System Design of a Parallel Direct Execution Device for Arithmetic Statements

SHUNZO MORISHITA*, YASUYOSHI INAGAKI* AND TERUO FUKUMURA*

1. *Introduction*

A design and an analysis are given for the device, named PADEM, capable of parallel direct execution (execution without a compiler) of FORTRAN or ALGOL arithmetic statements. This device can execute the syntax analysis operation and the arithmetic operation in parallel using two push down storages.

The processing time and the memory capacity of this device necessary for the parallel direct execution are evaluated theoretically, and the results are checked by the simulation experiment on the computer HITAC 5020E using about 50 typical arithmetic statements.

It can be expected that this device, a hardware version of "one-pass-load-and-go" compiler will decrease the overhead of software. The idea shown in this paper will be also applicable to the direct execution of other types of FORTRAN or ALGOL statements.

Bashkov [1], [2] has already proposed the direct execution circuit which is essentially sequential circuit realization of compiler and thus requires a large number of states and very complex processing circuits. Compairing with this circuit, the control system of our PADEM is exceedingly simplified by using two push down storages and restricting the statement to the arithmetic one.

2. *Design of Parallel Direct Execution Device*

The arithmetic statements under consideration contains only binary operators α, of which operational priorities $\pi(\alpha)$ are defined as seen in Table 1. The variable is denoted by v and given the priority $\pi(v)=7$.

The arithmetic statements are assumed to be delimited by the initial symbol

Table 1. Operators and operational priority.

α	¢, $	=	)	(	+, −	*, /	↑
$\tau(\alpha)$	0	1	2	3	4	5	6

+ (addition), − (subtraction)
* (multiplication), / (division)
↑ (power)

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 11, No. 7 (1970), pp. 400-410.
* Faculty of Engineering, Nagoya University.

¢ and the end symbol $. It is also assumed that all variables are represented by one character.

The devices used in PADEM are as follows:

(1) P_T: push down storage (*pds*) provided for the translation of the source statement to the inverse polish string

(2) P_E: *pds* provided for the execution of arithmetic operations

(3) R_T: temporal memory register used when an operator symbol is stored into *pds* R_T

(4) R_E: temporal memory register used when an operand is stored into *pds* P_E

(5) R_B: temporal memory register to store the next operator to be executed

(6) DCD: decorder of operator in R_B generating instruction

(7) N_P: depth counter of parenthesis in source statements

(8) F_1, F_2: flip-flop indicators; $F_1=1$ when PADEM is in execution mode and $F_1=0$ otherwise; $F_2=1$ when R_B is occupied by some operator and $F_2=0$ otherwise

The state transition diagram of PADEM thus composed of the above devices is shown in Fig. 1.

Notations and symbols in Fig. 1 are interpreted as follows:

(1) $f/\{X\}$: If the condition f holds, a set $\{X\}$ of instructions is generated. The symbol — in the expression $f/$— means no instruction.

(2) $v/\rightarrow R_B$, READ: If the symbol just read in is a variable v, it is transfered into R_B and the instruction READ to read the next symbol is generated.

(3) $\pi(P_T)<\pi(R_T)/R_T\rightarrow P_T$, READ: If the priority $\pi(P_T)$ of the symbol on the top of *pds* P_T is less than that of the symbol in R_T, then the content of R_T is stored into *pds* P_T and READ instruction is generated.

(4) $R_B\rightarrow$DCD (β): The operator stored in R_B is decoded by DCD, giving the instruction β to drive the corresponding execution device.

(5) $N_P\uparrow(N_P\downarrow)$: The content of the counter N_P is increased (decreased) by one.

(6) LOCK: By this instruction PADEM control system halts until the condition for the next transition is satisfied.

(7) START and END: The instruction START is given to PADEM by the outer control unit to start processing of a given statement, and by the instruction END, the PADEM signals its finishment of processing to the outer control unit.

(8) If any conditions indicated on edges emanating from a state do not hold, then the PADEM falls in the error state (state 26).

(9) The PADEM system returns unconditionally from the final state 25 to the initial state 0.

(10) The double circle indicates the possibility of temporal halt at the circled state.

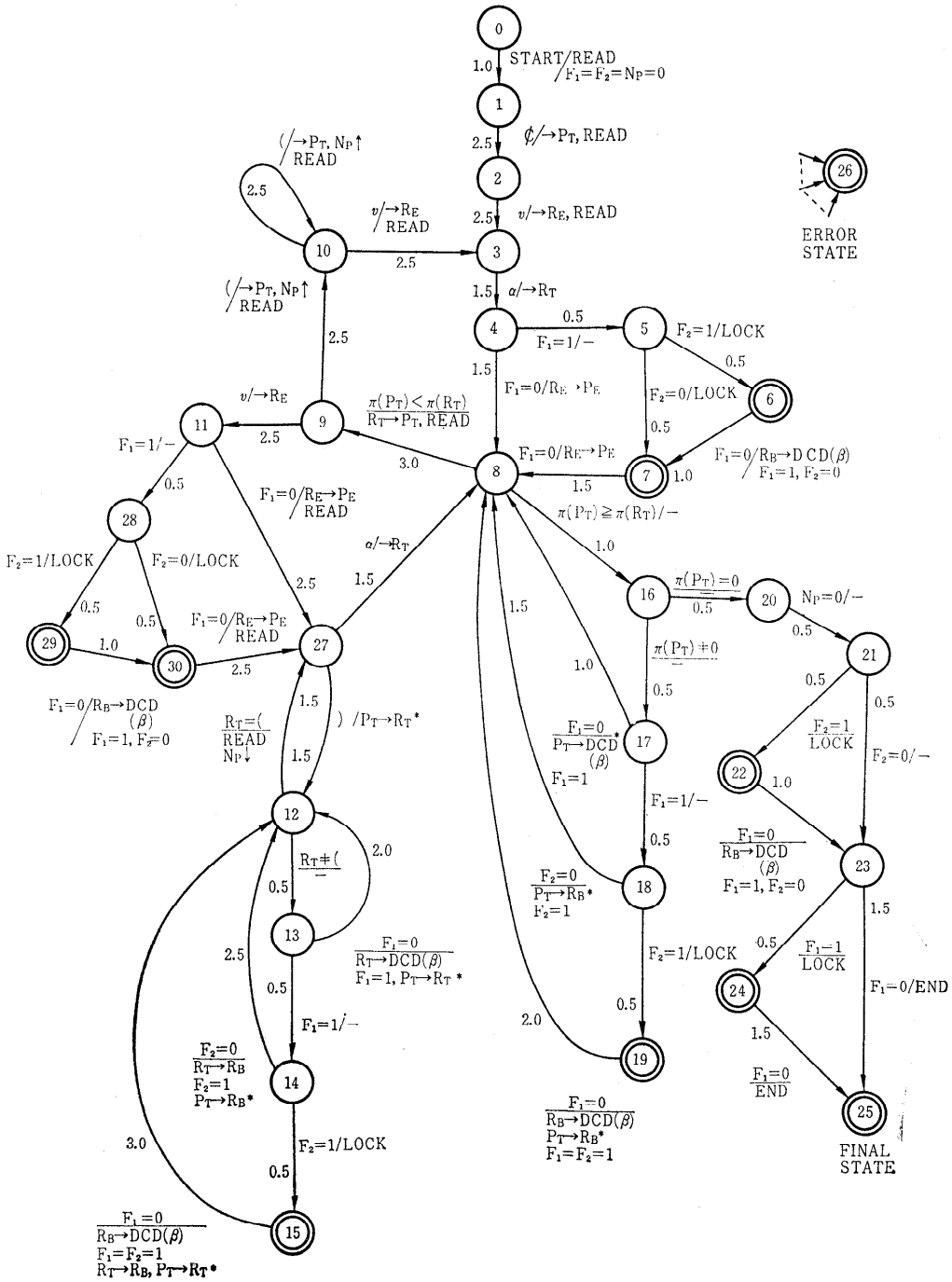
START/READ
F1=F2=NP=0
PT, READ
PT, NP↑ READ
RE READ
RE, READ
ERROR STATE
PT, NP↑ READ
RT
F2=1/LOCK
F2=0/LOCK
F1=1/−
F1=0/RE→PE
π(PT)<π(RT) RT→PT, READ
F1=0/RB→DCD(β) F1=1, F2=0
F1=0/RE→PE READ
π(PT)≧π(RT)/−
π(PT)=0
NP=0/−
π(PT)≠0
F1=0 PT→DCD* (β) F1=1
F2=1 LOCK
F2=0/−
F1=0/RB→DCD (β) F1=1, F2=0
RT=(READ NP↓
) /PT→RT*
RT≠(
F1=0 RT→DCD(β) F1=1, PT→RT*
F2=0 RT→RB F2=1 PT→RB*
F2=0 PT→RB* F2=1
F1=0 RB→DCD(β) PT→RB* F1=F2=1
F1=0 RB→DCD(β) F1=F2=1 RT→RB, PT→RT*
F1=1 LOCK
F1=0/END
F1=0 END
FINAL STATE

Fig. 1.

(11) Numbers attached to each edges indicates the required time for the corresponding state transition (reffer to sec. 4).

As an illustrative example the process of PADEM for processing the statement ¢ $a=b/c$ \$ is shown in the following:

0⇒1 (¢→P_T)⇒2 (a→R_E)⇒3 (=→R_T)⇒4 (R_E→P_E)⇒8 (R_T→P_T)⇒9 (b→R_E)⇒11 (R_E→P_E)⇒27 (/→R_T)⇒8 (R_T→P_T)⇒9 (c→R_E)⇒11 (R_E→P_E)⇒27 (\$→R_T)⇒8⇒16 (Division b/c begins.)⇒8⇒16⇒17⇒18 (=→R_B)⇒8⇒16⇒20⇒21⇒22 (Temporal halt until the execution $r_1 \triangleq b/c$ finishes. After then the execution $a=r_1$ begins.)⇒23⇒24 (Temporal halt until the execution $a=r_1$ finishes.)⇒25 (end)

Finally, it is noted that by a little modification of the state diagram of PADEM it becomes capable of treating the wider ranges of arithmetic statements including unary minus, subscripted variables and etc.

3. *Estimation of Memory Capacity Necessary for pds*

First we define the level of statement, i.e., level n of the statement k is defined as

$$n \triangleq \max_{\alpha \in S_k} \{\pi(\alpha)\} - \min_{\alpha \in S_k} \{\pi(\alpha)\} + 1 \qquad (1)$$

where S_k is the set of operators appearing in the statement k. Depth m of the nest is measured by the number of consecutive left parentheses. The maximum nest is the deepest one in a given arithmetic statement. Using these quantities and the allowable length l of an arithmetic statement, we estimate the memory capacities M_T and M_E of *pds* P_T and P_E, respectively, as follows: The memory capacities M_T and M_E required for processing an arithmetic statements with level n, allowable length l and maximum nest depth m_R are constrained by inequalities

$$M_T \leqq (n-3)m+n \qquad (2)$$

$$M_E \leqq (n-4)m+n \qquad (3)$$

where $m \triangleq \min\{m_0, m_R\}$ and $m_0 = [(l-4)/(2l-6)]$.

4. *Estimation of the Processing Time Improvement*

As already described, PADEM system performs translation and execution of arithmetic statements in parallel as far as possible. Thus the total processing time will decrease compairing to the serial processing. Degree of the improvement of the processing time will be discussed below by compairing our PADEM system with a hypothetical system, called serial system, which translates a given arithmetic statement to the corresponding polish string and then executes the arithmetic operations sequentially according to the polish string.

First, several definitions are given:

T_t: Total processing time for an arithmetic statement by PADEM system.

T_p: Proper execution time for an arithmetic statement by PADEM system, which cannot be embeded into the translation time.

$T=T_t-T_p$: Translation time for an arithmetic statement by PADEM system.

T_s: Time required to execute a sequence of arithmetic operations according to the polish string by the serial system.

Two measures representing the improvement of processing time are defined by

$$\varepsilon \triangleq T_s/T_p \tag{4}$$

and

$$\varepsilon' \triangleq (T_q+T_s)/T_t=(T_t-T_p)/T_t \tag{5}$$

Meaning of measure ε is obvious. The alternative measure ε' is provided to compare the total processing times of two systems.

Next we make a reasonable assumption on the execution time of elementary operations as shown in Table 2, where every execution time is given as ratio to that of addition time. They are estimated from the real data of several computers.

Using Table 2, we determine the transition time t_j^i along an edge d_j^i of the state transition diagram of PADEM. The time t_j^i is attached to the edge d_j^i in Fig. 1. We also determine the processing time for operators appeared in arithmetic statements as shown in Table 3, basing upon the execution time of elementary operations of Table 2.

Now by tracing the moves of PADEM for about fifty typical arithmetic statements on the state transition diagram of Fig. 1, we have obtained an estimate of T_p of an arithmetic statement as follows:

$$T_p=5.5N_3+1.5N_4 \tag{6}$$

where N_3 and N_4 are the numbers of operators "/" (division) and "=" (substitution), respectively. Thus, using eq. (4) we have an estimate of $\varepsilon=T_s/T_p$ as follows:

Table 2. Relative execution time of fundamental operations.

Operation	Time
Add, Sub, Load, Store	1.0
Mult	5.0
Div	10.0
Conditional Branch	0.5
Compare	1.0
Unconditional Branch	0.5
Shift	2.0
Indexing	1.0

Table 3. Operation time.

i	Operator	Operation time
1	+, −	3.5
2	*	7.5
3	/	12.5
4	=	2.5

$$\varepsilon_d \triangleq T_s/T_p=(3.5N_1+7.5N_2+12.5N_3+2.5N_4)/(5.5N_3+1.5N_4) \tag{7}$$

where N_1 is the number of +, − (addition, subtraction) operators and N_2 is the number of * (multiplication) operator in the arithmetic statement.

5. *Computer Simulation of PADEM System*

Table 4.* An example of a simulation result. ($V=A/B$)
Simulation of the direct execution circuit. Simulated stat.=1 8 2 8 7 8 1
¢$V=A/B$\$

PASS NODE	(T_t) TOTAL TIME	(T_p) EFF. EEX. TM.	(T_s) SER. SUM. TM.	LOCK TIME	LOCK NODE	
1	1.00	0.0	0.0			
2	3.50	0.0	0.0			¢→P_T
3	6.00	0.0	0.0			
4	7.50	0.0	0.0			
8	9.00	0.0	0.0			
9	12.00	0.0	0.0			=→P_T
11	14.50	0.0	0.0			
27	17.00	0.0	0.0			
8	18.50	0.0	0.0			
9	21.50	0.0	0.0			1→PT
11	24.00	0.0	0.0			
27	26.50	0.0	0.0			
8	28.00	0.0	0.0			
16	29.00	0.0	0.0			
17	29.50	0.0	0.0			
EXEC OF OPER IN PUSHT (LT) LT=3 IPUSHT (LT)=7 KK=IPUSHT (LT)=7						EXECUTION $A/B(\triangleq R_1)$ BEGINS
8	30.50	0.0	12.50			
16	31.50	0.0	12.50			
17	32.00	0.0	12.50			
18	32.50	0.0	12.50			
8	34.00	0.0	12.50			
16	35.00	0.0	12.50			
20	35.50	0.0	12.50			
21	36.00	0.0	12.50			
				5.50	22	
EXEC OF OPER IN BUFF REGISTOR JBUFF=2 M=7 IFF 2=1						EXECUTION $V=R_1$ BEGINS
22	42.00	5.50	15.00			
23	42.50	5.50	15.00			
24	44.00	6.50	15.00			
				1.00	24	
25	45.50	6.50	15.00			
EXECUTION IS FINISHED.						END

* This table except the last column is a reprint of line-printer output.

The simulation of PADEM system is made to check the logics and the performance of our PADEM system designed in sec. 1. The validities of our estimations of memory capacity and the processing time are also been confirmed by the simulation.

The outline of our simulation program is as follows: The main program, by which our simulation is carried out, has about 600 FORTRAN statements.

Table 5. The simulation result and theoretical evaluation of ε and ε'.

No.	Type of Arithmetic Statement	Number	T_t	T_p	T_s	ε'_s	ε_s	ε_d
1	$v=a$	627	25.00	0.00	2.50	1.10	∞	1.7
2	$v=a+b$	168	38.00	0.00	6.00	1.16	∞	4.0
3	$v=a*b$	40	41.00	1.50	10.00	1.21	6.7	6.7
4	$v=a/b$	59	46.00	6.50	15.00	1.18	2.3	2.1
5	$v=a*b+c$	19	51.00	0.00	13.50	1.26	∞	9.0
6	$v=a+b*c$	29	54.00	1.50	13.50	1.22	9.0	9.0
7	$v=a/b+c$	1	56.00	5.00	18.50	1.24	3.7	2.6
8	$v=a+b/c$	10	59.00	6.50	18.50	1.20	2.8	2.6
9	$v=a+b+c$	17	50.00	0.00	9.50	1.19	∞	6.3
10	$v=a*b/c$	8	59.00	6.50	22.50	1.27	3.5	3.2
11	$v=a/b*c$	1	59.00	6.50	22.50	1.27	3.5	3.5
12	$v=a+b*c*d$	1	67.00	1.50	21.00	1.29	14.0	14.0
13	$v=(a+b)/c$	4	62.50	6.50	18.50	1.19	2.8	2.6
14	$v=a+(b+c)$	1	54.50	0.00	9.50	1.17	∞	6.3
15	$v=a*b+c*d$	3	67.00	1.50	21.00	1.29	14.0	14.0
16	$v=a*(b+c)$	3	57.50	1.50	13.50	1.21	9.0	9.0
17	$v=a+b/c*d$	2	72.00	6.50	26.00	1.27	4.0	3.7
18	$v=a/b*c/d$	1	77.00	11.50	35.00	1.31	3.0	2.8
19	$v=a/b+c/d$	2	77.00	11.50	31.00	1.25	2.7	2.5
20	$v=a+b*c*d$	3	67.00	1.50	21.00	1.29	14.0	14.0
21	$v=a+b*c/d$	2	72.00	6.50	26.00	1.27	4.0	3.7
22	$v=a/(b*c)$	1	63.50	6.50	22.50	1.25	3.5	3.2
23	$v=(a+b)*c$	1	57.50	1.50	13.50	1.21	9.0	9.0
24	$v=a/(b+c)$	1	62.50	6.50	18.50	1.19	2.8	2.6
25	$v=a+b*c+d*e$	6	80.00	1.50	24.50	1.29	16.3	16.3
26	$v=(a+b*c)/d$	3	57.50	6.50	26.00	1.26	4.0	3.7
27	$v=a+(b+c)*d$	1	70.50	1.50	17.00	1.22	11.3	11.3
28	$v=a+b*(c+d)$	2	70.50	1.50	17.00	1.22	11.3	11.3
29	$v=a*(b+c/d)$	3	76.50	5.50	26.00	1.27	4.7	3.7
30	$v=(a+b+c)*d$	1	69.50	1.50	17.00	1.22	11.3	11.3
31	$v=a*(b+c)/d$	1	75.50	6.50	26.00	1.26	4.0	3.7
32	$v=(a+b)*c+d*e$	1	83.50	1.50	24.50	1.28	16.3	16.3
33	$v=(a*(b+c))/d$	1	79.00	6.50	26.00	1.25	4.0	3.7
34	$v=a*b+c*d+e*f$	3	93.00	1.50	32.00	1.33	21.3	21.3
35	$v=a*b*c+d*e*f$	2	93.00	1.50	36.00	1.37	24.0	24.0
36	$v=(a+b)/(c+d)$	2	79.00	6.50	22.00	1.20	3.4	3.1
37	$v=(a*b+c*d)*e+f$	1	93.50	0.00	32.00	1.34	∞	21.3
38	$v=(a+b/(c*d))*e$	1	91.00	3.50	33.50	1.33	9.6	4.8
39	$v=a+b*c/d+e*f/g$	1	114.50	10.50	49.50	1.34	4.7	4.0
40	$v=a*b+c*d*(e+f*g)$	1	109.50	1.50	39.50	1.35	26.3	26.3
41	$v=a*b+(c+d*e*f)*g$	1	108.50	1.50	39.50	1.35	26.3	26.3
42	$v=(a*b+c+d)*e/f+g$	1	110.50	5.00	40.50	1.32	8.1	5.8
43	$v=(a+b)*c*d+e*f+g$	1	106.50	0.00	35.50	1.33	∞	23.7
44	$v=((a+b)*c+d)*e+f$	1	98.00	0.00	28.00	1.29	∞	18.7
45	$v=a+(b+c*d+e)*f/g$	1	114.50	6.50	40.50	1.30	6.2	5.8
46	$v=a*b+c*(d+e/f)*g$	1	115.50	5.50	44.50	1.34	8.1	6.4
47	$v=a*b+c*d*e*f+g*h$	1	119.00	1.50	47.00	1.38	31.3	31.3
48	$v=(a+b)*c+(d+e)*f*g$	1	112.00	1.50	35.50	1.30	23.7	23.7
49	$v=(a*b+c)*d*e+f*g+h$	1	119.50	0.00	43.00	1.36	∞	28.7
50	$v=a+(b*c+d+e*f)*g/h$	1	126.50	6.50	48.00	1.33	7.4	6.9
51	$v=(a*(b+c*d))/(e*f)$	1	110.50	6.50	41.00	1.31	6.3	5.9
52	$v=(a+b*c*d*e)/(f+g*h*i)$	1	145.00	6.50	59.50	1.37	9.2	8.5
53	$v=(a+b/c*(d*e+f*g*h))/(i+j/k*l)$	1	194.50	12.50	88.00	1.39	7.0	4.9

Sixteen subprograms, which have about 400 FORTRAN statements in total, are used to compute the memory capacity and the processing time and also used to make tables and to treat input and output data.

The compiling time is about 6 seconds and the average time required for the simulation of an arithmetic statement is about 1.5 seconds.

The results of the simulation are given as the table of the state transition sequence, the elapsed time, and the conditions of execution and the temporaly halting time. Table 4 shows an example of such a simulation result given as the output of computer.

The arithmetic statements used for the simulation are 1,047 in number and gathered from scientific and technological computer programs appeared in the "Algorithms" of the journal "Information Processing" published from 1962 to 1965. They are grouped into 53 types of classes as shown in Table 5, where $+$ denotes $+$ or $-$.

The degree of the improvement of the processing time ε_s and ε'_s computed by using the simulation results in eqs. (4) and (5) are shown in Table 5 together with T_t, T_p, T_s and ε_d.

The results concerning the memory capacity of P_T and P_E are shown in Table 6 together with M_T and M_E of eqs. (2) and (3).

These tables show that we have good estimations of memory capacity and the processing time.

Table 6. The memory capacities M_T, M_E of two *pds*'s P_T, P_E.

m_R		0	1	2	9	19
Simulation results	M_T	4	7	7	31	60
	M_E	4	6	6	22	41
Theoretical results	M_T	5	8	11	43	83
	M_E	5	7	6	34	64

6. *Concluding remarks*

As an attempt to make hardware take over the load of software, we have proposed a parallel direct execution device for arithmetic statements, named PADEM. We have also estimated the memory capacities of *pds* P_T and P_E used in PADEM and the processing time. By the computer simulation of PADEM system validities of these results are verified experimentally.

References

[1] Bashkow, T. R.: A Sequential Circuit for Algebraic Statement Translation, *IEEE Trans., EC-13*, pp. 102—105 (1964).

[2] Bashkow, T. R., A. Sasson and A. Kronfield: System Design of a FORTRAN Machine, *IEEE Trans., EC-16*, pp. 485—499 (1967).