D 934/a

UNIVERSITE LIBRE DE BRUXELLES
Faculté des Sciences Appliquées

Laboratoires d'Electronique industrielle et d'Automatique

THE CONCEPT AND DESIGN OF INCREMENTAL COMPUTERS

ALGORITHMS AND DESIGN

Thèse de doctorat

Hassan SEIEDRAZI
Avril 1969

UNIVERSITE LIBRE DE BRUXELLES
Faculté des Sciences Appliquées

Laboratoires d'Electronique industrielle et d'Automatique

THE CONCEPT AND DESIGN OF INCREMENTAL COMPUTERS

ALGORITHMS AND DESIGN

Thèse de doctorat

Hassan SEIEDRAZI

Avril 1969

CHAPTER V

THE TOTAL ERROR AND THE CHOICE OF ALGORITHMS FOR INCREMENTAL COMPUTERS

5.1. Total error in the integration process in unitary or multiple increment computers, when the independent variable of integration X, is the independent variable t.

As we have seen in the preceding chapters, the actual value of integral s (t) in incremental computer is replaced by the approximated interpolated quantized rounded off and delayed value of integral $\tilde{s}^{*}_{QMD(k)}(t)$, which cause the error of method $\Gamma(t)$, the error of quantization ε_{tQ}, the round off error e_k and the error of transmission ε_{Tr}. The relation between these values is:

$$s_{(k)}(t) = s^{*}_{QMD(k)}(t) + [\Gamma(t) + \varepsilon_{tQ}(t) + e_k(t) + \varepsilon_{Tr}]_{(k)}$$

$$t \in (t_o, t_k) \qquad (5\text{-}1)$$

417086 The total error of process of integration ε_t is equal to:

$$\varepsilon_t = \Gamma(t) + \varepsilon_{tQ}(t) + e_k(t) + \varepsilon_{Tr} \qquad t \in (t_o, t_k) \tag{5-2}$$

so the equation (5-1) can be written as:

$$s_{(k)}(t) = s^{*}_{QMD(k)}(t) + \varepsilon_t(t) \qquad t \in (t_o, t_k) \tag{5-3}$$

where $s_k(t)$ is the actual value of integration in interval $t \in (t_o, t_k)$, $s^{*}_{QMD(k)}(t)$ is the approximated interpolated, quantized, rounded off and delayed value of integration which is the algorithm of the incremental machine, and $\varepsilon_t(t)$ is the total error of process of integration.

Now, we will calculate the total error of process of integration for the rectangular, trapezoidal, and three points method of integration with unitary or multiple increments.

As it is shown in the preceeding chapters, in the rectangular method of integration with unitary increments, the error of method is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{1}{2}(\Delta x)^2 \cdot y'(x) \qquad x \in (x_i, x_{i+1}) \tag{5-4}$$

$$\Gamma(t) = \frac{1}{2}\,\frac{(x_k - x_o)^2}{k}\, y'(\xi) \qquad \xi \in (x_o, x_k) \tag{5-5}$$

the error of quantization is:

$$\varepsilon_{tQ} < \Delta y\,(x_{kQ} - x_{oQ}) \tag{5-6}$$

The round off error is:

$$e_k < \Delta x \tag{5-7}$$

So the total error of process of integration in rectangular method of integration with unitary increments, can be found from equations (5-2), (5-4), (5-5), (5-6) and (5-7) as following:

$$\varepsilon_t < \frac{1}{2}\,\frac{(x_k - x_o)^2}{k}\,y'(\xi) + \Delta y\,(x_{kQ} - x_{oQ}) + \Delta x + \varepsilon_{Tr} \tag{5-8}$$

The approximated interpolated, quantized, rounded off value of integral $s^{*}_{QM(k)}$ (t), using the rectangular method of integration, which is the algorithm of incremental machine with unitary increments is:

$$s^{*}_{QM(k)}\ (t) = \sum_{i=1}^{k} y_{iQ} \cdot \Delta_{iQ} x \tag{5-9}$$

The error of method in rectangular method of integration with multiple increments ($\delta x = 2^r \cdot \Delta x$, $\delta y = 2^r \cdot \Delta y$, and $\delta s = 2^r \cdot \Delta s$) is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{1}{2}(\delta x)^2\,y'(x) \tag{5-10}$$

$$t \in (t_i,\ t_{i+1})$$

The error of quantization is:

$$\varepsilon_{tQ} < \Delta y \cdot 2^h (x_{kQ} - x_{oQ}) \tag{5-11}$$

The round off error is:

$$e_k < \delta x \tag{5-12}$$

so the total error of process of integration in rectangular method of integration with multiple increments can be found from equations (5-2) (5-10), (5-11) and (5-12) as following:

$$\varepsilon_t < \left[\sum_{i=1}^{k} \frac{1}{2} (\delta x)^2 y'(x) + \Delta y \cdot 2^h (x_{kQ} - x_{oQ}) + \right. \\ \left. + \delta x + \varepsilon_{Tr} \right] \tag{5-13}$$

and the approximated interpolated quantized, rounded off value of integral $s^{*}_{QM(k)}(t)$, using the rectangular method of integration, which is the algorithm of incremental computer with multiple increments is:

$$s^{*}_{QM(k)}(t) = \sum_{i=1}^{k} y_{iQ} \cdot \delta_{iQ} x \tag{5-14}$$

In the trapezoidal method of integration with unitary increments, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{1}{12} (\Delta x)^3 y''(x) \qquad x \in (x_i, x_{i+1}) \tag{5-15}$$

$$= \frac{1}{12} \frac{(x_k - x_o)^3}{k^2} \; y'' \; (\xi) \qquad \xi \in (x_o, x_k)$$

The quantization error ε_{tQ} (t) is:

$$\varepsilon_{tQ} \; (t) < \Delta y \; (x_{kQ} - x_{oQ}) \tag{5-16}$$

The round off error e_k (t) is:

$$e_k \; (t) < \Delta x \tag{5-17}$$

Therefore from equations (5-2), (5-15), (5-16) and (5-17), the total error of the process of integration in incremental machine with unitary increments is:

$$\varepsilon_t < \frac{1}{12} \frac{(x_{kQ} - x_{oQ})^3}{k^2} \; y'' \; (\xi) + \Delta y \; (x_{kQ} - x_{oQ}) + \\ + \Delta x + \varepsilon_{Tr} \tag{5-18}$$

The approximated interpolated, rounded off value of integration, s^{*}_{QM} (t), using the trapezoidal method of integration, which is the algorithm of the incremental computer with unitary increments is:

$$s^{*}_{QM(k)} \; (t) = \sum_{i=1}^{k} \left(y_{iQ} \cdot \Delta_{iQ}x + \frac{1}{2} \Delta_{iQ}x \cdot \Delta_{iQ}y \right) \tag{5-19}$$

In the trapezoidal method of integration, with multiple increments, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{1}{12} (\delta x)^3 y'' (x) \qquad x \in (x_i, x_{i+1}) \quad (5\text{-}20)$$

The quantization error ε_{tQ} (t) is:

$$\varepsilon_{tQ} (t) < \Delta y \cdot 2^h (x_{kQ} - x_{oQ}) \qquad (5\text{-}21)$$

The round off error e_k (t) is:

$$e_k (t) < \delta x \qquad (5\text{-}22)$$

Therefore from equations (5-2), (5-20), (5-21) and (5-22), the total error of process of integration ε_t in incremental computer with multiple increments is:

$$\varepsilon_t < \sum_{i=1}^{k} \frac{1}{12} (\delta x)^3 y'' (x) + \Delta y \cdot 2^h (x_{kQ} - x_{oQ}) + \\ + \delta x + \varepsilon_{Tr} \qquad (5\text{-}23)$$

and the approximated interpolated, rounded off value of integration $s^{*}_{QM(k)}$ (t) which is the algorithm of the incremental machine using the trapezoidal method of integration with multiple increments is:

$$s^{*}_{QM(k)} (t) = \sum_{i=1}^{k} (y_{iQ} \cdot \delta_{iQ}x + \delta_{iQ}y \cdot \delta_{iQ}x) \qquad (5\text{-}24)$$

In the three points method of integration, with unitary increments, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{1}{24} (\Delta x)^4 y'''(x) \qquad x \in (x_i, x_{i+1})$$

$$= \frac{1}{24} \frac{(x_k - x_o)^4}{k^3} y'''(\xi) \qquad \xi \in (x_o, x_k) \tag{5-25}$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < \Delta y \, (x_{kQ} - x_{oQ}) \tag{5-26}$$

The round off error e_k is:

$$e_k < \Delta x \tag{5-27}$$

Therefore, from equations (5-2), (5-25), (5-26) and (5-27), the total error ε_t of process of integration with unitary increments is:

$$\varepsilon_t < \frac{1}{24} \frac{(x_k - x_o)^4}{k^3} y'''(\xi) + \Delta y \, (x_{kQ} - x_{oQ}) + \Delta x + \varepsilon_{Tr} \tag{5-28}$$

The approximated interpolated, quantized and rounded off value of integration $s^{*}_{QM(k)}$ (t), using the three points method of integration which is the algorithm of incremental computer, with unitary increments is:

$$s^{*}_{QM(k)} (t) = \sum_{i=1}^{k} [y_{iQ} \cdot \Delta_{iQ}x + \frac{1}{2}\Delta_{iQ}x \cdot \Delta_{iQ}y + \frac{1}{12} \Delta_{iQ}x \, (\Delta_{iQ}y - \Delta_{(i-1)Q}y)] \tag{5-29}$$

In the three points method of integration, with multiple increments, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{1}{24} (\delta x)^4 y'''(x) \qquad x \in (x_i, x_{i+1}) \qquad (5\text{-}30)$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < \Delta y \cdot 2^h (x_{kQ} - x_{oQ}) \qquad (5\text{-}31)$$

The round off error e_k is:

$$e_k < \delta x \qquad (5\text{-}32)$$

Therefore, from equations (5-2), (5-30), (5-31) and (5-32), the total error ε_t of process of integration, with multiple increments is:

$$\varepsilon_t < \left[\sum_{i=1}^{k} \frac{1}{24} (\delta x)^4 y'''(x) + \Delta y \cdot 2^h (x_{kQ} - x_{oQ}) + + \delta x + \varepsilon_{Tr} \right] \qquad (5\text{-}33)$$

The approximated, interpolated, quantized and rounded off value of integration $s^{*}_{QM(k)}(t)$, using the three points method of integration, which is the algorithm of incremental machine with multiple increments is:

$$s^{*}_{QM(k)}(t) = \sum_{i=1}^{k} [y_{iQ} \cdot \delta_{iQ}x + \frac{1}{2} \delta_{iQ}x \cdot \delta_{iQ}y +$$

$$+\frac{1}{12}\,\delta_{iQ}x\;(\delta_{iQ}y - \delta_{(i-1)Q}y)] \qquad (5\text{-}34)$$

The block diagram of incremental computers is drawn in figure (5.1).

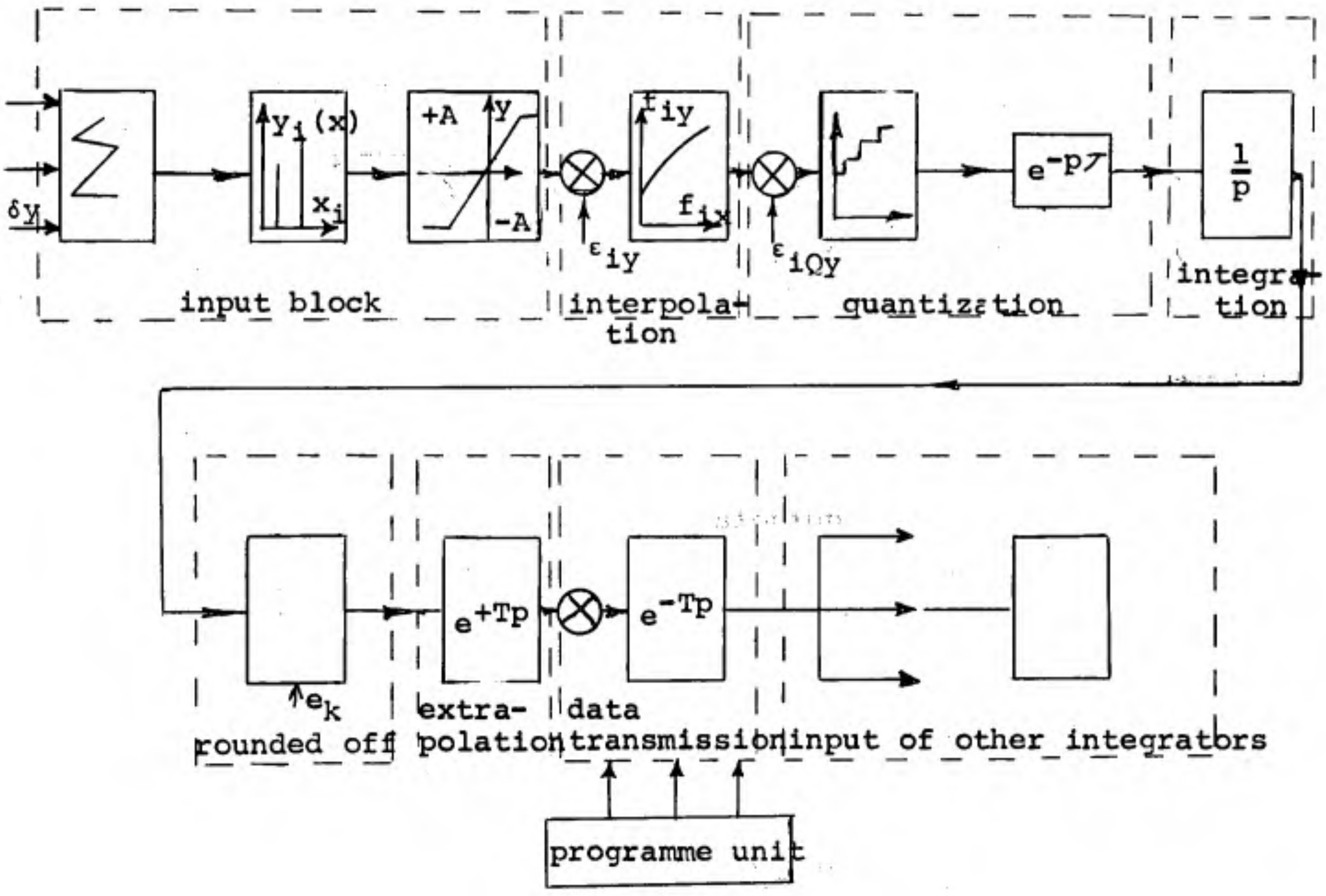


fig. 5.1.

Block diagram of incremental computer.

5.2. Total error in the integration process in unitary or multiple increment computers, when the independent variable of integration X is a function of the independent variable t.

As it was discussed earlier, the actual value of integral $s\ (t)$ is replaced with the approximated, interpolated, quantized rounded off and delayed value of integral $s^{*}_{QMD(k)}\ (t)$ as:

$$s\ (t) = s^{*}_{QMD(k)}\ (t) + [\Gamma(t) + \varepsilon_{tQ}\ (t) + e_k\ (t) + \varepsilon_{Tr}] \qquad (5-35)$$

$$s\ (t) = s^{*}_{QMD(k)}\ (t) + \varepsilon_t$$
$$\varepsilon_t = \Gamma(t) + \varepsilon_{tQ}\ (t) + e_k\ (t) + \varepsilon_{Tr} \qquad (5-36)$$

where ε_t is the total error of process of integration, which is equal to the sum of the error of method $\Gamma(t)$, the error of quantization $\varepsilon_{tQ}\ (t)$, the round off error $e_k\ (t)$ and the transmission error ε_{Tr}.

As we have seen in the preceeding chapter, the error of the rectangular method of integration $\Gamma(t)$, with unitary increments, when the variable of integration X is a function of the independent **variable t of machine:**

$$\Gamma(t) = \frac{1}{2} \sum_{i=1}^{k} (\Delta t)^2\ y'(t)\ x'(t) \qquad t \in (t_i, t_{i+1})$$
$$= \frac{1}{2} \frac{(t_k - t_o)^2}{k} y'(\xi)\ x'(\xi) \qquad \xi \in (x_k,\ x_o) \qquad (5-37)$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < \Delta y\ (x_{Qk} - x_k) + \Delta x\ (y_{Qk} - y_{ok}) \tag{5-38}$$

The round off error e_k is:

$$e_k < \Delta x \tag{5-39}$$

and ε_{Tr} depends on the programming.

So the total error ε_t of integration process in rectangular method, with unitary increment, when the independent variable of integral is a function of the independent variable t of machine, can be found by the equations (5-36), (5-37), (5-38) and (5-39) as:

$$\varepsilon_t < \left[\frac{1}{2}\ \frac{(t_k - t_o)^2}{k}\, y'(\xi)\cdot x'(\xi) + \right. \\ \left. + [\Delta y\ (x_{Qk} - x_{Qo}) + \Delta x\ (y_{ok} - y_{oQ})] + \Delta x + \varepsilon_{Tr}\right] \tag{5-40}$$

and the approximated interpolated, quantized, rounded off value of integration $s^{*}_{QM(k)}$ (t), using the unitary increments is:

$$s^{*}_{QM(k)}\ (t) = \sum_{i=1}^{k} y_{iQ}\cdot \Delta_{iQ}x \tag{5-41}$$

In the rectangular method of integration with multiple increments, when the independent variable of integral X is a function of the independent variable t of machine, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \frac{1}{2} \sum_{i=1}^{k} (\delta x)^2 \; y'(t) \; x'(t) \qquad t \in (t_i, t_{i+1}) \tag{5-42}$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < 2^h \cdot \Delta y \; (x_{Qk} - x_{ok}) + \Delta x \; y_{Qk} - y_{oQ}) \tag{5-43}$$

The round off error e_k is:

$$e_k < \delta x \tag{5-44}$$

and the transmission error ε_{Tr} depends on the programming.

The total error ε_t of the rectangular method of integration, with multiple increments when the independent variable of integral X is a function of the independent variable t of machine, can be find from equations (5-36), (5-42), (5-43) and (5-44) as following:

$$\varepsilon_t < \left| \frac{1}{2} \sum_{i=1}^{k} [(\delta t)^2 \; y'(t) \; x'(t)] + [2^h \cdot \Delta y \; (x_{Qk} - x_{oQ}) + \right.$$

$$\left. + \; \Delta x \; (y_{Qk} - y_{oQ})] + \delta x + \varepsilon_{Tr} \right| \tag{5-45}$$

and the approximated interpolated, quantized, and rounded off value of integration $s^{*}_{QM(k)}$ (t), with multiple increments ($\delta x = 2^r \cdot \Delta x$, $\delta y = 2^r \cdot \Delta y$ and $\delta s = 2^r \cdot \Delta s$) is:

$$s^{*}_{QM(k)} \; (t) = \sum_{i=1}^{k} \; y_{iQ} \cdot \delta_{iQ} x \tag{5-46}$$

In the trapezoidal method of integration, with unitary increments, when the independent variable of integral X is a function of the independent variable t of machine, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \sum_{i=1}^{k} (\Delta t)^3[y''(t)\ x'(t) - y'(t)\ x''(t)] \qquad (5\text{-}47)$$

$$x \in (x_i, x_{i+1})$$

$$= \frac{(t_k - t_o)^3}{k^2}[y''(\xi)\ x'(\xi) - y'(\xi)\ x''(\xi)],\ \xi \in (x_o, x_k)$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < \Delta y\ (x_{Qk} - x_{oQ}) + \Delta x\ (y_{Qk} - y_{oQ}) \qquad (5\text{-}48)$$

The round off error e_k is:

$$e_k < \Delta x \qquad (5\text{-}49)$$

and the transmission error ε_{Tr} depends on the programming.

So, the total error ε_t of trapezoidal method of integration, with unitary increments, when the independent variable of integral X is a function of the independent variable t of machine, can be found from equations (5-36), (5-47), (5-48) and (5-49) as following:

$$\varepsilon_t < \sum_{i=1}^{k} \frac{(\Delta t)^3}{12}\ [y''(t)\ x'(t) - y'(t)\ x''(t)] + \qquad (5\text{-}50)$$

$$+ [\Delta y\ (x_{Qk} - x_{oQ}) + \Delta x\ (y_{Qk} - y_{oQ})] + \Delta x + \varepsilon_{Tr}$$

The approximated interpolated, quantized and rounded off value of integration $s^{*}_{QM(k)}\ (t)$ with unitary increments is:

$$s^{*}_{QM(k)}\ (t) = \sum_{i=1}^{k} (y_{iQ} \cdot \Delta_{iQ}x + \frac{1}{2}\Delta_{iQ}y \cdot \Delta_{iQ}x) \qquad (5\text{-}51)$$

The error of method $\Gamma(t)$ in the trapezoidal method of integration, using the multiple increments, when the independent variable of integral X is a function of the independent variable t of machine, is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{(\delta t)^3}{12} [y''(t)\ x'(t) - y'(t)\ x''(t)] \qquad (5\text{-}52)$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < 2^h \cdot \Delta y\ (x_{Qk} - x_{oQ}) + \Delta x\ (y_{Qk} - y_{oQ}) \qquad (5\text{-}53)$$

The round off error e_k is:

$$e_k < \delta x \qquad (5\text{-}54)$$

and the transmission error ε_{Tr} depends on the programming.

So, the total error ε_t in the trapezoidal method of integration, using the multiple increments, when the independent variable of integral is a function of the independent variable t of machine, can be found from equations (5-36), (5-52), (5-53) and (5-54) as:

$$\varepsilon_t < \left[\sum_{i=1}^{k} \frac{(\delta t)^3}{12} [y''(t)\ x'(t) - y'(t)\ x''(t)] + \right. \tag{5-55}$$
$$\left. +[2^h \cdot \ \Delta y\ (x_{Qk} - x_{oQ}) + \ \Delta x\ (y_{Qk} - y_{oQ})] + \delta x + \varepsilon_{Tr} \right]$$

The algorithms of machine, which is the approximated interpolated, quantized and rounded off value of integration $s^{*}_{QM(k)}$ (t), using the multiple increments, when the independent variable of integral X is a function of the independent variable t of machine, is:

$$s^{*}_{QM(k)}\ (t) = \sum_{i=1}^{k} (y_{iQ} \cdot \delta_{iQ}x + \frac{1}{2}\delta_{iQ}x \cdot \delta_{iQ}y) \qquad x \in (x_i, x_{i+1}) \tag{5-56}$$

In the three points method of integration, using the unitary increment, when the independent variable of integral X is a function of the independent variable t of machine, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{(\Delta t)^4}{24} [y'''(t)\ x'(t) - y'(t)\ x'''(t)]$$
$$t \in (t_i,\ t_{i+1})$$
$$= \frac{(t_k - t_o)^4}{24k^3} \left[y'''(\xi)\ x'(\xi) - y'(\xi)\ x'''(\xi) \right] \tag{5-57}$$
$$\xi \in (t_o,\ t_k)$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < \Delta y \ (x_{Qk} - x_{oQ}) + \Delta x \ (y_{Qk} - y_{Qo}) \tag{5-58}$$

The round off error e_k is:

$$e_k < \Delta x \tag{5-59}$$

and the transmission error ε_{Tr} depends on the programming.

So the total error ε_t, in the three points method of integration, using the unitary increments, when the independent variable of integral X is a function of the independent variable of machine t, can be found from equations (5-36), (5-57), (5-58) and (5-59) as following:

$$\varepsilon_t < \left| \frac{(t_k - t_o)^4}{24k^3} [y'''(t) \ x'(t) - y'(t) \ x'''(t)] + \right. \\ \left. + [\ \Delta y \ (x_{Qk} - x_{oQ}) + \Delta x \ (y_{Qk} - y_{Qo})] + \Delta x + \varepsilon_{Tr} \right| \tag{5-60}$$

The algorithm of machine, which is the approximated, interpolated, quantized and rounded off value of integration $s^{*}_{QM(k)}$ (t), using the unitary increments, when the independent variable of integral X is a function of the independent variable t of machine, is:

$$s^{*}_{QM(k)} \ (t) = \sum_{i=1}^{k} [y_{iQ} \cdot \Delta_{iQ}x + \frac{1}{2} \Delta_{iQ}x \cdot \Delta_{iQ}y + \\ + \frac{1}{12} (\Delta_{iQ}y \cdot \Delta_{(i-1)Q}x - \Delta_{(i-1)Q}y \cdot \Delta_{iQ}x] \tag{5-61}$$

In the three points method of integration, using the multiple increments, when the independent variable of integral X is a function of the independent variable t of machine, the error of method $\Gamma(t)$ is:

$$\Gamma(t) = \sum_{i=1}^{k} \frac{(\delta t)^4}{24} [y'''(t)\ x'(t) - y'(t)\ x'''(t)] \qquad t \in (t_i,\ t_{i+1}) \tag{5-62}$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < 2^h \cdot \Delta y\ (x_{kQ} - x_{oQ}) + \Delta x\ (x_{kQ} - x_{oQ}) \tag{5-63}$$

The round off error e_k is:

$$e_k < \delta x \tag{5-64}$$

and the transmission error ε_{Tr} depends on the programming.

So the total error ε_t, in the three points method of integration using the multiple increments, when the independent variable of integral X is a function of the independent variable t of machine, can be found from equations (5-36), (5-62), (5-63) and (5-64) as:

$$\varepsilon_t < \left[\sum_{i=1}^{k} \frac{(\delta t)^4}{24} [y'''(t)\ x'(t) - y'(t)\ x'''(t)] + \right. \\ \left. + [2^h \cdot \Delta y\ (x_{kQ} - x_{oQ}) + \Delta x\ (x_{kQ} - x_{oQ})] + \delta x + \varepsilon_{Tr} \right] \tag{5-65}$$

The algorithms of machine, using the multiple increments, which are the approximated interpolated, quantized and rounded off value of integration $s^{*}_{QM(k)}(t)$ are:

$$s^{*}_{QM(k)}(t) = \sum_{i=1}^{k} [y_{iQ} \cdot \delta_{iQ}x + \frac{1}{2}\delta_{iQ}x \cdot \delta_{iQ}y + + \frac{1}{12}(\delta_{iQ}y \cdot \delta_{(i-1)Q}x - \delta_{(i-1)Q}y \cdot \delta_{iQ}x)] \tag{5-66}$$

The block diagram of the incremental computer is shown in figure (5.2).

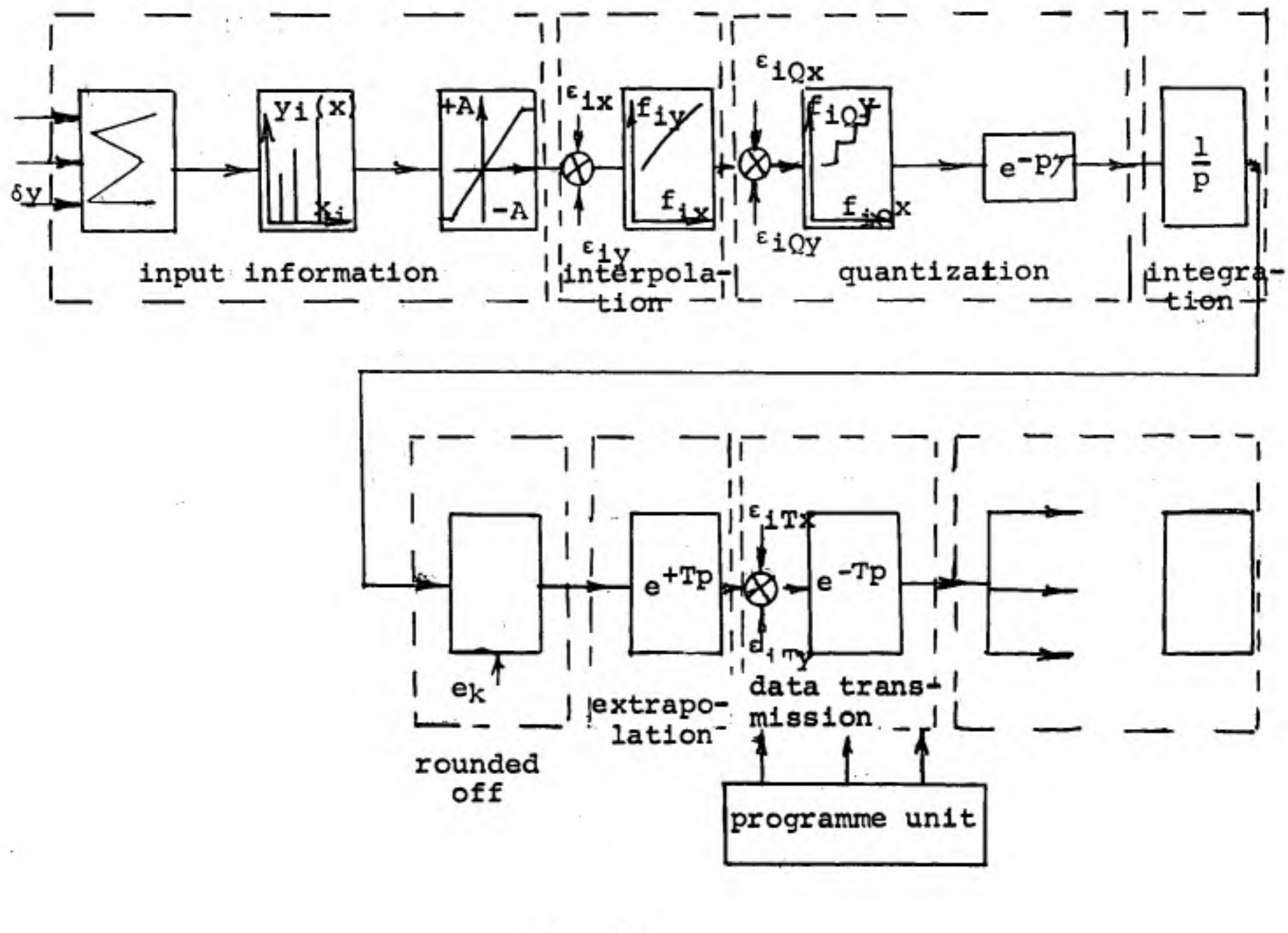


fig. 5.2.

Block diagram of incremental computer.

5.3. Relative error in the integration process in unitary or multiple increment computers.

We calculated the total error ε_t of integration process in each method of integration. But the most useful formula for evaluating the error in a interval $t \in (t_o, t_k)$ is the relative error γ, which is the ratio of the total absolute error ε_t, to the value of integration s (t) as:

$$\gamma = \frac{\varepsilon_t}{s\ (t)} \tag{5-67}$$

But the value of s (t) is not available befor the calculation, this problem can be solved by assuming:

$$s\ (t) = y_m\ (x_{kQ} - x_{oQ}) \tag{5-68}$$

where y_m is the mean value of y in interval $x \in (x_o, x_k)$, which in some cases can be presented as $\frac{1}{2}\ (y_{max} + y_{min})$.

The relative error γ from equations (5-67) and (5-68) can be found as following:

$$\gamma = \frac{\varepsilon_t}{y_m\ (x_{kQ} - x_{oQ})} \tag{5-69}$$

By using the total error ε_t from the preceeding paragraph, we can calculate the relative error γ in each method of integration with unitary and multiple increments.

From equations (5-8), (5-18), (5-28) and (5-69), the relative error γ of rectangular, trapezoidal and three points formula of integration with unitary increments, when the independent variable of integral γ is same as the independent variable t of machine, will be as following:

a - in rectangular method.

$$\gamma < \frac{1}{y_m}\left[\frac{1}{2}\,\frac{(x_k - x_o)}{k}\,y'(\xi) + \Delta y + \frac{\Delta x + \varepsilon_{Tr}}{x_k - x_o}\right] \quad (5\text{-}70)$$

b - in trapezoidal method.

$$\gamma < \frac{1}{y_m}\left[\frac{1}{12}\,\frac{(x_{Qk} - x_{Qo})^2}{k^2}\,y''(\xi) + \Delta y + \frac{\Delta x + \varepsilon_{Tr}}{x_k - x_o}\right] \quad (5\text{-}71)$$

c - in three points method.

$$\gamma < \frac{1}{y_m}\left[\frac{1}{24}\,\frac{(x_{Qk} - x_{Qo})^3}{k^3}\,y''(\xi) + \Delta y + \frac{\Delta x + \varepsilon_{Tr}}{x_k - x_o}\right] \quad (5\text{-}72)$$

The relative error γ, in rectangular, trapezoidal and three points method of integration, with multiple increments, when the independent variable of integral X is same as the independent variable t of machine, from equations (5-13), (5-23), (5-33) and (5-69), will be:

d - in rectangular method.

$$\gamma < \frac{1}{y_m} \left[\sum_{i=1}^{k} \frac{1}{2} \frac{(\delta x)^2}{x_{Qk}-x_{Qo}} \; y'(x) + \Delta y \cdot 2^h + \frac{\delta x + \varepsilon_{Tr}}{x_{Qk} - x_{Qo}} \right] \tag{5-73}$$

e - in trapezoidal method.

$$\gamma < \frac{1}{y_m} \left[\sum_{i=1}^{k} \frac{1}{12} \frac{(\delta x)^3}{x_{Qk}-x_{Qo}} \; y''(x) + \Delta y \cdot 2^h + \frac{\delta x + \varepsilon_{Tr}}{x_{Qk} - x_{Qo}} \right] \tag{5-74}$$

f - in three points formula.

$$\gamma < \frac{1}{y_m} \left[\sum_{i=1}^{k} \frac{1}{24} \frac{(\delta x)^4}{x_{Qk}-x_{Qo}} \; y'''(x) + \Delta y \cdot 2^h + \frac{\delta x + \varepsilon_{Tr}}{x_{Qk} - x_{Qo}} \right] \tag{5-75}$$

The relative error γ, in the rectangular, trapezoidal, and three points formula of integration with unitary increments, when the independent variable of integral X is a function of the independent variable t of machine, can be found from equations (5-40), (5-50), (5-61) and (5-69) as following:

a - in rectangular method.

$$\gamma < \frac{1}{y_m}\left[\frac{1}{2}\,\frac{(t_k - t_o)^2}{x_{Qk} - x_{Qo}}\, y'(\xi)\, x'(\xi) + \right. \tag{5-76}$$
$$\left. + [\Delta y_Q + \Delta x\, \frac{y_{Qk} - y_{oQ}}{x_{Qk} - x_{oQ}}] + \frac{\Delta x + \varepsilon_{Tr}}{x_{Qk} - x_{oQ}}\right]$$

b - in trapezoidal method.

$$\gamma < \frac{1}{y_m}\left[\frac{(t_{kQ} - t_{oQ})^3}{12k^2(x_{Qk} - x_{Qo})}\,[y''(\xi)\, x'(\xi) - y'(\xi)\, x''(\xi)] + \right. \tag{5-77}$$
$$\left. + [\Delta y + \frac{\Delta x(y_{Qk} - y_{oQ})}{x_{Qk} - x_{Qo}}] + [\frac{\Delta x + \varepsilon_{Tr}}{x_{Qk} - x_{Qo}}]\right]$$

c - in three points formula.

$$\gamma < \frac{1}{y_m}\left[\frac{(t_{kQ} - t_{oQ})^4}{24k^3(x_{Qk}-x_{Qo})}\,[y'''(t)\, x'(t) - y'(t)\, x'''(t)] + \right. \tag{5-78}$$
$$\left. + [\Delta y + \frac{\Delta x(y_{Qk} - y_{Qo})}{x_{Qk} - x_{Qo}}] + \frac{\Delta x + \varepsilon_{Tr}}{x_{Qk} - x_{Qo}}\right]$$

The relative error γ, in the rectangular, trapezoidal and three points formula, with multiple increments, when the independent variable of integral is a function of the independent variable t of machine, can be found from equations (5-45), (5-55), (5-65) and (5-69) as following:

d - in rectangular method.

$$\gamma < \frac{1}{y_m}\left[\left[\sum_{i=1}^{k} \frac{1}{2}\,\frac{(\delta t)^2}{x_{Qk}-x_{Qo}}\; y'(t)\; x'(t)\right] + \right. \tag{5-79}$$

$$\left. + \left[2^h \cdot \Delta_Q y + \frac{\Delta x(y_{Qk} - y_{oQ})}{x_{Qk} - x_{oQ}}\right] + \frac{\delta x + \varepsilon_{Tr}}{x_{Qk} - x_{oQ}} \right]$$

e - in trapezoidal method.

$$\gamma < \frac{1}{y_m}\left[\sum_{i=1}^{k} \frac{(\delta t)^3}{12(x_{Qk}-x_{oQ})}\left[y''(t)\; x'(t) - y'(t)\; x''(t)\right] + \right. \tag{5-80}$$

$$\left. + \left[2^h \cdot \Delta_Q y + \frac{\Delta x(y_{Qk} - y_{oQ})}{x_{Qk} - x_{oQ}}\right] + \frac{\delta x + \varepsilon_{Tr}}{x_{Qk} - x_{oQ}} \right]$$

f - in three points method.

$$\gamma < \frac{1}{y_m}\left[\sum_{i=1}^{k} \frac{(\delta t)^4}{24(x_{kQ}-x_{oQ})}\left[y'''(t)\; x'(t) - y'(t)\; x'''(t)\right] + \right.$$

$$+ \quad 2^h \cdot \Delta_Q y + \frac{\Delta x(y_{Qk} - y_{Qo})}{x_{Qk} - x_{Qo}} + \frac{\delta x + \varepsilon_{Tr}}{x_{Qk} - x_{Qo}} \Bigg]$$

As it is seen, with unitary or multiple increments computation, the relative error is inverse proportional to y_m. Therefore, in order to reduce the relative error γ, we should increase the value of y_m by choosing properly the scale factor of y register in the operation of integration.

By increasing the content of y register, we decrease the relative error, and increase the output rates. In general, it can be seen that any effect which increase the output rates will decrease the relative error.

5.4. Minimization of the transmission error in unitary or multiple increment computers.

As it was mentioned in chapter 1, by transmitting the output data of some integrators to the inputs dx and dy of integrator k, there is a delay of $(1+\lambda)T$ between the actual continuous functions X (t), Y (t) and the approximated interpolated, quantized and delayed functions f_{iQDx} (t), f_{iQDy} (t), which cause the error of quantization and the error of transmission. The functions X (t), Y (t) are defined as:

$$\begin{cases} X\ (t) = X\ [x_o,\ \delta x_1,\ \delta x_2, \ldots\ \ \delta_i x, \ldots\ (i=1,2,\ldots\infty] \\ Y\ (t) = Y\ [y_o,\ \delta y_1,\ \delta y_2, \ldots\ \ \delta_i y, \ldots\ (i=1,2,\ldots\infty] \end{cases} \tag{5-82}$$

and f_{iQDx} (t), f_{iQDy} (t) are expressed as following:

$$\begin{cases} f_{iQDx}\ (t) = f_{ix}\ [x_{oQ},\ \delta_{1Q}x,\ \delta_{2Q}x, \ldots\ \ \delta_{(i-\lambda-1)Q}x,\ t] \\ f_{iQDy}\ (t) = f_{iy}\ [y_{oQ},\ \delta_{1Q}y,\ \delta_{2Q}y, \ldots\ \ \delta_{(i-\lambda-1)Q}y,\ t] \end{cases} \tag{5-83}$$

As it was mentioned in quantization process, the delay $= \lambda T$ cause the quantization error ε_{iQx}, ε_{iQy} in interval $t \in (t_i,\ t_{i+1})$, which is the difference between the quantized function f_{iQx}, f_{iQy}, and the unquantized function f_{ix}, f_{iy} as following:

$$\begin{cases} \varepsilon_{iQx} = f_{ix}\ (t) - f_{iQx}\ (t) \\ \varepsilon_{iQy} = f_{iy}\ (t) - f_{iQy}\ (t) \end{cases} \tag{5-84}$$

where

$$\begin{cases} f_{ix}(t) = f_{ix}[x_o, \delta_1 x, \delta_2 x, \ldots\ldots \quad \delta_i x, t] \\ f_{iy}(t) = f_{iy}[y_o, \delta_1 y, \delta_2 y, \ldots\ldots \quad \delta_i y, t] \end{cases} \tag{5-85}$$

and

$$\begin{cases} f_{iQx}(t) = f_{iQx}[x_{oQ}, \delta_{1Q} x, \delta_{2Q} x, \ldots\ldots \quad \delta_{iQ} x, t] \\ f_{iQy}(t) = f_{iQy}[y_{oQ}, \delta_{1Q} y, \delta_{2Q} y, \ldots\ldots \quad \delta_{iQ} y, t] \end{cases} \tag{5-86}$$

The quantization error ε_{iQx}, ε_{iQy}, and the delay of quantization $\mathcal{T} = \lambda T$ can be minimized by reducting the weight of the quantums Δx and Δy.

The input data which should be available in the input of incremental computer, are the quantized value (x_{oQ}, y_{oQ}, $\delta_{1Q}x$, $\delta_{1Q}y$, $\delta_{2Q}x$, $\delta_{2Q}y$, ... $\delta_{iQ}x$, $\delta_{iQ}y$, t) But in parallel incremental computer, the data, which are available at the input of machine in the i th iteration, are the data of former iteration 1, 2,......i-1 as following:

$$\begin{matrix} [x_{oQ}, \delta_{1Q} x, \delta_{2Q} x, \ldots\ldots \quad , \delta_{(i-1)Q} x, t] \\ [y_{oQ}, \delta_{1Q} y, \delta_{2Q} y, \ldots\ldots \quad , \delta_{(i-1)Q} y, t] \end{matrix} \tag{5-87}$$

So instead of the approximated interpolated, quantized function f_{iQx}, f_{iQy}, there will be the approximated interpolated, quantized and delayed function f_{iQDx}, f_{iQDy}, whose data have a delay of T (iteration time of integral) with respect to the functions f_{iQx}, f_{iQy}, so the f_{iQDx}, f_{iQDy} are constructed in the input of the incremental computer as following:

$$\begin{cases} f_{iQDx}\ (t) = f_{iQDx}\ [x_{oQ},\ \delta_{1Q}x,\ \delta_{2Q}x, \ldots\ \delta_{(i-1)Q}x,\ t] \\ f_{iQDy}\ (t) = f_{iQDy}\ [y_{oQ},\ \delta_{1Q}y,\ \delta_{2Q}y, \ldots\ \delta_{(i-1)Q}y,\ t] \end{cases} \quad (5-88)$$

Therefore, in each iteration, there will be a transmission error ϵ_{Tx}, ϵ_{Ty} as following:

$$\begin{cases} \epsilon_{Tx} = f_{iQx}\ [x_{oQ},\ \delta_{1Q}x,\ \delta_{2Q}x, \ldots t] - f_{iQDx}\ [x_{oQ},\ \delta_{1Q}x, \\ \qquad ,\ \delta_{2Q}x, \ldots\ \delta_{(i-1)Q}x, t] \\ \epsilon_{Ty} = f_{iQy}\ [y_{oQ},\ \delta_{1Q}y,\ \delta_{2Q}y, \ldots t] - f_{iQDy}\ [y_{oQ},\ \delta_{1Q}y, \\ \qquad ,\ \delta_{2Q}y, \ldots\ \delta_{(i-1)Q}y, t] \end{cases} \quad (5-89)$$

which cause the total transmission error ϵ_{Tr}.

The block diagram of incremental computer, as it was discussed earlier, is shown in figure (5.3).

In order to suprime the delay T and the transmission error ϵ_{Tr} it is sufficient to extrapolate the output increment $\delta_{i-1}\mathcal{S}$ of each integrator in parallel incremental computer, in interval $t \in (t_i, t_{i+1})$. In this way, the input data of integrator k will be:

$$\begin{cases} x_{oQ},\ \delta_{1Q}x,\ \delta_{2Q}x, \ldots\ldots \quad ,\ \delta_{iQ}x,\ t \\ y_{oQ},\ \delta_{1Q}y,\ \delta_{2Q}y, \ldots\ldots \quad ,\ \delta_{iQ}y,\ t \end{cases} \quad (5-90)$$

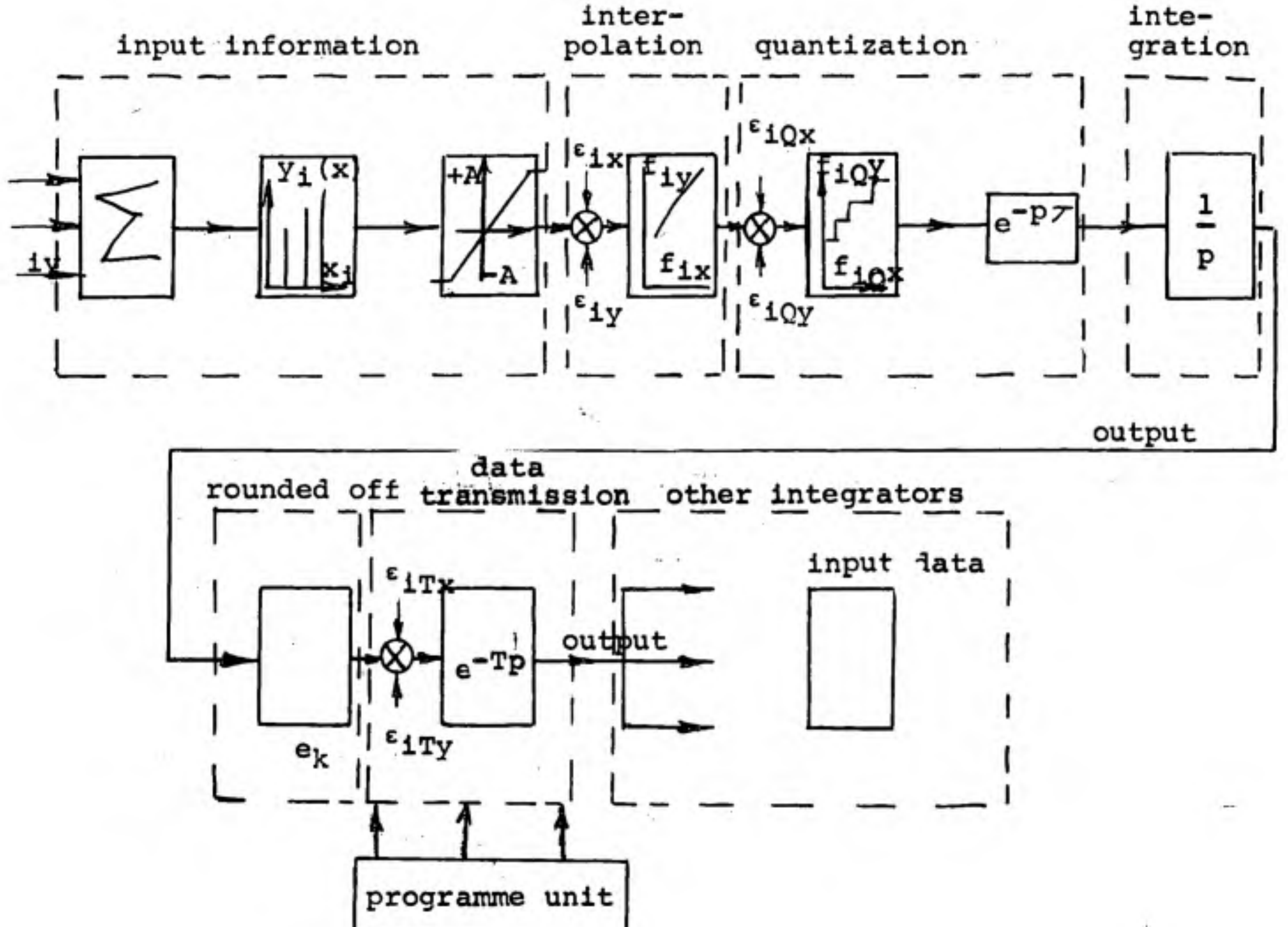


fig. 5.3.

Block diagram of I.C.

instead of the data:

$$\begin{cases} x_{oQ}, \delta_{1Q}x, \delta_{2Q}x, \ldots\ldots & , \delta_{(i-1)Q}x, t \\ y_{oQ}, \delta_{1Q}y, \delta_{2Q}y, \ldots\ldots & , \delta_{(i-1)Q}y, t \end{cases} \qquad (5\text{-}91)$$

The transfer function of extrapolators $F(p) = e^{pT}$, which use the output data of integrators $\delta_{1Q}s, \delta_{2Q}s, \ldots\ldots, \delta_{(i-1)Q}s$ is defined as:

$$F\ (p) = F\ (\delta_{1Q}s, \delta_{2Q}s, \delta_{3Q}s, \ldots\ldots, \delta_{(i-1)Q}s) \qquad (5\text{-}92)$$

The new block diagram of incremental machine is drawn in figure (5.4).

In serial incremental computer, in proper way of programming, it is possible to suppress the delay T and transmission error ε_{Tr} between integrators. If n integrators are connected in cascade (fig 5.5) and are proceeded in the same direction, then the input data of each integrator in i iteration (except the first one) will be as following:

$$\begin{cases} x_{oQ}, \delta_{1Q}x, \delta_{2Q}x, \ldots\ldots & , \delta_{iQ}x \\ y_{oQ}, \delta_{1Q}y, \delta_{2Q}y, \ldots\ldots & , \delta_{iQ}y \end{cases} \qquad (5\text{-}93)$$

For instant, the input δy of integrator 2 has the information of i^{th} iteration of integrator 1 as following:

$$(\delta_1 s)_i = (\delta_2 y)_i \qquad (5\text{-}94)$$

$$\begin{cases} f_{1QY} = T \\ f_{iQY} = f_{iQY}\,(Y_o,\ \delta_{1Q}Y,\ \delta_{2Q}Y, \ldots\ldots \quad ,\delta_{iQ}Y) \end{cases}$$

In this case, because the data of i th iteration exist in the input of all integrators (except the first one), there is no transmission error and there is no need of extrapolation block.

Example: In order to generate $Y = \cos \omega t$ the interconnection diagram is as figure (5.6).

In the figure (5.6), the integrators are connected in cascade and proceeded in the same direction. Therefore, the following difference equation may be written for each integrator.

integrator 1

$$\begin{cases} (\nabla I_1)_i = (Y)_i \cdot dt \\ Y_i = Y_{i-1} + (\nabla Y)_i \\ (\nabla Y)_i = -\ (\nabla I_3)_{i-1} \end{cases} \tag{5-95}$$

integrator 2

$$\begin{cases} (\nabla I_2)_i = \omega^2 \cdot (dx)_i \\ (dx)_i = (\nabla I_1)_i \end{cases} \tag{5-96}$$

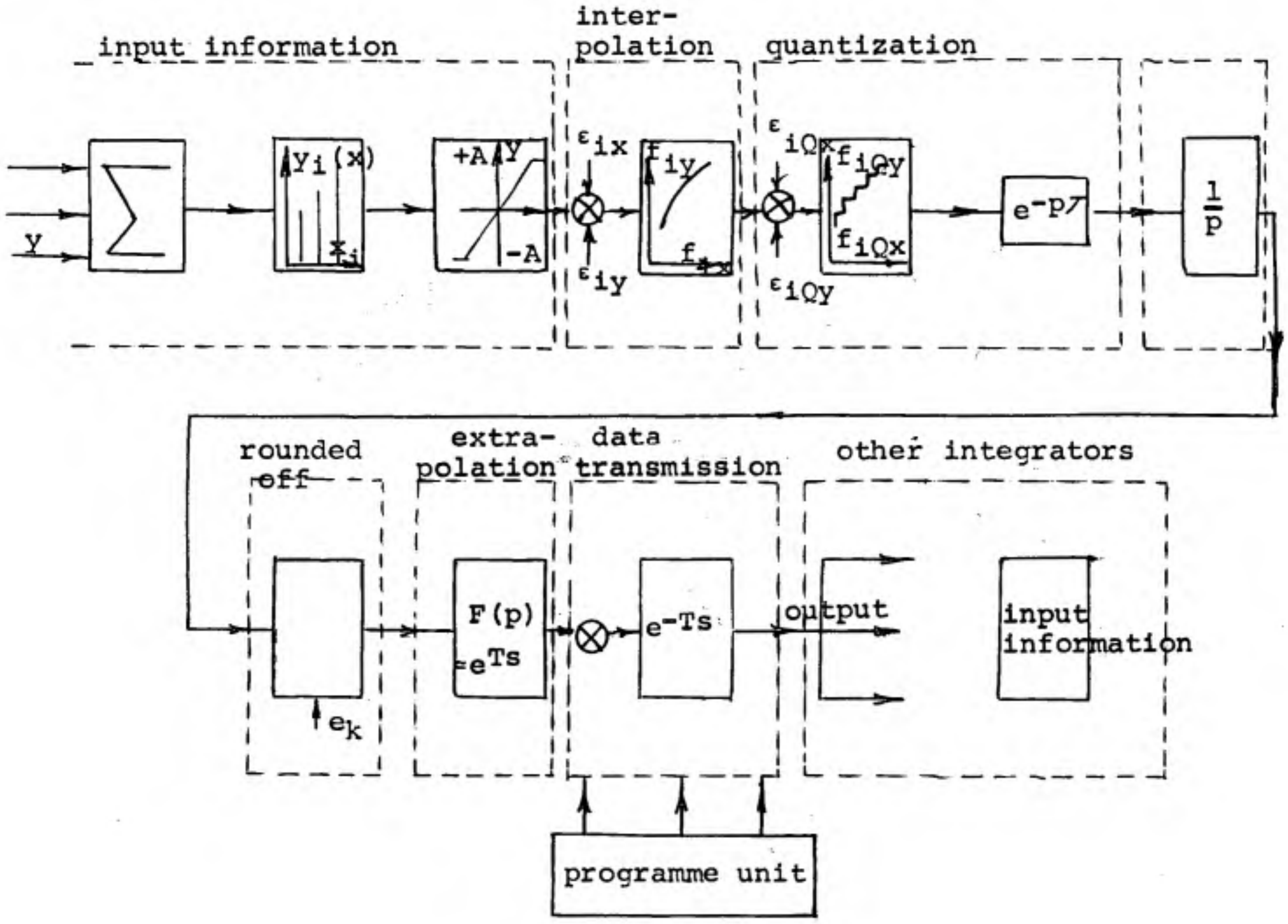


fig. 5.4.

Block diagram of I.C. by adding the extrapolation block X.

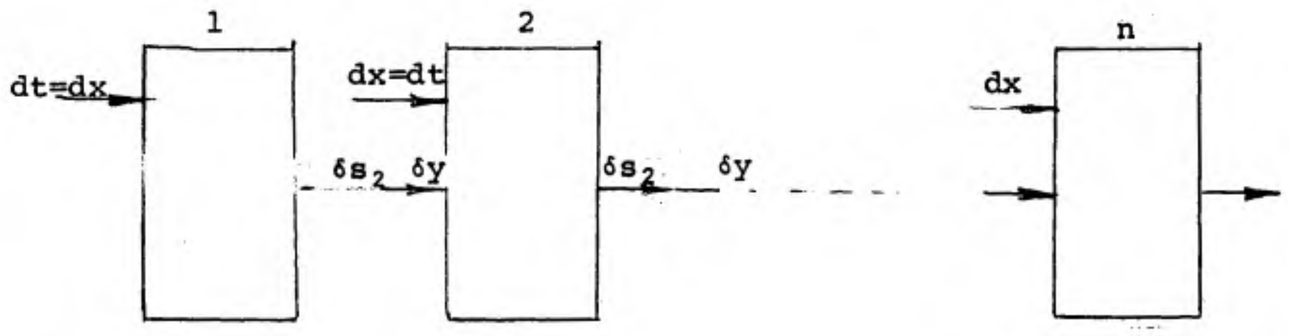


fig. 5.5.

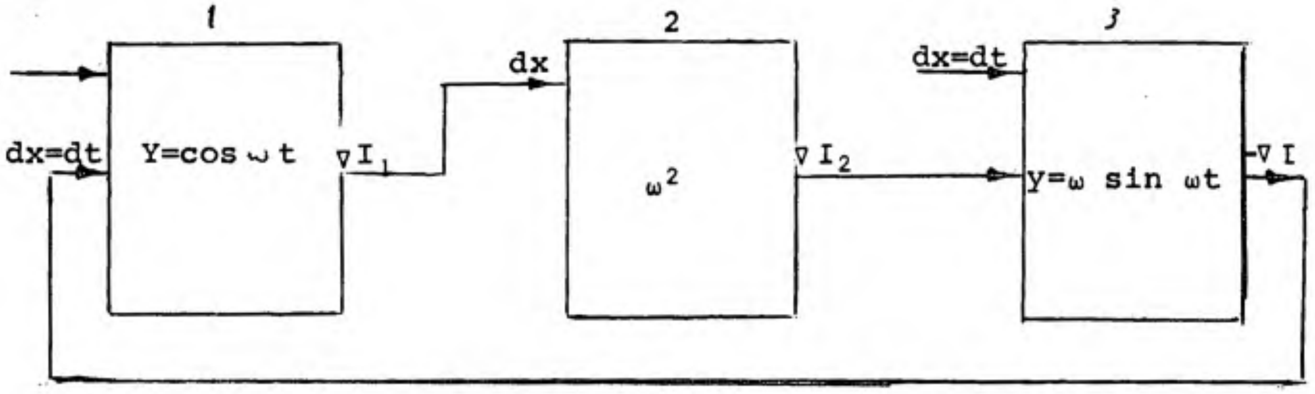


fig. 5.6.

and for integrator 3

$$\begin{cases} (\nabla I_3)_i = y_i \ (dt) \\ y_i = y_{i-1} + (\nabla y)_i \\ (\nabla y)_i = (\nabla I_2)_i \end{cases} \tag{5-97}$$

The equations (5-95), (5-96) and (5-97) can be written as:

$$\begin{cases} (\nabla I_1)_i = Y_i \cdot dt \\ Y_i = Y_{i-1} - (\nabla I_3)_{i-1} \end{cases} \tag{5-98}$$

$$(\nabla I_2)_i = \omega^2 \cdot (\nabla I_1)_i \tag{5-99}$$

$$\begin{cases} (\nabla I_3)_i = y_i \cdot dt \\ y_i = y_{i-1} + (\nabla I_2)_i \end{cases} \tag{5-100}$$

by finding the Y_i and taking its Z transform, from equations (5-98), (5-99) and (5-100), the Y (z) will be:

$$Y\ (z) = \frac{Y\ (0)\ (1 - z^{-1})}{1 - 2\ (1 - \omega^2\frac{T^2}{2})\ z^{-1} + z^{-2}} \tag{5-101}$$

The inverse Z transform of equation (5-101) will be:

$$\begin{cases} Y\ (iT) = \dfrac{Y\ (0)}{(1 - \omega^2\frac{T^2}{4})^{1/2}} \ \cos\ (i\Omega\ T + \lambda) \end{cases}$$

$$\begin{cases} \Omega = \dfrac{1}{T} \text{ arc tg } \dfrac{\omega T \left(1 - \omega^2 \frac{T^2}{4}\right)^{1/2}}{1 - \omega^2 \frac{T^2}{2}} \\ \\ \lambda = \text{arc tg } \dfrac{2 \omega T}{4 - \omega^2 T^2} \end{cases} \qquad (5\text{-}102)$$

As it can be seen from equation (5-102), if T becomes very small $T \rightarrow 0$. Then, amplitude, frequency and phase of the approximated function Y (iT) will be very near to the original function Y = cos ωT as following:

$$T = \rightarrow 0$$

$$\Omega = \left[\frac{1}{T} \text{ arc tg } \frac{\omega T \left(1 - \omega^2 \frac{T^2}{4}\right)^{1/2}}{1 - \omega^2 \frac{T^2}{2}} \right]_{T \rightarrow 0} \simeq \omega \qquad (5\text{-}104)$$

$$\left[\frac{Y\ (0)}{\left(1 - \omega^2 \frac{T^2}{4}\right)^{1/2}} \right]_{T \rightarrow 0} \simeq Y\ (0) \qquad (5\text{-}105)$$

$$\lambda = \left[\text{arc tg } \frac{2\, \omega\, T}{4 - \omega^2 T^2} \right]_{T \rightarrow 0} \simeq 0 \qquad (5\text{-}106)$$

so the equation (5-102) can be written as:

$$Y\ (iT) \simeq \cos\ (\omega i T) \qquad (5\text{-}107)$$

Of course, the small difference which exist between the original function $Y = \cos \omega t$ and the equation (5-102) is, because of the errors of method, the quantization, round off, of the integrators and also the transmission error in the first integrator.

In the same problem, if the integrators are series coscaded in reverse direction, the interconnection diagram will be as figure (5.7).

The increments $(\nabla I_1)_{i-1}$, $(\nabla I_2)_{i-1}$ and $(\nabla I_3)_{i-1}$ are in the input of integral, in the i^{th} iteration. So it is like the parallel incremental computer which use the data with the time delay of T (machine cycle). Therefore there will be the error of transmission in each integral as following:

in integrator 1
$$\begin{cases} \varepsilon_{Tx} = 0 \qquad (f_{iQx} = t) \\ \varepsilon_{Ty} = (\nabla I_2)_{i-1} - (\nabla I_2)_i \end{cases} \tag{5-108}$$

in integrator 2
$$\begin{cases} \varepsilon_{Tx} = (\nabla I_1)_{i-1} - (\nabla I_1)_i \\ \varepsilon_{Ty} = 0 \end{cases} \tag{5-109}$$

in integrator 3
$$\begin{cases} \varepsilon_{Tx} = 0 \\ \varepsilon_{Ty} = (\nabla I_3)_i - (\nabla I_3)_{i-1} \end{cases} \tag{5-110}$$

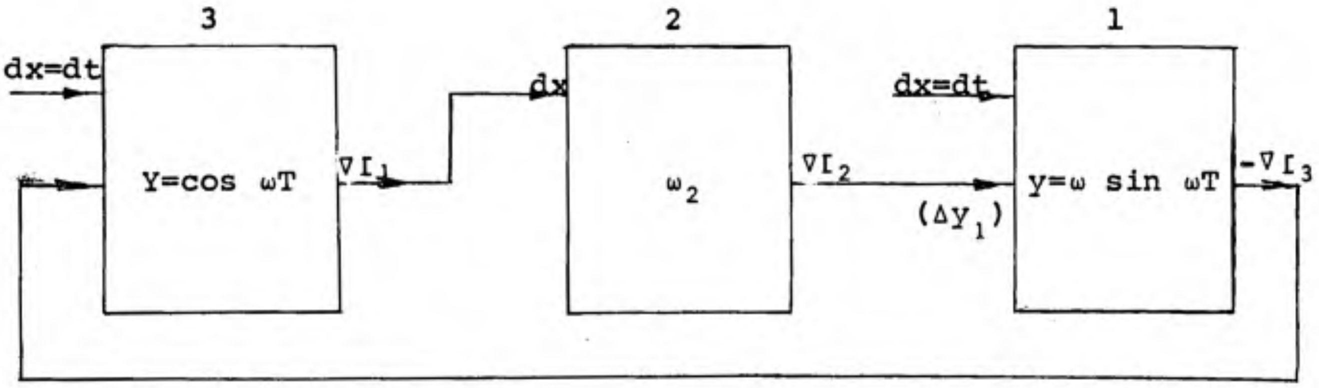


fig. 5.7.

The difference equation of the integrators will be:

integrator 1
$$\begin{cases} (\nabla I_3)_i = y_i \cdot (dt)_i \\ y_i = y_{i-1} + (\nabla y)_{i-1} \\ (\nabla y_1)_i = (\nabla I_2)_{i-1} \end{cases} \qquad (5\text{-}111)$$

integrator 2
$$\begin{cases} (\nabla I_2)_i = \omega^2 \cdot (dx_2)_i \\ (dx_2)_i = (\nabla I_1)_{i-1} \end{cases} \qquad (5\text{-}112)$$

integrator 3
$$\begin{cases} (\nabla I_1)_i = Y_i \cdot (dt)_i \\ Y_i = Y_{i-1} + (\nabla Y_3)_i \\ (\nabla Y_3)_i = -(\nabla I_3)_i \end{cases} \qquad (5\text{-}113)$$

by solving the equations (5-111), (5-112) and (5-113), and taking the Z transform of them, the result will be:

$$Y(z) = \frac{Y(0)\,(1 - z^{-1})}{z^{-2}\,(1 + \omega^2T^2) - 2z^{-1} + 1} \qquad (5\text{-}114)$$

and its inverse Z transform is expressed as:

$$Y(iT) = Y(0)\, e^{i \log \sqrt{1+\omega^2T^2}} \cos(i \operatorname{arc\,tg} \omega T) \qquad (5\text{-}115)$$

As it is seen from equation (5-115), the generation of function $Y(iT) = Y(0) \cos \omega T$, is followed by the generation of an exponential function $e^{i \log 1 + \omega^2 T^2}$, which is the result of accumulation of the transmission error ϵ_{Tx} and ϵ_{Ty} in each iteration.

It is seen, when the integrators are series coscades in the same direction which they are process, the data in the input of each integrator are:

$$\delta_{1Q}x, \ \delta_{2Q}x, \dots \ \delta_{iQ}x$$

and

$$\delta_{1Q}y, \ \delta_{2Q}y, \dots \ \delta_{iQ}y$$

so there is no delay T of transmission data which has existed in the parallel incremental computer, therefore, there is no transmission error ϵ_{Tr}.

In the second case, where the integrators are series coscaded in reverse direction, because in the input of each integrator in i th iteration, there is the information of former iteration: $\delta_{1Q}x, \ \delta_{2Q}x, \dots \ \delta_{(i-1)Q}x$ and $\delta_{1Q}y, \ \delta_{2Q}y, \dots \ \delta_{(i-1)Q}y$, there will be a delay of T in the transmission of data, which cause the transmission error ϵ_{Tx} and ϵ_{Ty} in each iteration as it was the case in the parallel incremental computer. The conclusion is that, in order to minimize the transmission error in serial machine, the programming should be in such a way that in the i th iteration, the input of most integrators use the data of the i th iteration. Those integrators which cannot receive the data of the i th iteration, should use an extrapolator unit as it was the case in the parallel incremental computer.

If the output extrapolation value of each integrator is called δ^{*}_{QE} s, then the above discussion can be formulated in the following form:

$$\delta^{*}_{QE}\, s_{ji} = A_{jK}\, \delta_{i}\, s^{*}_{Q} + B_{(j-K)}\, \delta_{QE}\, s^{*}_{i} \tag{5-116}$$

where

$$\begin{cases} A_{j-K} = 1 & \text{if} \quad j-K > 0 \\ A_{j-K} = 0 & \text{if} \quad j-K \leqslant 0 \end{cases} \tag{5-117}$$

$$\begin{cases} B_{j-K} = 1 & \text{if} \quad j-K \geqslant 0 \\ B_{j-K} = 0 & \text{if} \quad j-K < 0 \end{cases} \tag{5-118}$$

and $\delta_{i}\, s^{*}_{Q}$ is the approximated interpolated and quantized value of integral function. By taking into account the above discussion and the equations (5-116), (5-117) and (5-118), the block diagram of serial incremental computer can be represented as in the figure (5.8).

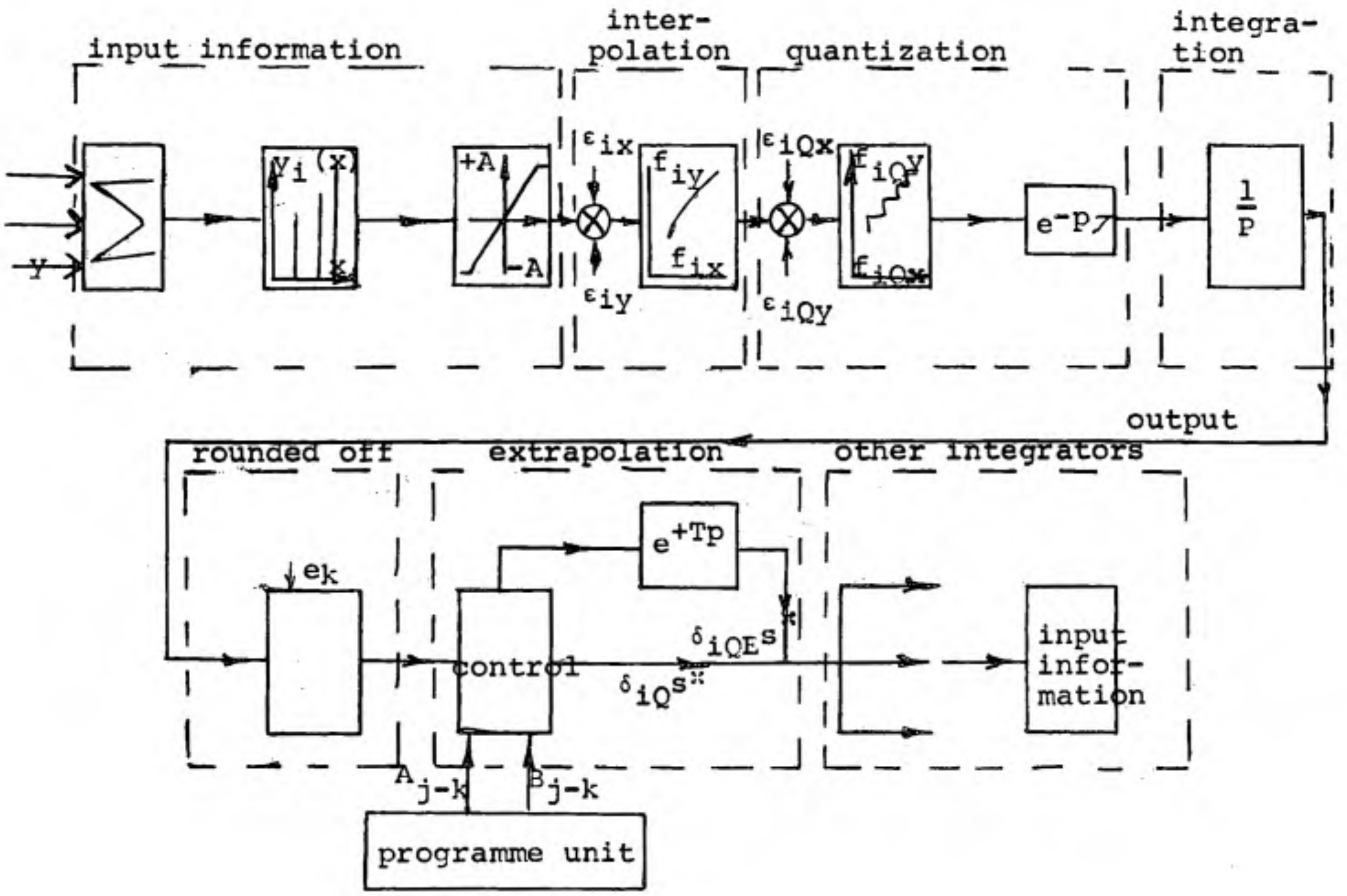


fig. 5.8.

The block diagram of serial I.C. with the extrapolation unit.

5.5 Choice of number of bits in multiple increment computer.

As it was seen, the value of integral function S (t) is replaced by the approximated interpolated quantized and rounded off value of integral $\overset{*}{S}_{Qm}(t)$ with the error ε_t as :

$$\varepsilon_t = S\ (t) - \overset{*}{S}_{Qm}(t) \qquad (5\text{-}119)$$

and the relative error is defined as :

$$\gamma = \frac{\varepsilon_t}{S\ (t)} \qquad (5\text{-}120)$$

If G(p) and $\overset{*}{G}$ (p) are transfer function of desired value of integration and approximated one, then the relative error γ(p) will be

$$\gamma(p) = \frac{G(p) - \overset{*}{G}(p)}{G\ (p)} = 1 - \frac{\overset{*}{G}(p)}{G(p)} \qquad (5\text{-}121)$$

for $p = j\omega$, the value of $G(\omega) = \frac{1}{\omega}$ so absolute value of relative error $\gamma(\omega)$ will be

$$\gamma(\omega) = 1 - \omega\ \overset{*}{G}\ (\omega) \qquad (5\text{-}122)$$

In order to find the relative error $\gamma(\omega)$, we should find the transfer function of approximated value of integration $\overset{*}{G}\ (\omega)$, for each method of integration.

As for multiple incremental computation, the degree of interpolation function is more than one, therefore we calculate the transfer function of first degree (trapezoidal) and higher degree interpolation formula for approximate integration.

The numerical realization of integration by the trapezoidal method, is:

$$\delta_i S^*_Q(t) = \int_{t-T}^{t} i(t) \circ dt \tag{5-123}$$

$$= \frac{T}{2}[i^*(t) - i^*(t - T)]$$

if the integral is realized by this numerical method, the calculations are performed by the following recurrence relations:

$$s^*_Q(t) = s^*_Q(t - T) + \frac{T}{2}[i^*(t) + i^*(t - T)] \tag{5-124}$$

its transfer function will be:

$$G^*(p) = \frac{T}{2} \circ \frac{1 + e^{-pT}}{1 - e^{-pT}} \tag{5-125}$$

substituting $j\omega$ for p and $e^{\pm j\omega t} = \cos \omega t \pm j \sin \omega t$ in equation (5-125), we get:

$$G^*(j\omega) = - j \frac{T}{2} \cot \frac{\omega T}{2} \tag{5-126}$$

$$A^*(\omega) = |G^*(\omega)|$$

$$= \left| \frac{T}{2} \cot \frac{\omega T}{2} \right| \tag{5-127}$$

$$\theta^{*}(\omega) = -\frac{\Pi}{2}$$

The ratio of transfer function of the approximate formula $A^{*}(\omega)$ (eq. 5-127) to the transfer function of the desired one

$$\frac{A^{*}(j\omega)}{G\ (j\omega)} = \frac{\frac{T}{2}\cot\frac{\omega T}{2}}{\frac{1}{\omega}}$$

$$= \frac{\omega T}{2} \cdot \cot\frac{\omega T}{2} \tag{5-128}$$

for three points formula of integration:

$$\delta_{i}S_{\ddot{Q}}^{*} = \int_{t-T}^{t} i\ (t) \cdot dt \tag{5-129}$$

$$\approx T\ [i^{*}(t) + \frac{1}{2}\Delta i(t) + \frac{1}{12}[\Delta i(t) - \Delta i(t+T)]\ ] \tag{5-130}$$

by putting:

$$\Delta i(t) = i(t-T) - i(t)$$

$$\Delta i(t+T) = i(t) - i(t+T) \tag{5-131}$$

in equation (5-130) we get:

$$\delta_i S_Q^* (t) = T \left[\frac{1}{3} i(t) + \frac{7}{12} i(t - T) + \frac{1}{12} i(t + T)\right] \quad (5\text{-}132)$$

Writting the recurance equation we will have:

$$s_Q^* (t) = s_Q^* (t - T) + T\left[\frac{1}{3} i(t) + \frac{7}{12} i(t - T) + \frac{1}{12} i(t + T)\right] \quad (5\text{-}133)$$

Taking the transfer function of equation (5-133) we get:

$$G^*(p) = \frac{s^*(p)}{I\ (p)} \quad (5\text{-}134)$$

$$= T\,\frac{\frac{1}{3} + \frac{1}{12}\, e^{pT} + \frac{7}{12}\, e^{-pT}}{1 - e^{-pT}}$$

by puting :

$$p = j\omega$$

and

$$e^{\pm j\omega t} = \cos \omega t \pm j \sin \omega t$$

the equation (5-134)can be written as:

$$A^*(j\omega) = |G^*(j\omega)| \quad (5\text{-}135)$$

$$= \frac{T}{24} \; \frac{\sqrt{28 \cos^2 \omega T + 16 \;\; \cos \omega T + 52}}{\sin \frac{\omega T}{2}}$$

The ratio of the approximate formula $A^{*}(j\omega)$, (eq.5-135 to the transfer function of the derired one G $(j\omega)$, is as following

$$\frac{A^{*}(j\omega)}{G\;(j\omega)} = \frac{\frac{T}{24} \frac{\sqrt{28 \cos^2 \omega T + 16 \;\; \cos \omega T + 52}}{\sin \omega T/2}}{\frac{1}{\omega}} \qquad (5\text{-}136)$$

$$= \frac{\omega T}{12} \; \frac{\sqrt{7 \cos^2 \omega T + \;\; 4 \cos \omega T + 13}}{\sin \frac{\omega T}{2}}$$

We will now consider Simpson's method of integration. This formula is obtained by integrating the second order interpolation polynomial which coincides with the abscissae at points t, t-T, t-2T and with discrete values of the input signal at these instants.

The difinite integral is calculated by the equation:

$$\delta_i \; S_Q^{*}(t) = \int_{t-2T}^{t} i(t) \cdot dt \qquad (5\text{-}137)$$

$$\approx \frac{T}{3}[i^{*}(t) + 4i^{*}(t - T) + i^{*}(t - 2T)]$$

If this method is employed, the following recurrence relation is used:

$$I^{*}(t) = I(t - 2T) + \frac{T}{3}[i(t) + 4i^{*}(t - T) + i^{*}(t - 2T)] \qquad (5 - 138)$$

The transfer function is:

$$G^{*}(p) = \frac{T}{3} \; \frac{1 + 4\,e^{-pT} + e^{-\,2pT}}{1 - e^{2pT}} \qquad (5 - 139$$

substituting $j\omega$ for p and $e^{j\omega t} = \cos \omega t + j \sin \omega t$ in formula (5 - 139), we get

$$G^{*}(j\omega) = - j \,\frac{T}{3} \cdot \frac{2 + \cos \omega T}{\sin \omega T} \qquad (5 - 140)$$

$$A^{*}(\omega) = G^{*}(j\omega)$$

$$= \left| \frac{T}{3} \cdot \frac{2 + \cos \omega T}{\sin \omega T} \right| \qquad (5 - 141)$$

$$\theta^{*}(\omega) = - \frac{\Pi}{2} \qquad (5 - 142)$$

The ratio of transfer function $A^{*}(\omega)$ of the approximate formula (eq.5-141 to the transfer function of the desired one is :

$$\frac{A^{*}(\omega)}{G\ (\omega)} = \frac{T\omega}{3} \cdot \frac{2 + \cos \omega T}{\sin \omega T} \qquad (5 - 143)$$

Then the relative error for each method of integration is:

1 - in trapezoidal method:

$$\gamma\ (\omega) = 1 - \frac{\omega t}{2}\ \cot \frac{\omega t}{2} \qquad (5\text{-}144)$$

2 - in three points method:

$$\gamma\ (\omega) = 1 - \frac{\omega t}{12}\ \frac{\sqrt{7\cos^2 \omega t + 4\cos \omega t + 13}}{\sin \frac{\omega t}{2}} \qquad (1\text{-}145)$$

3 - in Simpson's method:

$$\gamma\ (\omega) = 1 - \frac{\omega t}{3}\ \frac{2 + \cos \omega t}{\sin \omega t} \qquad (5\text{-}146)$$

If the ω_c is the maximum froquency that can pass in the linear part of characteristic frequency of incremental computer, without producing the distortion and phase shift at the output, then we will have:

$$y = A \cdot \sin \omega_c\ t \qquad (5\text{-}147)$$

so the increment δy, which is applied to the input of increment computer is:

$$\frac{\delta y}{\delta t} \simeq \frac{dy}{dt} \qquad (5\text{-}148)$$

$$= A \circ \omega \cdot \cos \omega_c t \qquad (5\text{-}149)$$

or

$$\delta y = A \cdot \omega \cdot \delta t \cos \omega_c t \qquad (5\text{-}150)$$

where δt is the machine cycle T, from equation (5-150), the y is maximum when $\cos \omega_c t = 1$

so

$$(\delta y)_{max} = (A)_{max} \cdot \omega_c \cdot T \qquad (5\text{-}151)$$

$(A)_{max}$ is the maximum capacity of Y register of the incremental computer, which is equal to:

$$(A)_{max} = 2^n - 1 \qquad (5\text{-}152)$$

where n is the number of bits of Y register. The maximum value of increments $(\delta y)_{max}$ is equal to:

$$(\delta y)_{max} = 2^h - 1 \qquad (5\text{-}153)$$

where h is the number of bits in incremental register δy. By putting the value of $(A)_{max}$ and $(\delta y)_{max}$ from equations (5-152), (5-153), in equation (5-151), we will have:

$$2^h - 1 = T \circ \omega_c (2 - 1) \qquad (5\text{-}154)$$

The maximum frequency ω_c can be found from equation (5-154) as following:

$$\omega_c = \frac{2^h - 1}{(2^n - 1)T} \tag{5-155}$$

The choice of the number of bits in incremental register h, can be made as high as it is desired in such a way that the error $\gamma(\omega)$ does not become greater than the quantum $\Delta y = 2^{-n}$, so:

$$\gamma(\omega) < 2^{-n} \tag{5-156}$$

By using the equation (5-156) for the error of different method of integration, the relation between h and n can be found by expanding the equation (5-144), (5-145) and (5-146) in ω and taking the first term of expansion as the approximation. By putting its maximum value ω_c from equation (5-155) in them, then for trapezoidal method of integration we will have:

$$2^h < \sqrt{3}\, 2^{\frac{n+2}{2}} \tag{5-157}$$

for $n = 10$

$h < 6$

For three points and Simpson's method, we will have:

$$2^h < 1 \cdot 15 \times 2^{\frac{3n+3}{4}} \tag{5-158}$$

for $n = 10$

$h < 8$

From the equations (5-157) and (5-158), it is seen that it is possible to increase the speed of integration by increasing the number of increment bits h from 1 to 8 bits for n = 10, in such a way that the error does not exceed the quantum Δy. But it is seen that the maximum value of h does not change too much from the first degree interpolation (trapezoidal method) to the second degree interpolation. The reason is that, when we increase the step of integration and the degree of interpolation, although we reduce the error of method, we do not reduce the quantization error ε_{tQ}.

It was discussed in chapter (3), the quantization error is practically same in all methods of integration. That is why, by using more accurate formula of integration, although the error of method is reduced, the total error ε_t will reduce slightly. So the remide of reducing the total error is reducing the quantization error as well. The best choice between the method of integration and the number of increment bits h, is to choose the error of method $\Gamma(t)$ approximately equal to the quantization error ε_{tQ} (t).

5.6. Choice of algorithms for unitary and multiple increment computers.

In order to choose the algorithms of unitary and multiple increments computers, with a desired accury, we should know the errors of the algorithms.

The error which exist in incremental computation is the sum of the error of method $\Gamma(t)$, the quantization error ε_{tQ}, the round off error e_k, and the transmission error ε_{Tr}. In the preceeding chapters, we have also seen how to reduce all these partial errors.

It is possible to reduce the error of method $\Gamma(t)$ as low as possible, by choosing the high degree interpolation formula. It was also seen that, it is not worthwhile to use higher than first degree interpolation formula for unitary incremental computation (ref. chapter 2).

The quantization error ε_{tQ} does not depend on the method of integration. It only depends on the quantums Δx, Δy and on the number of bits h in δX register.

The round off error e_k of increment computers can be minimized by the appropriate initial condition.

The transmission error ε_{Tr} can be minimized by adding the extrapolator blocks in parallel increment computers. In serial increment computers, the transmission error can be minimized by the proposed method of programming in the chapter 4. The total error ε_t is the sum of error of method $\Gamma(t)$, the quantization error ε_{tQ}, the round off error e_k, and the transmission error ε_{Tr}.

The choice of algorithms in incremental computation must be made in such a way that, with minimal equipment and calculating time, the total error ε_t does not exceed a certain limit.

We resume here the rectangular, trapezoidal methods, and their partial resulting errors, as well as the total errors and the algorithms.

In rectangular method of integration in unitary increment computers, the error of method $\Gamma(t)$ is:

$$\Gamma(t) < \frac{7}{2}(x_k - x_o)\ \Delta y$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < \Delta y\ (x_k - x_o)$$

Then the total error ε_t is:

$$\varepsilon_t < \frac{9}{2}\Delta y\ (x_k - x_o) + e_k + \varepsilon_{Tr}$$

and the algorithms of integration are:

$$s_Q^{*}\ (t) = \sum_{i=1}^{k} y_{iQ} \cdot \Delta_{iQ}x$$

$$y_{iQ} = y_{(i-1)Q} + \sum_{j=1}^{b} (\Delta_{iQ}y)_j$$

where b is the number of Δy input of incremental computer.

In trapezoidal method of integration in unitary increment computers, the error of method $\Gamma(t)$ is:

$$\Gamma(t) < \frac{7}{12} (x_k - x_o) \Delta y$$

The quantization error ε_{tQ} is:

$$\varepsilon_{tQ} < \Delta y (x_k - x_o)$$

Then the total error ε_t is:

$$\varepsilon_t < \frac{19}{12} \Delta y (x_k - x_o) + e_k + \varepsilon_{Tr}$$

and the algorithms of integration are:

$$\begin{cases} s_Q^{*} (t) = \sum_{i=1}^{k} (y_{iQ} \cdot \Delta_{iQ}x + \frac{1}{2} \Delta_{iQ}y \cdot \Delta_{iQ}x) \\ y_{iQ} = y_{(i-1)Q} + \sum_{j=1}^{b} (\Delta_{iQ}y)_j \end{cases}$$

In unitary increment computers, by comparing the total error ε_t in rectangular and trapezoidal method of integration, it is seen that, the total error ε_t in trapezoidal method is 2·5 time smaller than the rectangular method. It is also seen that, it is not worthwhile to use an higher degree interpolation formula than the first degree interpolation.

For instance by choosing the three points method of integration, we have:

$$s^{*}(t) = \sum_{i=1}^{k} \left[y_{iQ} \cdot \Delta_{iQ}x + \frac{1}{2}\Delta_{iQ}x \cdot \Delta_{iQ}y + \right.$$

$$\left. + \left(\frac{1}{12}\Delta_{iQ}y \cdot \Delta_{(i-1)Q}x - \frac{1}{12}\Delta_{(i-1)Q}y \cdot \Delta_{iQ}x\right) \right]$$

In this equation, the quantities in paranthesis are smaller than the quantum Δy of the function $y(x)$. Therefore, the computer cannot detect and operate on it. So, when accurate algorithms are needed for unitary incremental computation, the algorithms of the trapezoidal method of integration are more convenient.

In multiple incremental computation, the step of integration δx is 2^r time larger than ($\delta x = 2^r \cdot \Delta x$) unitary increment computation. Therefore, the algorithms of this computation should be chosen with higher precision.

The three points integration formula gives a good approximation for multiple increment computation. The algorithm of this method is:

$$\begin{cases} s_Q^*(t) = \sum_{i=1}^{k} [y_{iQ} \cdot \delta_{iQ}x + \frac{1}{2}\delta_{iQ}y \cdot \delta_{iQ}x + \\ + \frac{1}{12}(\delta_{iQ}y \cdot \delta_{(i-1)Q}x - \delta_{(i-1)Q}y \cdot \delta_{iQ}x)] \\ y_{iQ} = y_{(i-1)Q} + \delta_{iQ}y \end{cases}$$

We have seen in (5 - 5) that it is possible to increase the step of integration δx to six bits, for ten bits Y register, in such a way that, the total error ε_t does not exceed the quantum Δy.

For larger integration step δx, we can choose the third, fourth, and even higher degree interpolation formula. But, by increasing the degree of the interpolation formula, the algorithm becomes more complex, so it needs too much equipment and calculating time. On the other hand, by increasing the step of integration, the quantization error ε_{tQ} is increased as well, and in some cases, the quantization error ε_{tQ} may dominate the total error ε_t.

Therefore, by choosing the step of integration δx, the most economical and convenient way of choosing the algorithm of machine is, to select a degree of interpolation formula, and the quantum Δy, in such a way that, the error of method $\Gamma(t)$, becomes equal to the quantization error ε_{tQ}.

CHAPTER V]

A NEW TYPE OF UNIVERSAL INCREMENTAL COMPUTER.

DESIGN - DEVELOPMENT - CONSTRUCTION.

Introduction.

The principal concept of the design philosophy of a new type of incremental computer is the realization of a high speed digital machine which is operationally analogous to a analogue computer with the accuracy and reliability of a digital machine. This computer performs the mathematical and logical operation of digital computer as well.Therefore this machine, successfuly combines the principal operation and advantages of digital and analogue computer.

The basic operation of this machine is the integration, summation, and multiplication of functions.

It also operates various mathematical operations as addition, substraction, multiplication, division as well as logical decision making.

The incremental computers can effectively be used for solving a system of linear or nonlinear differential equations, as well as algebraic, integral, trigonometric equations or other problems, associated with calculation of continuous functions.

The incremental computers may be devided into two types, serial and parallel:

In serial machine, the integration and others operations are sequentially performed one after another in the serial form. In this computer, the common units are: the memory unit, arithmetic unit, control unit and input output units.

The common memory permits the continuous solution of a mathematical problem to be achived serially, that is, one operation after another by means of only one arithmetic unit. Such a serial method of solving a problem by means of a single integrator in an analogue computer is in principle possible, if a suitable analogue memory is available.

In serial incremental computer, the interconnection of integrators into a sequence required for a problem solution, is achieved by means of a programme which may be fixed or altered, according to the result obtained during the problem-solving process.

In other words, this machine may achieve conditional or unconditional transfer from one programme to another in a manner similar to that achieved in a conventional digital computer. From the standpoint of economy, serial machines are conveniently used for solving complex nonlinear problems, requiring a relatively large number of integrators. It possess wide possibility for execution of various logic operations. The speed of problem solution in a serial machine depends on the complexity of the problem.

A parallel incremental computer consists of a number of individual digital integrators, each integrator being a self-contained unit operating independently of the other integrators. In a parallel machine, it is not necessary to have a common memory unit and an arithmetic unit. Interconnection of individual integrators are performed by a plugboard. The advantage of the parallel incremental computer is high computing speed, which does not depend on the complexity of the problem. This speed advantage is achieved at the expense of more integrators and, consequently, leads to a considerable increase in the electronic equipment. Thus, the use of parallel incremental computer is advisable only in those cases where an extremely high computation rate is necessary.

Combination of the problem-solving principal used in analogue computers and the digital technique permits to achieving in both types of incremental computers, an essentially new type of digital computer, possessing advantages of both analogue and digital machines.

On the other hand, the use of incremental computers permits a substantial increase in accuracy of computation, in flexibility and general applicability of the computer, as well as in simplicity of design and economy.

The incremental computer which is devised by the author in the industrial electronic laboratory of Brussel University is a new type of serial incremental computers.

6-1. Organization of computer.

The incremental computer consists of five main units: Arithmetic unit, programming unit, memory unit, control unit and input output unit.

The arithmetic unit is an electronic device which is able to carry out the integration based on unitary or multiple incremental computation, as well as the summation and multiplication of the functions. In addition, it operate the basic arithmetic operations, such as additions, substractions, multiplications and divisions, as general purpose digital computer. The multiplication is done in one time of addition with a special algorithm.

The programming unit is based on the new system of transmitting the information between the integrators. The transmission of data from the output of the integrators to the input of a desired integrator is performed on the patch pannel as the analogue computer. The transmissions data are carried out by one cable for unitary and multiple incremental computations on the patch pannel. Therefore, there is no need of storing the programme instruction in the memory, and the programme is easily checked by visual observations.

The memory unit is used to contain the initial numerical data, calculated intermediate results, the final results, as well as the sets of instruction for arithmetic operations.

The control unit is also indispensable for the automatic functioning of the computers. In general, it coordinates the operation of the other four main parts. The block diagram of incremental computer is shown in figure (6-I)

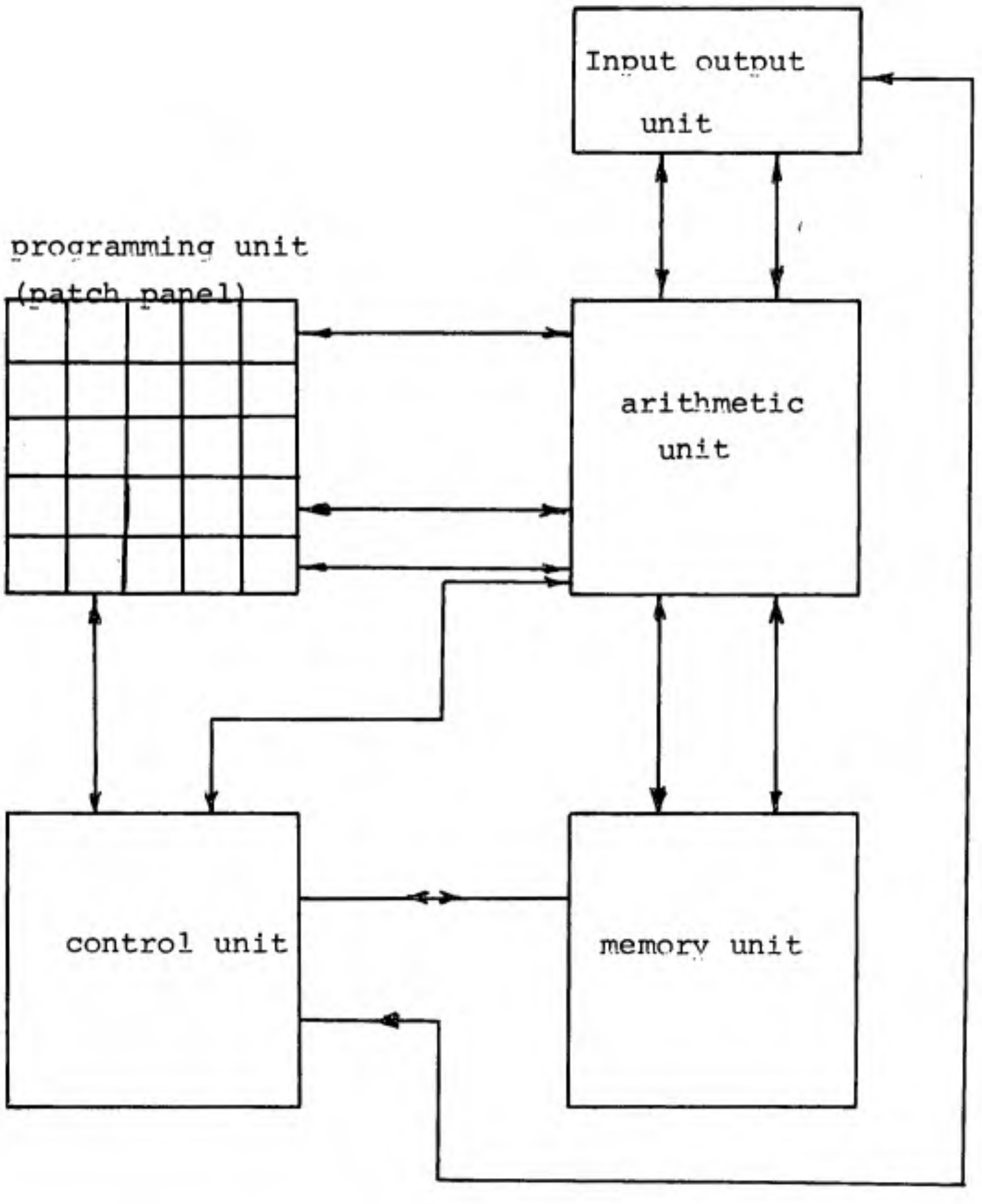

Input output unit
programming unit (patch panel)
arithmetic unit
control unit
memory unit

Fig 6-1

6 2. Arithmetic unit.

In serial incremental computer, the operations are proceeded sequentially in a common arithmetic unit.

The arithmetic unit is composed of six registers A, B, C, D, R, S, a multiplier M and some one bit adders, as they are shown in fig.(6-2). and(6-3). The logical design of this unit is described by building up the complete schematic from the basic logical element.

The principale operations are as following:

6 2.1. The incremental operations.

6.2.1.1. The integration operation based on the unitary incremental computation, by the rectangular method.

6 2.1.2. Integration operation based on the unitary incremental computation with trapezoidal method.

6 2.1.3. Integration operation based on the multiple incremental computation.

6.2.1.4. New system of multiplication of two functions.

6 2.1.5. Summation of the functions or digital servo.

6.2.1.6. Function generator.

6.2.2. Numerical operation.

6.2.2.1. Addition and substraction of two 10 bit numbers A, B.
$(B) := (B) \pm (A)$

6.2.2.2. Addition and substraction of two 20 bits number:

$(S,R) := (S,R) \pm (A,B$

6.2.2.3. Addition and substraction of three numbers A, B and R in one time of addition. $(R) := (R) + (B \pm A)$.

6.2.2.4. Multiplication of two numbers, in one time of addition

$(R,S) := (B) \circ (C)$

or

$(R,S) := (B \pm A) \circ (C)$

6.2.2.5. Division.

6.2.3. Logical and other auxiliary operations.

We explain now, the logic flow of each operation in more details.

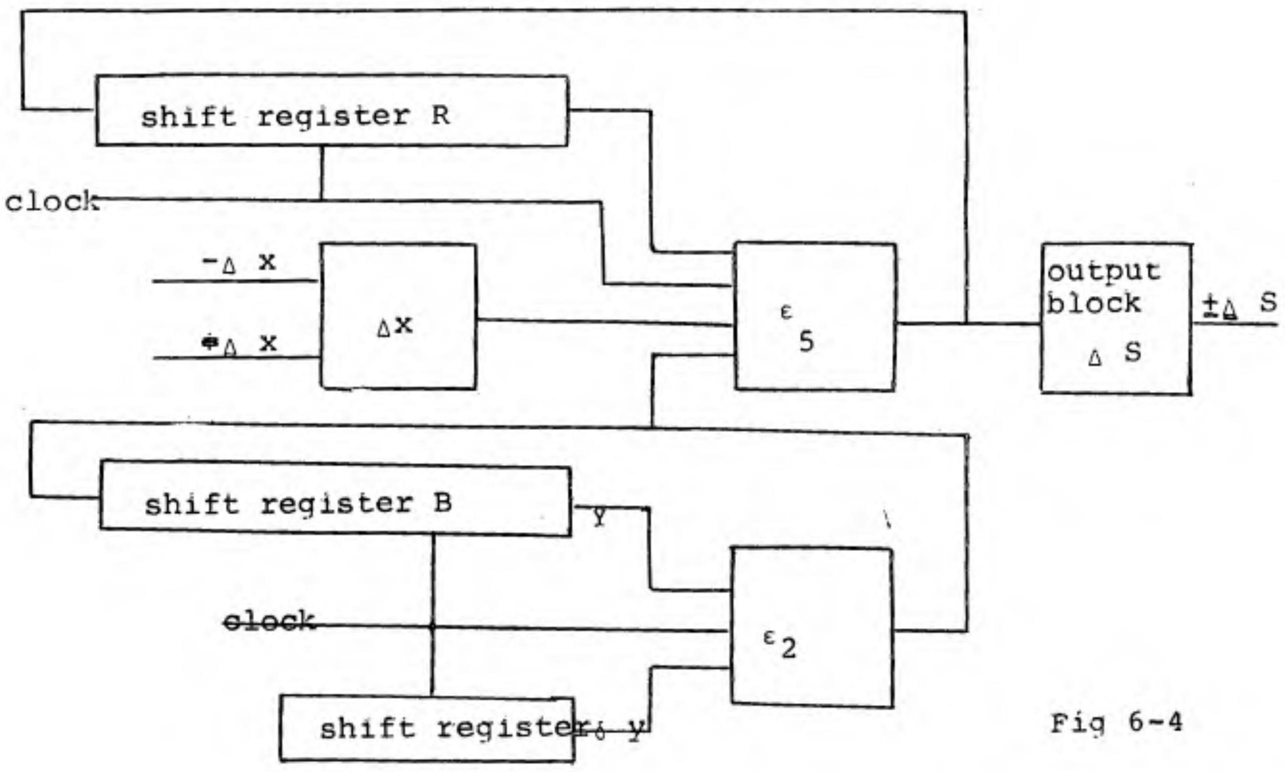
6.2.1.1. Integration operation, based on the unitary incremental computation, by the rectangular method.

The basic operation is formed by using three registers, δy, B, and R. As it was seen, the algorithm of this method is:

$$\begin{cases} y_{iQ} = y_{(i-1)Q} + \delta_{iQ}\, y \\ \Delta_{iQ} s = \dfrac{1}{2^n} \circ [y_{iQ} \circ \Delta_{iQ} x + s_{o(i-1)} - s_{oi}] \end{cases}$$

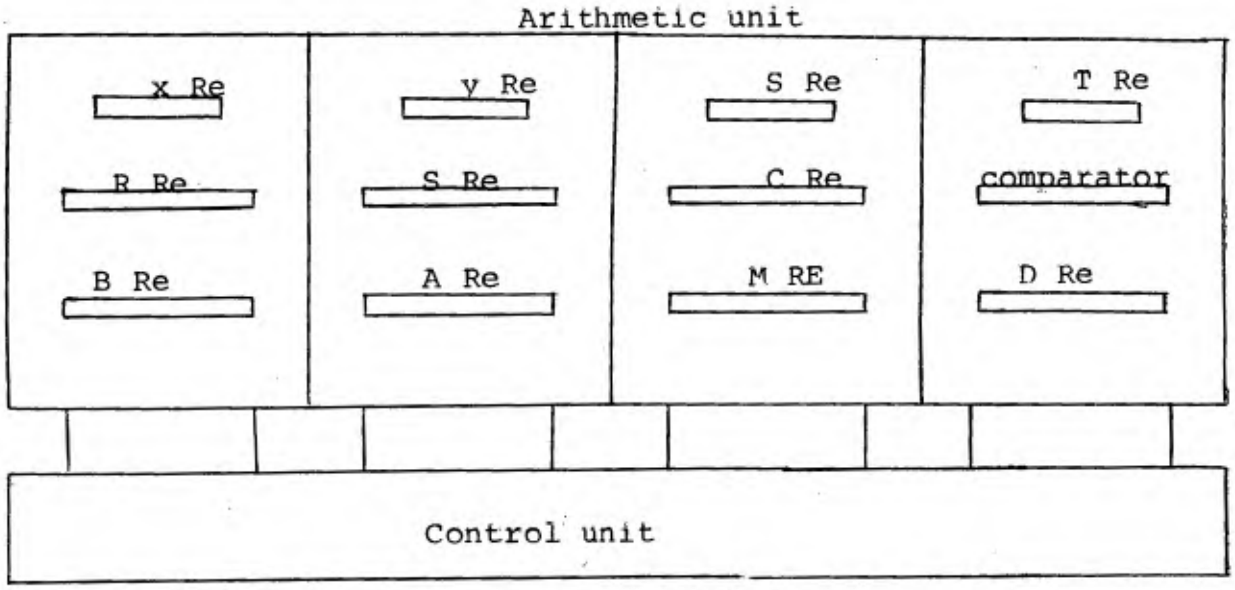
The value $\delta_i y$ comes from programming unit and is put in the register δy. The values of the functions $y_{(i-1)Q}$, and the rest of integral $s_{o(i-1)}$ which arrives from the memory, are put in the registers B and R. The first adding operation of the above equation is done by the registers B, $\delta_{iQ} y$ and the adder ε_2. The result is put in the register B. The output y_{iQ} of the adder ε_2 is sumed by the rest of integral $s_{o(i-1)}$ from register R in the adder ε_5 at the same time that the first operation is performed. If there is an overflow, the output block Δs gives the signe and the value of Δs which is stored in the increment memory of programming units for applying as the input of other integrators or output devices Fig. (6 - 4)

The communication system between the integrators, is the ternary system; that means an overflow Δs can have one of the three values ± 1 or 0. The actual significance of the "one" is of course, determined by the scale given to it during the programming of the machine. The ternary communication systems are well known and explained in the litrateaure, we explain only the algorithm by flow diagram. Fig.(6-5)



shift register R
clock
$-\Delta$ x
$+\Delta$ x
Δx
ε_5
output block
Δ S
$\pm\Delta$ S
shift register B
y
~~clock~~
ε_2
shift register$_0$ y

Fig 6-4



Arithmetic unit
x Re
R Re
B Re
y Re
S Re
A Re
S Re
C Re
M RE
T Re
comparator
D Re
Control unit

Fig 6 -3

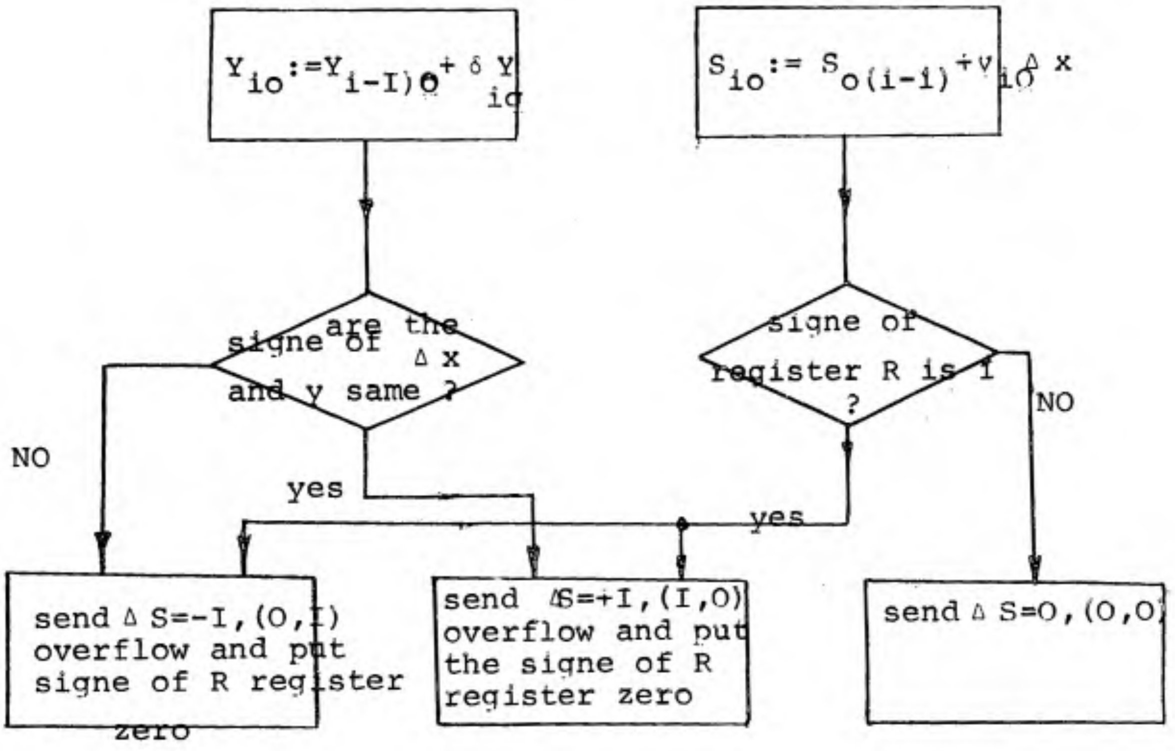

$Y_{io} := Y_{(i-I)o} + \delta Y_{io}$
$S_{io} := S_{o(i-1)} + v_{io} \Delta x$
are the signe of Δ x and y same ?
signe of register R is I ?
NO
yes
yes
NO
send Δ S=-I,(O,I) overflow and put signe of R register zero
send ΔS=+I,(I,O) overflow and put the signe of R register zero
send Δ S=O,(O,O)

fig 6-5

6.2.1.2. Integration operation based on the unitary incremental computation with the trapezoidal method of integration.

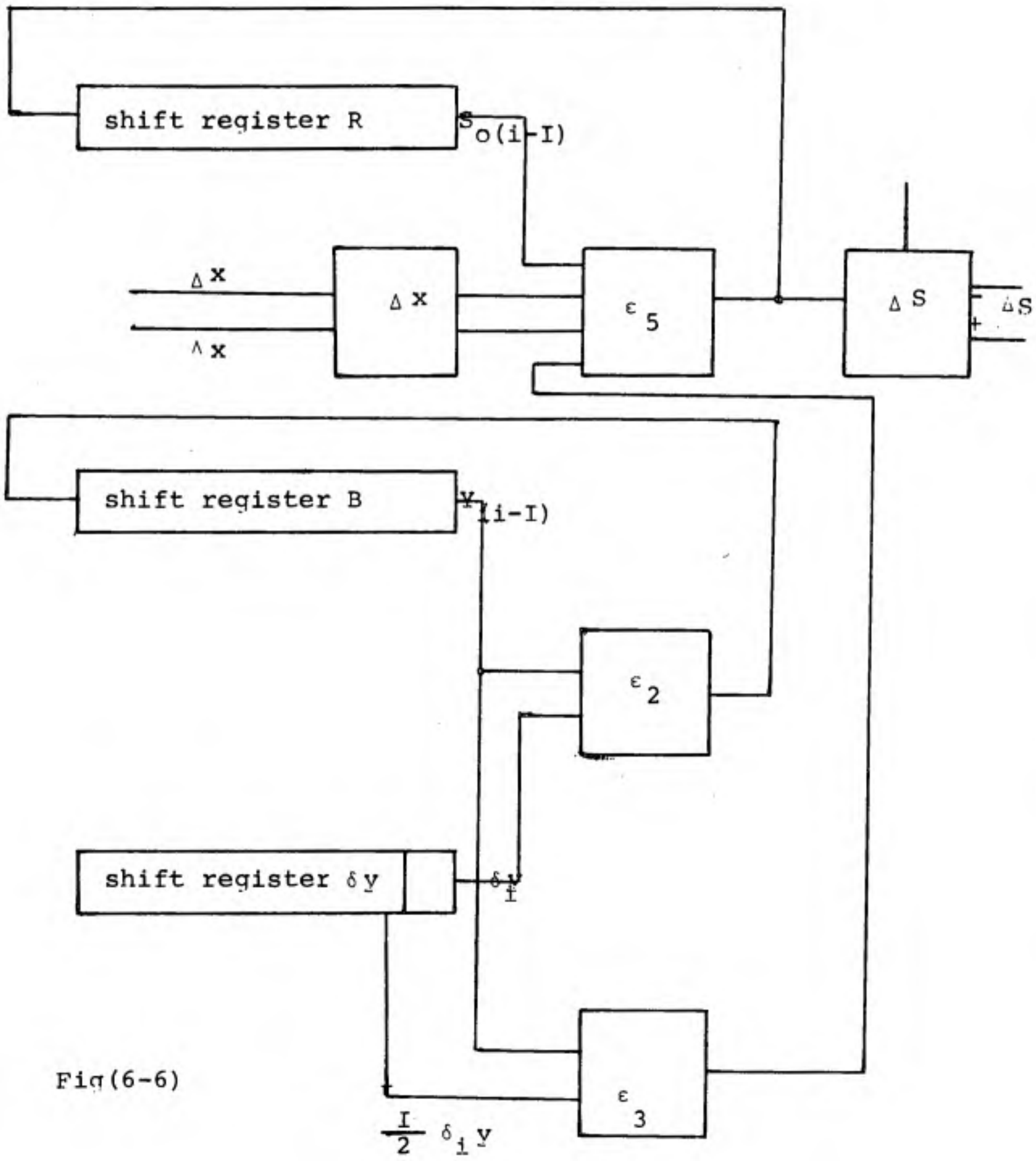
The algorithms of trapezoidal method of integration is:

$$\begin{cases} y_{iQ} = y_{(i-1)Q} + \delta_{iQ}y \\ \Delta_{iQ}s = \frac{1}{2} [(y_{(i-1)Q} + \frac{1}{2} \Delta_{iQ}y) \Delta_{iQ}x + \\ \qquad + s_{o(i-1)} - s_{oi}] \end{cases}$$

The logic flow of the system is shown in figure (6-6).
The first operation of the above equation is performed by the registers δy, B, and the adder $_{1}$. The adder ε_3 provide the sum $(y_{i-1} + \frac{1}{2} \Delta_{iQ}y)$. This sum is added to ($\Delta x = + 1$) or substracted from ($\Delta x = - 1$), inefected ($\Delta x = 0$) the rest of integral $s_{o(i-1)}$ by the adder ε_5.

The three addition operations are performed simultaneously. Therefore, it is very rapid and economical system.

The output overflow $\pm \Delta s$ will produce with the same algorithms, as rectangular method.



shift register R
$S_{o(i-I)}$
Δx
Δx
Λx
ε_5
ΔS
ΔS
shift register B
$y_{(i-I)}$
ε_2
shift register δy
δy_I
ε_3
$\frac{I}{2}\,\delta_i y$

Fig(6-6)

6.2.1.3. Integration based on multiple incremental computation.

As it was discussed earlier, by increasing the step of integration Δx in D.D.A. to $\delta x = 2^r \cdot \Delta x$, $(h>r>o)$ in multiple incremental computer, the speed of integration is increased by the factor of 2^r.

Assume y_{eq} is chosen in such a way that fulfil the following requirement:

$$\delta_i s_Q^{*} = y_{eq} \cdot \delta_{iQ} x$$

where

$$y_{eq} = f\,[y_{iQ},\ \delta_{iQ}x,\ \delta_{iQ}y,\ldots\ldots]$$

Therefore, if we put the value of y_{eq} in register y_{eq} (fig6-7) and multiply by the $\delta_{iQ}x$ in multiplier M, then the value of integral will be $y_{eq} \cdot \delta_{iQ}x$ in S register. If the number of bit in y_{eq} register is n, in δx register is h, then the number of bits of S register will be n + h. The rounded off value of integral $\delta_i s_{QM}^{*}$ is h most dignificant bits of S register from n + 1 to n + h. This value of integral is transmettid to the increment storage of programming unit as the output of integrator. The n less significant bits of S register goes to the memory for adding to the value of $y_{eq}\ \delta_{(i+1)Q}x$, in the next iteration (i+1).

The value of y_{eq} is formed from $\delta_{iQ}x$, $\delta_{iQ}y$ which comes

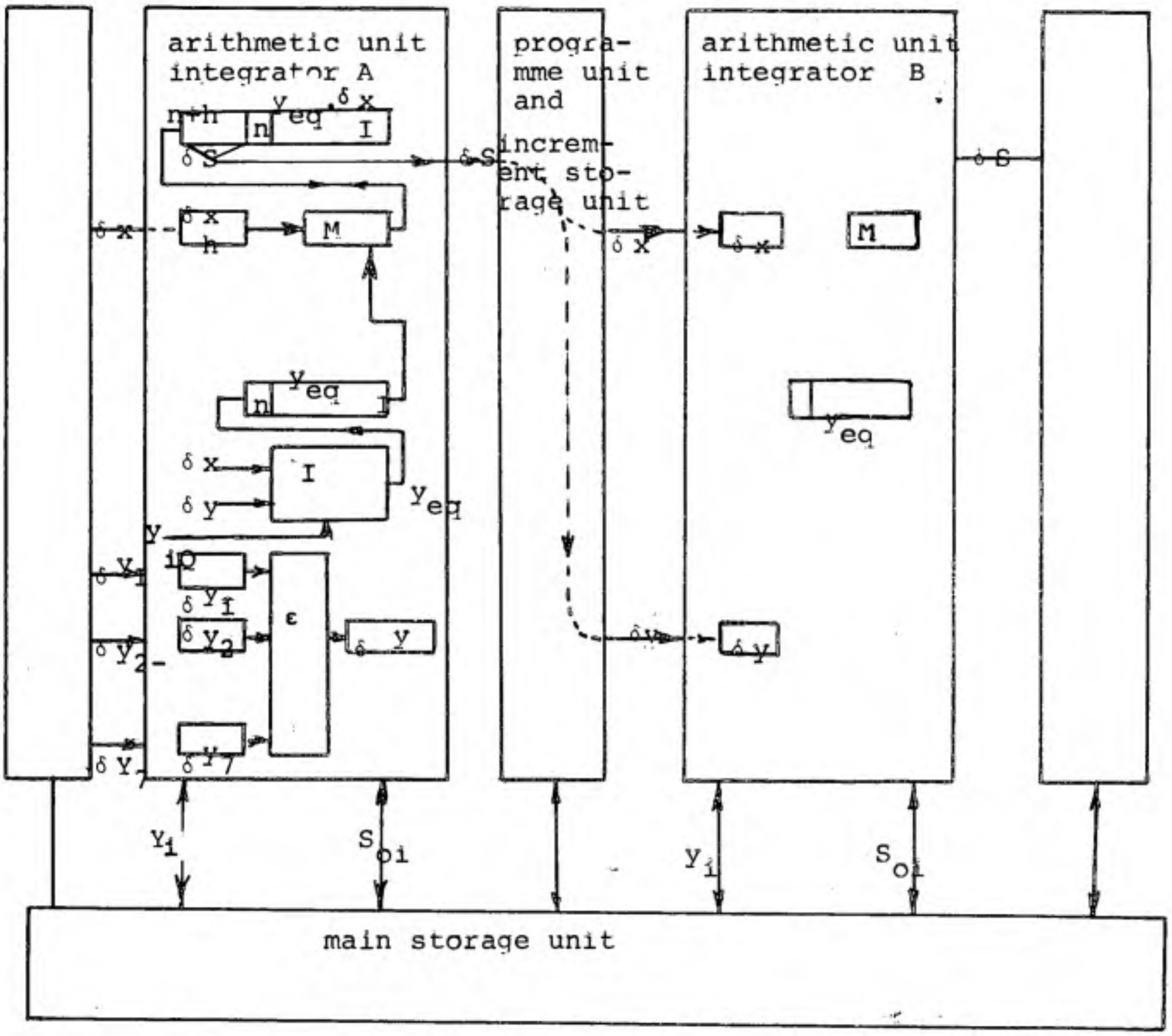

arithmetic unit integrator A
programme unit and increment storage unit
arithmetic unit integrator B
main storage unit
n+h
n
y_{eq}
δx
I
δS
δx
h
M
δx
δy
y_{eq}
y
δy_1
δy_2
δY_7
ε
δy
y_i
S_{oi}
δS

fig 6-7

from programming unit. The value of $\delta_{iQ}y$ is the sum of $\delta_1 y$, $\delta_2 y$, ... $\delta_7 y$, which comes from programming unit, and is added in the adder ε, that result is put in δy register.

We will now explain the operation of arithmetic unit of incremental computer of the Brussel University, which perform the integration on the basis of multiple increment.

The most economical and convenient algorithms of integration for multiple incremental computation as it was seen in chapter 5, are:

$$\begin{cases} \delta_{iQ}\ddot{s} = [y_{iQ} \circ \delta_{iQ}x + \frac{1}{2}\,\delta_{iQ}y \circ \delta_{iQ}x + \\ \qquad\qquad + \frac{1}{12}\,(\delta_{iQ}y \circ \delta_{(i-1)Q}x - \delta_{(i-1)Q}y \circ \delta_{iQ}x)] \\ y_{eq} = y_{iQ} + \frac{1}{2}\,\delta_{iQ}y + \frac{1}{12}\,(\delta_{iQ}y\,\frac{\delta_{(i-1)Q}x}{\delta_{iQ}x} - \delta_{(i-1)Q}y) \\ y_{iQ} = y_{(i-1)Q} + \sum_{j=1}^{7}\,(\delta_{iQ}y)_j \\ \delta x = 2^r \circ \Delta x \\ \delta y = 2^r \circ \Delta y \qquad\qquad h > r > o \\ \delta s = 2^r \circ \Delta s \end{cases}$$

\s the realization of terms $\frac{1}{12}$ $(\delta_{iQ}y \circ \delta_{(i-1)Q}x - \delta_{(i-1)Q}y \circ$ $\circ\ \delta_{iQ}x)$ need too much equipment for the factor $\frac{1}{12}$, so with a good approximation we can replace this term by $\frac{1}{16}$ $(\delta_{iQ}y \circ \delta_{(i-1)Q}x -$ $- \delta_{(i-1)Q}y \circ \delta_{iQ}x)$ which is easy to realize it. Therefore, $\delta_{iQ}s$ will be:

$$\delta_{iQ}s \overset{*}{=} [y_{iQ} \circ \delta_{iQ}x + \frac{1}{2}\delta_{iQ}y \circ \delta_{iQ}x + $$

$$+ \frac{1}{16}(\delta_{iQ}y \circ \delta_{(i-1)Q}x - \delta_{(i-1)Q}y \circ \delta_{iQ}x)]$$

The logic flow of the integration operation is shown in figure (6-8).

The values $(\delta_{iQ}y)_1$, $(\delta_{iQ}y)_2$, $(\delta_{iQ}y)_3$, $(\delta_{iQ}y)_4$, $(\delta_{iQ}y)_5$, $(\delta_{iQ}y)_6$, $(\delta_{iQ}y)_7$, and δx come from the programming unit, and are put in the 4 bit shift registers $\delta_1 y$, $\delta_2 y$,..... $\delta_7 y$. The increments $(\delta_{iQ}y)_1$, $(\delta_{iQ}y)_2$,..... $(\delta_{iQ}y)_7$, are summed by the adders ε_6, ε_7, ε_8, ε_9, ε_{10}, ε_{11} and the results are put in the δy register with 7 bits. The value of y_{iQ} is get from the output of the adder ε_2 and is put in the register B. The values of $y_{iQ} + \frac{1}{2}\delta_{iQ}y -$ $- \frac{1}{16}\delta_{(i-1)Q}y$ is found from the output of the adder ε_1, and is multiplied in multiplier M, by the value of $\delta_{iQ}x$. The output of the multiplier M is added to the value $\frac{1}{16}\delta_{(i-1)Q}x \circ \delta_{iQ}y$, which is the output of the multiplier M_2, and gives at its output the

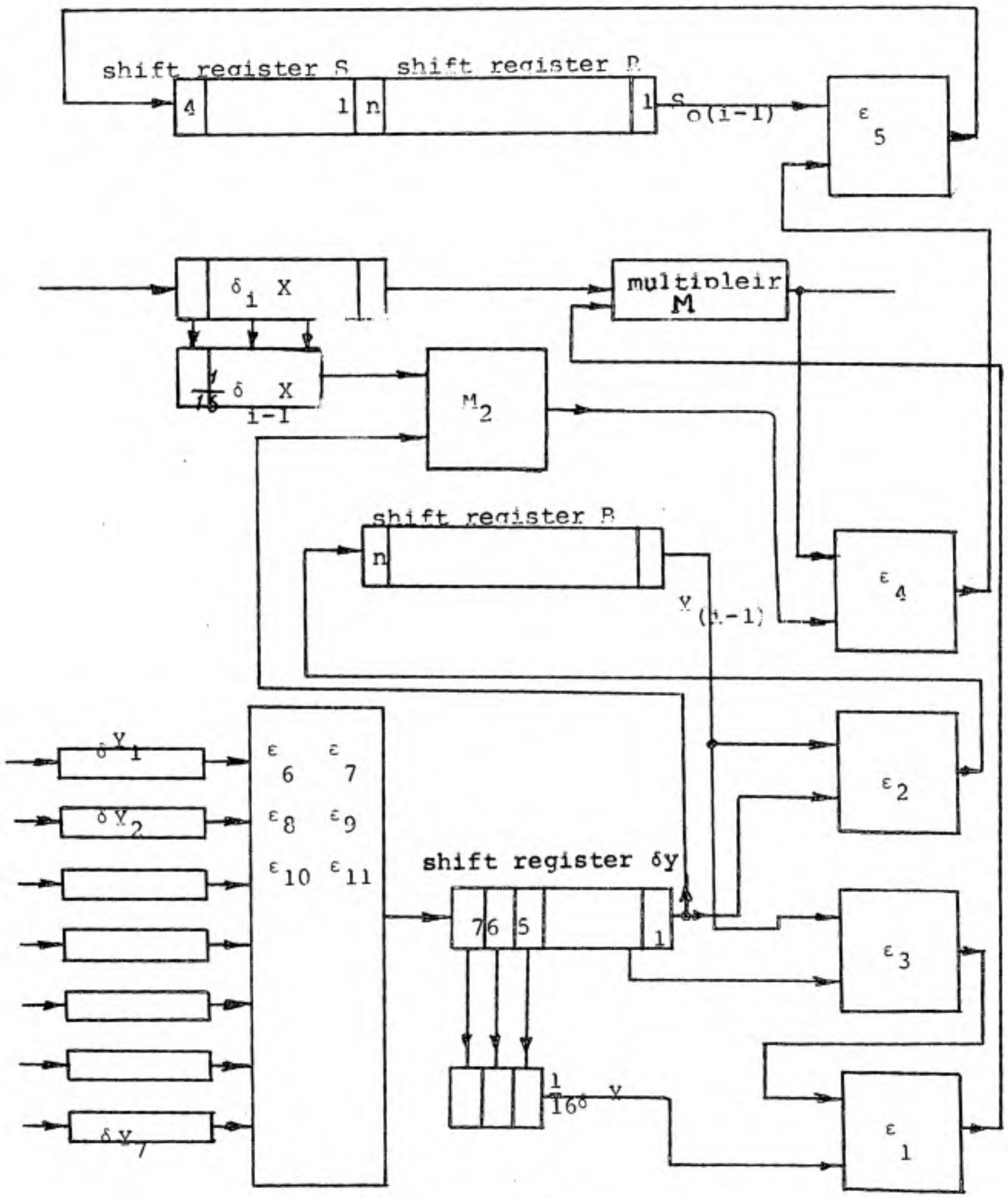

shift register S
shift register R
4
1
n
1
$S_{o(i-1)}$
ε_5
$\delta_i X$
multipleir M
$\frac{1}{16}\delta_{i-1} X$
M_2
shift register R
n
$Y_{(i-1)}$
ε_4
δY_1
δY_2
δY_7
ε_6 ε_7
ε_8 ε_9
ε_{10} ε_{11}
ε_2
shift register δy
7 6 5
1
ε_3
$\frac{1}{16}\delta Y$
ε_1

Fig 6-8

$$y_{iQ} \, \delta_{iQ}x + \frac{1}{2}\delta_{iQ}x \circ \delta_{iQ}y + \frac{1}{16} (\delta_{iQ}y \circ \delta_{(i-1)Q}x -$$

$- \delta_{(i-1)Q}y \circ \delta_{iQ}x)$. This sum is entered in the adder ϵ_5 and is summed with the rest of integral $s_{o(i-1)}$ of the preceeding iteration (i-1). The output of the adder ϵ_5 is:

$$[y_{iQ} \circ \delta_{iQ}x + \frac{1}{2}\delta_{iQ}y \circ \delta_{iQ}x + \frac{1}{16} (\delta_{iQ}y \circ \delta_{(i-1)Q}x -$$

$$- \delta_{(i-1)Q}y \circ \delta_{iQ}x] + s_{o(i-1)}.$$

As the value of y is put in 10 bits B register, and the value of $\delta_{iQ}x$ in four bits δx register, the value of $y_{iQ} \circ \delta_{iQ}x$ and the above bracket will have maximum 14 bits. This value is put in the 14 bits register (S,R), (10 bits of register R and 4 bits of register S). Four most significant bits of the register (S,R), are taken as the output $\delta_i s^{*}_{QM}$, which is the rounded off value of $\delta_i s^{*}_{Q}$. The increment of integration $\delta_i s^{*}_{QM}$ is sent to the increment memory in programming unit. The less significant bit (register R) is the new rest of the integral s_{oi}, which is sent to the memory with the value of y_{iQ} for the next operation.

6,2.1.4. New system of multiplication of two functions.

The multiplication of two functions x and y was achieved normally in a differential form.

$$d\,(x \cdot y) = x \cdot dy + y \cdot dx \qquad (1)$$

Thus, a minimum of two integrators and an adder (fig. 6-9) was required.

The exact value of $\Delta(x \cdot y)$ is:

$$(x_i + \Delta_i x)\ (y_i + \Delta_i y) = x_i y_i + x_i \cdot \Delta_i y + y_i \cdot \Delta_i x + \Delta_i x \cdot \Delta_i y \qquad (2)$$

$$\Delta(x_i \cdot y_i) = x_i \cdot \Delta_i x + y_i \cdot \Delta_i x + \Delta_i y \cdot \Delta_i x \qquad (3)$$

Normallv the third term in equation (3) was neglected and the equation (1) was used with three integrators, with the error of $\Delta_i x$, $\Delta_i y$.

But a new method i- just calculating the sum of the three terms in the equation (3) is:

$$\Delta(x \cdot y) = x_i \cdot \Delta y + y_{i+1} \cdot \Delta x \qquad (4)$$

Therefore, we can calculate the equation (4) in arithmetic unit, and transmitting the increment of function $\Delta(x \cdot y)$ as it was done in the integration, with the ternary communication system. In this system we can build the multiplier block in incremental computer, with the inputs Δx, Δy and the output $\Delta(x \cdot y)$, exactly with the communication system of integrators, and using one block instead of three integrators, with higher accuracy.

The logic flow of this operation is shown in fig. (6-I0)

There are five multiplier of function in the incremental computer of the industrial electronic laboratory of the Brussel University, which are designed by the author. The transmission of informations in input and output of the multiplier is done by only one cable on the patch pannel, as it was the case of the integration.

6.2.1.5. Summation of the functions or digital servo.

Though a separate accumulator or the Y accumulator of an integrator can accept two or more incremental inputs, and accumulate their sum, it cannot generate a rate equal to the sum of the input rates. The addition of rates may be performed by using an integrator programmed to function as a digital servo.

Considering first how a digital servo can produce an incremental output equal to the sum of two inputs.

Since its operation has been governed by the number system used, the description will be related specifically to the number system usually employed in the integrators with bineary communications.

This number system, shown in figure (6-I1), is described as

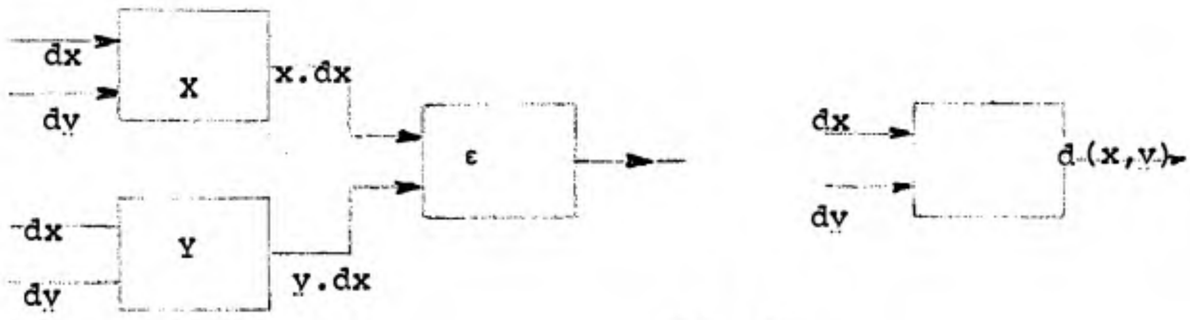


Fig 6-9

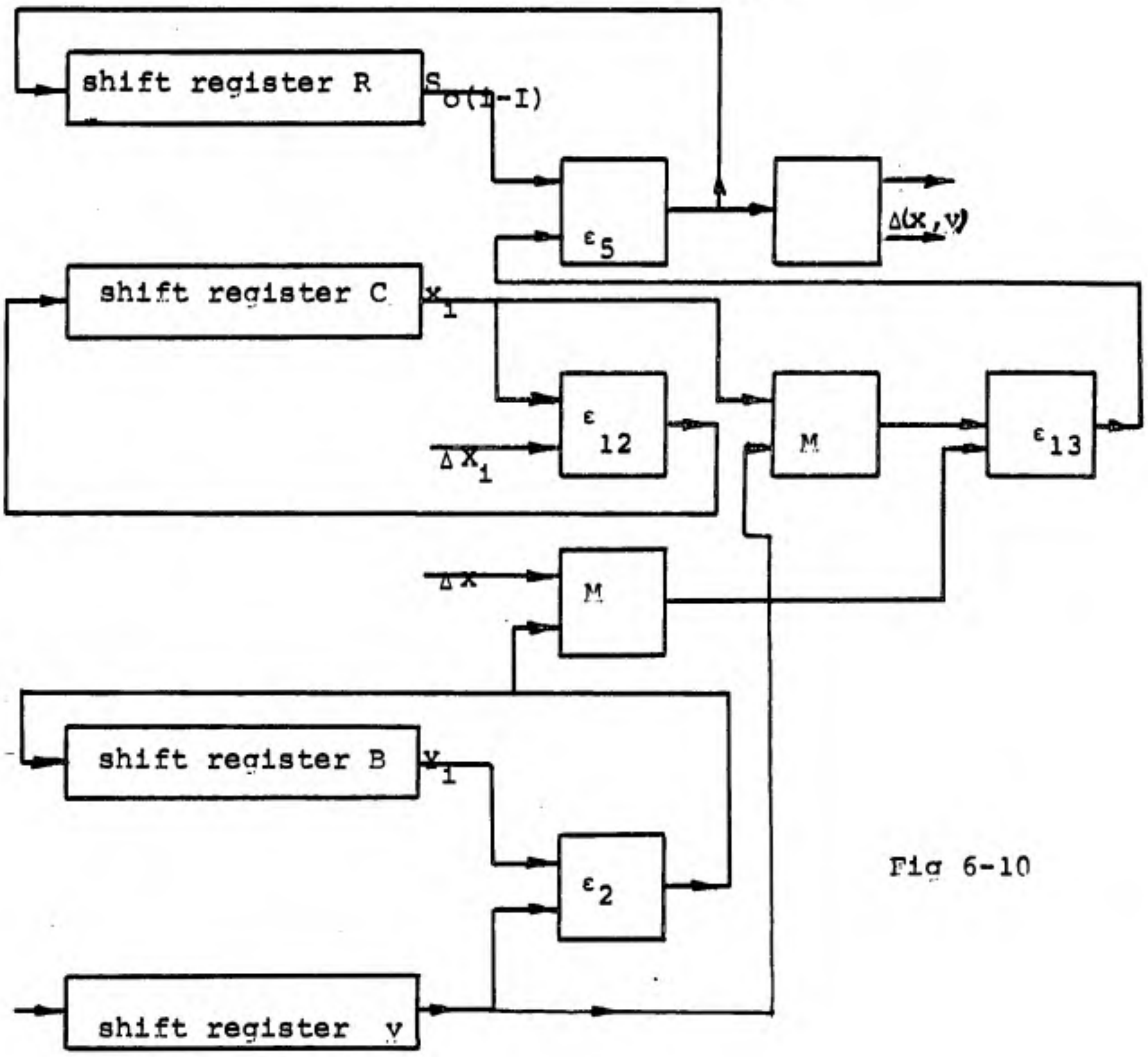
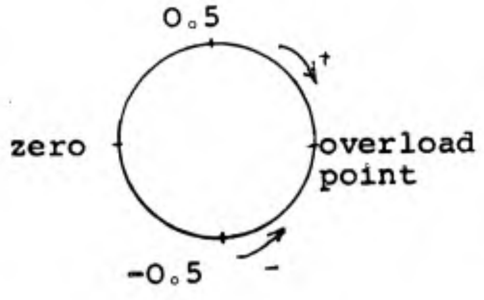


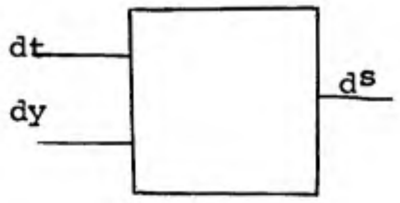
Fig 6-10

circular, because, when an increment is added to the representation of the maximum positive number, $1 - 2^{-n}$, the result is the representation of the maximum negative number -1, and conversely, substracting a single increment from -1, produces $1 - 2^{-n}$.

In servo integrator we put the maximum positive number of $1 - 2^{-n}$ in Y register of the integrator, and the independent variable is time. Under these conditions, the integrator will produce a 1 at its output during each iteration period. The sequence of 1's, thus generated, represents the the maximum positive output rate of the machine. If a single bit is added in the least significant bit, the value of the integral will change from $1 - 2^{-n}$ to -1 and the integrator will then generate a sequence of zeros, the maximum negative rate.

The initial value of y register is put to $1 - 2^{-n}$ (1.1111...1). If the output ds is feedback with the negative signe to the input dy, then by applying the quantities e in the other input of dy, the first output signal will be a positive rate (+1), and the value of y register becomes $1 - 2^{-n} + e$. When fed back, during the next iteration period it change from $1 - 2^{-n} + e$ to $1 - 2^{-n} + e - 1$. As a result the next output of the integrator will be a negative increment, which feeds back with opposite signe converting the value of integrand to $1 - 2^{-n} + e$, but as this value is a negative number, it gives at the output a negative increment, and





Binary number	Decimal equivalent
1.111	7/8
1.110	6/8
1.101	5/8
1.100	4/8
1.011	3/8
1.010	2/8
1.001	1/8
1.000	0
0.111	-1/8
0.110	-2/8
0.101	-3/8
0.100	-4/8
0.011	-5/8
0.010	-6/8
0.001	-7/8
0.000	-1

Fig 6-11

feeds back by (+1), so the new value of integrand will be $1 - 2^{-n} + e + 1$. This process continues until the value of Y register becomes $1 - 2^{-n}$, as its initial value, and the number of pulses at the output of the system becomes to e.

This system has the application when we want to send some pulses equal to a number e. The servo system can be used in feedback control systems, for generating a number of positive or negative pulses, depending on the application.

There are five servo systems with seven dy inputs , four outputs + Δs and $-\Delta S$ in the incremental computer of the industrial electronic laboratory. The systems of communication are bineary systems, and the interconnections between them and the integrators are done by one cable on the patch pannel.

6.2.1.6. Function generator.

In analogue computation, it is possible to generate any function with the linear method of interpolation, by the diode resistance function generator.

The variation of resistance R and the number of diode in the circuit can generate any function by the linear interpolation formula.

The same operation can be done by the increment principale by the variation of numerical value W and a (fig.6-13)

In this aim, if we use an integrator as a multiplier, the content of Y register of this integrator is a which comes from the memory. The value of W is put in the W register of the memory. The output dz of the integrator is:

$$dz = a \cdot dx$$

or

$$z = \sum dz$$

$$= a \cdot \sum dx = a \cdot x$$

The integrator continues to perform the integration operation for each dx input; until the number of dx input pulses which are count by the counter X, becomes equal to W. At this moment, the comparator gives the signal of the end of operation. The reversible counter (C) count the dz pulses, so its content will be $Z = a \cdot W$ at that instant Fig. (6 - 14)

By choosing the value of a in the Y register of integrator and w in the W register, it is possible to generate any linear curve with any slope (a) and bias (W).

So, by generating the curve OA, AB, BC,.... one after the other, we can produce the curve o A B C D ..... M N.Fig (6-13

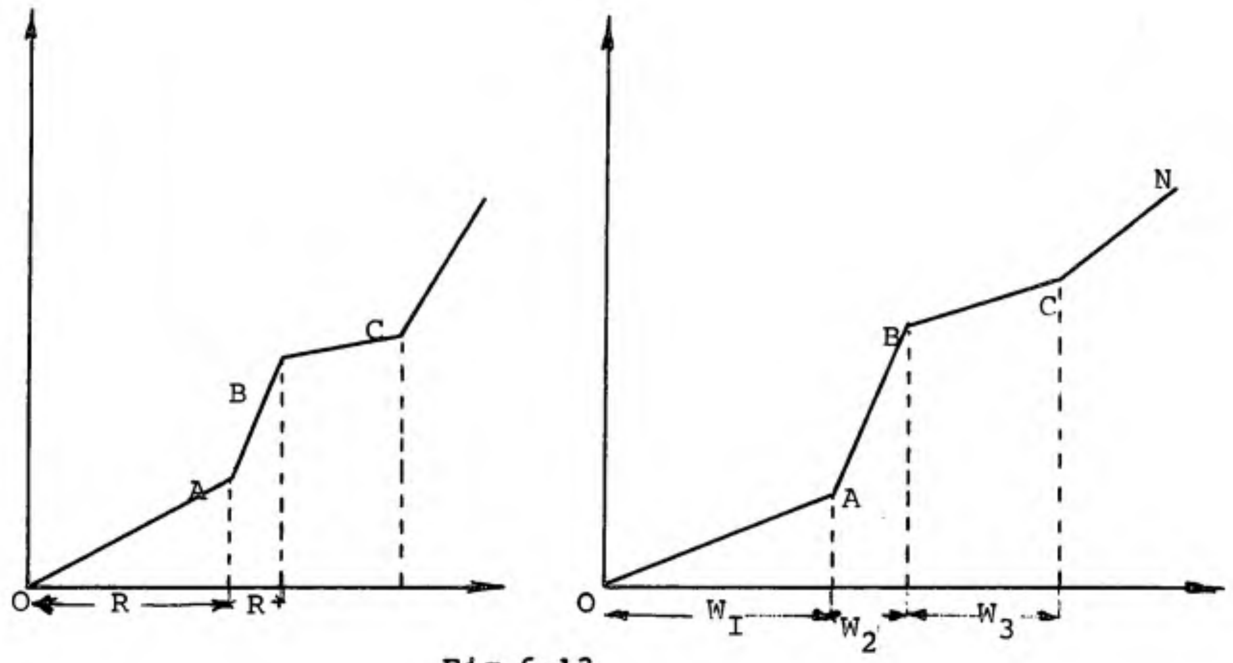


Fig 6-13

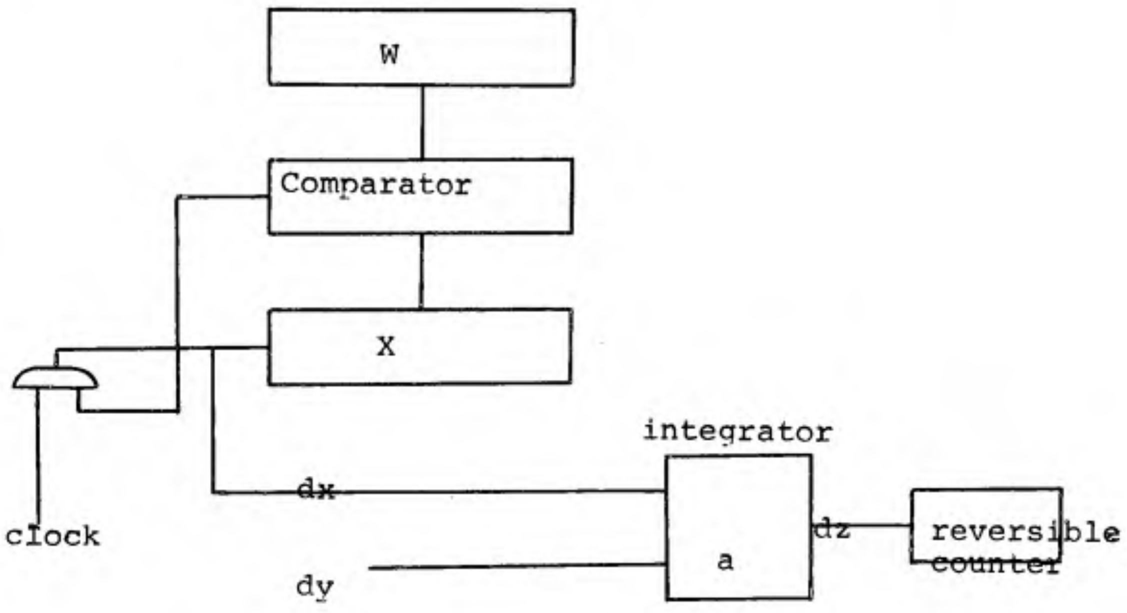


Fig 6-I4)

6.2.2. Numerical operations.

The additions and substractions are done b one bits adders and the registers A, B, R, S,.

6.2.2.1. Addition and substraction of two numbers A, B, (B): = (B) ± (A).

For this operation the shift registers A, B and the adder ε_2 are used. The results of operation are put in register B, as it is shown in figure (6 - 5)

6.2.2.2. Addition and substraction of two 20 bits number, (S,R): = (S,R) ± (A,B).

The addition and substraction of two numbers (S,R) and (A,B) are done in adder ε_5, and the results are put in register (S,R), as it is shown in figure (6 - 6)

6.2.2.3. Addition and substraction of three numbers A, B and R, (R): = (R) ± (A ± B).

The addition and substraction of three numbers R, A, B,

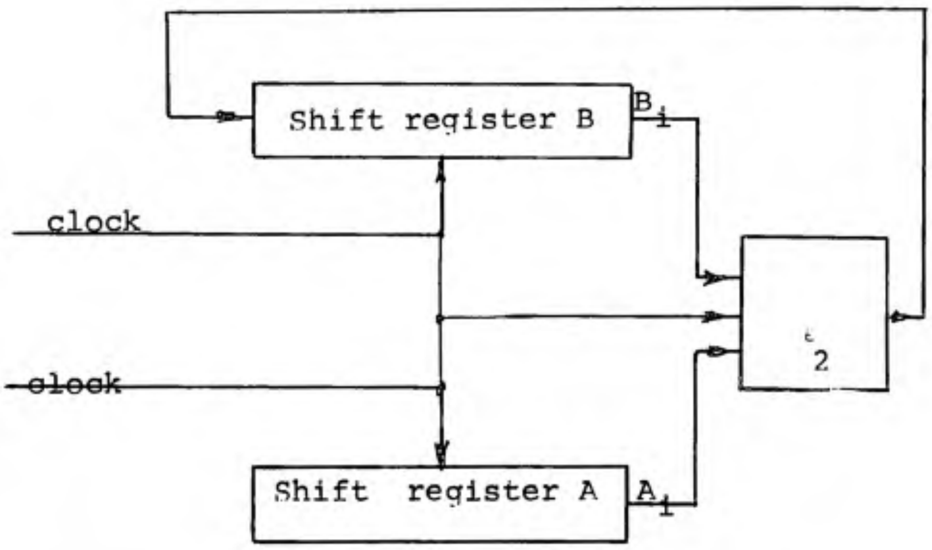
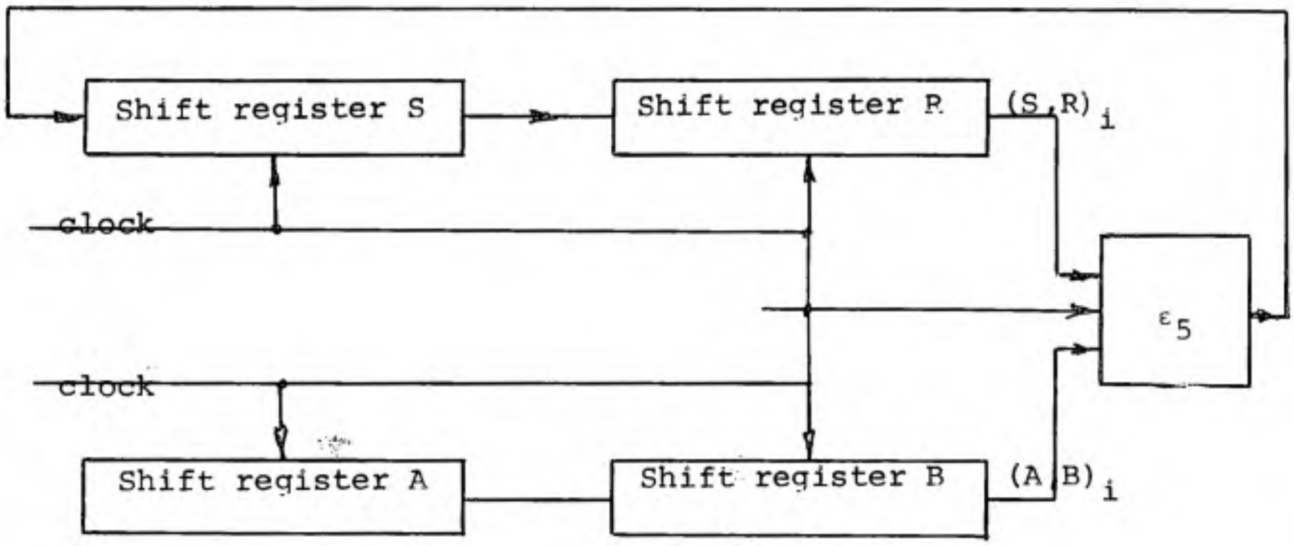


Fig (6- I5)



Fig(6-I6)

are done by the registers A, B, R and the adders ε_2, ε_5. The sum of these three numbers are done in one time of addition and the results are put in register R, as it is shown in figure (6-17)

6 2.2.4. Multiplication of two numbers in one time of addition. (S,R): = (B)·(C) or (S,R): = (B ± A)·(C).

The multiplication of two numbers B and C or the algebraic sum of (B ± A) to (C) is done with a special algorithm which is a kind of serial and parallel multiplication. The time of multiplication is equal to one time of addition.

If $C_1, C_2, \ldots\ldots, C_{10}$ are the outputs of C register bits, then the multiplication algorithm is:

$$(B)\cdot(C) = B\cdot 2^9\cdot C_{10} + B\cdot 2^8\cdot C_9 + B\cdot 2^7\cdot C_8 + \ldots\ldots + B\cdot C_1$$

The logic flow is shown in figure (6)

The multiplier M has 9 adders and one 1 bit memory for 10 bits C register. The output of one bit memory is $2^9\cdot C_{10}\cdot B_i$, the output of ε_9 is $2^8\cdot C_9\cdot B_i$, The output of ε_8 is $2^7\cdot C_8\cdot B_i$ and so on. Therefore, the output of each adder gives one term of above algorithm.

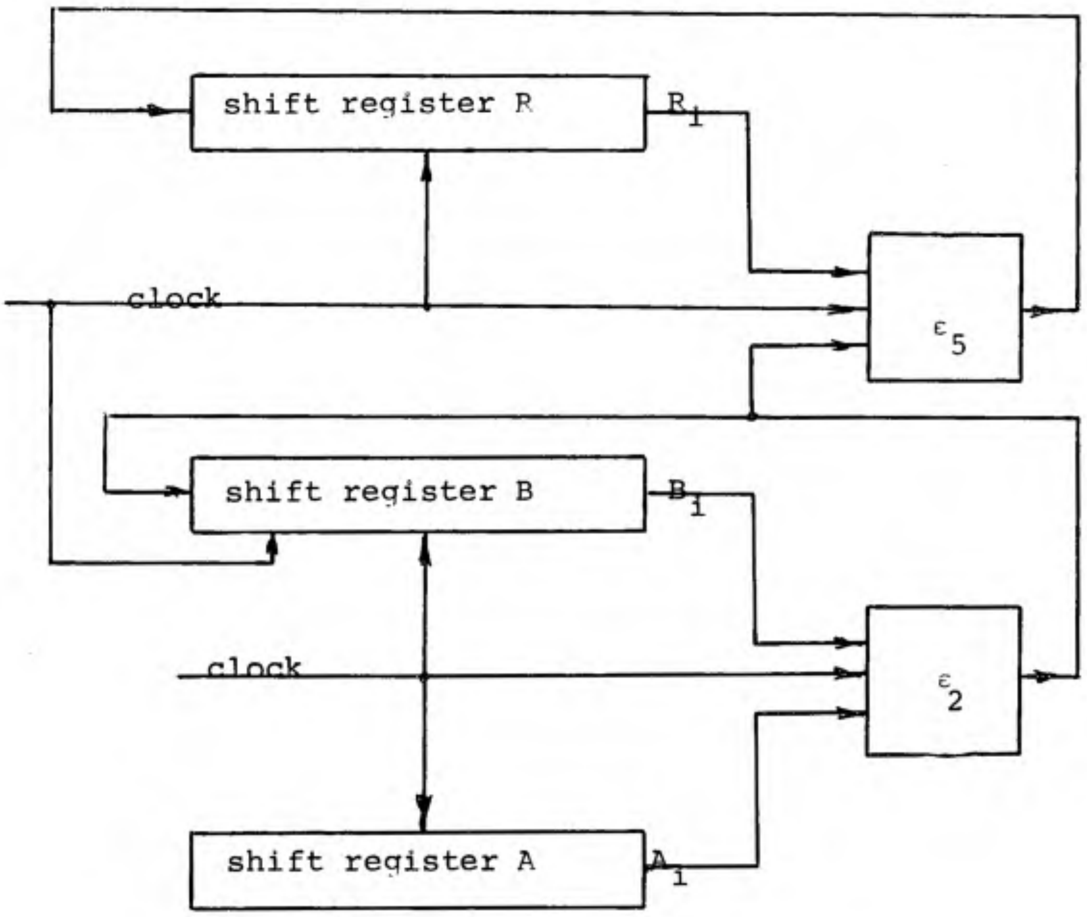

shift register R
R_i
clock
ε_5
shift register B
B_i
clock
ε_2
shift register A
A_i

Fig (6-I7)

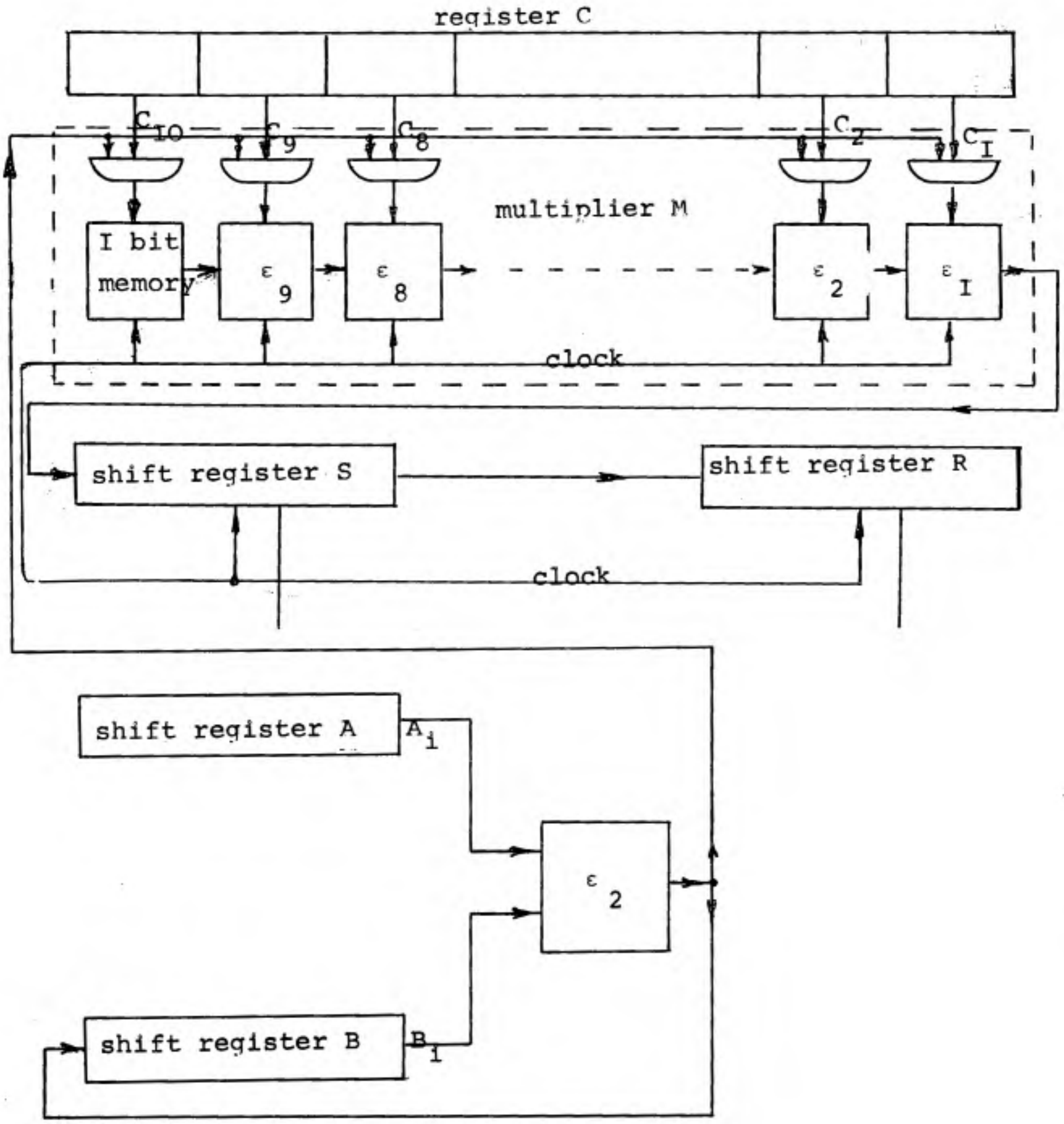

register C
C_{10}
C_9
C_8
C_2
C_1
multiplier M
I bit memory
ε_9
ε_8
ε_2
ε_1
clock
shift register S
shift register R
clock
shift register A
A_i
ε_2
shift register B
B_i

Fig (6-I8)

As it is seen, the value of B_i or $(B_{i+1}) := (B_i) \pm (A_i)$ arrives in serial form at the input of multiplier, and in the multiplier this value is multiplied by the value of C register in the parallel form. The output of multiplier is the output of adder ε_1 which is stored in the register (S, R).

The first advantage of this multiplier is that, it does the multiplication in one time of addition, and its input B_i can be the output of any adder or logical function. For instance here, the output of ε_2 is applied for the multiplication of $B_i \cdot C$ or $(B_i \pm A_i) \cdot (C)$. In the multiple incremental computation, the output of adder $\sum$ is applied to its input for the operation:

$$(y_i + \frac{1}{2}\delta_i y - \frac{1}{16}\delta_{i-1} y) \cdot \delta_i x$$

The other advantage is that, the amount of equipment depends on the number of bits in C register. For instance, if we want to multiply the four bits of C register to ten bits of B register, then we need only 3 adders ε_1, ε_2 and ε_3 instead of 9 adders.

6.2.2.5. Division.

If A is dividand, D is the divisor, q is the quotient and r is the rest of the division. Then, the division equation is:

$$\frac{A}{D} = q + \frac{r}{D}$$

In order to do the division of the number A over D, the number A should be put in the register (S,R), the number D in the register A.

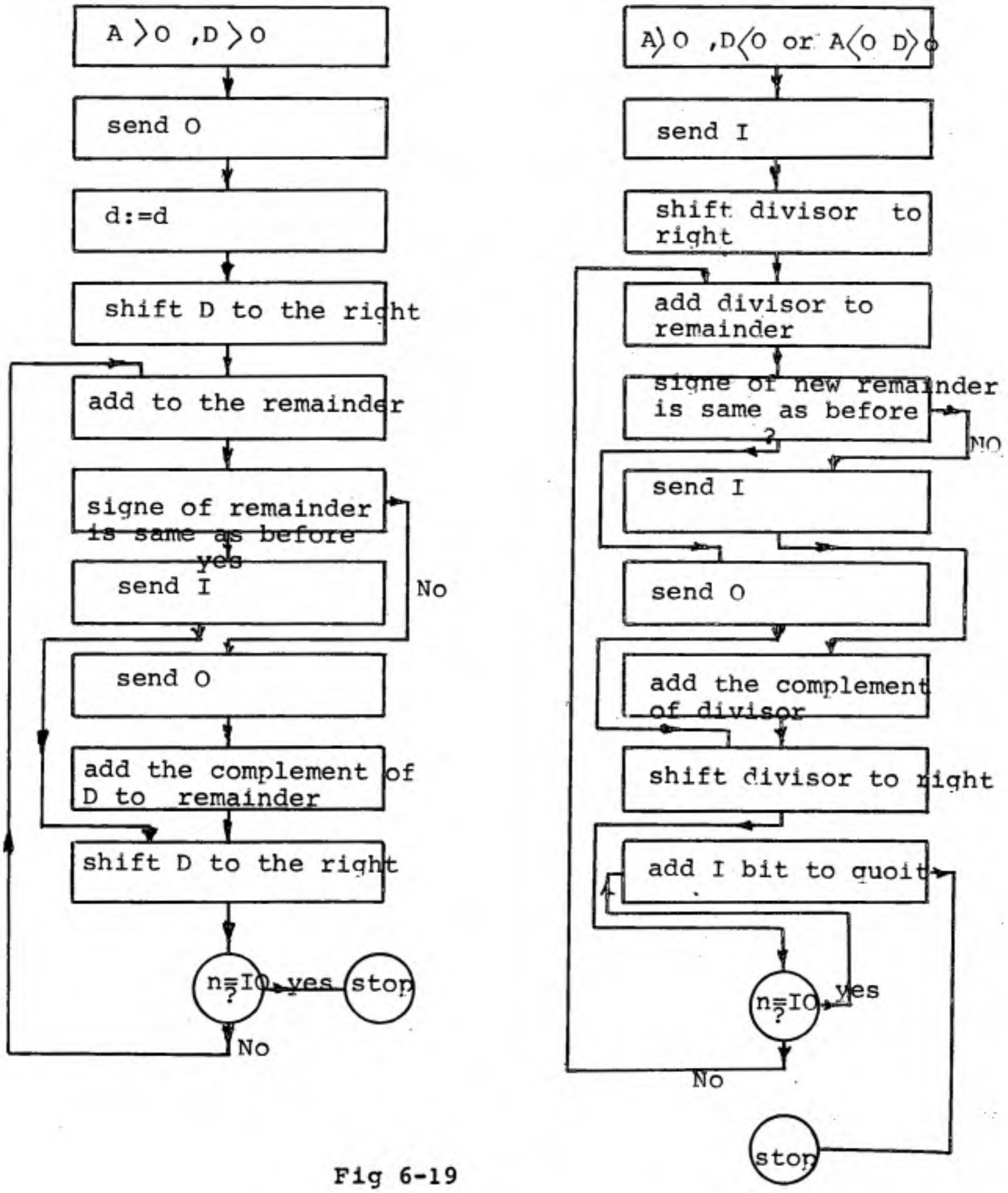
After the division operation, the quotient q comes in the register C, and the rest of integral in the register (S,R). The algorithm of the operation is shown in the flow diagram(6-19)

When $\begin{cases} A > o \text{ or } A < o, \\ D < o \quad D > o \end{cases}$ we should add one bit to the result

quotient q Therefore, another adder ε_{13} is necessary. The control unit gives all the control signal and clock pulses which are necessary for the division operation.

6 2.3. Logical and other auxilary operations.

Four 10 bits shift registers A, B, S, and R or two 20 bits shift registers (A, B) and (S, R), can be used independently, as the shift registers for any purpose of calculation, as they are



A > O ,D > O
send O
d:=d
shift D to the right
add to the remainder
signe of remainder is same as before
yes
send I
No
send O
add the complement of D to remainder
shift D to the right
n=IO ?
yes
stop
No
A > O ,D < O or A < O D > O
send I
shift divisor to right
add divisor to remainder
signe of new remainder is same as before ?
NO
send I
send O
add the complement of divisor
shift divisor to right
add I bit to quoit
n=IO ?
yes
No
stop

Fig 6-19

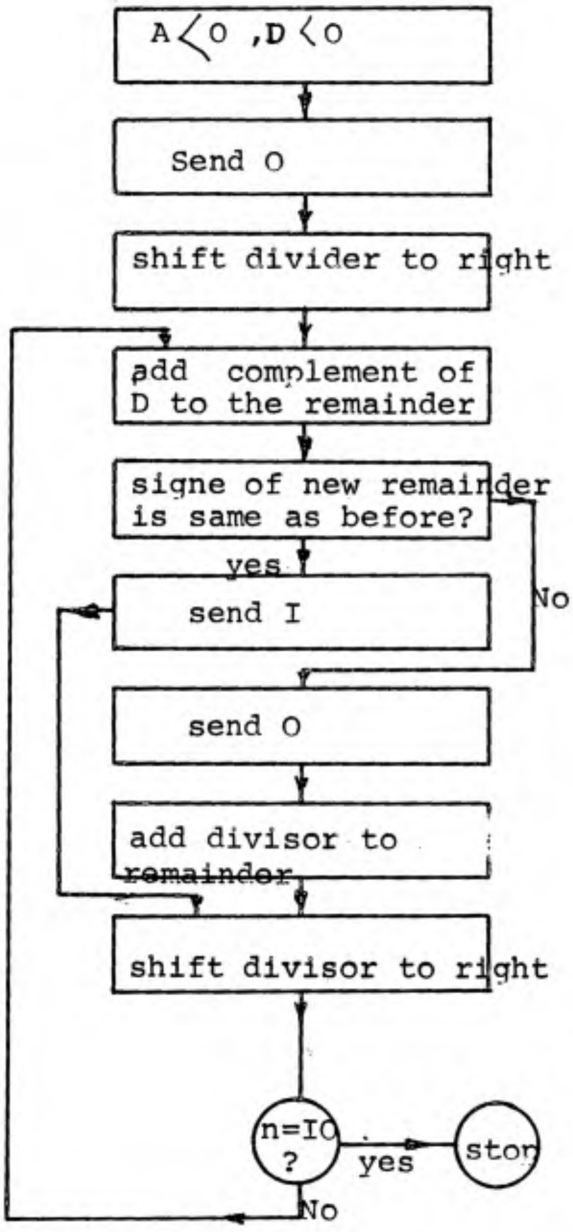

A < O , D < O
Send O
shift divider to right
add complement of D to the remainder
signe of new remainder is same as before?
yes
No
send I
send O
add divisor to remainder
shift divisor to right
n=IO ?
yes
No
stop

Fig 6-19

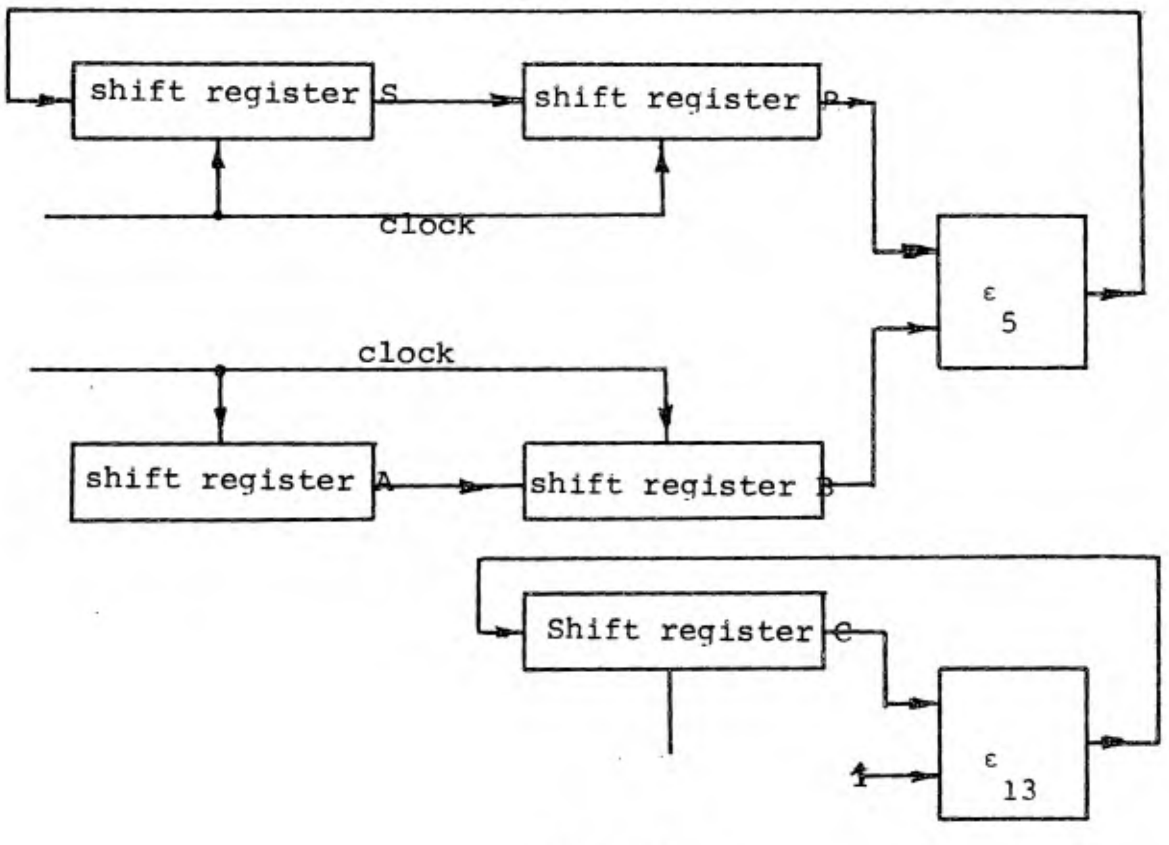

shift register
S
shift register
P
clock
ε
5
clock
shift register
A
shift register
B
Shift register
C
1
ε
13

Fig 6-19

shown in figure (6-20).

There is a reversible counter which is called register D. The flipflops which are used in this counter are the T type flipflops. The input equations of each bit are:

$$Y_1 = C + D$$

$$Y_2 = Y_1 \circ C + Y'_1 \circ D$$

$$Y_3 = Y_1 \circ Y_2 \circ C + Y'_1 \circ Y'_2 \circ D$$

$$Y_4 = Y_1 \circ Y_2 \circ Y_3 \circ C + Y'_1 \circ Y'_2 \circ Y'_3 \circ D$$

$$Y_{10} = Y_1 \circ Y_2 \circ Y_3 \circ Y_4 \circ Y_5 \circ Y_6 \circ Y_7 \circ Y_8 \circ Y_9 \circ C$$
$$+ Y'_1 \circ Y'_2 \circ Y'_3 \circ Y'_4 \circ Y'_5 \circ Y'_6 \circ Y'_7 \circ Y'_8 \circ Y'_9 \circ D$$

$$Y_{11} = Y_1 \circ Y_2 \circ Y_3 \circ Y_4 \circ Y_5 \circ Y_6 \circ Y_7 \circ Y_8 \circ Y_9 \circ Y_{10} \circ C +$$
$$+ Y'_1 \circ Y'_2 \circ Y'_3 \circ Y'_4 \circ Y'_5 \circ Y'_6 \circ Y'_7 \circ Y'_8 \circ Y'_9 \circ Y'_{10} \circ D$$

There is a comperator which indicate whether two numbers C and D of ten bits are equal or not. The logical equation is:

$$E_i = C_i \circ D_i \circ E_{i+1} + C'_i \circ D'_i \circ E_{i+1}$$

There is also a simulator for various logical problem.

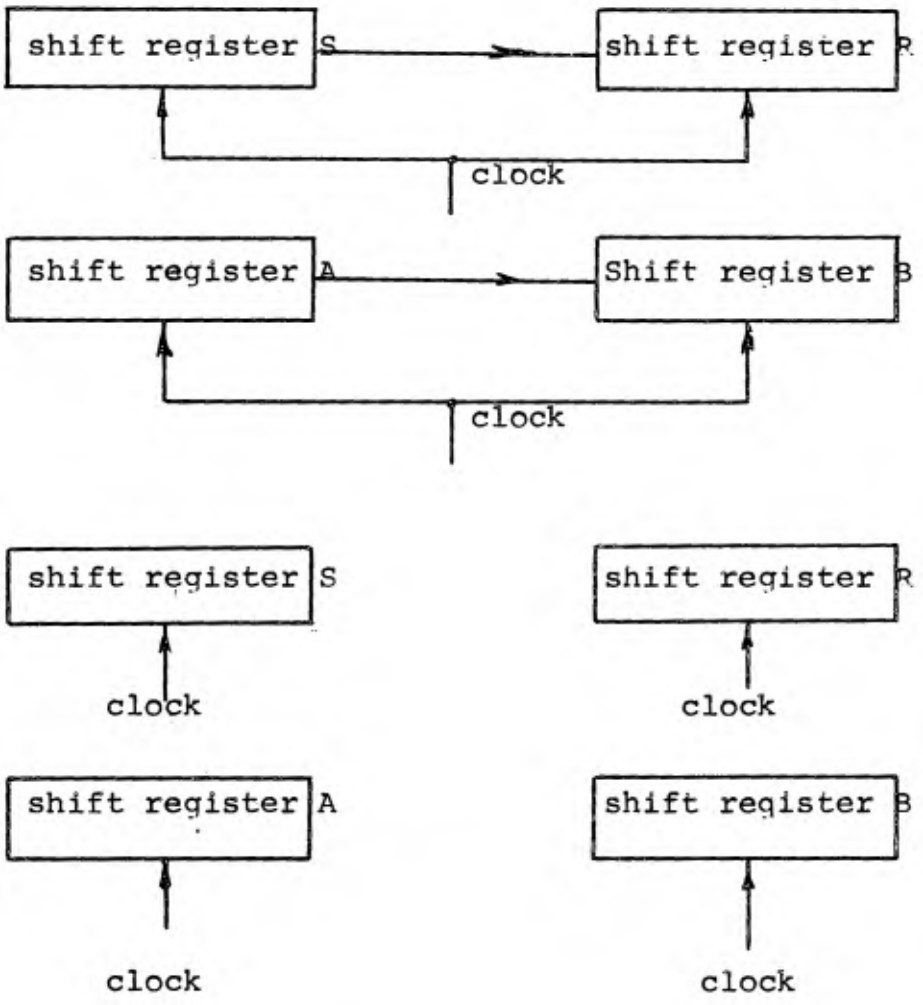

shift register S
shift register R
clock
shift register A
Shift register B
clock
shift register S
shift register R
clock
clock
shift register A
shift register B
clock
clock

Fig 6-20)

6 3. A new concept for programming unit in incremental computer.

In a serial unitary incremental computer, the programme of a problem is generally stored in the memory, like in the case of serial D.D.A.

In parallel D.D.A., a patch panel used to interconnect the integrators by two leads for one bit unitary increment plus signe.

The new programming unit, use a patch panel for serial and parallel machines, which need only one lead for transmitting the information between the integrators.

The main advantage of this system is that, there is no need of storing the programme instructions in the memory, which need too much memory storage and operation time. Therefore, the programme of a problem is realized on the patch panel, exactly as an analog computer, with only one lead between the positive or negative output, and the dx or dy input of the integrators.

This new system of transmitting the information between the integrators, can be used either for unitary or for multiple increment computation.

The programming unit of the serial incremental computer, which is designed by the author in the industrial electronic laboratory of Brussel university, has two parts.

The former, built on two pannels, offers the possibility of interconnecting sixty integrators working with the unitary increment principle.

The latter, permits of interconnecting ten integrators working with the multiple increment principle. On the same pannel, there are five functions adder and functions multiplier.

On the patch pannel, each integrator has a block of sixteen sockets. There are one dx input, seven dy inputs, four positive ds and four negative ds outputs. The inputs and outputs of each integrator are shown in figure (6-21)

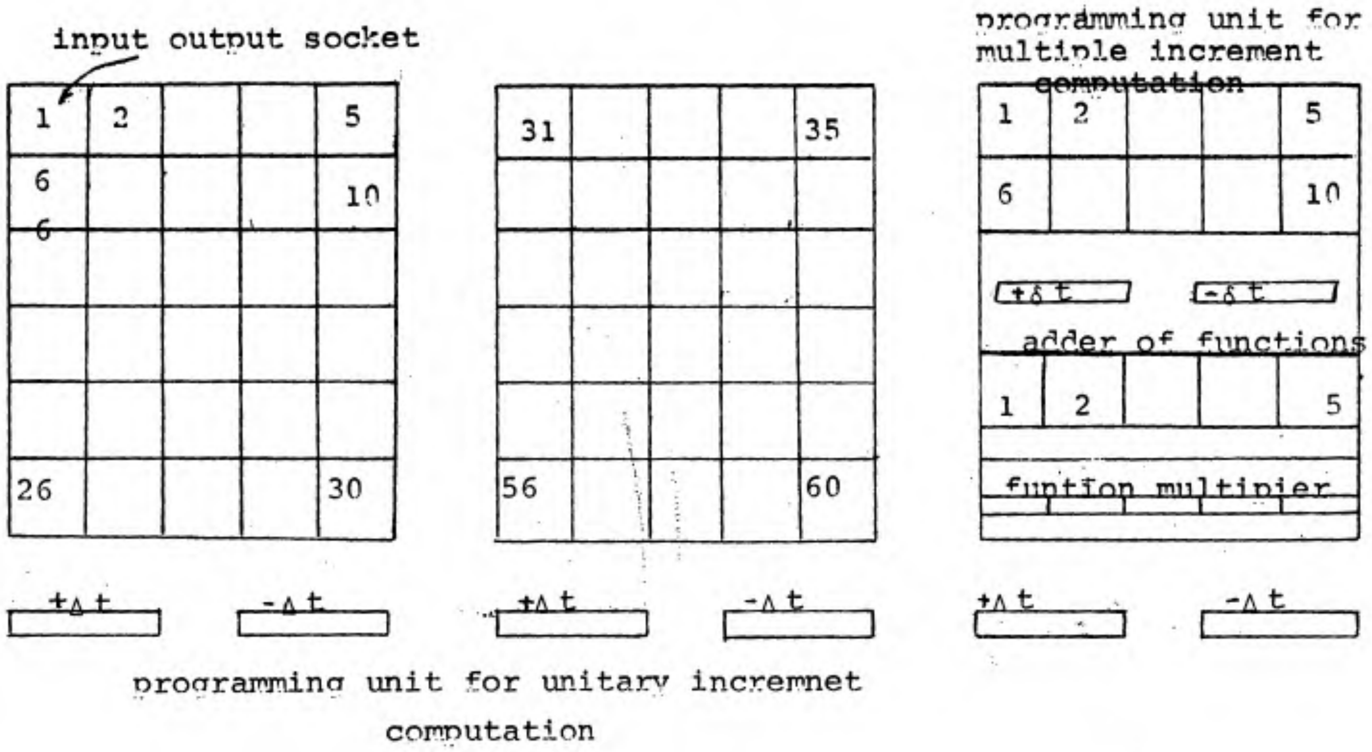
The algorithms of the system are as following:

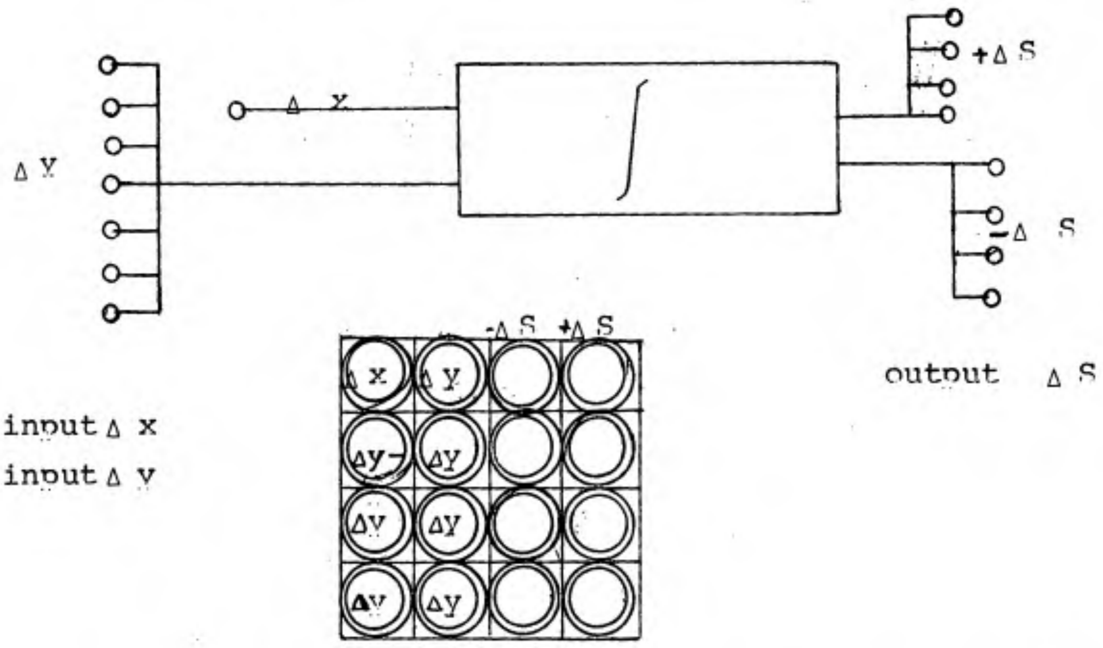
The sixty integrators are operated one after another, by the sixty signals A_1 till A_{60} which arrive from the selector of the control unit.

When the control unit gives the signal A_j, the following operations take place:

a) The informations of the integrator j, which were stored in the memory, are transfered to the arithmetic unit.

b) The programme unit takes the input informations δx, δy_1, δy_2, ... δy_7 from the output of integrators or other devices.





input output in each integrator in programmin unit

Fig 6- 21)

c) The arithmetic unit performs the integration operation and gives the output increment of integral δs.

d) The increment δs is stored in the increment memory, which in unitary increment computations is 1 bit memory and in multiple increment computations is h bit memory.

e) The new values of function y_{iQ} and the rest of integral s_{io} are stored in the memory of the computer.

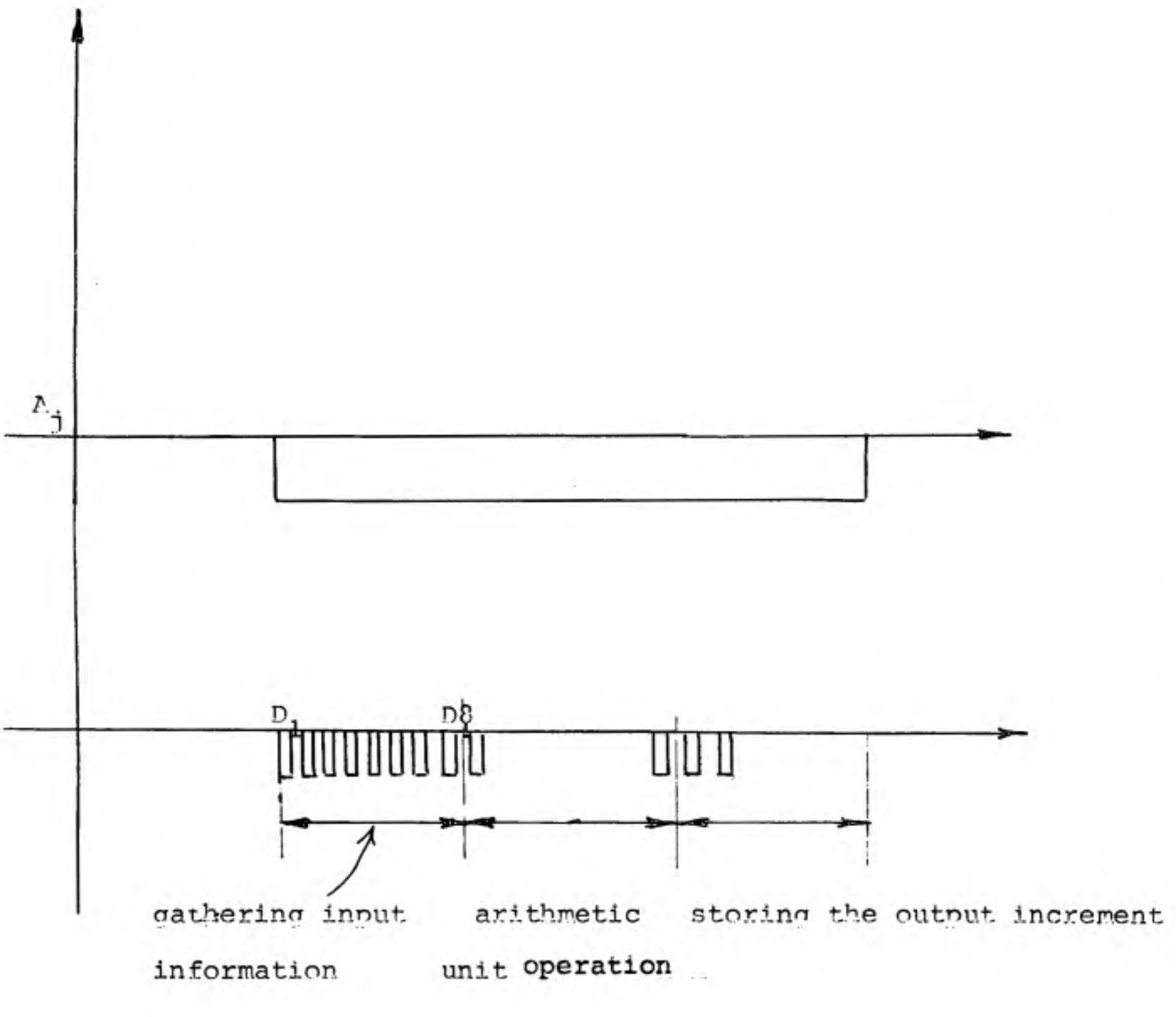
The task of programming unit is first, taking the input information δx and $\sum_{j=1}^{7} (\delta y)_J$ from the output of different integrators and sending it to the arithmetic unit. The timing diagram of the arithmetic unit and programming unit is shown in figure (6-22).

The operation of transmitting the information δx, δv_1, δy_2,.... ... δy_7 to the input of integrator j at A_j is done as following:

Let's suppose, the three inputs δx, δy_4. δy_7, of integration j are connected to the output of integrator i, p, Q, on the patch pannel, fig (6-23).

At the integrator period A_j, the integrator j asks the information to its input by sending eight pulses $A_j D_1$, $A_j D_2$, $A_j D_3$, $A_j D_4$, $A_j D_5$, $A_j D_6$, $A_j D_7$, $A_j D_8$ from its inputs δx, $\delta_1 y$, $\delta_2 y$, $\delta_3 y$, $\delta_4 y$, $\delta_5 y$, $\delta_6 y$, $\delta_7 y$,

In our example, the integrator j sends the $A_j D_1$ signal from its input δx by the external lead on the patch pannel. This signal



A_j
D_1
D_8
gathering input information
arithmetic unit operation
storing the output increment

Fig 6-22

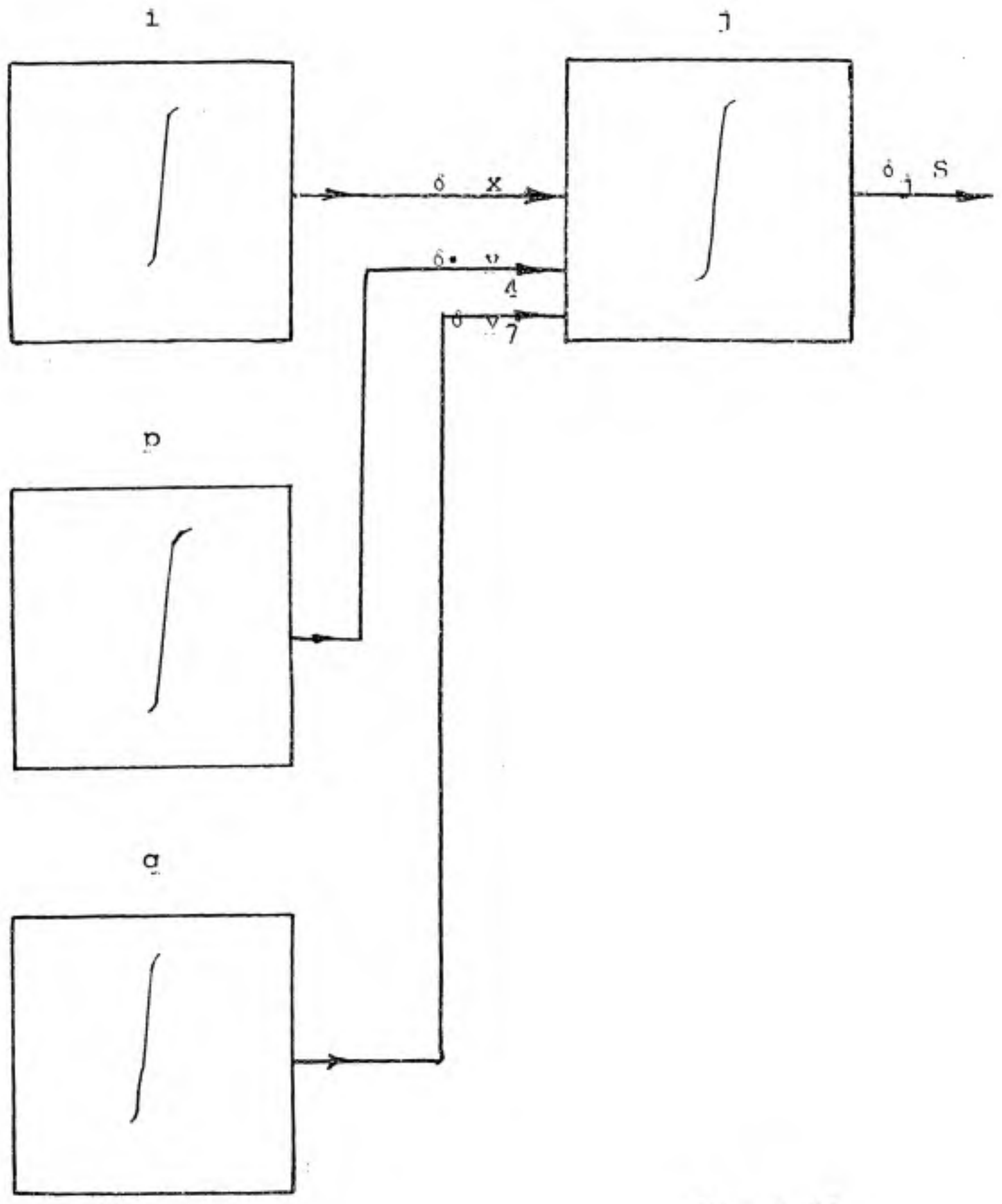


Fig 6-23

A_j D_1 pass via an O R circuit ($+\Delta s_i$) and gives the order of transfering the value of $\delta_i s$ to the arithmetic unit,fig (6-24).

The signals A_j D_4, and A_j D_7 which are send by the input δy_3 and δy_7 of the integrator j, transfer the increment $\delta_p s$ and $\delta_q s$ to the arithmetic unit.

The internal connection of input output socket of each integrator j, with the increment memory and the arithmetic unit, is shown in figure (6-25)

By this system, when the integrator j begins to operate by sending 8 pulses A_j D_1 to A_j D_8, via the leads on the patch pannel, it gathers its input information in the arithmetic unit. Then the integral operation is done in the arithmetic unit, at the end, the output δs is send to the memory incremental storage $\delta_j s$ by the signal A_j D_{24}.

This system has the great adventages of transmitting the informations between the integrators by only one lead on the patch pannel, for the unitary and multiple increment computation.

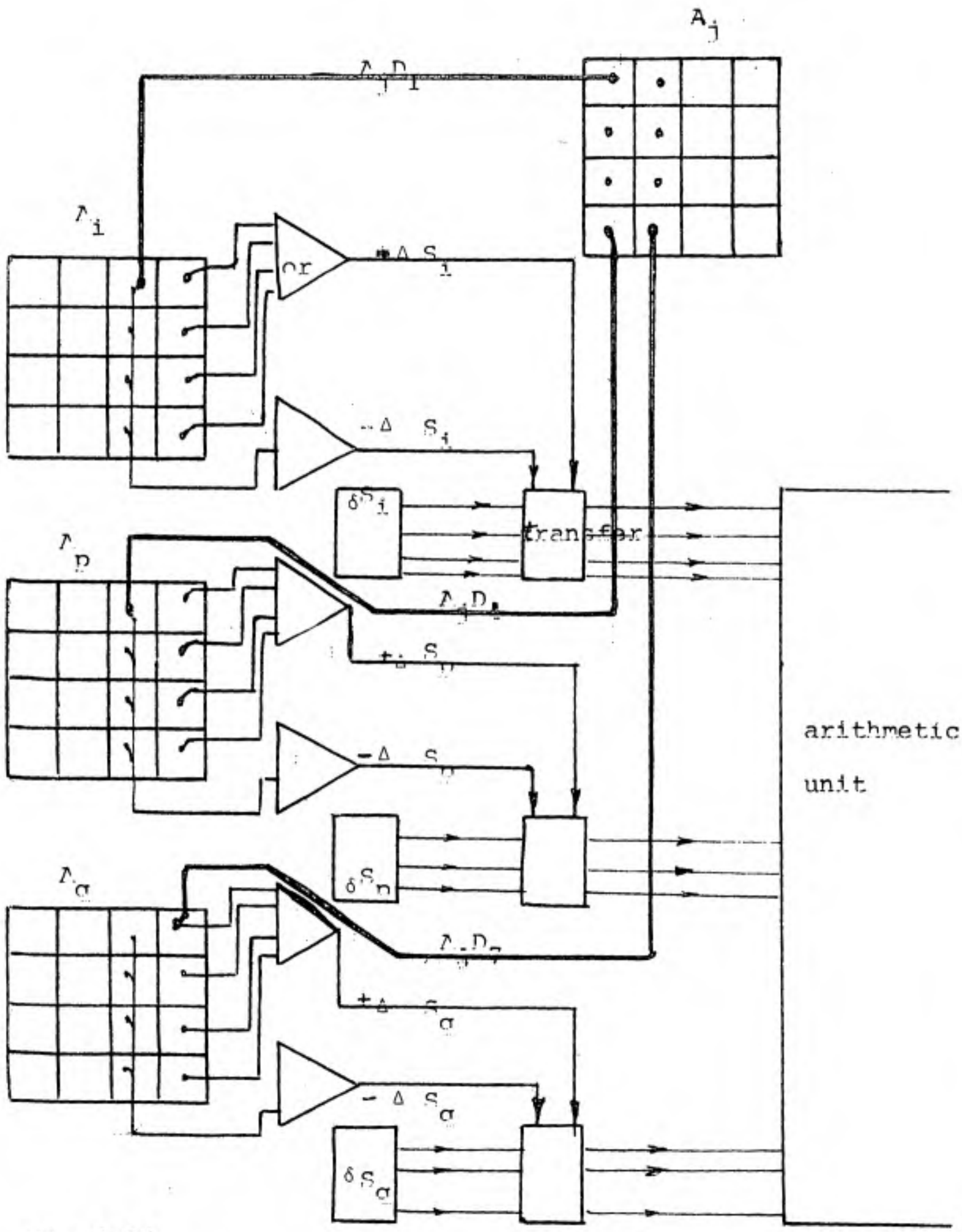

A_j
A_j D_1
A_i
or
+Δ S_i
-Δ S_i
δS_i
transfer
A_p
A_j D_2
+Δ S_p
-Δ S_p
δS_p
A_σ
A_j D_3
+Δ S_σ
-Δ S_σ
δS_σ
arithmetic unit

Fig 6-24

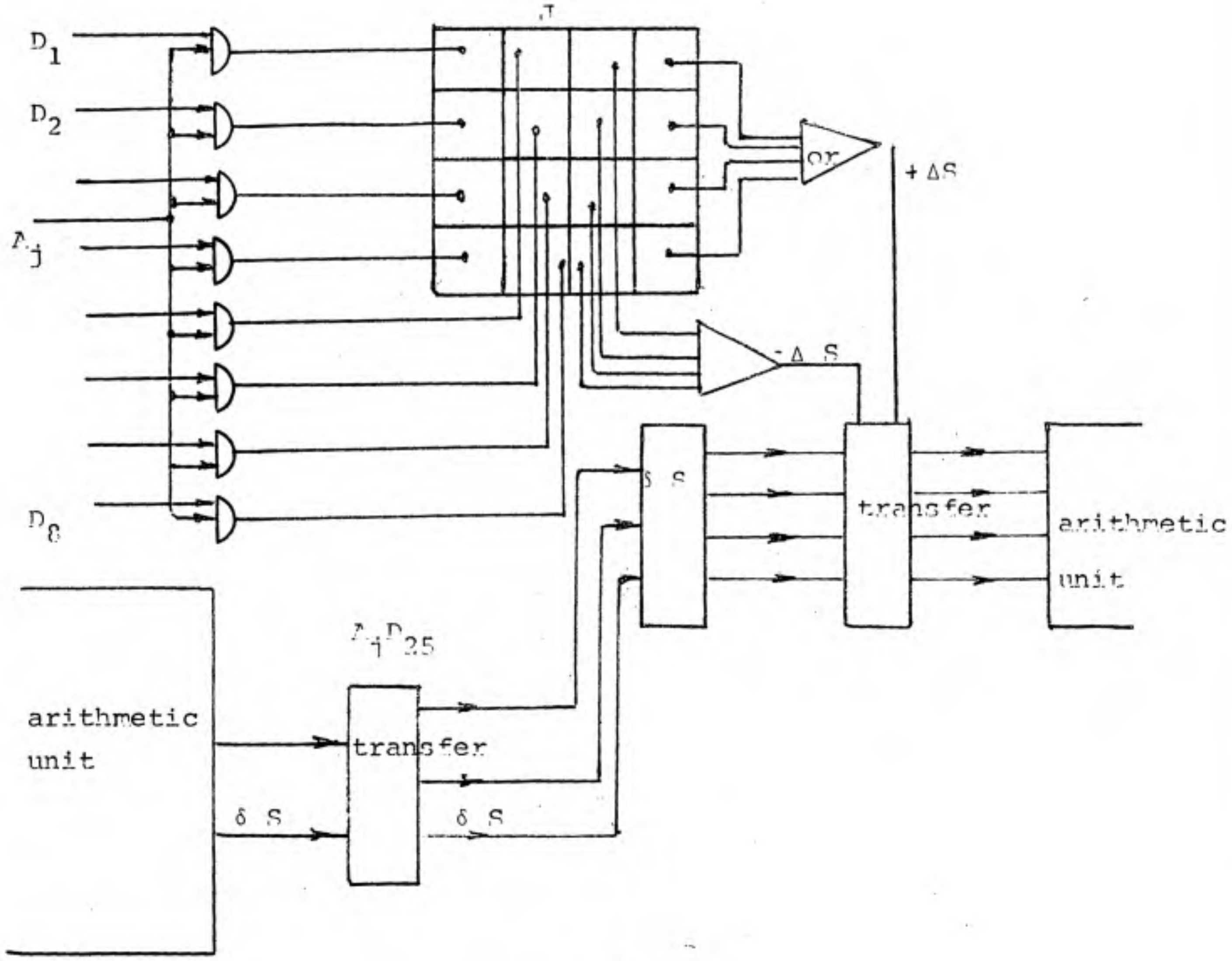

J
D_1
D_2
A_j
D_8
or
$+\Delta S$
$-\Delta S$
δS
transfer
arithmetic unit
A_jD_{25}
arithmetic unit
transfer
δS
δS

Fig. 6-25

6.4. Main store of incremental computer.

In this article, a description is given of the main store of the incremental computer, which is designed by the author in the industrial electronic laboratory of the Brussel University.

The basic methematical elements of the computer are a set of sixty integrators which are based on the unitary increment computations and ten integrators which are based on the multiple increment computations. Each of them are represented by a pair of b neary numbers, Y_i and s_{oi} ($i = 1, 2, \ldots 60$). These numbers of ten bineary digits plus signe, are stored on two 11 x 140 magnetic core matrix, from which they are read together in parallel, i.c. The Y_i and s_{oi} numbers corresponding to one integrator are treated within, the store as a single word of 22 bits. In one complet machine cycle, all sixty words are read in the sequence $A_1, A_2, \ldots\ldots A_j$. During the period A_j, the number pair Y_i and s_{oi} are transformed to the arithmetic units. In arithmetic unit, the numbers are operated upon according to a programme, producing two new numbers y_{i+1}, $\Delta s_{(i+1)o}$. These numbers are returned to the locations previously occupied by Y_i, s_{oi}, in the main store, to be read as a new Y_i, s_{oi} during the succeeding machine cycle.

The two memories can also be used completely independent of the incremental computer, they can be programmed and addressed manualy by the signals or by the programme, for any mathematical and logical operation. The values of registers A, B, C, D, R, S of arithmetic unit can be written in the two memories, and any value which is read in the memory, can be put in the registers A, B, C, D, R, S.

The figure (6-26)shows the interconnections between memory, arithmetic unit and control unit.

Each memory is devided into memory locations. The memory location has the capacity to store one word, consisting of 11 bits. The numbers of bits are same for all locations of the memory. When a word is to be stored, that word is sent to the memory and at the same time an address is given to it. The word is then stored in that address.

This procedure is called writing in the memory. The opposite procedure, is called reading from the memory, again an address is given, whereupon the contents of that address are returned at the arithmetic unit. A store operation in an address automatically involves the erasur of the previous contents of that address.

Information in core storage is stored by using the magnetic properties of ferrite cores. Each ferrite core can store a single bit of information, either a logical 1 or a logical 0.

A ferrite core is a bi-stable storage device. The ferromagnetic properties of the core permit it to be megnetized by applying an internal magnetizing force. Two wires X and Y, carrying electric current, inserted through the core, provide this force. After the core is magnetized, it retains the magnetic flux even though the magnetizing

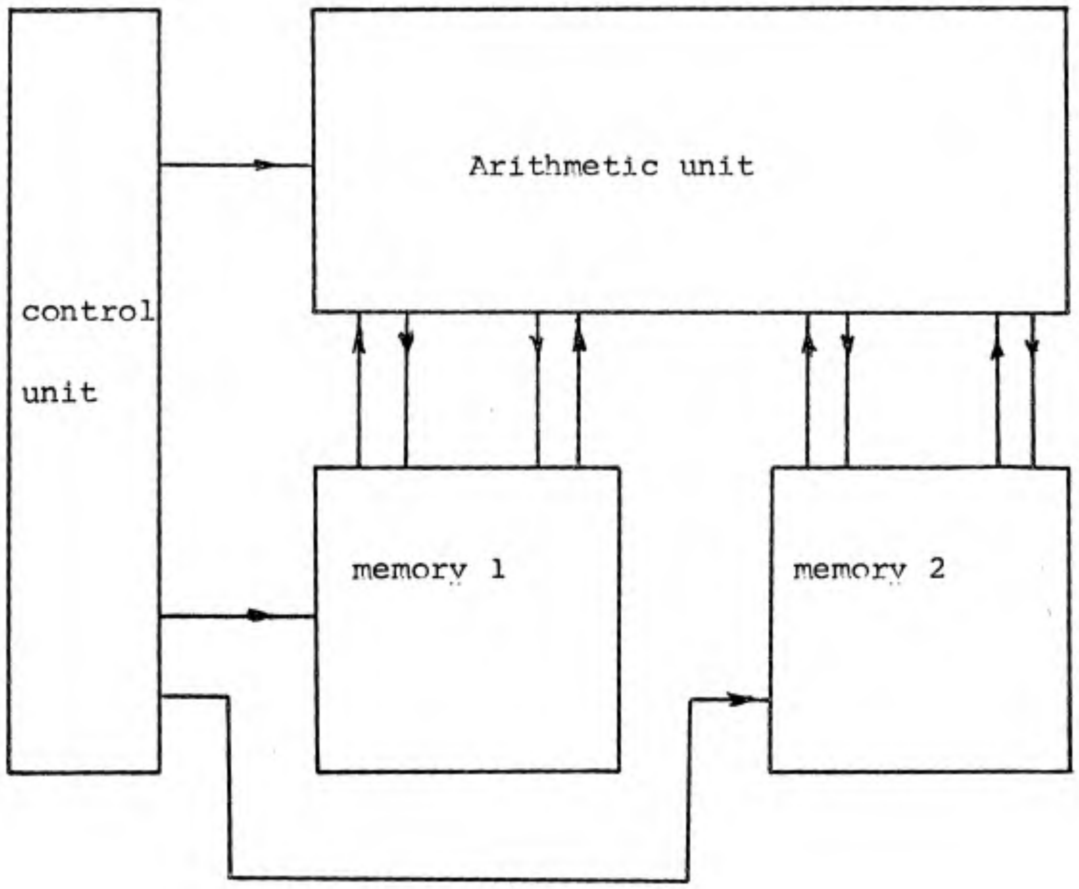


Fig 6-26

force is removed. It can be magnetized in either a north-to-south or a south-to-north direction in much same way that a trigger can be set in two states, hence, the term bi-stable. These two states are shown as points A and D in figure (6-27).

Magnetic force H is plotted horizentally, flux density B, is plotted vertically. If a large positive H, represented by the current I is applied, the core retains a flux dendity D, after the magnetizing force has returned to zero. Similary, if a large negative I is applied to the core, the flux follows the path D, E, F. Then the core retains a residual magnetizing equivalent to A after the negative I has been removed. Current greater than I or-I have little effect on the flux density B, because the core is then saturated.

Point D has been arbitary assigned to stand for logical one, and point A for a logical zero.

In figure (6-27) it is evident that $\frac{1}{2}$ I has little effect on the flux density B of the core. For example, if the core were in the logical one state, point D, and only - $\frac{1}{2}$ I were applied, the magnetizing force produced would be insufficient to cause the flux density to change beyond point E. The core would therefore not change its bit status from the logical one to zero as it did when full -I was applied.

The switching system used in core storage, uses this ability of the core to discriminate between $\frac{1}{2}$ I and full I. Considering the X and Y wires running parallel through the core and each carrying 250 miliamps of current in the same direction. The resulting force from each current is additive and equivalent to the force produced by one current of 500 milliamps. Assuming that 500 milliamps is the -I required to switch the status of the core, the X and Y wires are each considered to be carrying $-\frac{1}{2}$ I.

Another wire, called the inhibit wire, passes through the core as shown in fig(6-28). It run parallel to the X wire and, like the X wire, can carry only $\frac{1}{2}$ I. At the appropriate time, it carries $\frac{1}{2}$ I in the direction opposite $\frac{1}{2}$ I in the X wire. The net results are to cancel the effect of the magnetizing force produced by the X wire. The inhibit line is energized when it is not desirable to change the bit status of the core. When -I is applied, it is possible to detect the original state of the core by means of an additional wire, through the core called the sense wire. If the core is in state D when -I is applied, the change of flux, from point D to A, induces a voltage in the sense wire. This voltage pulse indicates that the core contained a one. If the core is at point A when the -I is applied, there is no flux change and no voltage is induced in the sense wire. This absence of induced voltage indicates that the core contained a zero. The core planes with sense amplifies, inhibits drivers and switches are shown in fig(6-29).

The figure (6-29) refers to the 11 planes used for 11 bits of memory. It shows that one X drive line goes through 14 cores in a given row of a plane, and continues through the corresponding row of 14 cores in each of the other planes. In a 10 x 14 core array, one X drive line

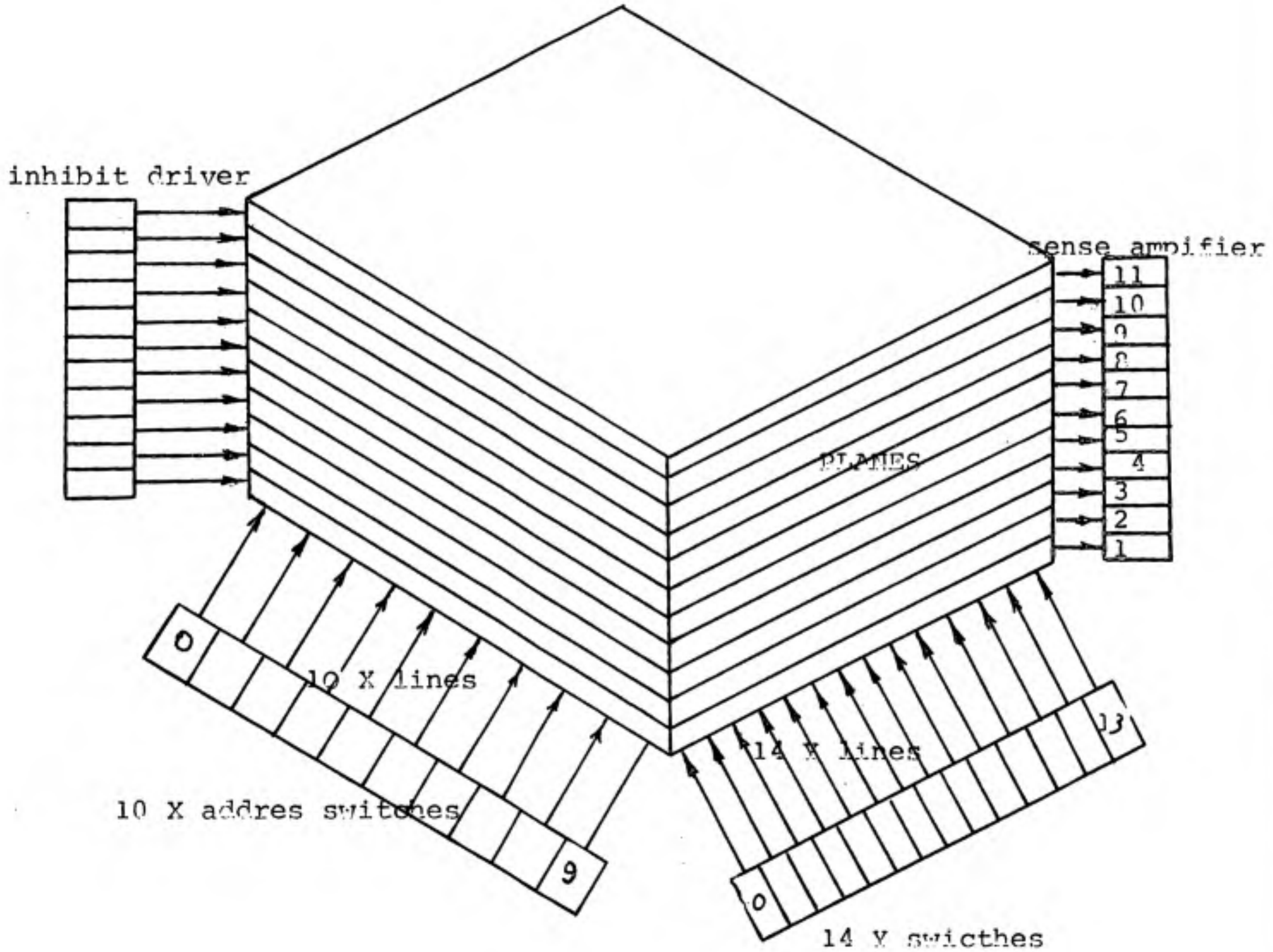
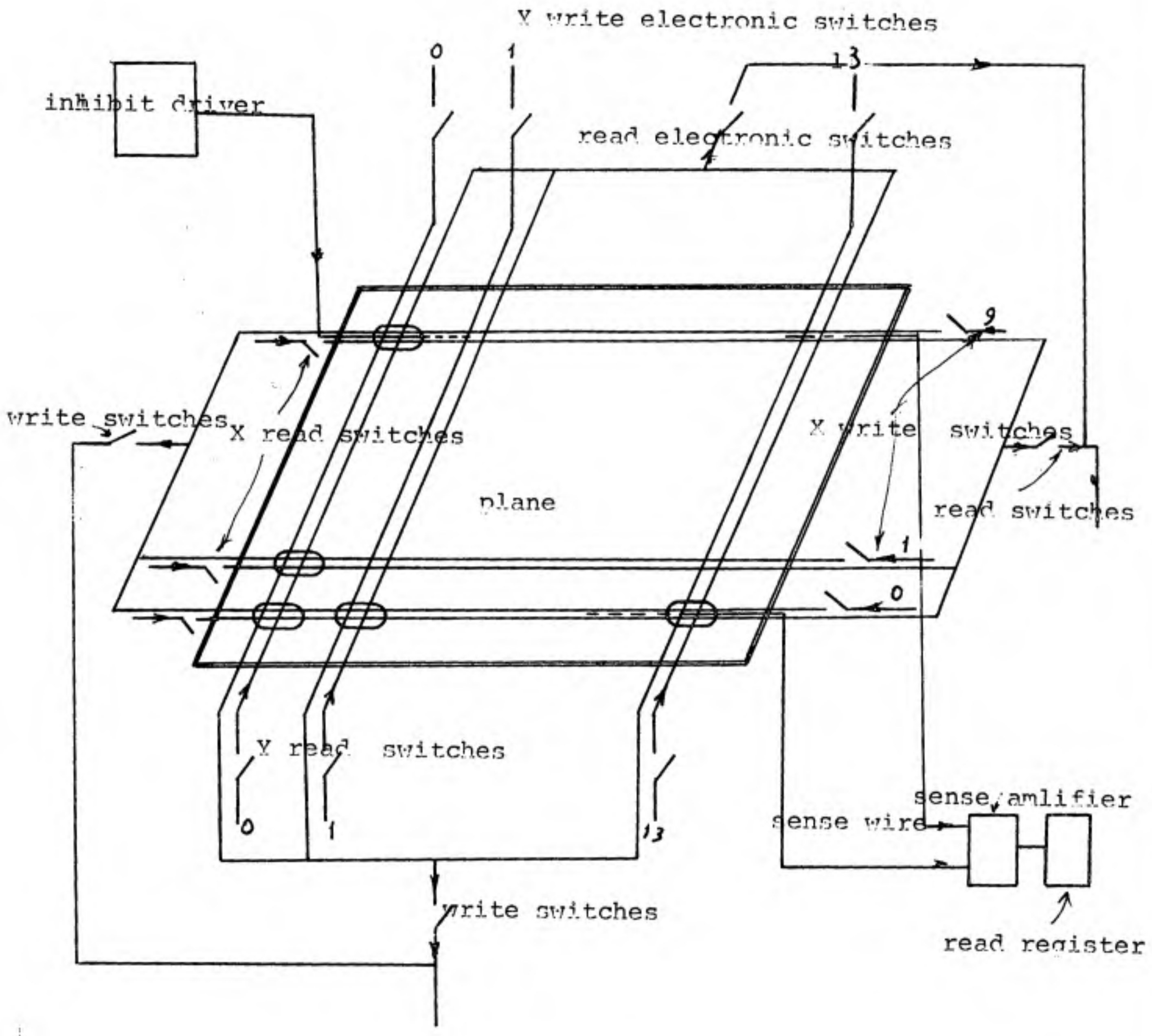

inhibit driver
sense ampifier
11
10
9
8
7
6
5
4
3
2
1
PLANES
0
10 X lines
9
10 X addres switches
14 Y lines
13
0
14 Y swicthes

Fig 6-29


Y write electronic switches
0
1
13
inhibit driver
read electronic switches
9
write switches
X read switches
X write switches
plane
read switches
1
0
Y read switches
0
1
13
sense amlifier
sense wire
write switches
read register

fig(6-29)

then passes through 14 (cores per row per plane) X 11 (planes); or 154 cores.

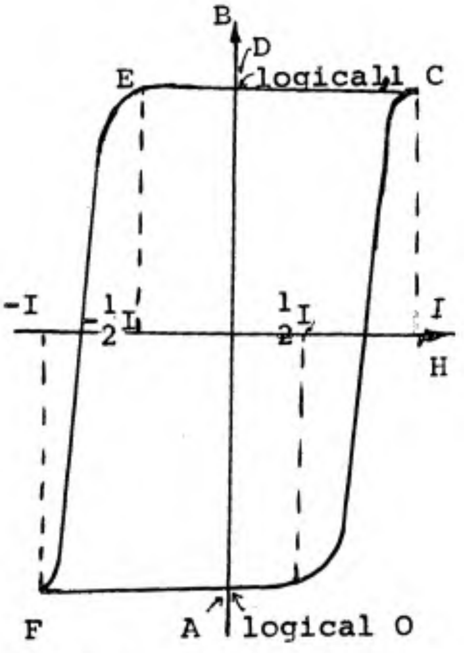
Similary, one Y drive line goes through all 10 cores in a given row of a plane, and continues through the corresponding 10 cores in each of the other planes. The Y lines crosses the X line at an angle of 90°, in each core plane. In a 10 x 14 core array, one Y drive line then passes through 10 (cores per row per plane) x 11 (planes), or 110 cores.

An inhibit drive line is associated with each bit plane.

A sense line is threaded parallel to the Y drive line through all cores in each core plane. Each core plane has one sense line. It is crossed over at the middle of the core plane to cancel the effect of the mutual coupling between it and the Y drive line.

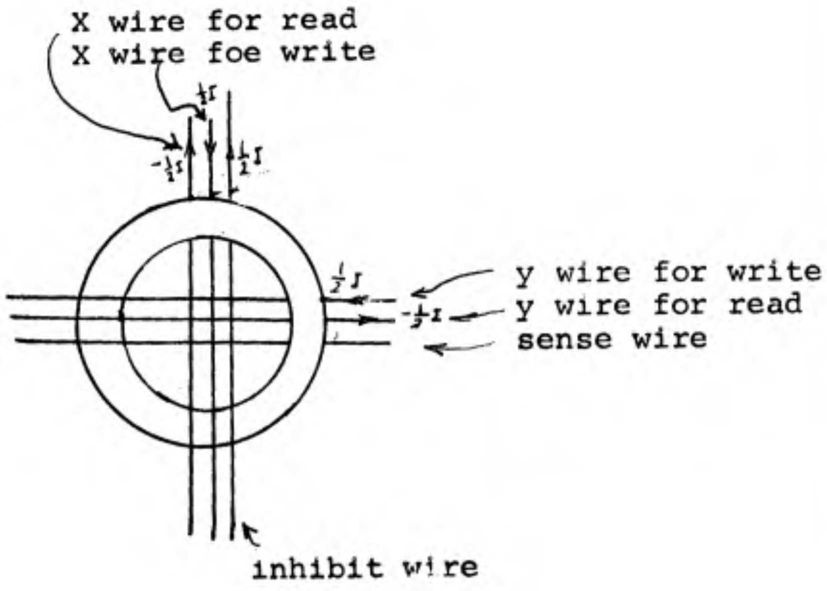
The logic flow from and to main core storage, is shown in figure (6-30) The sense lines from core storage must pass through sense amplifiers. The induced voltage pulse resulting from the read-out of a logical 1 is of such low amplitude (40 to 50 miltivolts) that it must be amplified to set the associated read register.

To select the induced voltage pulses from noise on the sense line, a short pulse (strobe) gates the sense amplifier just when the ferrite core is being read out.



Ferrite-Core Hysteresisloop

Fig 6-27



ferrite core

Fig 6-28

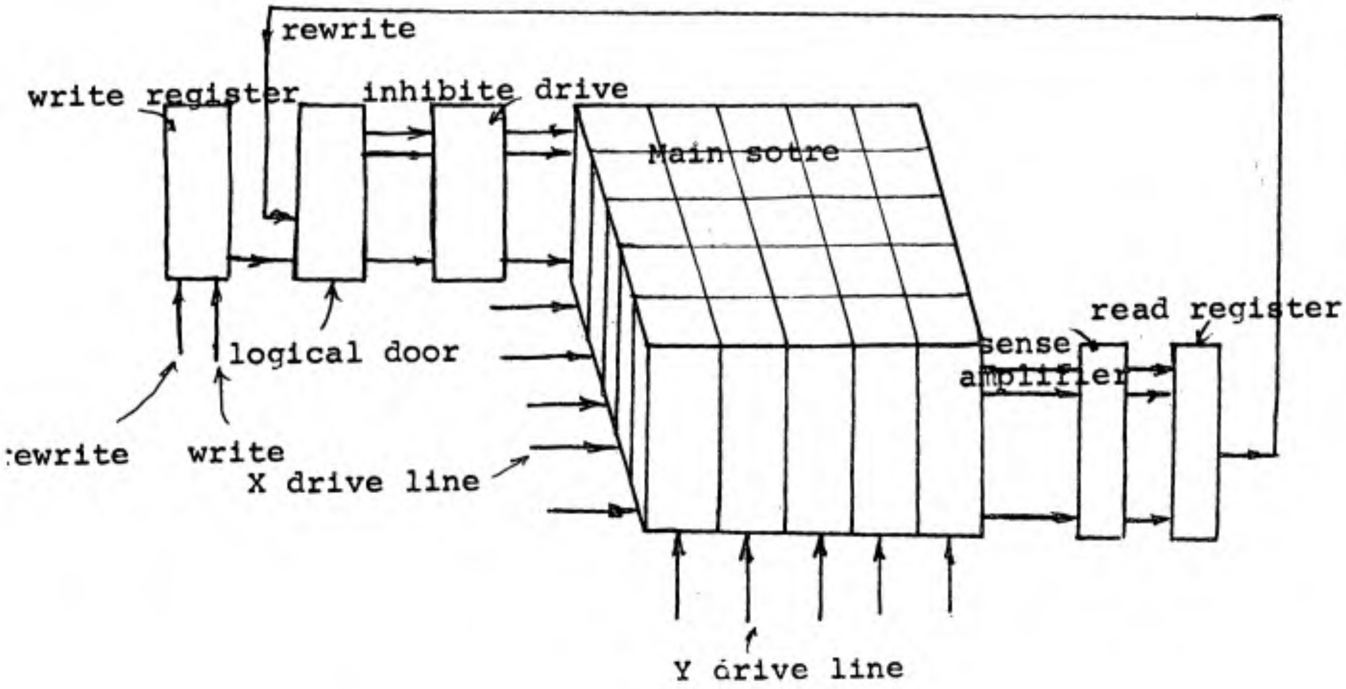


Fig 6-30

The read register holds the information read from storage. This information can be written back into the cores by the path indicated.

The inhibit drivers control the writing into cores. Because write current is in a direction opposite to read current, all cores would be set to 1 if the inhibit drives were not activated during write time. The inhibit drivers prevent writing a one in a core by allowing a half current to flow in the inhibit line that opposes the write current in the X drive line.

The core storage cycle (11.5 microseconds) is divided into two parts: The read portion and the write portion. The read portion of the cycle makes available to the system information contained in the core storage unit. The write portion of the cycle allows information contained in other unit to be entered into the core storage.

These operations of memory are explained in some words as following:

Read: On a read portion of storage cycle, the same relative core in each core plane is impulsed with X and Y drive current in the read direction. This current sets all cores at a particular address to 0. Any ones present during read time are picked up on the sense lines, amplified by the sense amplifiers, and used to set the read register. Cores that contain zero have no induced voltage in the sense line; therefore, those bits in read register rest inchanged (0).

Write: During the last part of the storage cylce, the X and Y drive lines of writing are impulsed in the opposite direction or write direction. This write current is in a direction to place a 1 in the cores being addressed. However, if any of the cores are to remain zero, the corresponding inhibit gates turn on their associated inhibit drives. With the inhibit drivers on, inhibit current flows in the inhibit line in a direction opposite to the current in the X drive line. Therefore, the X drive current is cancelled and the core remain at zero. The read write signal the inhibit signals and output signal of sense amplifiers, are shown in figure (6-3I).

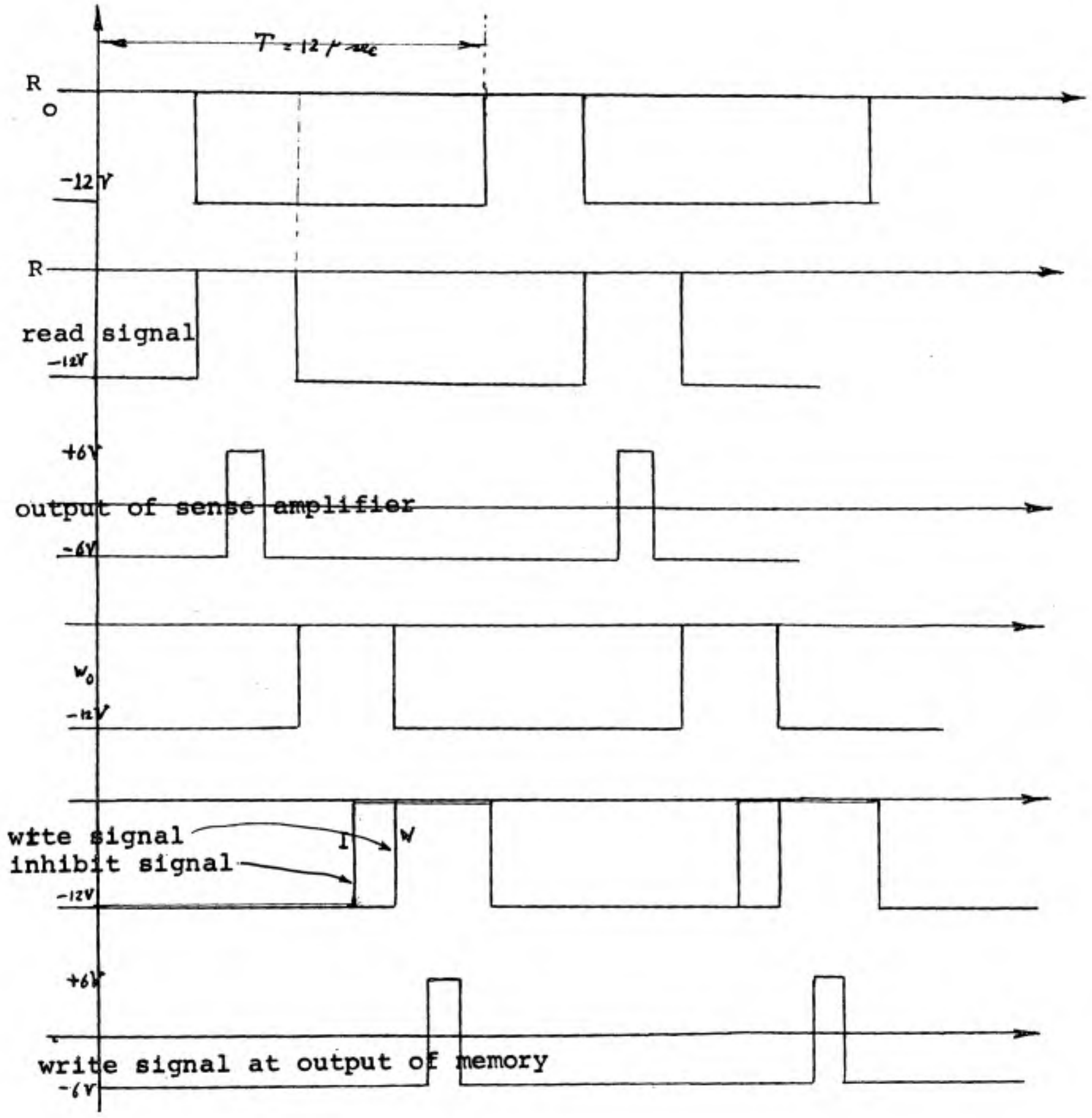

T = 12 μsec
R
O
-12V
R
read signal
-12V
+6V
output of sense amplifier
-6V
W₀
-12V
write signal
inhibit signal
I
W
-12V
+6V
write signal at output of memory
-6V

Fig 6-30

6.5. Conclusions.

This investigatin has led us to develop a new method of incremental computation, with multiple increments, which has the advantage of speed, versatility, flexability and short slewing time for jump function over the unitary incremental computation (D.D.A.).

A new effective method has presented for the computation and comparison of various errors in unitary and multiple incremental computation. This was necessary in order to choose the algorithms of the machine in the most economical and convenient way with the desired accuracy in results. It was shown that by choosing more accurate interpolation formula for incremental computation, it is possible to reduce the error of method as low as possible. But from calculation of this error, it was shown that, it is not worthwhile to use higher than first degree interpolation formula (trapezoidal) for unitary increment computation. It was also concluded that, the second degree interpolation formula (three points method) is a good approximation for multiple increment computation.

From the computation of quantization error, it was seen that, the quantization error is same for each method of integration but it depends to the quantums Δx, Δy and the number of bits h in multiple increments. The study of this error conducts us to choose the register's length and the speed of the computers. The study of round off and transmission errors showed the way of minimizing them in incremental computers.

The calculation of the total error enabled us to compare the error for different methods of integration in unitary and multiple increment computation. From the comparison of these error, we deduced the choice of algorithms for unitary and amultiple increment computer, with the desired accuracy.

We explained the design, algorithm and development of a new type of transistor incremental computer which is desvised by the author ir the Laboratory of Brussels University. This computer performs the integral operation on the basis of unitary and multiple increment computation. In addition, this machine is capable of doing the basic mathematical and other combined operations. By a new algorithm, the interconnection between the integrators was realized on the patch panel only by one lead. A new method of multiplication of two functions is represented which use only one integrator with higher accuracy instead of three integrators which were used before.

The memories of computer can be used for incremental or numerical operation of computer or works as the independent memory device.
The selector of control unit gives the order of integration sequence to the arithmetic unit in only desired sequences. The time of computing the integration in general purpose is 50 msec. (with clock frequency of 1 MHz) in unitary increment computer is 40 μsec. (with clock frequency of 500 KHz), so the speed of this machine is 2.500 higher than general purpose. Still the multiple increment computation increase the speed of integration by 2^4 (with same clock frequency),

so the speed of integration is increased by $2.500 \cdot 2^4 = 40.000$ compared to general purpose digital computer.

From above discussion, it appears that future investigation would be directed to the application of incremental computation in the Direct Digital Control (D.D.C.), optimization and simulation of modern automatic control which requires high dynamic quality, accuracy, flexibility, speed and reliability.

Bibliography.

I. Electronic computers and logical systems.

1. J. FLORINE, Automatismes à sequences et commandes numériques. Bibliothèque de l'automaticien DUNOD. Paris, 1969.

2. J. FLORINE, La synthèse des machines logiques et son automatisation. Presses acadèmiques européennes. Bruxelles, 1964.

3. J. FLORINE, Introduction à la recherche optimale de fonctions logiques. Revue A, No. 2. 1964, pp. 76-82.

4. S.P. FRANKEL, The logical design of a simple general purpose computer Trans., I.R.E. on electronic computer. Vol. EC.6. No.1 pp. 5-14. March 1957.

5. G.A. MALEY & J.E. ARLE, The logic design of transistor digital computers. Englewood cliffs, N.J. 1963. Prentice-Hall, INC, London.

6. D. MAC CRAKEN, Digital computer programming. John Wiley & sons. New York. 1957.

7. M. PHISTER. Logical design of digital computers. John Wiley & sons. New York. 1956.

8. M.V. WILKES. Automatic digital computers. John Wiley & sons. New York. 1956.

II. Incremental computers.

1. BROWN, E.L. Digital Computer Design. Academic Press. New York. 1963.
2. BRIT, J. A directly-coupled serial adder, designed for use in D.D.A. I.R.E. Vol. 23. April 1962.
3. DONAN, J.F. The serial memory D.D.A. math tables aides to computation. No. 38, pp. 102-112. April 1952.
4. DICKINSON, M.M. A comparaison of D.D.A. and G.P.C. in guidance systems. A.I.E.E. January 1961.
5. GEAR, C.W. The numerical integration of ordinary differential equations, math tables aides to computation, pp. 146. April 1967.
6. WILF, HERBERT. S. An open formula for the numerical integration of first order differential equation. Math Tables Aides to Computation. January 1958.
7. HILLS F.B. A study of incremental computation by difference equations. M.I.T. Thesis 1958. Cambridge Massachusetts.
8. HAMER, E. and PALMER, W. Comparaison of computational speeds of D.D.A. and G.P.C. I.E.E.E. Trans. Vol. EC-13. June 1964. pp. 307-308.
9. KNUDSEN, H.K. The scaling of D.D.A. I.E.E.E. Trans. Vol. EC.14. pp. 383-390. August 1965.

10. LAMSON, R.C. A division Algorithm for D.D.A. I.E.E.E. Trans. Vol. EC.13. pp. 54-55. February 1964.

11. LAMB, D. Design and application of D.D.A. Proceeding of I.R.E. Australia. April 1961.

12. MAYOROV, F.V. Electronic Digital integrating Computer. Iliffe Books. London 1964.

13. MONROE, A.J. Digital Process for sampled data systems. John Wiley & Sons. New York 1962.

14. PALEVSKY and J.F. HONWELL. Digital differential equation solner. Instr. and Control system. Vol. 63. pp. 118-121. April 1963.

15. RAYMOND, F.H. Vice président de l'A.I.C.A. Réflexions sur les machines à calculer. A.I.C.A. Février 1966.

16. RAWLEY, G.C. Transistor Circuits for D.D.A. Proc. I.E.E.E. Vol 106. Part. B. suppl. No 16. pp. 685-698. May 1959.

17. REARILH.Function generator of digital differential Analyzer. I.R.E. Trans. Telemetry and remote control. Vol TRC-1. No3 pp. 8-12. August 1955.

18. SHABELUND, D. The numerial process of a binary D.D.A. University of Utah. Salt Lake City 1, Utah. U.S.A. August 1953.

19. STOLLER.L, MORRESON. D. A method for numerical integration of differential equation. Math Table. Aides to computation. October 1958.

20. SHILEIKO, A.V. Digital differential Analyzers . Peramon Press. Oxford 1964.

21. TURTLE, O.C. Incremental Computer error Analysis. I.E.E.E. Trans. on Commun and electronic. Vol. 82. pp.422-425. Septmber 1963.

22. TOOTILL,G.C. The incremental digital Computer, first principals of operation and use. Process control and Automation. Septmber 1958.

23. SPRAGUE, R.F. Fundamental Concepts of D.D.A. Math Tables. Aides to computation. Vol. 6. No. 37. pp. 41-49. January 1952.

24. HANGE ,O.error analysis of D.D.A internal raport T.H.Achen Germany. September 1960.

III. Automatic Control.

1. BERTAN. The effect of quantization in sampled data feedback system. A.I.E.E. pp. 177. September 1958.

2. BELLMAN, R. KABABA,R. Dynamic Programming and feedback control. 1St IFAC CONGRESS. pp. 460-464.

3. CURRY, E.E. The analysis of round off and truncation error in a Hybrid control systems. I.E.E.E. Trans. on Automatic Control. pp. 601. October 1961. Vol AC.-12 No 5.

4. HOUSEHOLDER, A.S. Principal of numerical Analysis. M.C. Graw-Hill. New York 1953.

5. FALB, A.M. Optimization Control on introduction to the theory and its application. M.C. Graw-Hill. New York. 1966.

6. KORN G.A. and KORN T.N. Electronic Analog Computers. M.C. Graw-Hill. New York 1956.

7. LARSON, R.E. Optimum Quantization in Dynamic systems. I.E.E.E. Trans. on Automatic Central. Vol AC-12 No 2. pp.162. April 1967.

8. JOHNSON, G.W. Upper bound on Dynamic quantization error in cøntrol system via the direct method of Liaquenov. I.E.E.E. Trans. on Automatic Control. Vol AC-10 No 4. pp.432. October 1965.

9. JOHNSON, G.W. and G.N.T. LACK. Comment on "Upper Bound on

Dynamic quantization error in Digital Control systems via the the Direct method of Liaquenov. I.E.E.E. Trans. on Automatic Control. Vol. AC-11 No 2. pp.331. April 1966.

10. MAITRA, K.K. and P.E. Sarachich. Digital compensation of continious data beedback control system. Trans. A.I.E.E. Vol. 75. Pt II. pp. 107-114. 1956

11. MISKLEIN & BRAUN. Adaptive Control systems. M.C. Graw-Hill.

12. MONROE,A.J. Automatic Control digital processes for sampled data systems. John Wiley & Sons Inc. New York 1962.

13. PERETZ,R. Analyse et Synthèse de processus Automatiques. Revue A. No 1. pp. 27-37. 1963.

14. PERETZ,R. Le rôle du régulateur dans les processus industriels. Revue A. No2. pp. 86-96. Avril 1961.

15. PERETZ,R. Opérateurs Hybrides. 4th International Analogue Computer Meeting.

16. JOSEPH, Peter D. Quantization in linear control systems. I.E.E.E. Trans. on Automatic Control. Oct. 1965, Vol AC-10, n°4, p. 439.

17. PETERSON, G.P. Basic Analog Computation. University of Arizona. The Mac Millan Co., New York 1967.

18. RAGAZZINI, J.R. & FRANKLIN, G.F. Sampled data control systems. McGraw-Hill, New York, 1958.

19. RAGAZZINI, J.R. Digital computers in feedback systems. I.R.E. Convention record. Vol.5, Pt.II, 1957, pp.33-42.

20. RYDER, J.D. Engineering electronics with industrial application and control. Mc Graw-Hill Company, LTD, NewYork, 1957.

21. SILBERBERG, M.Y. Dynamic optimization and control in computer control of industrial processes. Mc Graw-Hill, 1965.

22. SKLOVSKY, J. & RAGAZZINI, J.R. Analysis of errors in Sampled data feedback systems. A.I.E.E., Vol.74, New York, 1957, pp.5-7.

IV. Electronics.

1. GOLDSTICK, G.H. Comparison of saturated and nonsaturated switching circuits techniques. I.R.E.Trans. Electronic Computer, June 1960.
2. HENBE, R.A. & WALSH, J.M. The application of transistors to computers, Proc. I.R.E., June 1958.
3. HUNTER, L.P. Handbook of semiconductor electronics, Mc Graw-Hill, New York, 1956.
4. HURLEY, R.B. Junction Transistor electronics, John Wiley & Sons, New York, 1958.
5. JOYCE & CLARKE. Transistor circuit analysis, Addison-Wesley Publishing company, Inc., London, 1963.
6. LECAN, C., HART, K. & DE RUYTER, C. The junction transistor as a switching device. Philips Technical Library, Reinhold Publishing Corp., New York, 1962.
7. STOVER, Wm.A. Integrated circuits, Texas Instruments Technical Seminar Series, 1966.
8. PRESSMAN, A.I. Design of Transistor circuit for digital computers. John F. Rider Publisher Inc. New York, 1959.
9. SMITHS, J. Transistor Physics. Philips International Institute Publications, Eindhoven-Holland, 1966.