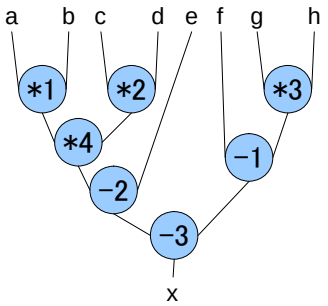

図2. DFG

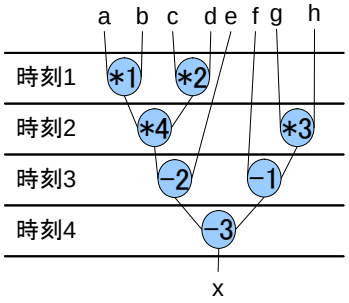

図4. ALAPスケジューリング

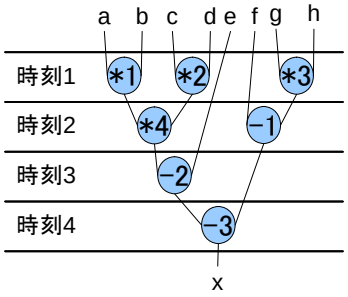

図3. ASAPスケジューリング

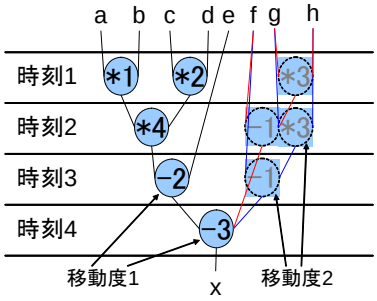

図 5. 移動度

3.1. ASAP・ALAPスケジューリング

代表的なスケジューリングの中で基本となるスケジューリングが二つある．それが ASAP (As Soon As Possible)スケジューリングと ALAP (As Late As Possible) スケジューリングである．ASAP スケジューリングとは，各演算操作を可能な限り早い時刻に割り当てるスケジューリングである (図 3)．ALAP スケジューリングとは，ASAP とは逆に各演算操作を可能な限り遅い時刻に割り当てるスケジューリングである (図 4)．例として使用した式の g と h の乗算部分に着目すると，ASAP スケジューリングでは時刻 1 に割り当てられ，ALAP スケジューリングでは時刻 2 に割り当てられることがわかる．同様に f と g*h の減算部分に着目すると ASAP スケジューリングでは時刻 2 に割り当てられ，ALAP スケジューリングでは時刻 3 に割り当てられる．このように演算を割り当てる際の余裕を移動度と言い，ASAP スケジューリングと ALAP スケジューリングの差により求めることができる．a と b の乗算操作のように ASAP スケジューリングと ALAP スケジューリングの時刻差が 0 の場合を移動度 1 とし，同様に時刻差が 1 の場合を移動度 2，時刻差が n の場合を移動度 n ＋1 とする (図 5)．

各時刻の演算操作の個数に着目すると，演算器の必要個数に差があることがわかる．乗算器に着目すると，ASAP スケジューリングでは時刻 1 に 3 個，時刻 2 に 1 個となり，乗算器の必要個数が 3 個となる．一方 ALAP スケジューリングでは時刻 1 に 2 個，時刻 2 に 2 個となり，乗算器の必要個数が 2 個となる．今回の例では乗算器に関しては ALAP スケジューリングの方が ASAP スケジューリングに比べ個数が 1 個少ない結果となり優秀に見えるが，続いて減算器の個数に着目してみる． ASAP スケジューリングでは，時刻 2 に 1 個，時刻 3 に 1 個，時刻 4 に 1 個となり．減算器の必要個数が 1 個となる．一方 ALAP スケジューリングでは，時刻 3 に 2 個，時刻 4 に 1 個となり，減算器の必要個数が 2 個となる．乗算器に着目した場合とは逆に，ASAP スケジューリングの方が ALAP スケジューリングに比べ個数が 1 個少ない結果となる．以上のことから，スケジューリングの方法で演算操作が割り当てられる時刻や演算器の個数が大きく異なることがわかる．

3.2. フォースディレクティッドスケジューリング

ASAP スケジューリングと ALAP スケジューリングより求められる移動度を利用するスケジューリングとしてフォースディレクティッドスケジューリングがある．フォースディレクティッドスケジューリングは各演算操作が移動度内の各時刻に等確率でスケジュールされると仮定し，期待値の最大値がなるべく小さくなるようにスケジューリングする手法である．図 6 にアルゴリズムを示す．FDS は以下の Step1 から Step8 の操作で構成される．

Step1：ASAP スケジューリングで各演算操作の時刻を取得
Step2：ALAP スケジューリングで各演算操作の時刻を取得
Step3：Step1 と Step2 で求めた各演算操作の時刻の差から，各演算操作の移動度を取得
Step4:移動度 1 の演算操作は割り当て時刻が一意に決まるため割り当て
Step5:各演算操作が ASAP 時刻から ALAP 時刻の各時刻に等確率で割り当てられると仮定し，各演算操作に期待値を設定
Step6:全ての演算操作がスケジューリングされた場合は終了
Step7:時刻ごとに演算操作の期待値の合計値を取得
Step8:まだ時刻が割り当てられていない演算操作の中で期待値が最も大きい演算操作を，Step7 で取得した時刻ごとの期待値の最も小さい時刻に割り当て Step5 へ

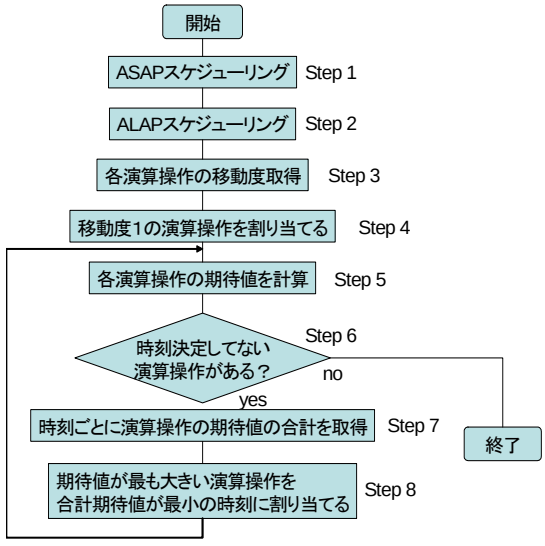

図 6．フォースディレクティッドスケジューリングアルゴリズムの処理フロー

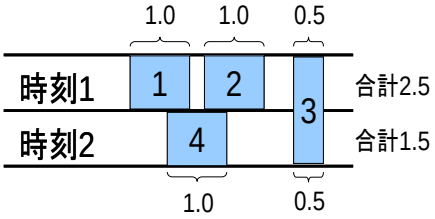

図 7．乗算操作の移動度

ASAP や ALAP 同様に図 2 の DFG をスケジューリングすると仮定しその中で乗算処理のみを例として説明する．なお Step1 と Step2 は 3．1 で述べたため説明を省く．Step3 で乗算操作のみを対象に移動度を取得したものが図 7 である．Step4 で移動度 1 となる乗算操作は時刻が割り当てられる．よって乗算操作 1・2 は時刻 1 に，乗算操作 4 は時刻 2 となる．Step5 で期待値を求める．乗算操作 1・2・4 は移動度が 1 のためそれぞれの期待値は 1 となる．一方，乗算操作 3 は移動度 2 となり，各時刻に等確率で割り当てられると仮定すると時刻 1 と時刻 2 にそれぞれ期待値 0.5 となる．Step6 の終了条件では，乗算操作 3 にまだ時刻が割り当てられていないためスケジューリングを続ける．Step7 で時刻 1 の期待値合計が 1+1+0．5=2．5，時刻 2 の期待値合計が 1+0.5=1.5 と決まる．Step8 でまだ時刻が割り当てられていない乗算操作 3 が割り当て対象に選択され，時刻 1 と時刻 2 では時刻ごとの期待値合計が小さい時刻 2 が最も小さい時刻となるため，乗算操作 3 が時刻 2 に割り当てられ Step5 に戻る．Step5 で期待値が再計算され乗算操作 3 の期待値が 1 に変更される．Step6 で全ての乗算操作に時刻が割り当てられたため終了となる．今回は乗算操作のみを例にスケジューリングをしたが，全ての演算操作がスケジューリングされた時点で FDS は終了となる．

4. 実験方法および実験結果

加算操作と乗算操作を用いた適当な式に各スケジューリングアルゴリズムを適用し，演算器個数の違いについて検証した．C 言語ベースの動作記述を入力とし，スケジューリングまでの作業はプログラミングで行った．またバインディングに関しては面積最小となる演算器の割り当てとなるように，使用演算器

表 1. 実験結果

式	スケジューリング	加算器の必要数	乗算器の必要数
x=(a+b)+c*d +(e+f)*g+h	ASAP	2	1
	ALAP	2	2
	FDS	1	1
x=a*b+(c+d) *(e+f)+g*h	ASAP	2	1
	ALAP	2	2
	FDS	2	1
x=(a+b)*(c+d) *(e+f)*(g+h)*(i+j)	ASAP	5	1
	ALAP	2	1
	FDS	2	1

数が最小となるように手で行った．実験結果を表 1 に示す．表 1 に示した三つの式では，フォースディレクティッドスケジューリングを用いたスケジューリングが面積最小の結果となった．また ASAP と ALAP に関しては式ごとに演算個数が多い場合と少ない場合があり，どちらのスケジューリングアルゴリズムが優秀とは言えない結果となった．

5. おわりに

本稿では，動作合成のスケジューリングアルゴリズムについて検討した．時間の関係上三つのスケジューリングアルゴリズムについて三つの式で検証することとなったが，十分に実験できなかった．また本稿で紹介したスケジューリングアルゴリズムの他にリストスケジューリングや ILP を用いたスケジューリングなどの実装もできなかった．今後の課題としては，使用演算操作が多い式での検証や，未実装のスケジューリングアルゴリズムの実装が挙げられる．

「参考文献」

1) 藤原秀雄：ディジタルシステムの設計とテスト，工学図書株式会社，2004

2) Daniel D. Gajski, Nikil D. Dutt, Allen C-H Wu, and Steve Y-L Lin, “HIGH-LEVEL SYNTHESIS Introdu-ction to Chip and System Design”, Kluwer Academic Publisher, 1992.

3) 宇佐美公良，内山邦男，岡島義憲，黒坂均，高見沢一彦，田口眞男，三木良雄，山口雅之，山本一郎，吉田正昭，若林一敏：2002 年度システム LSI 設計 ―LSI 設計編― 分冊 1，株式会社半導体理工学研究センター，2003