

## 資源割り当て駆動スケジューリングにおけるレジスタ間転送の自動挿入

小畑 貴之<sup>†</sup> 金子 峰雄<sup>†</sup> 田湯 智<sup>†</sup>

<sup>†</sup> 北陸先端科学技術大学院大学 情報科学科 〒923-1292 石川県能美郡辰口町旭台 1-1

E-mail: [†{t-obata,mkaneko,tayu}@jaist.ac.jp](mailto:†{t-obata,mkaneko,tayu}@jaist.ac.jp)

**あらまし** ループパイプラインスケジューリングにおいて、データのレジスタ間転送はスケジューリング可能性の増大、繰り返し周期短縮に寄与し得る技術である。演算スケジューリング後に資源割り当てを行う場合は、先に決まるデータの生存期間を見て、適宜データのレジスタ間転送を挿入しながらデータのレジスタへの割り当てを行えるが、資源割り当てスケジューリングにおいては、陽にデータのレジスタ間転送を考慮したスケジューリングが必要となる。本稿では資源割り当て駆動ループパイプラインスケジューリングにおいてデータのレジスタ間転送の自動挿入を考慮して、その転送回数を最小化する問題に対する整数線形計画問題としての定式化・解法を提案する。

**キーワード** 高位合成, スケジューリング, パイプライン実行, レジスタ間転送, 整数線形計画法

## Automatic Register-to-Register Transfer Insertion in Assignment Driven Scheduling

Takayuki OBATA<sup>†</sup>, Mineo KANEKO<sup>†</sup>, and Satoshi TAYU<sup>†</sup>

<sup>†</sup> School of Information Science, Japan Advanced Institute of Science And Technology 1-1 Asahidai  
Tatunokuti, Ishikawa, 923-1292 Japan

E-mail: [†{t-obata,mkaneko,tayu}@jaist.ac.jp](mailto:†{t-obata,mkaneko,tayu}@jaist.ac.jp)

**Abstract** With respect to loop pipeline schedule, register-to-register data transfer is mandatory in some cases, and helps to make iteration period shorter in other case. For a synthesis system in which the operation scheduling precedes binding, register-to-register data transfers can be easily inserted by checking lifetime of each data set which is determined by the scheduling, and we need not give a care to register-to-register data transfers in scheduling phase. On the other hand, for a binding-driven synthesis approach and/or post-floorplan scheduling, resource binding is fixed (temporary or finally) before scheduling, and we need to include register-to-register data transfers into a formulation of scheduling problem explicitly. In this paper, we propose a integer linear programming based approach to the loop pipeline scheduling minimizing the number of register-to-register data transfers under specified resource assignment and iteration period.

**Key words** High-Level Synthesis, Scheduling, Pipeline Execution, Register-to-Register Transfer, Integer Linear Programming.

### 1. ま え が き

近年における集積回路の大規模化高集積化に伴い、集積回路設計は大規模化・複雑化の一途をたどっている。また、多種多様な目的に対する専用ハードウェアの設計要求などにより、計算機を用いた設計支援や自動化技術の開発が重要となってきた。大規模な回路を設計するためには、抽象度の高い記述から抽象度の低い記述へ段階的に変換する階層化設計が用いられる。高位合成は、こうした階層化設計の比較的上位に位置し、計算アルゴリズム記述を入力として、これを抽象度のより低い

RT レベル記述へ変換する技術である。この変換の中で、演算のスケジューリングやハードウェア資源への割り当てが決定されることになる。

信号処理や画像処理などの部分的に同一な処理を繰り返し実行する計算アルゴリズムでは、繰り返し周期が回路性能の支配的要因となる。繰り返し周期を小さくするための手法としては、一回の繰り返し演算が終了する前に次の繰り返し演算を開始するパイプライン実行が用いられる。繰り返し実行を行うときには同一の演算が複数回実行されるが、回路構造や回路の制御機構を単純化するため、同一の演算はどの周期でも同じ演算器で

実行され、同一の演算の生成するデータはどの周期でも同じレジスタに書き込まれるものとする。この条件下でパイプライン実行を行う場合、一つの演算の異なる周期での実行によって生成されるデータ同士は生存期間に重なりが存在してはならないので、繰り返し周期が制限されることがある。こうした状況下で繰り返し周期をより積極的に短縮するための方法として、データのレジスタ間転送の導入が考えられる。

高位合成へのアプローチとして、スケジュールを先に行い、その結果を受けて資源割り当てを行う従来方式では、演算スケジュール時に陽にレジスタ間転送を考慮することはなく、結果として得られるデータの生存期間を見て、適宜レジスタ間転送を挿入すれば良い。しかし反面、レジスタ間転送回数を考慮しない分、転送に付随する電力消費の最小化はなされない。一方資源割り当てを先に行い、それに対してスケジュールを決める資源割り当て駆動合成方式[2]や Post-Floorplan schedule[7]においては、陽にレジスタ間転送を考慮したスケジュールの定式化を行わない限り、先に述べた繰り返し周期が制限される問題が発生する。

本稿では、資源割り当て駆動スケジューリングによる解空間探索を支援するために、資源割り当てから演算、及びレジスタ間転送のスケジュールを求める手法を開発する。繰り返し周期と資源割り当てが与えられたとき、一つのデータに関してはレジスタ間転送を繰り返し周期ごとにしか行わないという条件の下でスケジュールが存在するならば、レジスタ間転送数最小のスケジュールを求めることができるアルゴリズムを開発した。実験により、従来の資源割り当て駆動スケジューリングで用いられていたレジスタ間転送を考慮していないスケジューリングよりも短い繰り返し周期でスケジュール可能な計算アルゴリズムが存在することを確認した。

## 2. データフローグラフとスケジュール

本論文では高位合成の入力アルゴリズムはデータフローグラフによって表されており、その全体を周期  $T_r \in \mathbb{Z}^+$  で繰り返し実行するものとする。データフローグラフは有向グラフ  $G = (V, A)$  であり、頂点集合  $V$  は演算を、辺集合  $A$  は演算間のデータ依存関係を表現している。演算  $u$  の  $i$  回目の繰り返し実行を  $u^{(i)}$  で表し、 $u^{(i)}$  の生成するデータを  $d(u^{(i)})$  で表す。図 1(a) では演算  $a$  から  $c$  への辺に重み 2 が付されているが、これは  $a, c$  間の遅延が 2 であることを表している。遅延は  $D: A \rightarrow \mathbb{Z}$  によって表され、辺  $(u, v)$  があって遅延が  $D(u, v)$  のとき、 $u^{(i)}$  の生成するデータを  $v^{(i+D(u, v))}$  が使用することを意味している。図中で重みを省略した場合は、遅延 0 を表すものとする。演算に対するスケジュールを  $\sigma: V \rightarrow \mathbb{Z}$  によって表す。 $u^{(i)}$  の実行開始コントロールステップは  $\sigma(u) + iT_r$  であり、 $\sigma(u^{(i)})$  で表す。各演算の実行に要するコントロールステップ数を  $e: V \rightarrow \mathbb{Z}^+$  により表す。 $u$  の演算結果が使用可能となるコントロールステップは  $\sigma(u) + e(u)$  である。

演算は使用するデータが生成された後に実行を開始しなければならないので、全ての  $(u, v) \in A$  について次式が成り立つ。

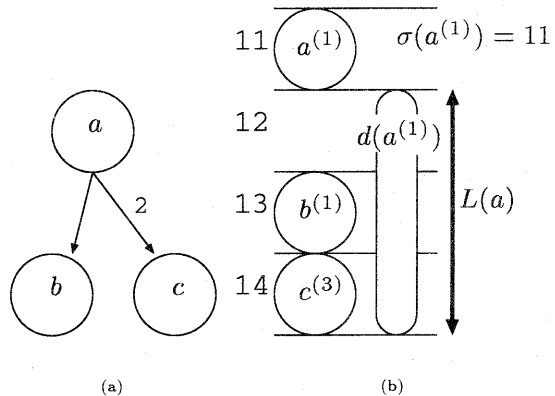


図 1 データフローグラフとスケジュールの例

$$\sigma(v) + D(u, v)T_r \geq \sigma(u) + e(u) \quad (1)$$

演算器がパイプライン化されているとき、他の演算を実行しているか否かに関わらず毎コントロールステップで演算を開始することができる。また、演算の入力データは実行開始コントロールステップにおいて利用可能であれば良い。本論文では全ての演算器がパイプライン化されているものとする。

データが生成されてから最後に使用されるまでのコントロールステップをそのデータの生存期間と呼ぶ。 $L: V \rightarrow \mathbb{Z}^+$  によって演算が生成するデータの生存期間のコントロールステップ数をあらわす。データの生存期間の定義より  $(u, v) \in A$  である全ての  $u, v$  について、 $L(u)$  および  $\sigma(u), \sigma(v)$  は次式を満たす。

$$\sigma(u) + e(u) + L(u) - 1 \geq \sigma(v) + D(u, v)T_r \quad (2)$$

## 3. パイプライン実行におけるレジスタ間転送

異なる周期での同一の演算の実行により生成されるデータが一つのレジスタに書き込まれる資源割り当てを行う場合、データのレジスタ間転送を許さないスケジューリングにおけるデータの生存期間は繰り返し周期以下でなければならない。レジスタ間転送を許すスケジューリングにおいては、ある演算の実行  $u^{(i)}$  により生成されたデータの生存期間が繰り返し周期  $T_r$  よりも大きい場合、少なくとも  $T_r$  ごとにデータを他のレジスタへ転送しなければならない。

図 2(a) に示すデータフローグラフ  $G$  は、演算  $b$  の実行時間が 2 であるため、どのようにスケジュールしても  $a$  により生成されたデータの生存期間は 3 以上である。したがって、レジスタ間転送を用いないスケジューリングでは、繰り返し周期は 3 以上となる。 $G$  をレジスタ間転送を用いて繰り返し周期  $T_r = 2$  でパイプライン実行した例を図 2(b) に示す。 $d(u^{(1)})$  と  $d(u^{(2)})$  は同一のレジスタへ書き込まれるため、 $d(u^{(2)})$  がコントロールステップ 4 でレジスタ  $R_1$  に書き込まれるときに、 $d(u^{(1)})$  はレジスタ  $R_1$  から  $R_2$  へ転送され、コントロールステップ 4 で  $c^{(1)}$  に使用される。同様に、コントロールステップ 6 では、 $d(u^{(2)})$  が  $R_2$  へ転送され、 $d(u^{(3)})$  が  $R_1$  へ書き込まれる。

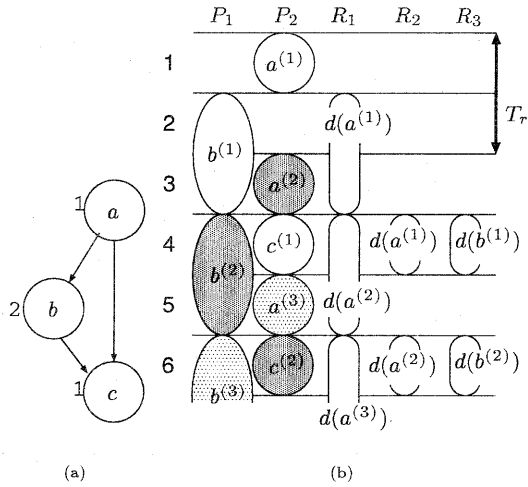


図2 データフローグラフとレジスタ間転送付スケジュールの例

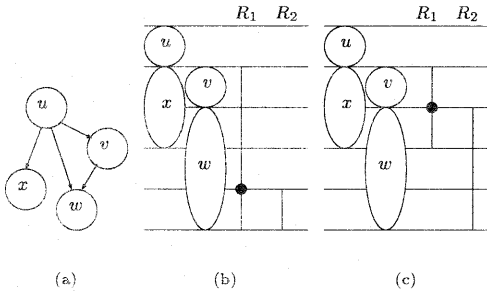


図3 非パイプライン演算器を使用する場合のスケジューリングとレジスタ間転送

演算器がパイプライン化されていないとき、演算の実行中入力データは同じレジスタに保持されていなければならない。したがってレジスタ間転送を行うコントロールステップに実行中の演算があれば、その演算が終わるまで転送元レジスタへは新たにデータを書き込むことができない。図3(a)に示すデータフローグラフが図3(b)のようにスケジュールされたとする。レジスタ  $R_1$  上の演算  $u$  の生成するデータを第4コントロールステップでレジスタ  $R_2$  へ転送しているが、演算  $w$  が実行中であるためにレジスタ  $R_1$  へは  $w$  の実行完了後の第6コントロールステップまで新たなデータを書き込むことができない。周期4以下で繰り返し実行するためには図3(c)のように  $w$  の実行開始以前の第2コントロールステップでレジスタ間転送をしなければならない。その場合も演算  $v$  が実行中であるために第3コントロールステップまでは  $R_1$  に新たなデータを書き込むことはできない。演算器がパイプライン化されていない場合はこのように繰り返し周期短縮のためのレジスタ間転送を行う場所がスケジュールによって制限される。

演算器がパイプライン化されているとき、実行開始後の1コントロールステップの間だけ入力と同じレジスタ上に保持され

ていれば良く、レジスタ間転送が行われた直後に転送元レジスタのデータが書き換えられても正しい演算結果が得られる。そのためデータを利用する演算のスケジュールに関わらず繰り返し周期と同じ時間が経過するごとにレジスタ間転送を行うものとするができる。

演算スケジュールを先に行い、その結果を受けて資源割り当てを行う従来の高位合成方式では、演算スケジュール時に陽にレジスタ間転送を考慮する必要はなく、結果として得られるデータの生存期間を見て、適宜、レジスタ間転送を挿入すればよい。一方資源割り当てを先に行い、それに対してスケジュールを決める割り当て駆動合成方式や Post-Floorplan Schedule においては、陽にレジスタ間転送を考慮して、スケジュールの定式化を行う必要がある。

#### 4. 資源割り当て駆動パイプラインスケジューリング

$\mathcal{R}$  と  $\mathcal{F}$  をそれぞれを使用するレジスタおよび演算器の集合とする。演算のレジスタおよび演算器への写像と演算の演算器への割り当て  $\xi: V \rightarrow \mathcal{R}$ ,  $\rho: V \rightarrow \mathcal{F}$  の対  $(\xi, \rho)$  を資源割り当てと呼ぶ。資源割り当て駆動合成方式 [2] では、資源割り当てから成る解空間を探索し、探索した解に対する評価は実行可能なスケジュール  $\sigma$  を生成して行う。本節ではレジスタ間転送を許さない条件下での実行可能なスケジュールの条件を制約式により与える。従来手法では、この制約式をもとに分枝限定法や整数線形計画法を用いてスケジュールを求めている。

資源制約下でのスケジューリング問題は以下のように表される。

入力: データフローグラフ  $G = (V, A)$ , 繰り返し周期  $T_r$ , 資源割り当て  $(\xi, \rho)$ , 演算の実行時間  $e: \mathbb{Z}^+$

出力: 実行可能なスケジュール  $\sigma: V \rightarrow \mathbb{Z}$

レジスタ間転送を行わないとき、どのデータの生存期間の長さも繰り返し周期を越えることはできない。したがって全ての演算  $u$  について次式が成り立つ。

$$L(u) \leq T_r \quad (3)$$

どの演算器も同時に高々1つの演算を実行開始可能である。これは同じ演算器に割り当てられた全ての2つの演算  $u, v$  の間の次のような制約として記述できる。

ある整数  $K_{uv}$  が存在し、 $u^{(i)}$  の実行開始コントロールステップは  $v^{(i+K_{uv})}$  の実行開始コントロールステップよりも小さく、 $u^{(i+1)}$  の実行開始コントロールステップは  $v^{(i+K_{uv})}$  の実行開始コントロールステップよりも大きい (図4)。

すなわち次のような制約式が成り立つことになる。

$$\sigma(u) < \sigma(v) + K_{uv}T_r \quad (4)$$

$$\sigma(v) + K_{uv}T_r < \sigma(u) + T_r \quad (5)$$

$$\text{for } \rho(u) = \rho(v)$$

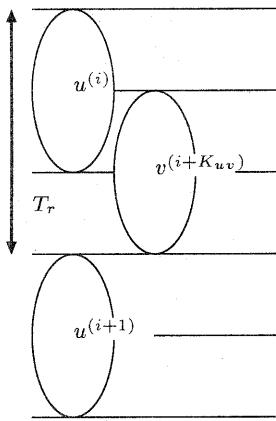


図4 同じ演算器に割り当てられた演算間の制約

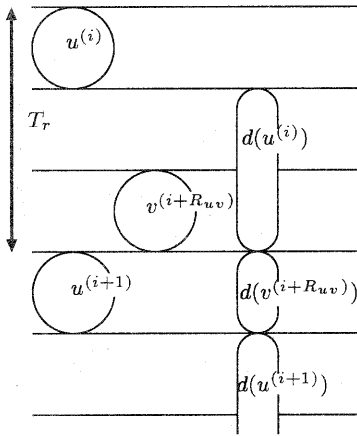


図5 出力するデータが同じレジスタに割り当てられた演算間の制約

どのレジスタも同時に高々1つのデータを保持することができる。これは結果が同じレジスタに書き込まれる全ての2つの演算  $u, v$  間の次のような制約として記述できる。

ある整数  $R_{uv}$  が存在し、 $u^{(i)}$  の生成するデータの生存期間の終了コントロールステップは  $v^{(i+R_{uv})}$  の生成するデータの生存期間の開始コントロールステップよりも小さく、 $v^{(i+R_{uv})}$  の生成するデータの生存期間の終了コントロールステップは  $u^{(i+1)}$  の生成するデータの生存期間の開始コントロールステップよりも小さい (図5)。

したがって次のような制約式が成り立つことになる。

$$\sigma(u) + e(u) + L(u) - 1 < \sigma(v) + R_{uv}T_r + e(v) \quad (6)$$

$$\sigma(v) + R_{uv}T_r + e(v) + L(v) - 1 < \sigma(u) + T_r + e(u) \quad (7)$$

$$\text{for } \xi(u) = \xi(v)$$

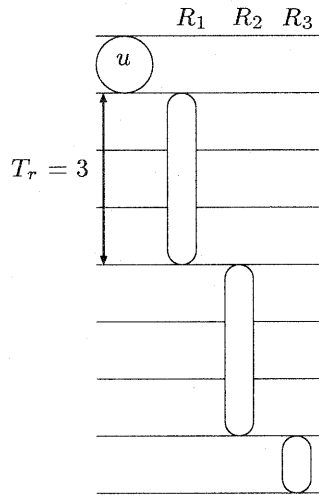


図6 レジスタ間転送の様子

## 5. 資源割り当て制約下のレジスタ間転送付スケジューリング

演算  $u$  が生成するデータの生存期間の長さが繰り返し周期より大きい場合は、データがレジスタに書き込まれてから繰り返し周期以内に他のレジスタへ転送することでスケジュールを実現できる。ここでは特に以下の仮定の下で、レジスタ間転送付スケジュールの定式化を行う。

(1) 生存期間が繰り返し周期より大きいデータに対して、生成時を起点に、繰り返し周期ごとに別のレジスタへ転送されるものとする。

(2) 演算の生成するデータのレジスタ割り当て  $\xi(u)$  は、そのデータが最後に使うレジスタを指定するものとする。

図6に繰り返し周期3の場合に生存期間が7のデータのレジスタ間転送例を示す。この場合、演算  $u$  の生成するデータは3つのレジスタを使用することになる。ここで用いる資源割り当て駆動スケジュールでは、 $u$  が割り当てられるレジスタ  $\xi(u)$  は与えられているが、この場合、 $\xi(u)$  は、データが最後に転送される先のレジスタ  $R3$  が  $\xi(u)$  となるものとする。 $u$  が生成するデータの生存期間が繰り返し周期以下の場合は、レジスタ間転送は行われず、 $\xi(u)$  に保持されるものとする。この条件のもとでは、レジスタ間転送数を最小化することで、レジスタ数と結線数を最小化するスケジュールが得られる。

演算  $u$  の生成するデータがレジスタ  $\xi(u)$  上に存在しなければならないコントロールステップ数を  $l(u)$ 、 $u$  の生成するデータに対して行われるレジスタ間転送数を  $r(u)$  とすると、次式が成り立つ。

$$0 < l(u) \leq T_r \quad (8)$$

また  $L(u) = r(u)T_r + l(u)$  となるので式 (1), (2) から次式を得る。

$$\sigma(v) + D(u, v)T_r \geq \sigma(u) + e(u) \quad (9)$$

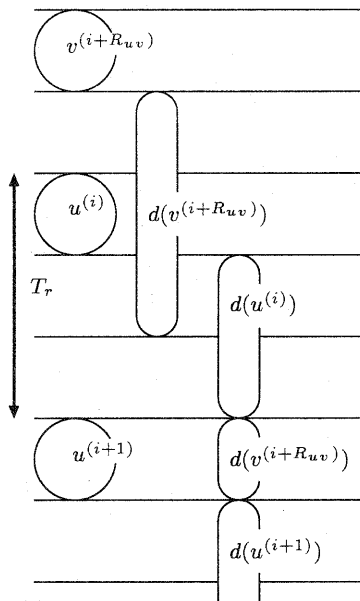


図7 出力するデータが同じレジスタに割り当てられた演算間の制約

$$\sigma(u) + e(u) + r(u)R_r + l(u) - 1 \geq \sigma(v) + D(u, v)T_r \quad (10)$$

生成するデータが同じレジスタに割り当てられた演算同士の制約は次ように記述できる。

ある整数  $R_{uv}$  が存在し、 $u^{(i)}$  の生成するデータの生存期間の終了コントロールステップは  $v^{(i+R_{uv})}$  の生成するデータが  $\xi(v)$  に書き込まれるコントロールステップよりも小さく、 $v^{(i+R_{uv})}$  の生成するデータの生存期間の終了コントロールステップは  $u^{(i+1)}$  の生成するデータが  $\xi(u)$  に書き込まれるコントロールステップよりも小さい (図7)。

したがって次の2式が成り立つ。

$$\begin{aligned} \sigma(u) + e(u) + r(u)T_r + l(u) - 1 < \\ \sigma(v) + R_{uv}T_r + e(v) + r(v)T_r \end{aligned} \quad (11)$$

$$\begin{aligned} \sigma(v) + R_{uv}T_r + e(v) + r(v)T_r + l(v) - 1 < \\ \sigma(u) + T_r + e(u) + r(u)T_r \end{aligned} \quad (12)$$

for  $\xi(u) = \xi(v)$

同じ演算器に割り当てられた演算同士の制約はレジスタ間転送を考慮しない場合と同様であり、式(4),(5)によって与えられる。

$\sigma(v), K_{uv}, R_{uv}, r(v), l(v)$  を変数とし、式(4)–(5),(8)–(12)を制約式、最適化目標を  $\sum_{v \in V} r(v)$  の最小化とする整数線形計画問題を解くことによりレジスタ間転送数最小のスケジュールを得ることができる。

## 6. 実験

図8に示すデータフローグラフについて表1に示す割り当

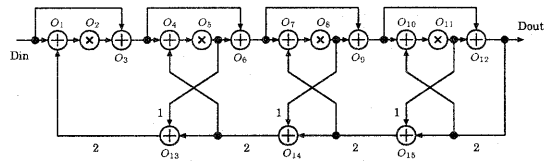


図8 実験インスタンス

|       |     |     |     |     |     |     |
|-------|-----|-----|-----|-----|-----|-----|
| 演算名   | O1  | O2  | O3  | O4  | O5  | O6  |
| レジスタ名 | R1  | R2  | R3  | R1  | R5  | R4  |
| 演算器名  | a1  | m20 | a2  | a1  | m21 | a2  |
| 演算名   | O7  | O8  | O9  | O10 | O11 | O12 |
| レジスタ名 | R2  | R8  | R9  | R9  | R11 | R12 |
| 演算器名  | a1  | m22 | a3  | a4  | m23 | a5  |
| 演算名   | O13 | O14 | O15 |     |     |     |
| レジスタ名 | R13 | R13 | R3  |     |     |     |
| 演算器名  | a2  | a3  | a3  |     |     |     |

表1 図8に対する資源割り当て

での制約下で従来の資源割り当て制約スケジューリングと提案手法とでスケジューリングを行った。加算は1コントロールステップ、乗算は2コントロールステップを要するものとした。

データフローグラフと資源割り当て、繰り返し周期を入力として整数線形計画問題を出力するプログラムをCにより作成し、生成された整数線形計画問題の解をlp\_solve 3.2を用いて求めた。lp\_solveの実行時間はAthlon 650MHzのPC上で14.8秒である。

従来の資源割り当て制約スケジューリングでの最短繰り返し周期  $T_r = 4$  を、提案手法では  $T_r = 3$  に短縮することができた。この時のスケジュールおよび各演算の生成するデータに対するレジスタ間転送数を図9に示す。図9ではデータを表す関数  $d$  を省略している。使用レジスタ数の与えられた資源割り当てからの増加は3である。

## 7. まとめ

繰り返し実行されるアルゴリズムに対する高位合成において、演算によって生成されるデータの生存期間により繰り返し周期が制限される場合に、レジスタ間転送によって繰り返し周期を短縮できることに着目し、資源割り当て駆動パイプラインスケジューリングにおいてレジスタ間転送を自動的に挿入する手法を提案した。

提案手法では、レジスタ間転送付スケジューリング問題をレジスタ間転送数の合計を最小化する整数線形計画問題として定式化することにより、レジスタ間転送の導入に伴う使用レジスタ数と結線数、消費電力の増大を抑えている。なお、より詳細な最適化のためには、レジスタ間転送を「繰り返し周期の間隔ごと」に限定することなく、 $T_r$  以下の任意の間隔で行えるようにすることが必要であり、これについては今後の課題となっている。

## 8. 謝辞

本研究の一部は、日本学術振興会科学研究費補助金 基盤 (C)

# 文 献

- [1] P.Michel, U.Lauther, and P.Duzzy, "The synthesis approach to digital system design", Kluwer Academic, 1992.
- [2] T.Yorozuya, K.Ohashi, and M.Kaneko, "Assignment-Driven Loop Pipeline Scheduling and Its Application to Data-Path Synthesis," IEICE Trans. Fundamentals, vol.E85-A, no.4 pp.819-826, April 2002.
- [3] K.Ohashi, M.Kaneko, and S.Tayu, "Assignment-space exploration approach to concurrent data-path/floorplan synthesis," Proc.IEEE Int. Conf. On Computer Design(ICCD), pp.370-375, 2000.
- [4] T.Kim, K.-S.Chung, and C.L.Liu, "A stepwise refinement synthesis of digital systems for testability enhancement," IEICE Trans. Fundamentals, vol.E82-A, no.6, pp.1070-1081, June 1999.
- [5] K.Ito and H.Kunieda, "VLSI system compiler for digital signal processin: Modulation and synchronization," IEEE Trans. Circuits&Syst., vol.38, no.4, pp.422-433, 1991.
- [6] X.Hu, S.C.Bass, and R.G.Harber, "Minimizing the Number of Delay Buffers in the Synchronization of Pipelined Systems," IEEE Trans. Comp.-Aided Design of Circuits and Systems, VOL.13,NO.12,pp.1441-1449, DEC.1994
- [7] M.Kaneko, K.Ohashi, "Post-Floorplan Control Schedule Under MAX/MIN Logic Interconnect Delays," The 16th Workshop on Circuit and Systems in Karuizawa, 2003.

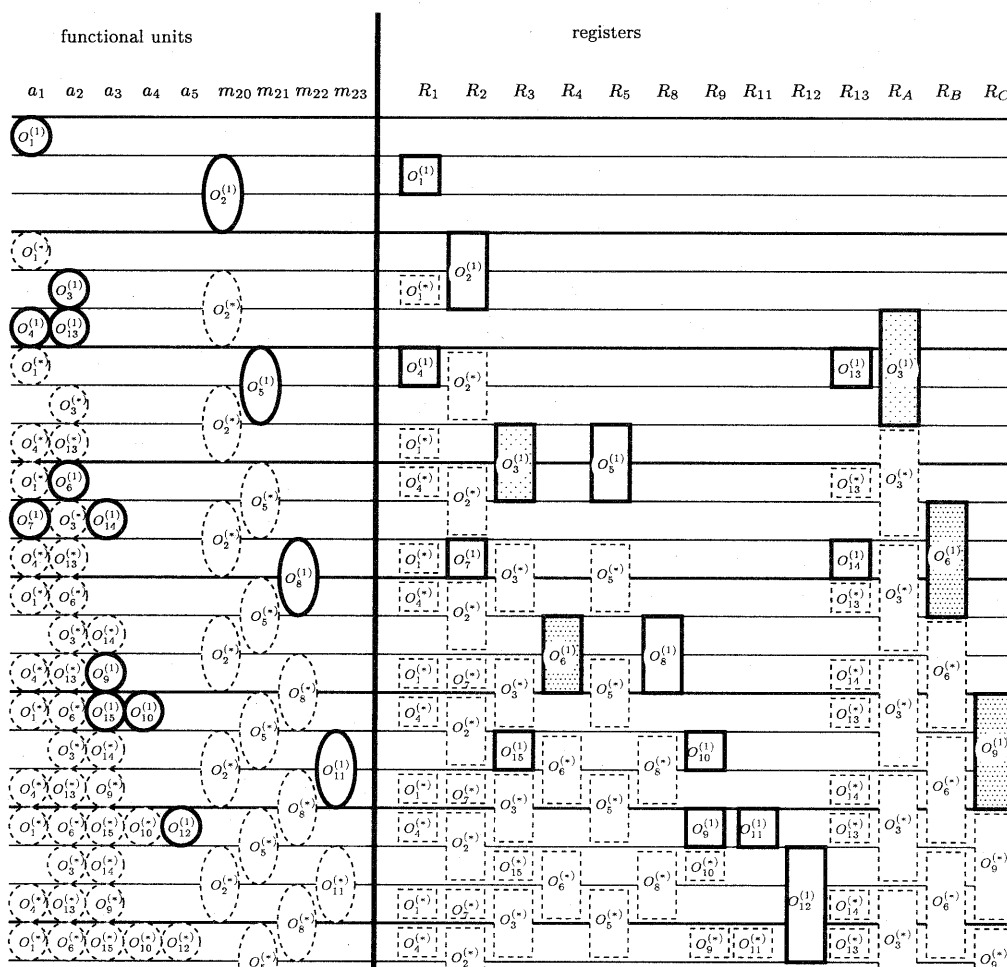


図 9 図 8 および表 1 に対するレジスタ間転送スケジュール