計算機アーキテクチャ　114－2
（１９９５．　１０．　１９）

実時間可変構造パイプライン計算機

長谷川　誠
静岡大学 工学部 知能情報工学科
(現在, 情報学部 情報科学科)
hasegawa@cs.inf.shizuoka.ac.jp

機能ユニット間の接続形態を変更することによってではなく，パイプライン中の縦続接続されている各演算ステージの演算機能そのものを実時間で動的に再定義することによって，実効的に可変構造としたパイプライン計算機について示す．このような方式を採用することにより，パイプライン計算機の本質的弱点と考えられてきた通過遅延時間を事実上無視することが可能となる．動画や音声などの連続ストリーム型のデータの柔軟な処理に適する．ウエハー・スケール・インテグレーションを効果的に利用して実現することが期待できる．

キーワード: VLSIアーキテクチャ,パイプライン,ウエファスケール・インテグレーション,可変構造,実時間機能再定義

Realtime Restructuring Pipeline Machine with on-the-fly Function Redefinitions

Makoto HASEGAWA
Comp. Sci. Dept. Fac. of Info. SHIZUOKA Univ.
hasegawa@cs.inf.shizuoka.ac.jp

On-the-fly function redefinition of pipeline segments makes it possible to novel reconfigurable machine architecture without reconfiguring the interconnections. We can avoid the startup latency of pipeline stages. It's especially suitable for processing the stream style data, ex. sound, image etc. The wafer scale integration is the another challengeable aspect of this architecture.

Keywords: VLSI architecture, pipeline, wafer scale integration, reconfigurable, on the fly function redefinition

1．まえがき

既に，VLSIプロセッサの集積度はチップ上に300万個以上のトランジスタを集積した製品の量産出荷が行われている段階にある．加減算と論理演算を実現する程度の簡単なALUならば1000～4000トランジスタ程度で実現できるはずであるから，チップ上に1000個程度のALUを並べることは，さほど困難なことではない．その一つ一つのALUにプログラムの1ステップを対応させることができるとすれば，1000ステップのプログラムを一つのチップ上に詰め込むことが可能である．

ウエファーを丸ごと一つこの目的に充てても構わないならば，1990年の時点で600万トランジスタ以上を実現した例が実在するのであるから[HORI95]，現時点でも10万～20万ステップ程度のプログラムをハードウエアで実現することができる．10年後には，集積度の向上の効果で，1000万ステップ以上のプログラムを実現することは決して困難なこととは思えない．

したがって，ある程度の大きさのプログラムをハードウエア上で直接実行させる「プログラムカウンタの無い計算機」を想定することは決して夢物語ではない．しかも，個々のプログラムステップを実現する各ALUはパイプライン動作をさせることも可能であるから，音声データや画像データのようなストリーム形のデータの処理に非常に好都合な計算機となる可能性がある．

さて，そのような計算機を実現するための手段としてどのようなものが考えられるであろうか？一つの有力な候補は，ハードウエア記述言語を使っての計算アルゴリズムのハードウエアによる実現の手段として実績のあるFPGAを用いることであろう[AMAN95,MITS94,NUMA94]．しかし，現在のFPGAの最大規模のものの性能はゲート数にして15000～20000程度，個々のゲートの動作速度が数十～百MHzで，価格にして数万円である．まず何よりも，先程の予測に比べて余りにも利用できるゲート数が少ないことに気づく．これは，実はFPGAの内部構造(図1)からすると無理からぬ点もある．まずチップ上で配線領域に専有されている部分の面積がかなり大きい．また，配線が長くなりやすいこととスイッチを経由せねばならないことが，動作速度に影響を与える．

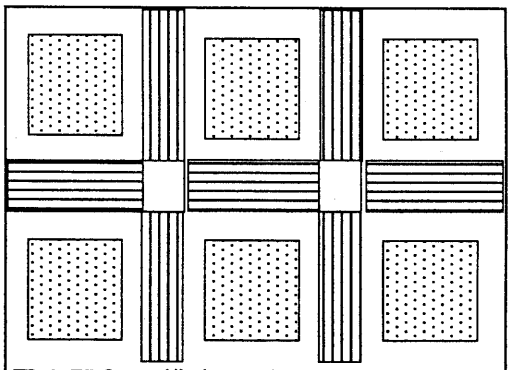

図1 FPGAの構造の一例

もう一つ別の方策として，従来のプログラム言語をなるべくそのまま使うあるいはハードウエア記述言語の機能を少し取り入れたハイブリッド言語での記述を考えるなら，パイプライン構造の上にマッピングすることが考えられる．このようにしたと

き，パイプライン計算機のスループット向上の要はパイプラインの各ステージのセグメント毎に費やされる時間の縮減にある．これを実現するために取りうる２つの方策がある．1) 各ステージの動作速度を高速化する --- これは主に素子技術と実装に依存する．2) パイプライン関数の分割をより細分化すること．方策1)は副作用なく性能向上が期待できる．方策2)を単純に実現したのでは通過時間(latency)の増大を引き起こし，ベクトル長を十分に確保できない場合の性能低下を引き起こす．ところが，パイプライン・セグメントの機能を即時的に再定義していくことにより[HASE81,HASE82]，通過時間を実効的にある定数に維持することができる．可能な分割の最小単位はビット加減算レベルであり，このとき加算器の隣接桁へのキャリー伝搬のみが限界速度を決定する．これは，1)の点でも改善となることを意味している．

局所通信だけで構成されていること，および高度の規則性を有することから，この方式はVLSI環境下で大きな可能性を有している．ただし要求される素子数は非常に多い．

今までは，演算とデータとは相互に関連はするもののそれぞれが別々の実体であるとしてとらえられてきた．これに対してオペランドのセットに対してオペレーションが対応し作用すると考えることにより，パイプライン中でのオペランドの進行に相伴って各セグメント・ステージの演算機能を即時的に再定義(on the fly redefinition)することが考えられる．物理的なパイプライン通過時間はいぜんとして残っているものの，もはやこれがオーバヘッドの源泉としての立ち上がり遅延として作用することは実効的には無くなってしまう．また，簡単にパイプライン機能の再構成を行うことができるようになるわけだから，対象となる問題との間でのマッピングの柔軟性を高めること，そしてより高度な演算を実現することが容易となる．

以下では，機能ユニット間の接続形態を変更することによってではなく，演算機能そのものを動的に再定義することによってその構造を問題に適合して実時間で再構成し(restructuring)続けるパイプライン計算機について述べる．まず，この考えの背景について説明し，関数パイプラインの構成について記し，その有用性について例示する．現時点から展望した実現のための手法と予想される性能についても述べるが，これについてはまだ大まかな議論しかできない．大部分をこれからのVLSI技術の進歩に依存しているからである．

２．成長に対応できるアーキテクチャ

アーキテクチャ実現のためのハードウエア環境を固定的に考えてはならない．現在の状況は変化を続けることこそが定常状態とさえ言ってよい．激しく進化を続けるVLSI世界の中にあってアーキテクチャもまた継続的に成長を続けることができるものでなければならない．具体的には，ある特定の集積度の時点における特定のハードウエア量だけを前提としたときに有効性を発揮するものではなく，進化していく集積度のそれぞれの局面においてチップ上に上手にマッピングできるような実現法が存在す

るものを考え出さねばならない．そのような柔軟性に富んだ基本アーキテクチャこそが強く求められている．たとえ，いくら強力ではあっても，集積度の一時点でしか有効でないようなリジッドなアーキテクチャであってはならない．このことは，大きなプログラムを作るときにとられる戦略と似通っている．明確な基本方針の下に仕様のフルセットを想定しておき，実際にはその時点の制約条件に合わせて適切なサブセットを作ることから始めて，順々に機能を拡大していって最終目標を達成する．これは必ずしも容易に実行できることではない．しかし，プログラム・インタフェースが朝令暮改されたらどういうことが生ずるであろうか．アーキテクチャにおいても利用者に与える影響はまったく同じである．

3．実時間可変構造パイプライン

長さが無限大のパイプラインのスループットは最大セグメント時間の逆数として与えられる．したがって，これをいかにして小さくするかが焦点となるが，単純にパイプラインの分割数を増すことだけによって実現したのではパイプラインの通過に要する遅延時間(Latency)が増加してしまう．通常のパイプライン計算機では，これがそのまま立ち上がり時間として性能に影響する．

パイプラインの各ステージに分割する前の演算部分における総所要時間をP，ラッチ挿入による遅延をL，分割数をmとすることにする(図2)．このとき，演算時間とラッチ時間を加え合わせたセグメント所要時間Tsegは，

$$Tseg = P/m + L \qquad (1)$$

長さmのベクトルに対するパイプライン中での総消費時間T(n)は，

$$T(n) = Tseg(n-1) + P + m$$
$$= P(m+n-1)/m + (m+n-1)L \qquad (2)$$

であり．各要素当たりの実行時間をt(n)とすると，

$$t(n) = P(m+n-1)/mn + L(m+n-1)/n \qquad (3)$$

となる．$n \to \infty$とした極限が無限長ベクトルに対する実行時間であり．

$$t(\infty) = P/m + L \qquad (4)$$

長さの短いベクトルに対する性能はn=1で代表することができ，

$$t(1) = P + Lm \qquad (5)$$

これは通過時間と一致している．このことは現在処理中のタスクがパイプラインを通過し終えるまでは他のタスクを割り付けるわけにはいかないことと対応している．

性能向上を求めて複数のパイプラインを連結(Chaining)して一本のパイプラインとして制御することが行われている．このとき，どのような変化が生ずるのだろうか．長さが無限大のベクトルのスループットは向上するが，実は通過時間もそれと同じ割合で増加してしまう．積極的な演算動作を

しない機能ユニット結合用スイッチによる通過遅延が更に付け加わることを考え合わせると，演算用パイプだけを考えた場合，短ベクトルに対する性能向上はさほど期待できない．このようなことから，短ベクトルに関する性能向上は演算素子の発展と実装技術に期待すべきであって，アーキテクチャの面からはなすすべのないものと考えられていた[HOCK81].

ところが，ベクトル・データのパイプライン中での進行に伴って，必要に応じてパイプライン・セグメントの演算機能を順番に再定義していくことを可能とするならば，これらの問題に対する解決を与えることができる．そのような，実時間で機能の再定義が可能な計算機を実時間可変構造パイプライン計算機と呼ぶこととしよう．これは，実はFeed Foward Machine[SHIG80]の一つの実現形態なのである．

簡単のために同じ長さのベクトル演算が連続する場合を考えると，要素当たりの演算時間は，次のようになる．

$$t(n) = P/m + L \qquad (6)$$

4．必要なハードウエア規模

ここでは，だいたいの見当を付ける目的で，加減算セグメントの実現のためにMeadの本[MEAD80]に載っている程度の簡単なALUを採用する場合を想定する．パス・トランジスタを使った実現では演算データ幅1bit当たり約100トランジスタを要するから，演算データ幅を16bitとするならば必要ハードウエア量は1600トランジスタ．また，演算データ幅を32bitとするならば3200トランジスタを必要とする．

チップ上での占有面積に関しては，パス・トランジスタを使った実現であるためトランジスタ数の割には小さな面積で済むはずである．

この方法では桁上げ先見加算器としての実現に比べて，一つの加算セグメントのみを見ると桁上げの点に関して速度的に不利なように見える．これに対してはSigned-Digit演算を導入することによってこれを回避し，桁上げ先見加算器よりも高速とすることが可能である．

ごくおおまかにではあるが，性能を予測してみるならば，このようなシステムの演算セグメント部の速度は10τ～20τ程度になる．あとは，システムクロックの分配に要する時間で性能が決定づけられることになる．　本システムは本質的に高い規則性を有しており，これが設計と実現を容易にする．ビット加算レベル，加算セグメントレベルでまったく同機能の要素を繰り返すことによって構成されているからである．このことにより，パイプラインにおける通信の局所性を最大限に生かして実現することができる．本システム実現のためのハードウエア量が膨大なのは確かだが，高い規則性故にVLSI上の面積要求量としては興味深い結果が得られるかも知れない．VLSIチップのテストのことを考えても，非常にモジュラリティの高いこと，各単位セグメントに含まれる機能は単純であってその組合せ方によって複雑な機能を実現している

こと，しかもチップレベルの機能そのものが規則性に富むことなどは大変に有利な点である．

5．Signed-Digit演算

通常のSigned-Digit表現による並列加減算では基数rが2よりも大きいことを要求される．ここでは2進表現のまま話を進めるためにr=2のSD加減算を与える．修正SD算法と呼ばれるものであり，2桁にまたがるキャリーが存在しえ，かつ和Sはr+1値をとる．これ以外の点では本質的にSD算法と一致する．これは以下のようにして実行することができる．

a) $z_i' + y_i' = r\, t_{i-1}' + w_i$

b) $w_i' + t_i' = r\, t_{i-1}' + w_i''$

c) $s_i' = w_i'' + z_i''$

z_i', y_i', s_i' は修正SD表現のディジットである．a,b,cの動作にそれぞれ対応する加算器を3段縦続接続し，その間にラッチを挟んだ形に実現できる．この個々の加算器は10 τ 程度の時間で動作することが期待できる．

6．適した用途

いわゆるマルチメディアのストリーム形データの処理を得意とするが，複数の文字コードが混在しているネットワーク環境におけるコード変換にも大いに適している．このような処理はキャッシュに大きく頼ったプロセッサでは1回しか使われないデータによるwipe out のために必ずしも高速処理を保証することができない．また，行列演算を得意とすることもあり，3次元グラフィックス用エンジンなどにも大きな適性が期待できることであろう．

7．結論

実時間でパイプライン・セグメントの機能を再定義することで，実効的に可変構造を実現する計算機について示した．本方式は非常に大量のハードウエアを必要とするとはいえ，近い将来においては十分に実現可能な規模である．

謝辞

この研究の最初の理解者であった佐藤利三郎東北大教授と重井芳治東北大教授に深く感謝します．著者在学時に議論の相手を努めて下さった東北大学重井研・佐藤研・木村研の皆さんにお礼申し上げます．

参考文献

[AMAN95] 天野英晴: FPGAとそのインパクト，bit Vol.27-10, pp11-21 (Oct. 1995).

[HORI95] 堀口進: ウェーハ規模超集積コンピュータの自律再構成方式に関する研究, 平成6年科研費研究成果報告書(1995)

[KANA80] 金井，長谷川，中村，重井: "フィード・フォワード計算機の構造", 電子通信学会電子計算機研究会技術報告EC80-34(1980).

[SHIG80] 重井，長谷川，中村: "Feed Foward 計算機の提案"， 情報処理学会第21回全国大会2J5 (1980).

[HASE82] M.Hasegawa: An Interconnection Scheme for Computing Sytems. Doctoral dissertation, Dep. Info. Sci., Tohoku univ. (Jan.1982).

[HASE83] 長谷川，重井: "関数パイプ",電子通信学会電子計算機研究会技術報告EC83-42(1983).

[HOCK81] R.W.Hockney and C.R.Jesshope: Parallel computers, Adam Hilgher Ltd. (1981).

[MEAD80] C.Mead and L.Conway: Introduction to VLSI systems. Addison-Wesley (1980).

[MITS94] 身次茂: FPGAの現状と将来，情報処理 Vol.35-6，pp.505-510(Jun. 1994).

[NUMA94] 沼昌弘: FPGAを利用したアーキテクチャとシステム設計，情報処理Vol.35-6，pp.511-518 (Jun. 1994).

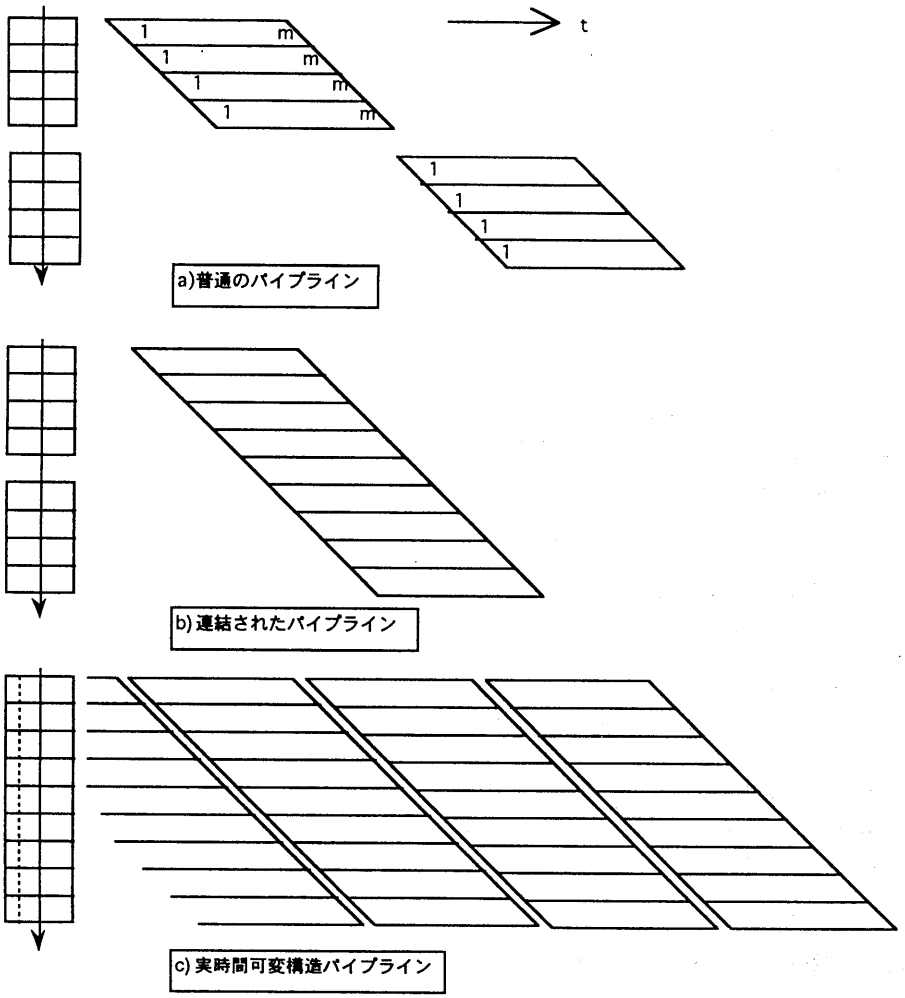

図２．従来のパイプラインと実時間可変構造パイプラインの対比