計算機アーキテクチャ 73－1
（1988. 10. 31, 11. 1）

二進木計算機による関数型言語の並列処理

槌谷 一　　高橋 義造

徳島大学工学部

本稿では，当研究室で開発中の関数型言語ＦＬの並列処理方式，および内部処理方式について述べる．最初に関数型言語ＦＬの概要を述べる．次にＦＬシステムのシステム構成について特徴を述べる．特に，言語処理系の内部処理方式について述べる．最後に，プロセスの発生，プロセス・スケジューリング等，並列処理方式を説明した後，机上シミュレーションによる並列処理結果について報告する．

Parallel Processing of a Functional Language on the Binary-tree Computer

Hajime TSUCHITANI　Yoshizo TAKAHASHI

University of Tokushima
2-1 Minami jyousanjima-cho,
Tokushima 770, Japan

This paper presents the aspect of parallel processing and implementation method of functional language FL which we are currently developing.

We will first describe the outline of a functional language FL, and then focus on the organization of FL system, especially on its implementation method.

After describing a prallel processing method, the process generation and a process scheduling method, the result of hand simulation are described.

1．はじめに

近年，計算機の発達およびソフトウェアへの負担の増大に伴って，並列計算機および並列処理言語に注目が集まっている．並列処理プログラムの開発においては，並列計算機のトポロジー，つまりハードウェアを意識して開発する必要がある．しがしながら，その様な方法ではプログラムの生産性，移植性の面で多くの問題がある．そこで，並列計算機を意識しないでも使用できる言語，ソフトウェア環境が必要となってくる．関数型言語はプログラムの参照透明性により関数の並列実行が可能で，プログラムに内在する並列性を抽出し易い．しかしながら，従来のノイマン型計算機では関数型言語の実行効率はよいものとは言えない．そこで，並列処理を意識せずに書かれたプログラムを効率よく並列処理することを目標とし，また，並列処理することでどの位実行速度が上がるかを評価するため，当研究室で開発した二進木並列計算機Ｃｏｒａ１６８Ｋ[1][4]上に関数型言語の並列処理系のインプリメントを進めている．以前Ｃｏｒａ１６８Ｋの前進である，Ｃｏｒａ１'８３[5]においてBacusのFP[7]がインプリメント[6]されていたが，今回は，FPよりも高水準な関数型言語としてHendersonが[2]において用いている純関数型言語(Purely Functional Language 以下PFLと呼ぶ)を拡張した関数型言語ＦＬの並列処理系の設計，試作を進めている．

本稿では，当研究室で開発中の関数型言語ＦＬの並列処理方式，および内部処理方式について述べる．最初に関数型言語ＦＬの概要を述べる．次にＦＬシステムのシステム構成について特徴を述べる．特に，言語処理系の内部処理方式について述べる．最後に，プロセスの発生，プロセス・スケジューリング等，並列処理方式を説明した後，机上シミュレーションによる並列処理結果について報告する．

2．関数型言語ＦＬ

関数型言語ＦＬは，HendersonのＰＦＬに様々なinfix演算子,prefix演算子，および集合演算機能等を付け加え，副作用，型付け機能を持たない言語である．同様なプログラミング・スタイルとしてLandinのISWIM[3]などがある．ＦＬによるプログラム例を以下に示す．

```
1:  qsort(x) =
2:     if x == nil then nil
3:     else {qsort([b|b<-y;b<=a]) ++ [a]
4:              ++ qsort([b|b<-y;b>a]) where a:y = x}
```

プログラム１　Quicksort program

プログラム１において，'++'はappend関数，':'はcons関数のinfix演算子である．また，3,4行目の関数qsortの再帰呼び出しの引数において集合演算の機能を用いている．

3．ＦＬシステム

現在，試作中のＦＬシステムにおいては，関数の評価方式として，先行評価

(eager evaluation)を用い，引数の並列評価のみを行う．しかし全ての引数を並列処理すると，並列処理の粒度があまりにも小さくなり，オーバーヘッドが大きくなるので，ユーザ定義関数についてのみ引数の並列評価を行う．ＦＬ第２版においては怠け評価(lazy evaluation)による，関数本体の並列実行を実現する予定である．

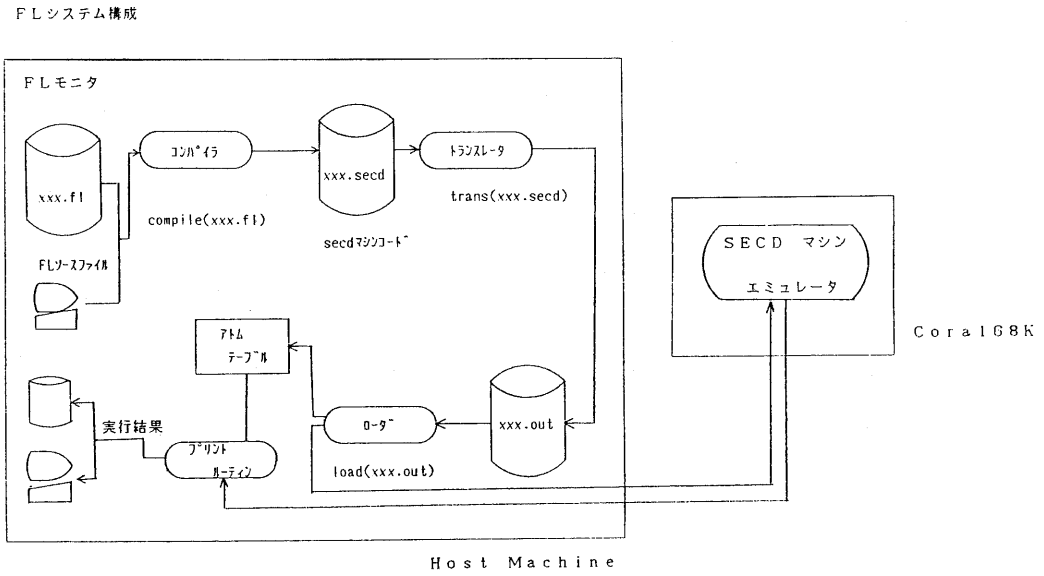

図１　ＦＬシステム

表１　Ｃｏｒａｌ６８Ｋの仕様

要素	項目	Coral68K
ホスト計算機	モデル	シャープＩＸ－５
	CPU	MC68000（10MHz）
	メモリ	4MB
	ﾊｰﾄﾞﾃﾞｨｽｸ	60MB
	OS	UNIX SYSTEMV
プロセッサ要素	台数	63台
	CPU	MC68000（10MHz）
	ROM	16KB
	RAM	512KB
転送速度	HOST-PE間	37KB／秒
	PE-PE間	2．0MB／秒

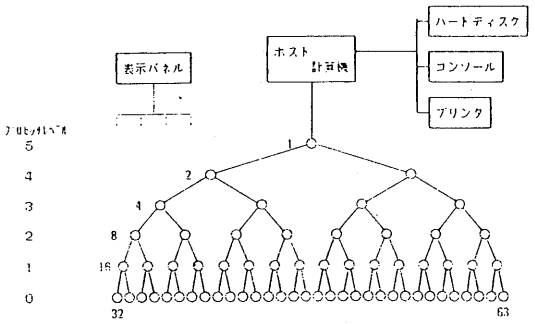

図２　Ｃｏｒａｌ６８Ｋの構成図

3.1　ＦＬの処理方式

ＦＬ処理系では，図１のようにホスト計算機上でソースプログラムを仮想機械コードにコンパイルし，それをトランスレータによって実行可能なバイナリーコードに変換する．そして，それを二進木並列計算機Ｃｏｒａｌ６８Ｋ上にローデ

ィングし，Ｃｏｒａｌ上で仮想機械をエミュレートする．

仮想機械としては，Landinの考案したSECD機械[3]に並列処理機能を付け加えたものである．

3.2 Ｃｏｒａｌ６８Ｋ

Ｃｏｒａｌ６８Ｋは図２に示すように二進木状に結合された６３台のプロセッサ要素と，その根に接続されたホスト計算機より構成されている．Ｃｏｒａｌ６８Ｋの構成要素と仕様を表１にまとめて示す[4]．

3.3 ＳＥＣＤ機械

Landinがその原型を考案したSECD機械の名前は，それを構成する次の４つの主要レジスタの名前に由来している．

Ｓ（stack）式の値を計算するときの中間結果を入れる．

Ｅ（environment）計算の途中で各変数に束縛される値を入れる．

Ｃ（control list）実行されるべき機械語プログラムを入れておく．

Ｄ（dump）新しい関数呼び出しが起こったときに，他のレジスタの内容をしまっておくスタックの一種．

ＦＬではその他に，どちらの方向のプロセッサからプロセスを受け取ったかを記憶しておくレジスタなどが必要となる．

3.4 SECD機械命令

基本命令２１個，遅延評価を実行する命令３個，計２４個の命令がある．機械語プログラムはS式で表される[2]．（図３，４参照）今回のｲﾝﾌﾟﾘﾒﾝﾄでは基本命令のみを用い，AP命令をﾌｪｯﾁすることにより，並列ﾌﾟﾛｾｽを発生する．

・基本命令

LD	(load)	LDC	(load constant)
LDF	(load function)	AP	(apply function)
RTN	(return)	DUM	(create dummy environment)
RAP	(recursive apply)	SEL	(select subcontrol)
JOIN	(rejoin main control)		
CAR	スタックの上端の要素のcarをとる．		
CDR	スタックの上端の要素のcdrをとる．		
ATOM	スタックの上端の要素にatom述語をとる．		
CONS	スタックの上端の２つの要素のconsを作る．		
EQ	スタックの上端の２つの要素にeq述語を適用する．		
ADD SUB MUL DIV REM	スタックの上端の要素に算術演算を適用する．		
STOP	停止		

・遅延評価命令

LDE	(load expression:式をﾛｰﾄﾞする)
APO	(apply prameterless function: 引数なし関数を適用する)
UPD	(return and update: 復帰して更新する)

図３　ＳＥＣＤ機械命令

フィボナッチ数列

```
{fib(6) whererec fib = λ(x) if n < 3 then 1 else
                              fib(n-1) + fib(n-2)}
                        ↓ ｺﾝﾊﾟｲﾙ
1: (DUM LDC NIL
2:    LDF (LD (0 . 0) LDC 2 LEQ SEL (LDC 1 JOIN)
3:          (LDC NIL LD (0 . 0) LDC 1 SUB CONS
4:           LD (1 . 0) AP
5:           LDC NIL LD (0 . 0) LDC 2 SUB CONS
6:           LD (1 . 0) AP
7:           ADD JOIN) RTN)
8:    CONS LDF (LDC NIL LDC 2 CONS LD (0 . 0)
9:              AP RTN) RAP)
```

図４　コンパイル例

3.5　データ構造

リスト構造を用いると処理系も簡潔になり，実行効率も良いと思われるが，プロセッサ間の通信においてはオーバーヘッドが大きい．ゆえに本処理系においては，通信のオーバーヘッドを緩和するために列(sequence)がデータ構造として採用されている．

3.5.1　列の表現

本処理系においては，Ｓ式で表された機械語命令を効率よく処理するため，上述のトランスレータ（図１参照）によって列構造を持った中間コードに変換し，Ｃｏｒａ１６８Ｋ上でエミュレートする．以下に変換例を示す．

((abc) (a . b))

↓　トランスレート

LIST	01*	02	abc	01	02	a	.	b	01	00

* ﾈｽﾃｨﾝｸﾞﾚﾍﾞﾙ

図５　中間コードの変換例

3.6　エミュレータの内部処理方式

エミュレータ内での処理方式としては，リスト操作のオーバーヘッドを緩和するため，次のような方式を採っている．

ヒープメモリ内に各レジスタの内容を格納する．各レジスタは格納すべきリストの各要素に対するポインタを格納する．この様にする事により，列構造データに対するリスト操作(cons car cdr等)を効率よく処理できる．以下にCONS命令における処理例を示す．

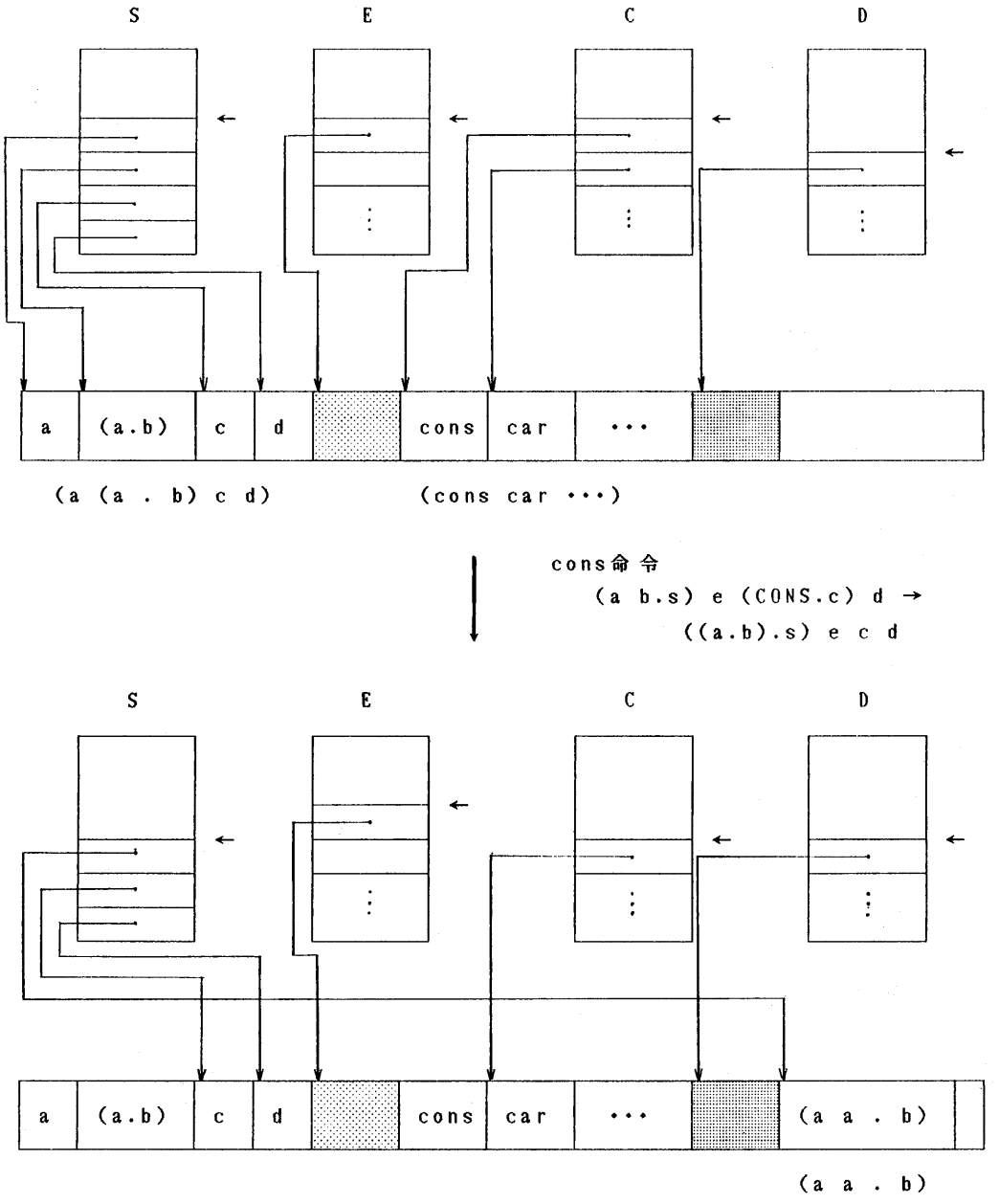

図６　エミュレータの内部処理

４．並列処理方式

4.1　並列プロセスの発生

機械語命令AP(apply function)に出くわすと，周りのプロセッサが暇かどうか調べ，手が空いているプロセッサがあれば，そのプロセッサに現時点における環

境および実行しようとしている関数へのポインタを渡すことにより，並列プロセスを発生する．プロセスを委ねたプロセッサはそのプロセスから返ってくる値を格納するＳレジスタの場所にマークを付け，自分は次の実行可能なステップに進んでいく．一方，プロセスをもらったプロセッサはもらった方向をレジスタに記憶しておき，プロセスが終了したらもらった方向のプロセッサに割り込みをかけ値を返す．（図７）

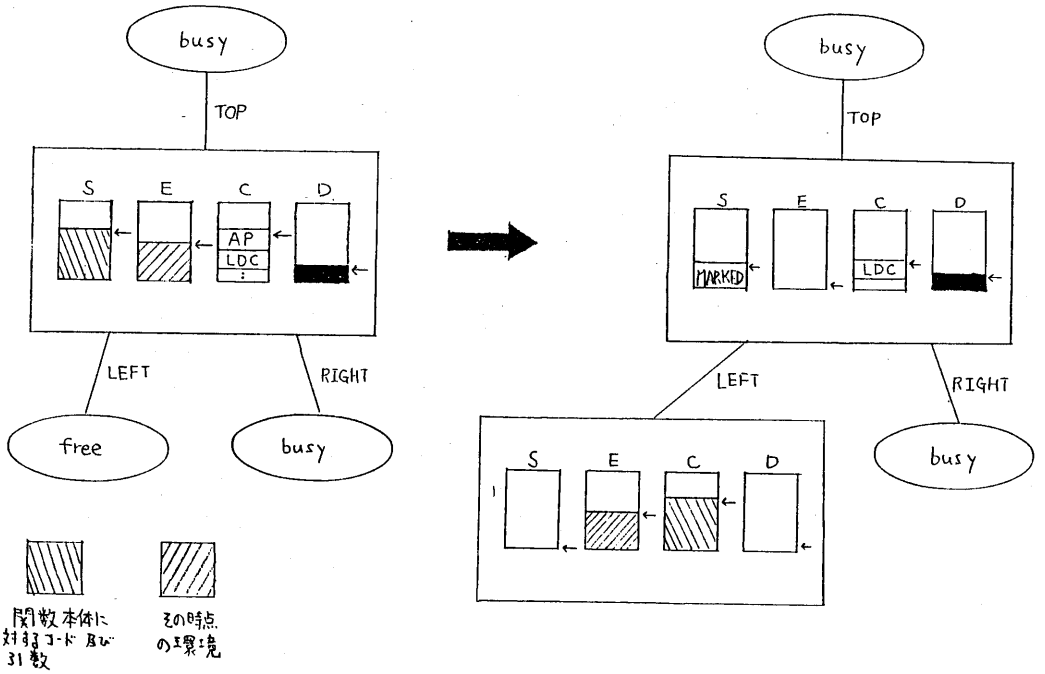

図７　プロセスの発生と割当

4.2　プロセス・スケジューリング

並列実行可能なプロセスに出くわすと，top,left,right三方向のいずれかにプロセスの実行を委ねる．プロセス割当の優先順位によって，プログラムの実行時間およびプロセッサ間の通信回数が異なってくるのでプロセス割当の優先順位は調整できるようにした．

4.3　机上シミュレーションによる評価

フィボナッチ数を求める机上シミュレーションを行った．関数一つ当りの処理をt時間とすると，フィボナッチ数６の場合，プロセッサ１台で実行すると38t時間かかり，プロセッサ７台使い，プロセス割当の優先順位をtop,left,own,rightとした場合12t時間，通信回数は10回，優先順位をtop,left,right,ownにした場合は11t時間，通信回数は14回であった．プロセッサ使用率は約50%弱であり，並列処理を意識せずに書かれたものとしてはまずまずの効率である．（図８）

フィボナッチ数列　プロセス発生例　　12/38　通信回数10

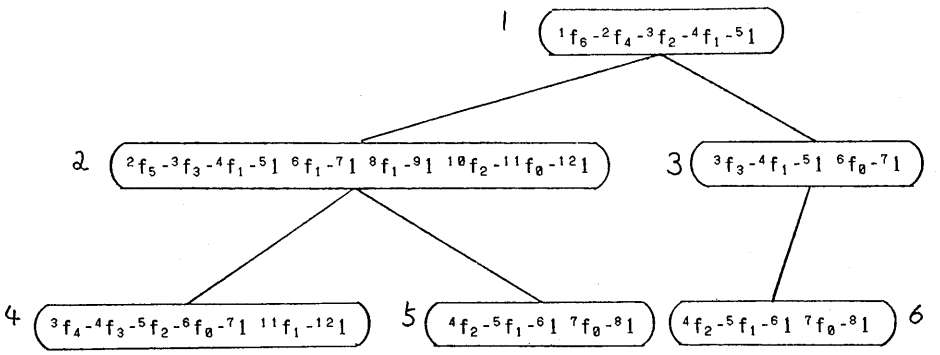

図８　フィボナッチ数列シミュレーション

５．おわりに

本報告においては，関数型言語ＦＬの並列処理方式の概要をを述べた．3.6で述べた方式においては，ｶﾞﾍﾞｰｼﾞ・ｺﾚｸｼｮﾝをする必要がある．今後，並列ＧＣについて考察する．なお現在プロトタイプＦＬをインプリメント中である．

参考文献

[1]遠藤俊雄，松尾賢二，白方新洋，槌谷　一，高橋義造：２進木マシンＣｏｒａ１６８Ｋのシステム構成と性能評価，計算機アーキテクチャ研究会66-5(1987-7)

[2]P.Henderson:FUNCTIONAL PROGRAMMING Application and Implementation, Prentice-Hall International,INC.(1980)（訳）関数型プログラミング，日本コンピュータ協会(1985)

[3]P.J.Landin:The next 700 programming languages. CACM,9,157-164(1966)

[4]高橋義造，松尾賢二：二進木計算機ＣＯＲＡＬ６８Ｋのプログラム実行制御方式，オペレーティング・システム研究会40-3(1988-9)

[5]Y.Takahashi,N.Wakabayashi,Y.Nobutomo: "A Binary Tree Multiprocessor : CORAL" Journal of Information Processing Vol.13,No4 PP280-287

[6]西山和義，高橋義造：２進木構造ﾏﾙﾁﾌﾟﾛｾｯｻCORAL上での関数型言語FPの並列処理，情報処理学会第２５回全国大会

[7]J.Bacus,:Can programming be liberated from the von Neumann style? A functional style and its algebra of programs.CACM 21(8) 613-41(1978)