# AN ON-LINE SQUARE ROOTING ALGORITHM

Miloš D. Ercegovac

UCLA Computer Science Department
University of California
Los Angeles, CA 90024

Abstract

An on-line algorithm for computing square roots in a radix 2, normalized floating-point number system with the redundant digit set {-1, 0, 1} is described. The algorithm has on-line delay of one and it is amenable for modular implementation. A systematic approach, used in deriving this algorithm, is presented in detail.

## 1. Introduction

We present here an on-line algorithm for computing square roots in radix 2 number representation system with a redundant digit set {-1, 0, 1}. An on-line algorithm is characterized by the following properties:

i) The result is computed in a digit-by-digit fashion, beginning with the most significant digit.

ii) The j-th digit of the result is computed as soon as $(j+\delta)$ digits of the operands are available. The on-line delay $\delta$ is a small positive integer.

The on-line algorithms for the basic arithmetic operations are discussed in [1] and [2]. These algorithms provide an effective way of speeding up sequences of operations by maximizing their overlap. Moreover, the on-line algorithms minimize the required interconnection bandwidth and the number of input/output connections per module. The system implications and organization of on-line computational structures are discussed in [2] and [3].

In Section 2 the square root algorithm for normalized fractions is derived. The implementation and performance are discussed in Section 3.

## 2. Derivation of the Algorithm

### 2.1 Definitions

The argument

$$x = \sum_{i=1}^{m} x_i 2^{-i}$$

is in the range $[2^{-p}, 2^{-p+1})$ where the range scaling factor p is a positive integer to be determined later. The argument x is represented in the redundant binary system with the symmetric digit set $\{-1 = \bar{1}, 0, 1\}$. Its on-line representation is defined as follows:

$$X_j = X_{j-1} + x_{\delta+j} 2^{-\delta-j}, \; j = 1, 2, \ldots, m \qquad (1)$$

where

$\delta \geq 0$ is the on-line delay,

$X_0 = \sum_{i=1}^{\delta} x_i 2^{-i}$, and

$x_k = 0$ for $k > m - \delta$.

Clearly $X_m = x$. The result

$$y = \sum_{i=1}^{m} y_i 2^{-i}$$

is in the range $[2^{-p/2}, 2^{(1-p)/2})$ and satisfies

$$| \sqrt{x} - y | < 2^{-m+1} \qquad (2)$$

Its on-line representation in the redundant binary system with the symmetric digit set {-1, 0, 1} is defined as:

$$Y_j = Y_{j-1} + y_j 2^{-j}, \; j = 1, 2, \ldots, m$$

and $\qquad (3)$

$$Y_0 = 0$$

In order to facilitate computation of the result digits, a scaled remainder is introduced:

$$R_j = 2^j (X_j - Y_j^2) \qquad (4)$$

The error condition (2) and the digit-by digit computation of on-line algorithms imply that the remainders should be bounded as follows:

$$| R_j | < 1 \qquad (5)$$

*This research was supported in part by the National Science Foundation, Grant No. MCS72-03633 A04.

The remainders can be determined recursively:

$$R_j = 2R_{j-1} + x_{\delta+j}2^{-\delta} - 2Y_{j-1}y_j - y_j^2 2^{-j}$$

$$= 2R_{j-1} + x_{\delta+j}\,2^{-\delta} - (Y_{j-1} + Y_j)y_j,$$

$$j = 1, 2, \ldots, m, \qquad (6)$$

where

$$R_0 = X_0.$$

At each step one result digit is obtained as a function of the remainder, i.e.,

$$y_j = S(R_{j-1})$$

The selection function S (the selection rule) should be the same in all steps. Since the redundant number representation system is used, it is expected that the selection can be performed using the estimates $\hat{R}_j$ of the remainders [4, 5, 6]. Consequently the remainders can be computed in a redundant representation in time independent of the length of the operands. They can be used for the selection without performing the full-precision conversion to the conventional form, i.e.,

$$y_j = S(\hat{R}_{j-1})$$

## 2.2 Selection Function

In order to determine the selection function $y_j = S(R_{j-1})$ we first establish three intervals $I_k = [a_k, b_k]$, $k = -1, 0, 1$, in the range of $R_{j-1}$ such that if $R_{j-1} \in I_k$ then $y_j = k$ is a valid digit choice. In general, these intervals must satisfy the following conditions:

i) The continuity condition

$$I_k \cap I_{k+1} \neq \emptyset,\ k = -1, 0 \qquad (7)$$

i.e., the intervals may not be disjoint. Alternatively,

$$a_k \leq b_{k-1},\ k = 0, 1 \qquad (8)$$

ii) The containment condition:

$$R_j \in I_{-1} \cup I_0 \cup I_1, \qquad (9)$$

i.e., the remainders must always be contained in the selection intervals. Alternatively,

$$a_{-1} \leq -1$$

and (10)

$$b_1 \geq 1.$$

These conditions and the selection intervals are illustrated in the next figure:

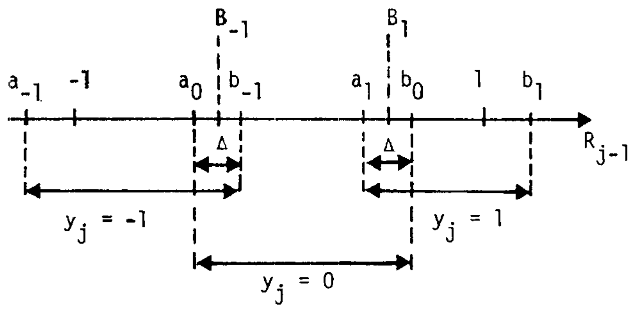

Figure 1: Selection Intervals

The intervals $I_k$ are directly obtained by evaluating

$$R_{j-1} = \frac{1}{2} R_j - x_{j+\delta}2^{-\delta-1} + Y_{j-1}y_j + y_j^2 2^{-j-1} \qquad (11)$$

for $R_j = \pm 1$ and $y_j = -1, 0,$ and $1$, following the approach used in [7]. In order to simplify notation let

$$u = x_{\delta+j}2^{-\delta-1} \qquad (12)$$

$$v = Y_{j-1} + 2^{-j-1}$$

$$w = -Y_{j-1} + 2^{-j-1}$$

Then the selection intervals are:

$$I_1 = (-\frac{1}{2} - u + v, \frac{1}{2} - u + v)$$

$$I_0 = (-\frac{1}{2} - u, \frac{1}{2} - u) \qquad (13)$$

$$I_{-1} = (-\frac{1}{2} - u + w, \frac{1}{2} - u + w)$$

The containment condition (9) implies that

$$u \leq v - \frac{1}{2} \qquad (14)$$

and

$$-u \leq -w - \frac{1}{2} \qquad (15)$$

Since $x_{\delta+j} \in \{\bar{1}, 0, 1\}$, these conditions imply, in the worst case when $x_{\delta+j} = \bar{1}$, that:

$$2^{-\delta} \leq 2Y_{j-1} - 2^{-j} - 1 \qquad (16)$$

The continuity condition (8) requires that the overlaps $\Delta$ between the selection intervals are positive:

$$\Delta_{01} = b_0 - a_1 = 1 - v > 0 \qquad (17)$$

$$\Delta_{10} = b_{-1} - a_0 = 1 + w > 0 \qquad (18)$$

Assuming that $\Delta \geq 2^{-t}$, the condition (8) becomes in the worst case:

$$2^{-t} \leq 1 - Y_{j-1} - 2^{-j-1} \qquad (19)$$

Combining (16) and (19) we obtain:

$$2^{-\delta} + 2^{-t} \leq Y_{j-1} - 2^{-j-1} \qquad (20)$$

From (19) and (20):

$$2^{-\delta} + 2^{-t} + 2^{-j-1} \leq Y_{j-1} \leq 1 - 2^{-t} - 2^{-j-1} \qquad (21)$$

Since $y \in [2^{-p/2}, 2^{(1-p)/2})$

$$2^{(1-p)/2} \leq 1 - 2^{-t} - 2^{-j-1}, \qquad (22)$$

which implies that the range scale factor p should be greater than one.

The choice of p is made as follows. Let $z \cdot 2^{e_z}$ be the argument represented in the radix 2 normalized floating point number system with the fraction $z \in [1/2, 1)$ and the exponent $e_z$ an integer. Similarly $w \cdot 2^{e_w}$ represents the result $(z \cdot 2^{e_z})^{1/2}$ in the same number system. Since only integer exponents are allowed,

$$w = (z)^{1/2}, \; e_w = e_z/2 \text{ if } e_z \text{ is even} \qquad (23)$$

and

$$w = (z/2)^{1/2}, \; e_w = (e_z + 1)/2 \text{ if } e_z \text{ is odd} \qquad (24)$$

where

$w \in [1/2, 1)$, i.e., the result appears in the normalized form. Also, in (23) and (24) p = 1 and p = 2, respectively. Because of the condition (22) which requires that p > 1 we chose the following argument scaling.

$$w = 2(z/4)^{1/2}, \; e_w = e_z/2 \text{ if } e_z \text{ is even.}$$

and $\qquad (25)$

$$w = (z/2)^{1/2}, \; e_w = (e_z + 1)/2 \text{ if } e_z \text{ is odd.}$$

Then the algorithm operates on

$$x = z/4 \text{ if } e_z \text{ is even.}$$

and $\qquad (26)$

$$x = z/2 \text{ if } e_z \text{ is odd.}$$

If $x \in [1/8, 1/4)$, then $y \in [1/2\sqrt{2}, 1/2)$

so that

$$y_1 = 0 \text{ and } y_2 = 1. \qquad (27)$$

Therefore, the worst case in the condition (20) occurs for j = 3:

$$2^{-\delta} + 2^{-t} + 2^{-4} \leq \frac{1}{2\sqrt{2}} \qquad (28)$$

or

$$2^{-\delta} + 2^{-t} \leq \frac{8-\sqrt{2}}{16\sqrt{2}} \qquad (29)$$

which is satisfied for $\delta \geq 2$ and $t > 5$. If $x \in [1/4, 1/2)$, then $y \in [1/2, \sqrt{2}/2)$ and $y_1 = 1$. The condition (20), for j = 2, gives

$$2^{-\delta} + 2^{-t} + 2^{-3} \leq 1/2$$

or $\qquad (30)$

$$2^{-\delta} + 2^{-t} \leq 3/8$$

which is satisfied for $\delta \geq 2$ and $t \geq 3$.

Therefore, we choose the minimum on-line delay $\delta = 2$ which guarantees that the overlap intervals are of the size $\Delta \geq 2^{-5}$. It will be seen that the on-line delay is $\delta = 1$ with respect to the floating point argument $z \cdot 2^{e_z}$.

We now determine two comparison points $B_{-1} \in [a_0, b_{-1}]$ and $B_1 \in [a_1, b_0]$. These points should lie in the middle of the overlap intervals so that the required precision of the remainder estimate $\hat{R}_j$ is minimized. We obtain:

$$-\frac{1}{2} - x_{2+j}2^{-3} + \frac{1}{64} + Y_{j-1} + 2^{-j-1} < B_1 < \frac{1}{2} - x_{2+j}2^{-3} - \frac{1}{64}$$

or $\qquad (31)$

$$\frac{15}{64} - x_{2+j}2^{-3} < B_1 < \frac{31}{64} - x_{2+j}2^{-3}$$

and choose

$$B_1 = \frac{1}{4} - x_{2+j}2^{-3} = (2 - x_{2+j})/8 \qquad (32)$$

Similarly

$$B_{-1} = -\frac{1}{4} - x_{2+j}2^{-3} = (-2 - x_{2+j})/8 \qquad (33)$$

This completes the specification of the selection function:

$$y_j = S(\hat{R}_{j-1}) = \begin{cases} 1 & \text{if } \hat{R}_{j-1} \geq B_1 \\ 0 & \text{if } B_{-1} < \hat{R}_{j-1} < B_1 \\ \bar{1} & \text{if } \hat{R}_{j-1} \leq B_{-1} \end{cases} \qquad (34)$$

where $\hat{R}_{j-1}$ is an estimate of $R_{j-1}$ satisfying

$$| R_{j-1} - \hat{R}_{j-1} | \leq 2^{-6} \qquad (35)$$

i.e., the remainder $R_{j-1}$, computed in a redundant form, is converted for the selection purposes into

$\hat{R}_{j-1}$. The estimate $\hat{R}_{j-1}$ is in the conventional form and has no more than seven bits including the sign.

The previous derivations are summarized as:

Algorithm A:

```
1. $Y_0 \leftarrow 0;\ R_0 \leftarrow x_1 \cdot 2^{-1} + x_2 \cdot 2^{-2}$
2. For $j = 1, 2, \ldots m + 1$ do:
    2.1 $y_j \leftarrow S(\hat{R}_{j-1}, x_{2+j})$
    2.2 $Y_j \leftarrow Y_{j-1} + y_j\, 2^{-j}$;
        $R_j \leftarrow 2R_{j-1} + x_{2+j}\, 2^{-2} - (2Y_{j-1} + y_j 2^{-j})y_j$
```

where

$$\hat{R}_j = \lceil 64R_j \rceil / 64$$

is the estimate of the remainder and

$$S(\hat{R}_{j-1}, x_{2+j}) = \begin{cases} 1 & \text{if } \hat{R}_{j-1} \geq (2 - x_{2+j})/8 \\ \bar{1} & \text{if } \hat{R}_{j-1} \leq (-2 - x_{2+j})/8 \\ 0 & \text{otherwise} \end{cases}$$

is the selection function.

Our main objective is to apply the on-line square rooting algorithm on the normalized floating-point argument $z \cdot 2^{e_z}$ and generate the result $w \cdot 2^{e_w}$. We now establish, using (25) and (26), that:

(a) For $e_z$ even:

$$x_1 = x_2 = 0,$$
$$x_i = z_{i-2},\ i = 3, 4, \ldots, m + 2$$
$$x_{m+3} = 0 \qquad (36)$$

and

$$w_i = y_{i+1},\ i = 1, \ldots, m.$$

(b) For $e_z$ odd:

$$x_1 = 0,$$
$$x_i = z_{i-1},\ i = 2, 3, \ldots, m + 1,$$
$$x_{m+2} = x_{m+3} = 0 \qquad (37)$$

and

$$w_i = y_i,\ i = 1, \ldots, m.$$

The algorithm A is now restated in a form convenient for implementation:

Algorithm B:

```
1. $W \leftarrow 0;\ R \leftarrow 1/4$
2. For $j = 1, 2, \ldots, m$ do:
    2.1 $w_j \leftarrow S(\hat{R}, z_{j+1})$
    2.2 $W \leftarrow W + w_j 2^{-k}$;
        $R \leftarrow 2R + z_{j+1} 2^{-2} - 2W \cdot w_j - w_j^2\, 2^{-k}$
```

where

$$k = \begin{cases} j & \text{if } e_z \text{ is odd;} \\ j + 1 & \text{if } e_z \text{ is even;} \end{cases}$$

W and R denote the result and the scaled remainder; $z_{j+1}$ is the digit of the argument mantissa received at the j-th step in which $w_j$ digit of the result mantissa is generated.

An example of square root evaluation is given in Figure 2.

## 3. On Implementation

The proposed on-line algorithm has two properties important for a modular implementation. First, a small number of inputs and outputs is needed because of the digit-by-digit mode of operation. Second, the primitive operations (limited carry propagation addition and concatenation) can be efficiently implemented and allow for easy partitioning.

The block diagram of the data section of the Basic Module (BM) for square rooting in on-line fashion is shown in Figure 3. The structure is based on the Algorithm B. Since W at the i-th step contains i-1 most significant digits, $W = W + w_i 2^{-1}$ can be obtained by concatentation, i.e.,

$$W \leftarrow (W, w_i)$$

Similarly,

$$R \leftarrow ((2R + z_{i+1} 2^{-2} - 2Ww_i), - w_i^2)$$

The data section of the BM is d digits wide and it consists of the following parts:

1. A limited carry-borrow propagation adder with the operands and the result represented in the redundant form.
2. Two registers R and W for the remainder and the result. Due to the redundant representation, each register requires 2d bits.

Argument:

$z = .65767815$ $e_z$ - even

Result:

$w = .81097358$ $e_w = e_z/2$

Number of bits: $m = 24$

| j | $z_{j+1}$ | $w_j$ | 2W | R | $\sqrt{z}$ - 2W |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | .25 | .81097358 |
| 1 | 1 | 1 | .5 | .5 | .310973586 |
| 2 | -1 | 1 | .75 | .125 | .060973586 |
| 3 | 0 | 0 | .75 | .25 | .060973586 |
| 4 | 1 | 1 | .8125 | -.03125 | -.001526414 |
| 5 | 0 | 0 | .8125 | -.0625 | -.001526414 |
| 6 | 0 | 0 | .8125 | -.125 | -.001526414 |
| 7 | 1 | 0 | .8125 | .0 | -.001526414 |
| 8 | -1 | 0 | .8125 | -.25 | -.001526414 |
| 9 | -1 | -1 | .810546875 | .0615234375 | .000426711 |
| 10 | 0 | 0 | .810546875 | .123046875 | .000426711 |
| 11 | 1 | 0 | .810546875 | .49609375 | .000426711 |
| 12 | 1 | 1 | .8107910156 | .43151856 | .000182570 |
| 13 | 1 | 1 | .8109130858 | .30218507 | .000060500 |
| 14 | 0 | 1 | .8109741208 | -.2065734633 | -.000000534 |
| 15 | 1 | 0 | .8109741208 | -.1631469266 | -.000000534 |
| 16 | 1 | 0 | .8109741208 | -.0762938532 | -.000000534 |
| 17 | 1 | 0 | .8109741208 | .0974122936 | -.000000534 |
| 18 | -1 | 0 | .8109741208 | -.0551754128 | -.000000534 |
| 19 | -1 | 0 | .8109741208 | -.3603508256 | -.000000534 |
| 20 | 1 | 0 | .8109741208 | -.4707016512 | -.000000534 |
| 21 | 0 | -1 | .810973644 | -.13042942 | -.000000058 |
| 22 | 0 | 0 | .810973644 | -.26085884 | -.000000058 |
| 23 | 0 | -1 | .8109735248 | .2892559044 | .000000061 |
| 24 | 1 | 1 | .8109735844 | .0175382542 | .000000001 |

Error: $$\left| \left( \sum_{i=0}^{25} z_i 2^{-i} \right)^{1/2} - \sum_{i=1}^{24} w_i \, 2^{-i} \right| < .5 \times 10^{-8} < 2^{-24}$$

Figure 2: Example

3. The result digit selection block S which implements the selection function $S(\hat{R}, z_{j+1})$. Since only the most significant module performs the selection, the output of the S block is controlled by a Select Enable (SE) signal. If the output of the S block is tri-state, then the terminal $w_j$ can also be used as input when BM is not in the most significant position. This is illustrated in Figure 4.

4. Two selection networks SN and the distribution (demultiplexing) network DN perform the multiplication by $(-w_j)$ and the concatenation of the term $w_j 2^{-k}$, respectively. In order to reduce the number of terminals on the module and make it independent of the working precision, the distribution network is controlled by a d - bit shift register.

5. The conversion block C generates an estimate $\hat{R}$ in the conventional form of the redundant remainder R.

The inputs to the BM are:

1. Transfer Inputs ($T_{IN}$)

   $T_{IN} = (C_{IN}, R_{IN}, W_{IN})$

where

$C_{IN}$ is the adder transfer digit;

$R_{IN}$ is the leftmost digit of the remainder; and

$W_{IN}$ is the leftmost digit of the partial result, generated by the next lower significant module.

2. Digit Input ($D_{IN}$)

$$D_{IN} = \begin{cases} z_{j+1} & \text{for the most significant BM;} \\ 0 & \text{otherwise.} \end{cases}$$

3. Select Enable (SE)

$$SE = \begin{cases} 1 & \text{for the most significant BM;} \\ 0 & \text{otherwise.} \end{cases}$$

4. Distribution Control Output ($DC_{IN}$)

For $BM_i$:

$$DC_{IN} = \begin{cases} 1 & \text{in step } j = (i-1)d; \\ 0 & \text{otherwise.} \end{cases}$$

In each subsequent step this 1 is shifted one position to the right, enabling the distribution of $w_j$ in the positions 2, 3,..., d of the module. After d steps, $DC_{OUT} = 1$ which enables the distribution of $w_j$ in the next lower significant module.

The outputs of the BM are:

1. Transfer Output ($T_{OUT}$)

$T_{OUT} = (C_{OUT}, R_{OUT}, W_{OUT})$,

defined analogously to $T_{IN}$.

2. Digit Output ($D_{OUT}$)

$$D_{OUT} = \begin{cases} w_j & \text{for the most significant BM;} \\ \text{used as input otherwise.} \end{cases}$$

3. Distribution Control Output ($DC_{OUT}$)

For $BM_i$:

$$DC_{OUT} = \begin{cases} 1 & \text{in step } j = i \cdot d; \\ 0 & \text{otherwise.} \end{cases}$$

We conclude that the basic module would require about 20 terminals for logic signals, including the clock and the initilization signals. Note that the number of terminals is independent of the working precision d of the module.

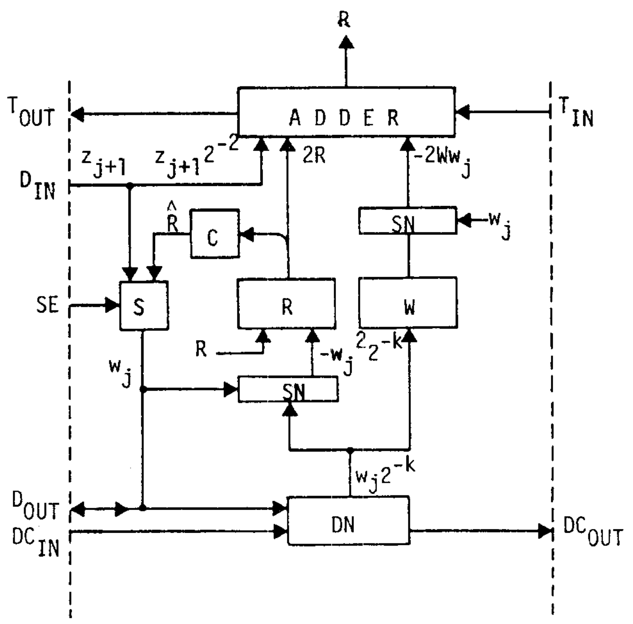

Figure 3: Data Selection of the Basic Module (BM)

The control of the BM requires an initialization signal (IS) to set the registers W and R according to Step 1 of Algorithm B. The recursion is controlled by a desired number of clock pulses (CP). These signals are issued in parallel to all modules. The beginning of operation is identified by an input synchronization signal ($S_{IN}$). The presence of this signal, which appears synchronously with CP, indicates a valid input digit DI. As soon as the first result digit is generated, the output synchronization signal ($S_{OUT}$) is issued so that any subsequent on-line operation using this result may proceed. As shown in Figure 4, a control module (CM) receives and generates the synchronization, initialization and clock signals. In addition, CM generates the result exponent $e_w$.

The time required to perform the basic recursion (Steps 2.1 and 2.2) is

$$t_o = t_A + t_S + t_T = O(10tg)$$

where $t_A$ is the limited carry borrow propagation addition time, $t_S$ is the selection time, $t_T$ is the register transfer (set-up) time and $t_g$ is the gate delay.

In order to obtain m most significant digits of the result mantissa, it is sufficient to have

$n = \left\lceil \frac{m+6}{2d} \right\rceil$ basic modules under assumption that

(a) A module has d bits of precision, and

(b) The estimate of the remainder used in the selection satisfies $|R - \hat{R}| \leq 2^{-6}$.

Clearly, $n = \left\lceil \frac{m}{d} \right\rceil$ modules are required for multi-precision operation. A multi-module organization is illustrated in Figure 4.

## 4. Concluding Remarks

An on-line algorithm for computing square roots in the normalizaed radix 2 floating-point system with the redundant (mantissa) digit set is presented. The algorithm has the on-line delay $\delta = 1$ and a simple step-invariant selection rule. Several implementation apsects, such as the basic module (BM), a multi-module organization and the control and interface, are discussed in some detail. The number of terminals of BM is independent of the working precision of the module. The modules are connected in a cascade fashion to provide for a desired precision, but the time to generate a digit (the step time $t_0$) is independent of the number of modules. Since this is a digit-by-digit algorithm, it takes m steps to generate the complete result. However, a subsequent operation with an on-line delay $\delta_k$ which uses this result can proceed after $\delta_k$ digits are generated. This provides for an overall speed-up of inter-dependent computational sequences.

The approach used in deriving the algorithm for r = 2 can be directly applied in a higher radix case. The existence of the selection function for a radix r can be easily shown and the corresponding selection rules obtained in a straightforward manner.

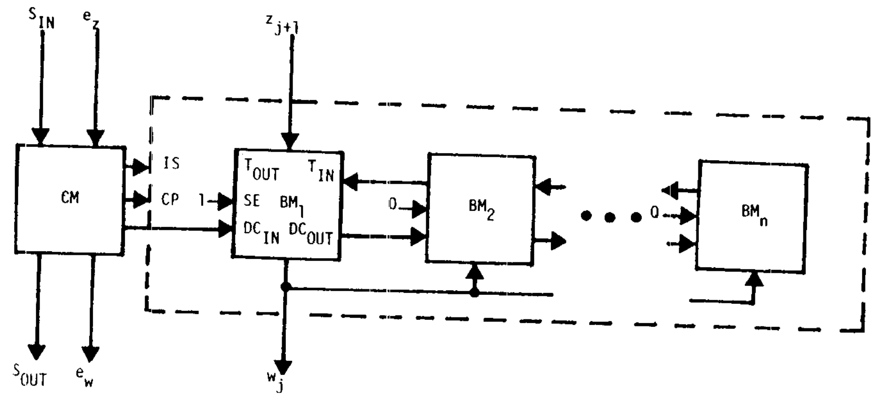

Figure 4: Mult:ple BM Organization

## References

1. K.S. Trivedi and M.D. Ercegovac, "On-line Algorithms for Division and Multiplication", IEEE Trans. Comput., Vol. C-26, No. 7, pp. 681-687, July 1977.

2. M.J. Irwin, "An Arithmetic Unit for On-line Computation", (Ph.D. Thesis), Report No. 873, Department of Computer Science, University of Illinois at Urbana-Champaign, May 1977.

3. A.A. Avizienis, "On a Flexible Impelementation of Digital Computer Arithmetic", Proceedings of IFIP, pp. 664-668, 1962.

4. J.E. Robertson, "A New Class of Digital Division Methods", IRE Trans. Electron. Comput., Vol. EC-7, pp. 218-222, September 1958.

5. A.A. Avizienis, "Signed-Digit Number Representation for Fast Parallel Arithmetic", IRE Trans. Electron. Comput., Vol. EC-10, pp. 389-400, September 1961.

6. M.D. Ercegovac, "A General Hardware-Oriented Method for Evaluation of Functions and Computations in a Digital Computer", IEEE Trans. Comput., Vol. C-26, No. 7, pp. 667-680, July 1977.

7. M.D. Ercegovac, "Radix-16 Evaluation of Certain Elementary Functions", IEEE Trans. Comput., Vol. C-22, No. 6, pp. 561-566, June 1973.