CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r. 82/08)

Canada

STORED SQUARE DIGITAL FILTERING

by

TARUNA TJAHJADI

A thesis
presented to the School of Graduate Studies and Research
of the University of Ottawa
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in Electrical Engineering

Ottawa, Canada, 1982

To My

Parents, Brothers and Sisters

University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

A digital filter realization based on the stored square multiplication technique replacing conventional digital multipliers is presented.

The error analyses are performed and compared with the accuracy obtained in direct multiplications.

The topics of hardware realization and speed are discussed and the split address technique is introduced to reduce the memory size required, for address wordlength longer than 11 bits.

The implementation of the stored square digital filter with pipelined configuration is described. With this structure a sampling frequency in excess of 15 MHz can be achieved.

A potential application of such a filter to adaptive filtering is evaluated.

ACKNOWLEDGEMENTS

The author wishes to express his deepest gratitude to his thesis supervisor Professor Willem Steenaart for his patient guidance and encouragement throughout this work.

Special thanks is extended to Miss Tjeng Kiem Tjoe for her help in preparing the drawings of this thesis.

Thanks to the National Research Council and to the University of Ottawa for financial assistance throughout the duration of the work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I

INTRODUCTION

1.1 GENERAL

A digital filter is a signal processor that enhances certain classes of signals and attenuates others digitally. In general, the implementation consists of a sequence of binary multiplications and additions. Since, in most cases the multiplication is the slowest operation in any digital filter realization, therefore, it is of the best interest to find a method of increasing the multiplication throughput rate in order to increase the sampling frequency without sacrificing the filter's accuracy. Although fast digital multipliers are available at present, however, that particular choice of solution to the problem is not that favourable due to the high cost of a fast digital multiplier and the application of several in a circuit.

One solution of achieving a fast multiplication rate with lower cost is to use the ROM look-up table technique. As larger ROMs at lower cost are now available, it is expected that in the future digital storage will prevail over digital arithmetic in terms of cost, speed and availability.

The use of ROM elements replacing multipliers in digital filters has been considered using distributed arithmetic [1] [2] and the stored product technique [3] [4] [8] [9]. These techniques require the a priori knowledge of the filter coefficients to determine the stored values. In other words, for every different set of coefficients a different set of values have to be stored and this property limits the flexibility of the filter.

In applications where multiplexing and variability of the coefficients are necessary, the use of the stored square ROM multiplication technique is the answer, where the squares of the sums and the squares of the differences of input signal and coefficient are stored, hence, it has the advantages of having identical ROM contents and independent of the multiplier coefficients. This flexibility makes application such as adaptive filtering possible. In addition, with a pipelined configuration a stored square digital filter can be implemented with a sampling frequency in excess of 15 MHz.

In the process of circuit integration the fact that the contents of all ROM elements can be the same and can be inscribed at the time of circuit integration is a distinct advantage over the other two methods using ROMs, as quoted above.

The performance evaluation of the stored square ROM multiplication technique in digital filtering has not been reported. The major objective in undertaking this research work is to explore the insights in depth, the viability of the stored square digital filtering technique.

1.2 SUMMARY

This thesis deals with the analysis of the error on the stored square ROM multiplier as a candidate for replacing the conventional digital multiplier in filtering. The error analysis is given for fixed point as well as for floating point arithmetic. It will be shown that the use of this technique for floating point arithmetic is not as favourable as for fixed point. In practice the fixed point arithmetic mode is the most commonly used for real time application where speed is of prime importance. Based on this evaluation the realization of digital FIR (Finite Impulse Response) filters using stored square ROM multipliers is presented. The performance of the filters is evaluated and a pipelined structure is proposed to enhance the speed. Finally the application of pipelined stored square digital FIR filters to adaptive filtering is described.

Chapter II presents the stored square ROM multiplier concept. An analysis is carried out and the results are compared with those of a digital multiplier for fixed point and

for floating point arithmetic. A method of reducing the required ROM storage size is described. The hardware realization of stored square ROM multiplier is discussed and the pipelined structure is proposed to improve the multiplication throughput rate, hence, taking advantage of the availability of fast ROM access time.

Chapter III is devoted to the design and analysis of the digital FIR filter and its implementation based on the pipelined SSRM (stored square ROM multiplier) with fixed point arithmetic.

An evaluation of the FIR filter performance and the effect of quantization and roundoff noise, carried out by computer simulation, is compared with the same properties of digital filters using conventional binary multipliers.

In chapter IV the implementation of adaptive digital filter using the stored square technique is described. The coefficients are adapted based on the Least Mean Square (LMS) algorithm. Again, simulations are carried out to evaluate the performance of such an implementation.

Finally, conclusions and suggestions for further studies are given in chapter V.

Chapter II

STORED SQUARE ROM MULTIPLIER (SSRM)

2.1 INTRODUCTION

The use of a stored square ROM as a multiplication technique was recently suggested [5] [6]. For use in digital filtering this technique requires additions before and after the ROM table look-up. Since the adders are available at lower cost as compared to the ROMs, therefore, the hardware cost of the SSRM is practically dependent on the cost of the ROMs and consequently the required ROM storage size.

An analysis is given of the accuracy obtainable with this technique, as compared with that of binary multiplication, for fixed point and for floating point arithmetic, assuming product roundoff error and two's complement numbers are used for negative values.

It is found that a compatible accuracy can be attained by this technique and also signal and coefficient quantization error can be shown to be equal to that obtained in conventional digital multiplier [10], [22].

The size of the ROM required for various wordlengths can then be formulated as a function of the required accuracy,

and of the signal wordlength. In the case of a very long signal and coefficient wordlength, it is necessary to reduce the required size of the ROM and this can be accomplished by using the split-address technique. To enhance the multiplication throughput rate a pipelined structure is proposed.

2.2 CONCEPT

A multiplication can be expressed as [6] :

$$x(n).c = \left[\frac{(x(n)+c)^2}{4} - \frac{(x(n)-c)^2}{4}\right] \qquad (2.1)$$

where x(n) and c are signal sample and coefficient respectively. Based on (2.1) a Stored Square ROM Multiplier (SSRM) is developed. The process involves storing the squared sum of signal and coefficient divided by 4 and the squared difference divided by 4. The quantities x(n)+c and x(n)-c will be used as addresses to the ROM. The overall multiplication consists of additions, subtractions and squaring, where the squares are obtained by a ROM lookup table technique. This technique will be evaluated for roundoff error performance and compared with digital multiplication for both fixed point and for floating point arithmetic.

2.3 ERROR ANALYSIS ON FIXED POINT ARITHMETIC

Consider a multiplication of two binary numbers using fixed point arithmetic with the binary point located between the sign bit and the first fraction bit. Assume that the signal, the coefficient and the output are each represented by N bits and the stored squares are represented by M bits ($N \leq M \leq 2(N+1)$) with sign bit excluded, all numbers rounded and two's complement numbers used for negative values. The block diagram of this multiplication process is shown in Fig. 1. The sum and difference of x(n) and c may result in a value of N+1 bits. "Absolute value" circuits (ABS) are used to change the sign of negative address numbers, and thus keep the number of stored bits to a minimum. The roundoff errors generated in the process are represented in Fig. 2. Note that for $M \leq N$, there is no output roundoff error (ε_{r3}).

The rounding errors of the squared numbers are bounded by :

$$-\frac{2^{-M}}{2} \leq \varepsilon_{ri} \leq \frac{2^{-M}}{2} \quad ; \; i=1,2 \qquad (2.2)$$

The roundoff error of the output is bounded by :

$$-\frac{2^{-N}}{2} \leq \varepsilon_{r3} \leq \frac{2^{-N}}{2} \qquad (2.3)$$

Using (2.1), the actual output is :

$$\begin{aligned} y(n) &= x(n).c + \varepsilon_{r1}(n) - \varepsilon_{r2}(n) + \varepsilon_{r3}(n) \\ &= P(n) + e_0(n) \end{aligned} \qquad (2.4)$$

with the output error :

$$e_0(n) = \varepsilon_{r1}(n) - \varepsilon_{r2}(n) + \varepsilon_{r3}(n) \qquad (2.5)$$

and the product :

$$P(n) = x(n).c \qquad (2.6)$$

The maximum output error to maximum product ratio is, with $P(n)_{max} = 1$:

$$\frac{\max.|e_0|}{\max\ |P|} = \begin{cases} \frac{2^{-N}}{2} + 2^{-M} & ; \text{ for } M>N \\ 2^{-M} & ; \text{ for } M \leqslant N \end{cases} \qquad (2.7)$$

Assuming that the roundoff error sources are statistically independent and uniformly distributed, the variance of the output error will be :

$$\sigma^2_{e_0} = \sigma^2_{\varepsilon_{r1}} + \sigma^2_{\varepsilon_{r2}} + \sigma^2_{\varepsilon_{r3}} \qquad (2.8)$$

where, $\sigma^2_{\varepsilon_{r1}} = \sigma^2_{\varepsilon_{r2}} = \frac{2^{-2M}}{12}$ and $\sigma^2_{\varepsilon_{r3}} = \frac{2^{-2N}}{12}$

Hence,

$$\sigma^2_{e_0} = \begin{cases} \frac{2^{-2N}}{12} + \frac{2^{-2M}}{6} & ; \text{ for } M>N \\ \frac{2^{-2M}}{6} & ; \text{ for } M \leqslant N \end{cases} \qquad (2.9)$$

With x(n) and c assumed to be uncorrelated, white and each of uniform probability density over (-1,+1), the variance of the product is :

$$\sigma^2_P = \sigma^2_x \cdot \sigma^2_c \qquad (2.10)$$

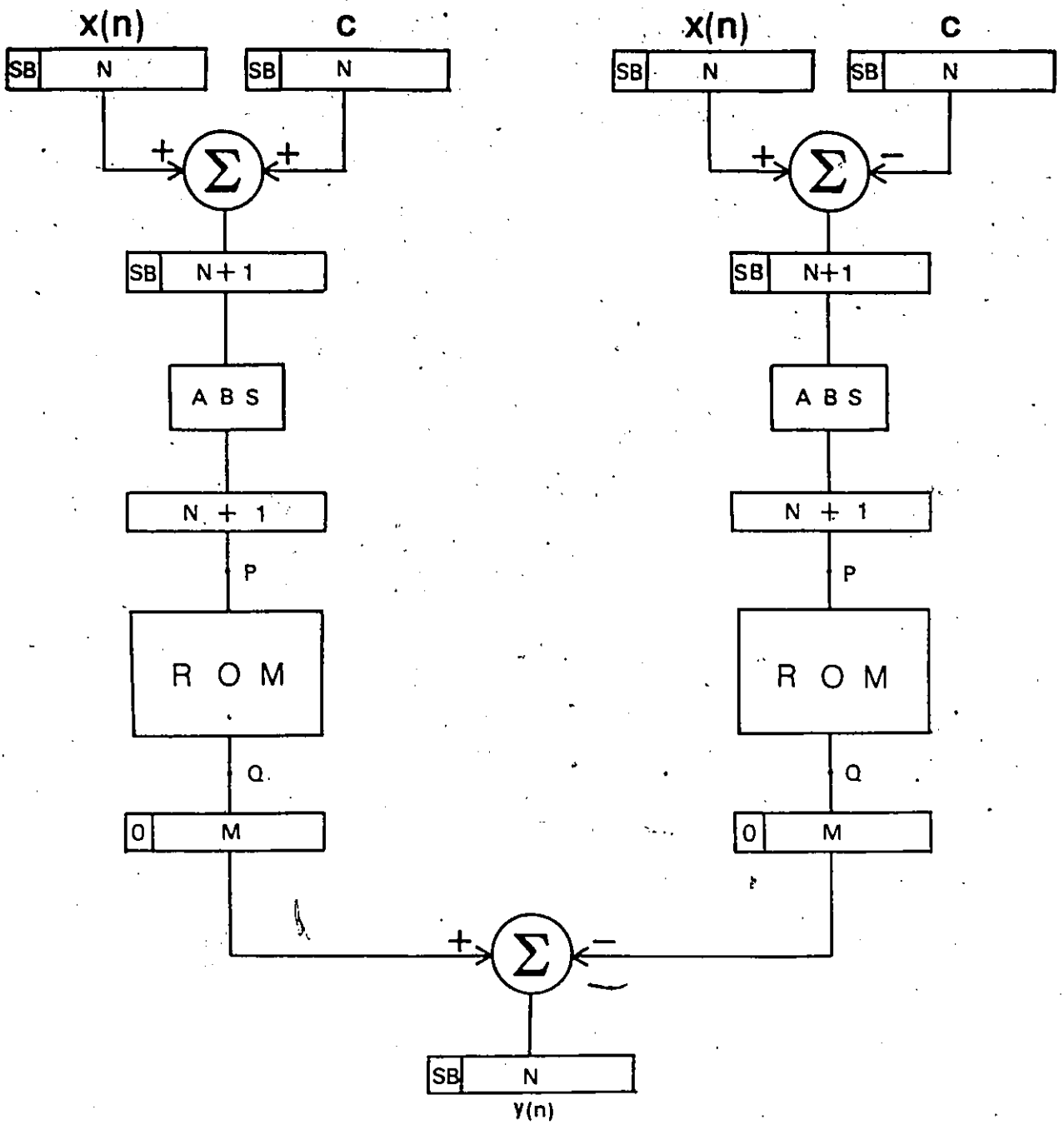


Figure 1: Block diagram of fixed point stored square ROM multiplier.

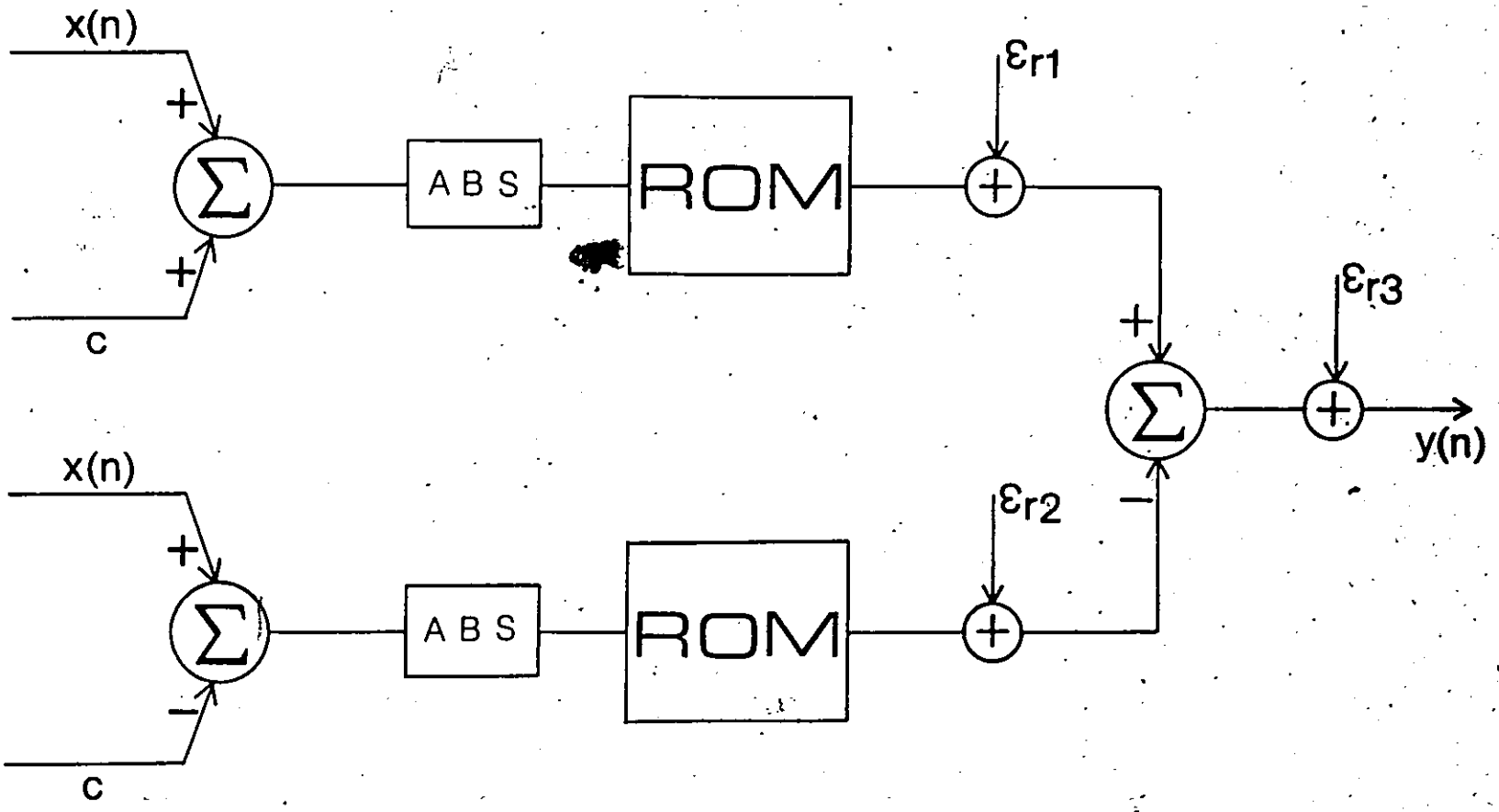


Figure 2: Block diagram of the multiplication process with roundoff errors, for fixed point arithmetic.

and

$$\sigma_P^2 = \left(\frac{1}{3}\right)\left(\frac{1}{3}\right) = \frac{1}{9} \tag{2.11}$$

So, the noise to product power ratio is :

$$\frac{\sigma_P^2}{\sigma_{e_0}^2} = \begin{cases} \frac{3}{4}\,2^{-2N} + \frac{3}{2}\,2^{-2M} & ; \text{ for } M>N \\ \frac{3}{2}\,2^{-2M} & ; \text{ for } M \leqslant N \end{cases} \tag{2.12}$$

In direct multiplication, the roundoff error of the product will be :

$$e_{Om} = \varepsilon_{rm} \tag{2.13}$$

where ε_{rm} is uniformly distributed between $-\frac{2^{-N}}{2}$ to $\frac{2^{-N}}{2}$.

Then, the maximum output error to maximum product is :

$$\frac{\max\ |e_{Om}|}{\max\ |P|} = \frac{2^{-N}}{2} \tag{2.14}$$

The variance of the output error is :

$$\sigma_{e_{Om}}^2 = \frac{2^{-2N}}{12} \tag{2.15}$$

and the output noise to product power ratio is :

$$\frac{\sigma_{e_{Om}}^2}{\sigma_P^2} = \frac{3}{4}\,2^{-2N} \tag{2.16}$$

From (2.7),(2.12),(2.14) and (2.16) it is concluded that the maximum output error to maximum product ratio and the output noise to product power ratio of the SSRM will approach to that of digital multiplier if M is larger than N and it is found that in general, M>N+3 for the product rounding to be comparable as shown in Fig. 3 and Fig. 4. The computer simulation results are plotted in the figures as well.

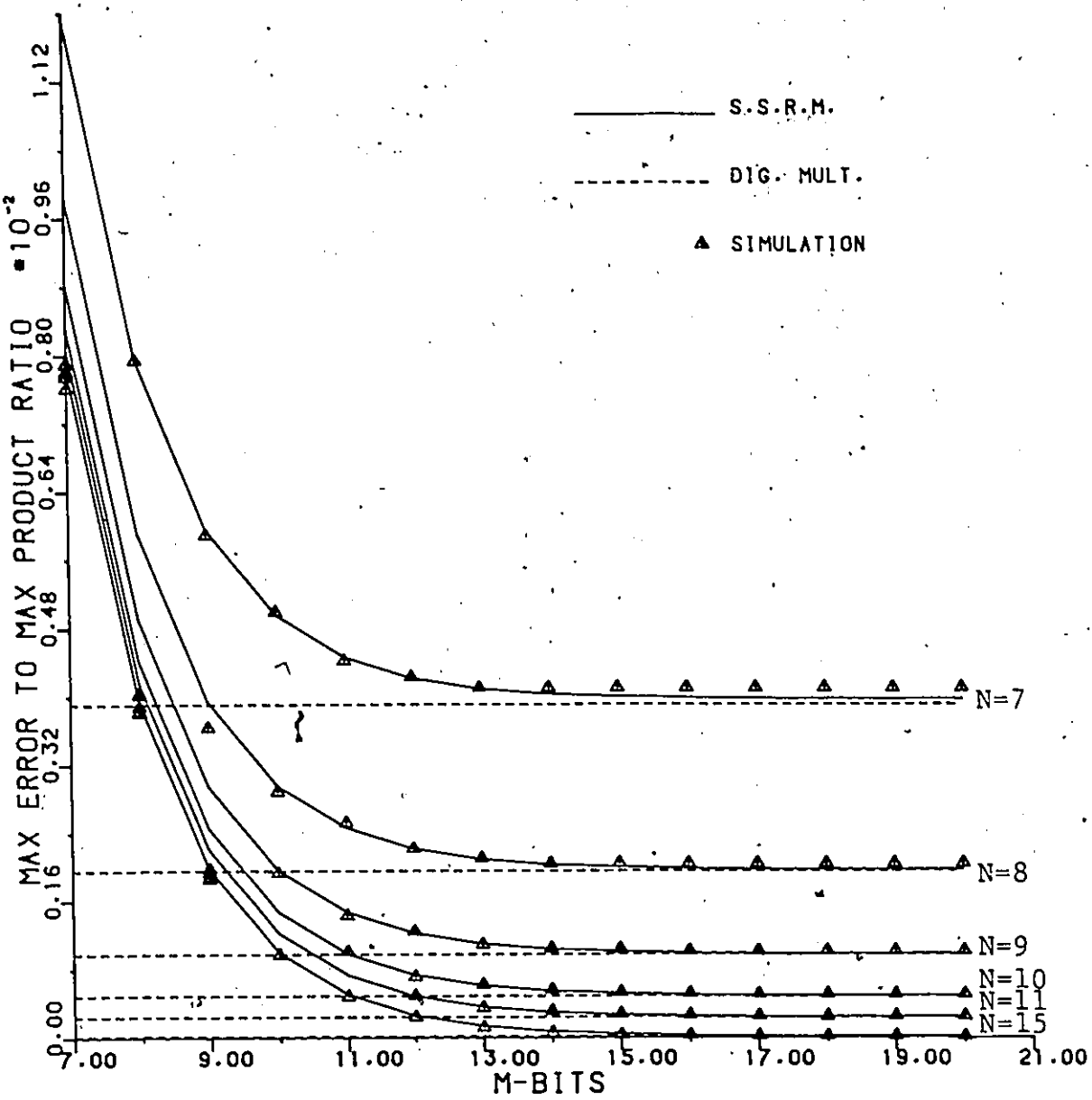


Figure 3: Maximum output error to maximum product ratio of fixed point plotted as a function M for various N.

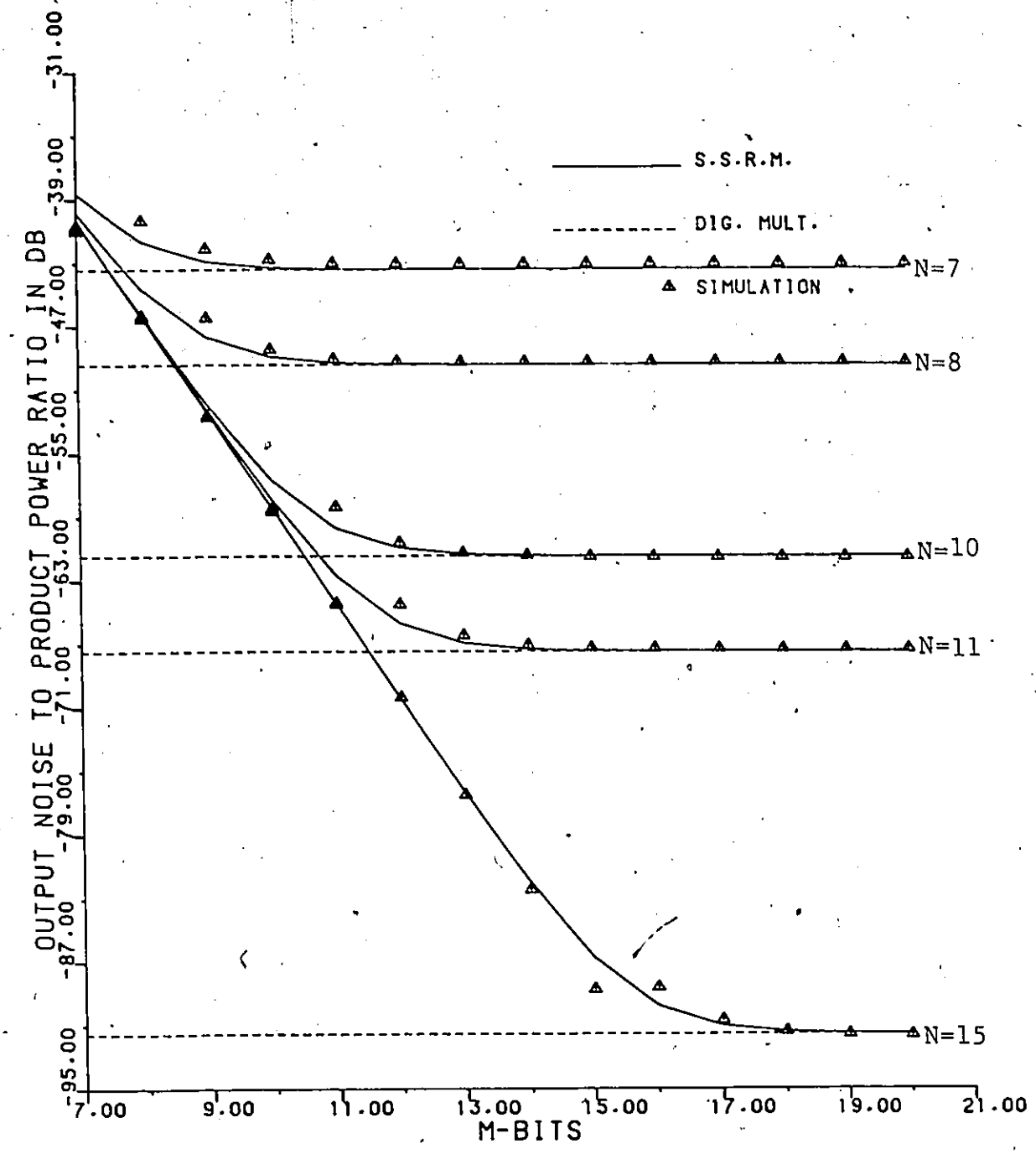


Figure 4: Output noise to product power ratio of fixed point plotted as a function M for various N.

2.4 ERROR ANALYSIS ON FLOATING POINT ARITHMETIC

In floating point arithmetic a number is represented by mantissa and exponent. The mantissa has a normalized magnitude in the range between 0.5 and 1. Let the mantissa of the signal and of the coefficient be N bits and the exponent be E bits, and let the mantissa of the stored squares be M bits. Fig. 5 shows the process in block diagram form. Note that the address to the ROM is (N-1) bits, since the address will always have a positive value between 0.5 and 1.

The errors that are generated in the process are

1. The errors due to addition or subtraction operation, denoted by e_{ai}.
2. The errors due to rounding, denoted by e_{ri}.

The multiplication proccess, along with the errors generated, is represented in Fig. 6.

Notice that the roundoff errors produced in the floating point arithmetic are not solely due to rounding the product, but also due to addition or subtraction process since the exponent and consequently the mantissa have to be adjusted accordingly before the operation.

In the following it is assumed that all quantization is by rounding and relative errors are used, denoted by ε. All errors are assumed to have the characteristics of white noise, and have uniform probability density.

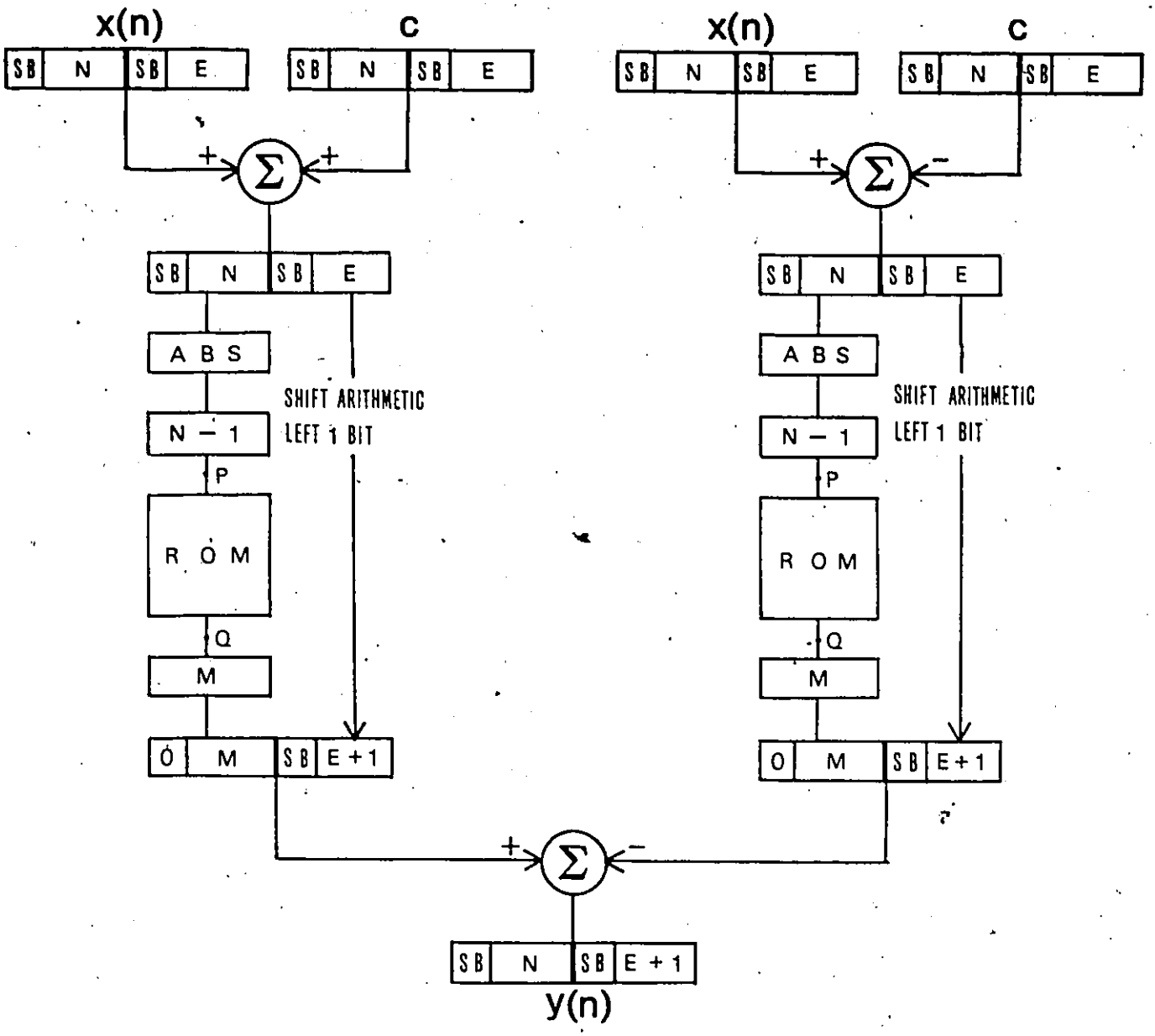


Figure 5: Block diagram of floating point stored square ROM multiplier.

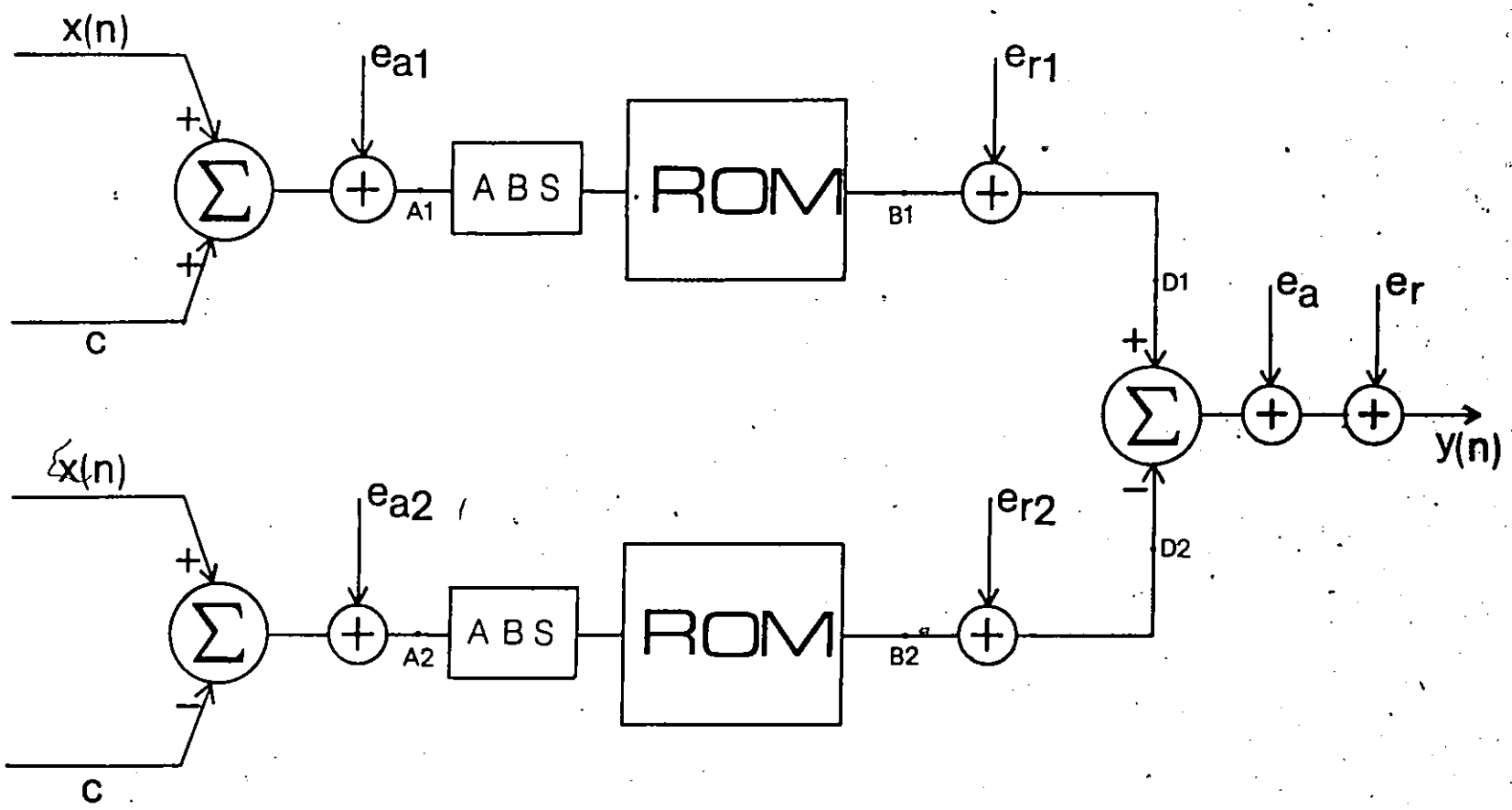


Figure 6: Block diagram of the multiplication with roundoff errors, for floating point arithmetic.

At nodes A1, A2 :

$$a_i(n) = (x(n)\pm c)\,[1 + \varepsilon_{ai}(n)] \;;\; i=1,2 \qquad (2.17)$$

At nodes B1, B2 :

$$b_i(n) = \frac{(x(n)\pm c)^2}{4}\,[1 + \varepsilon_{ai}(n)]^2$$

$$\approx \frac{(x(n)\pm c)^2}{4}\,[1 + 2\,\varepsilon_{ai}(n)] \;;\; i=1,2 \qquad (2.18)$$

At nodes D1, D2 :

$$d_i(n) = \frac{(x(n)\pm c)^2}{4}\,[1 + 2\,\varepsilon_{ai}(n)]\,[1 + \varepsilon_{ri}(n)]$$

$$\approx \frac{(x(n)\pm c)^2}{4}\,[1 + 2\,\varepsilon_{ai}(n) + \varepsilon_{ri}(n)] \;;\; i=1,2 \qquad (2.19)$$

The actual product is :

$$y(n) = \frac{(x(n)+c)^2}{4}[1 + 2\,\varepsilon_{a1}(n) + \varepsilon_{r1}(n) + \varepsilon_a(n) + \varepsilon_r(n)]$$

$$- \frac{(x(n)-c)^2}{4}[1 + 2\,\varepsilon_{a2}(n) + \varepsilon_{r2}(n) + \varepsilon_a(n) + \varepsilon_r(n)] \qquad (2.20)$$

Let $\varepsilon_{Ti}(n) = 2\,\varepsilon_{ai}(n) + \varepsilon_{ri}(n) + \varepsilon_a(n) + \varepsilon_r(n)$; i=1,2

The output error is :

$$e_0(n) = \frac{(x(n)+c)^2}{4}\varepsilon_{T1}(n) - \frac{(x(n)-c)^2}{4}\varepsilon_{T2}(n) \qquad (2.21)$$

The errors ε_{a1} , ε_{a2} and ε_r are bounded by :

$$-2^{-N} \leqslant \varepsilon_{a1}, \varepsilon_{a2}, \varepsilon_r \leqslant 2^{-N} \qquad (2.22)$$

The errors ε_{r1} , ε_{r2} and ε_a are bounded by :

$$-2^{-M} \leqslant \varepsilon_{r1}, \varepsilon_{r2}, \varepsilon_a \leqslant 2^{-M} \qquad (2.23)$$

Since, $\max [\varepsilon_{T1}(n)] = \max [\varepsilon_{T2}(n)]$, so the maximum output error is :

$$\max |e_0| = \max [\varepsilon_{Ti}(n)].\max |P|$$

$$= [3.2^{-N} + 2^{-M+1}].\max |P| \qquad (2.24)$$

and the maximum output error to maximum product ratio is :

$$\frac{\max |e_0|}{\max |P|} = \begin{cases} 3.2^{-N} + 2^{-M+1} & ; \text{ for } M > N \\ 2.2^{-N} + 2^{-M+1} & ; \text{ for } M \leqslant N \end{cases} \qquad (2.25)$$

Let $\frac{(x(n)+c)^2}{4} = \alpha(n)$ and $\frac{(x(n)-c)^2}{4} = \beta(n)$.

Assuming that $\alpha(n)$ and $\beta(n)$ are uncorrelated with $\varepsilon_{T1}(n)$ and $\varepsilon_{T2}(n)$. Assume also that x(n) and c are uniformly distributed between -q and q, it can be shown [7] that (see Appendix A) the mean and mean square of $\alpha(n)$ and $\beta(n)$ are :

$$\overline{\alpha} = \frac{q^2}{6} \quad ; \quad \overline{\beta} = \frac{q^2}{6}$$

$$\overline{\alpha^2} = \frac{q^4}{15} \quad ; \quad \overline{\beta^2} = \frac{q^4}{15} \qquad (2.26)$$

From (2.21) it follows that :

$$\sigma^2_{e_0} = \sigma^2_{\varepsilon_{T1}} \overline{a^2} + \sigma^2_{\varepsilon_{T2}} \overline{\beta^2} \qquad (2.27)$$

Since $\sigma^2_{\varepsilon_{T1}} = \sigma^2_{\varepsilon_{T2}}$, so

$$\sigma^2_{e_0} = \sigma^2_{\varepsilon_{T1}} (\overline{a^2} + \overline{\beta^2}) \qquad (2.28)$$

With

$$P(n) = a(n) - \beta(n) \qquad (2.29)$$

and

$$\overline{P} = \overline{a} - \overline{\beta} = 0 \qquad (2.30)$$

$$\overline{P^2} = \sigma^2_P = \overline{a^2} + \overline{\beta^2} - 2\overline{a\beta} = \frac{q^4}{9} \qquad (2.31)$$

Then (2.28) becomes :

$$\sigma^2_{e_0} = \sigma^2_{\varepsilon_{T1}} (\sigma^2_P + 2\overline{a\beta})$$

$$= \sigma^2_{\varepsilon_{T1}} \cdot \sigma^2_P (1 + \frac{2\overline{a\beta}}{\sigma^2_P}) \qquad (2.32)$$

From (2.26), (2.31) and (2.32), the output noise to product power ratio is :

$$\frac{\sigma^2_{e_0}}{\sigma^2_P} = \begin{cases} 2.2^{-2N} + \frac{4}{5} 2^{-2M} & ; \text{ for } M > N \\ \frac{8}{5} 2^{-2N} + \frac{4}{5} 2^{-2M} & ; \text{ for } M \leqslant N \end{cases} \qquad (2.33)$$

With direct multiplication, the maximum output error to maximum product ratio will be :

$$\frac{\max |e_{0m}|}{\max |P|} = 2^{-N} \qquad (2.34)$$

and the output noise to product power ratio is :

$$\frac{\sigma^2_{e_{0m}}}{\sigma^2_P} = \frac{1}{3} 2^{-2N} \qquad (2.35)$$

The maximum output error to maximum product ratio and the output noise to product power ratio are plotted in Fig. 7 and Fig. 8 and compared with the errors obtained in direct multiplication.

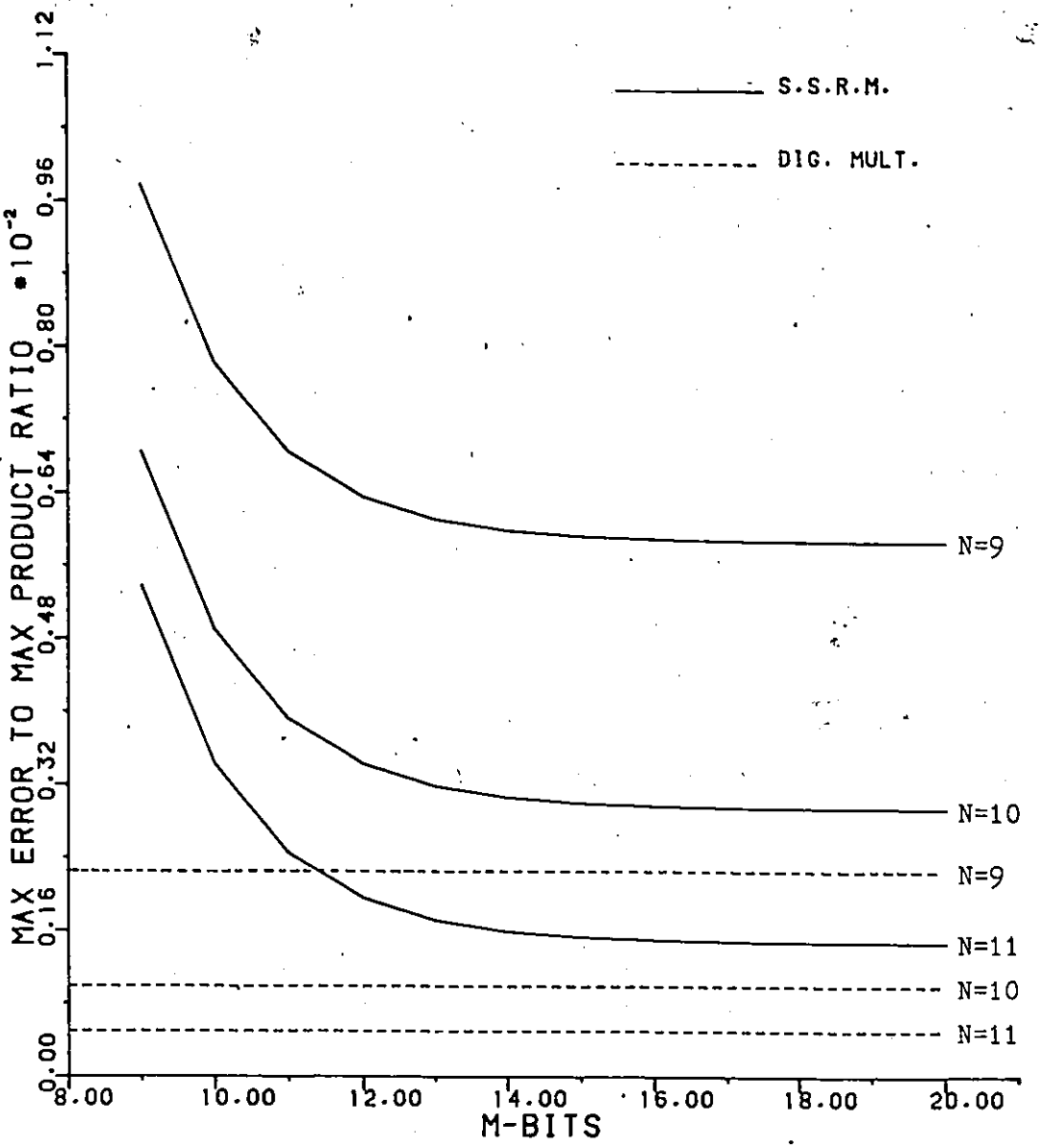


Figure 7: Maximum output error to maximum product ratio of floating point plotted as a function M for various N.

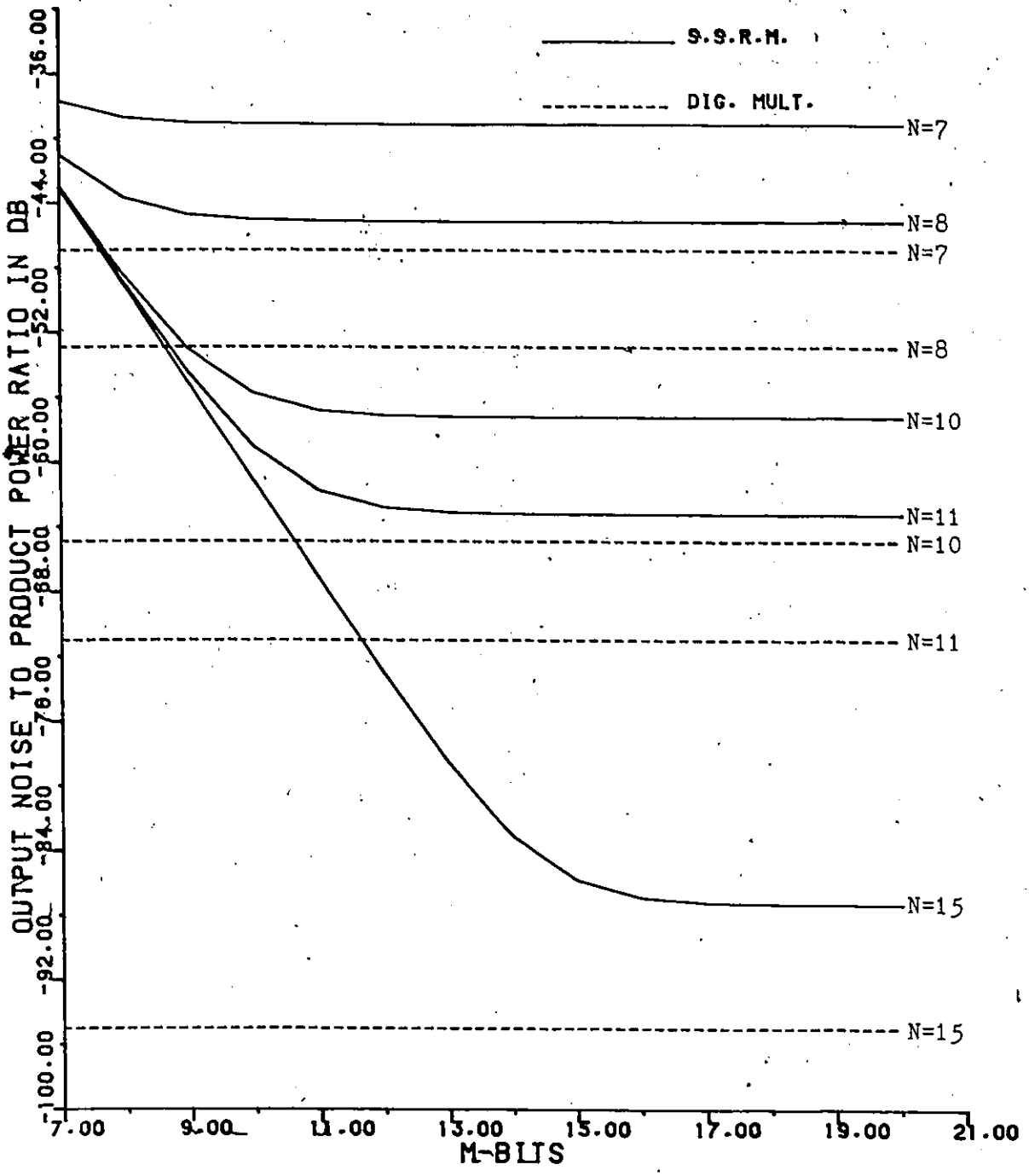


Figure 8: Output noise to product power ratio of floating point plotted as a function M for various N.

2.5 QUANTIZATION EFFECT ON THE FIXED POINT SSRM

In the previous sections, we assumed that the input signal and the coefficient are quantized, here we will take into account the quantization errors and compare these with errors obtained in direct multiplication. The analysis is based on the equations derived in section 2.3.

In the following discussion, let $x(n)$ and c be the continuous input signal samples and coefficient respectively and $x_q(n)$ and c_q denote the quantized values.

With input signal and coeficient quantized, the actual product (2.4) becomes :

$$y(n) = \frac{(x_q(n)+c_q)^2}{4} - \frac{(x_q(n)-c_q)^2}{4}$$

$$+ \varepsilon_{r1}(n) - \varepsilon_{r2}(n) + \varepsilon_{r3}(n) \qquad (2.36)$$

but $x_q(n) = x(n) + \varepsilon_x(n)$ and $c_q = c + \varepsilon_c$

where $\varepsilon_x(n)$ and ε_c are the quantization errors. Then,

$$y(n)=x(n).c+x(n).\varepsilon_c+c.\varepsilon_x(n)+\varepsilon_x(n).\varepsilon_c+\varepsilon_{r1}(n)-\varepsilon_{r2}(n)+\varepsilon_{r3}(n) \qquad (2.37)$$

The error introduced due to input signal and coefficient quantization is :

$$e_q(n) = x(n).\varepsilon_c + c.\varepsilon_x(n) + \varepsilon_x(n).\varepsilon_c \qquad (2.38)$$

In direct multiplication, the product of $x_q(n).c_q$ is :

$$y_m(n) = (x(n) + \varepsilon_x(n))\ (c + \varepsilon_c) + \varepsilon_{rm}$$

$$= x(n).c + x(n).\varepsilon_c + c.\varepsilon_x(n) + \varepsilon_x(n).\varepsilon_c + \varepsilon_{rm} \qquad (2.39)$$

So, the error due to input signal and coefficient on fixed point SSRM is equal to that obtained with direct multiplication.

2.6 QUANTIZATION EFFECT ON THE FLOATING POINT SSRM

A similar analysis can be carried out for floating point arithmetic. Fig. 9 shows the block diagram of floating point SSRM with quantization and roundoff noise.

At node A1, A2 :

$$a_i(n) = [x(n)(1 + \varepsilon_x(n)) \pm c\ (1 + \varepsilon_c)]\ [1 + \varepsilon_{ai}(n)]$$

$$= [x_q(n) \pm c_q]\ [1 + \varepsilon_{ai}(n)]\ ;\ i=1,2 \qquad (2.40)$$

At node D1, D2 :

$$d_i(n) \approx \frac{(x_q(n) \pm c_q)^2}{4}\ [1 + 2\ \varepsilon_{a1}(n) + \varepsilon_{ri}(n)]\ ;\ i=1,2 \qquad (2.41)$$

The actual product is :

$$y(n) = \frac{(x_q(n)+c_q)^2}{4}\ [1 + \varepsilon_{T1}(n)] - \frac{(x_q(n)-c_q)^2}{4}\ [1 + \varepsilon_{T2}(n)] \qquad (2.42)$$

with $\varepsilon_{Ti}(n) = 2\ \varepsilon_{ai}(n) + \varepsilon_{ri}(n) + \varepsilon_a(n) + \varepsilon_r(n)\ ;\ i=1,2$

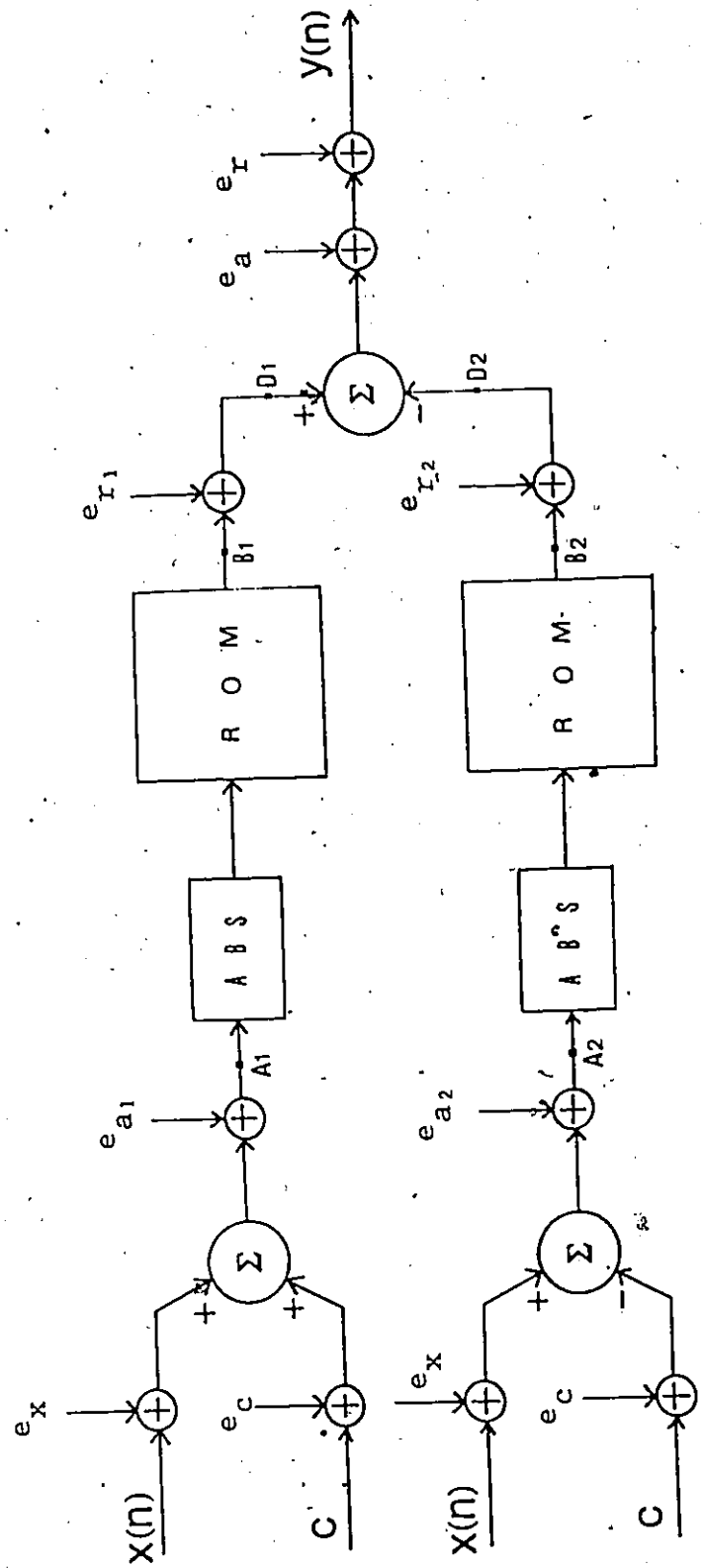


Figure 9: Block diagram of floating point SSRM with quantization and roundoff errors.

Equation (2.42) can be rewritten as :

$$y(n) = x(n).c\ (1 + \varepsilon_x(n))(1 + \varepsilon_c) + \frac{(x_q(n)+c_q)^2}{4}\ \varepsilon_{T1}(n) - \frac{(x_q(n)-c_q)^2}{4}\ \varepsilon_{T2}(n) \qquad (2.43)$$

Since, only the input signal and coefficient quantization effect that is to be observed, so it is necessary to assume that there is no error introduced in addition or subtraction process and assume the stored squares have wordlength long enough so there is no rounding error and only the final product is rounded, then

$$\varepsilon_{T1}(n) = \varepsilon_{T2}(n) = \varepsilon_r(n) \qquad (2.44)$$

Hence,

$$y(n) = x(n).c\ (1 + \varepsilon_x(n))(1 + \varepsilon_c) + \varepsilon_r(n)\left[\frac{(x_q(n)+c_q)^2}{4} - \frac{(x_q(n)-c_q)^2}{4}\right]$$

$$= x(n).c\ (1+\varepsilon_x(n))(1+\varepsilon_c) + \varepsilon_r(n)[x(n).c\ (1+\varepsilon_x(n))(1+\varepsilon_c)]$$

$$= x(n).c\ (1+\varepsilon_x(n))(1+\varepsilon_c)(1+\varepsilon_r(n))$$

$$\approx x(n).c\ (1 + \varepsilon_x(n) + \varepsilon_c + \varepsilon_r(n)) \qquad (2.45)$$

So, the error resulted due to input signal and coefficient quantization is :

$$e_q = x(n).c\ [\varepsilon_x(n) + \varepsilon_c] \qquad (2.46)$$

In direct multiplication, the actual product is :

$$y_m(n) = x_q(n).c_q\ (1 + \varepsilon_r(n))$$

$$= x(n)\ (1 + \varepsilon_r(n)).c\ (1 + \varepsilon_c)\ (1 + \varepsilon_r(n))$$

$$x(n).c\ (1 + \varepsilon_x(n) + \varepsilon_c + \varepsilon_r(n)) \qquad (2.47)$$

which has the same effect on quantization noise as in the stored square ROM multiplier.

2.7 COMPARISON BETWEEN FIXED POINT AND FLOATING POINT ERROR

The error analysis on the stored square ROM multiplier has been evaluated for fixed point and for floating point arithmetic. In this section those two errors will be compared.

If we let $M \longrightarrow \infty$, then

For fixed point :

$$\frac{\max\ |e_o|}{\max\ |P|} = \frac{1}{2}\,2^{-N} \qquad (2.48)$$

$$\frac{\sigma^2_{e_0}}{\sigma^2_P} = \frac{3}{4}\,2^{-2N} \qquad (2.49)$$

which has the same maximum output error to maximum product ratio and the output noise to product power ratio as in digital multiplier. In general, M>N+3 for the fixed point SSRM errors to be comparable to the one obtainable by direct multiplication.

For floating point :

$$\frac{\max |e_0|}{\max |P|} = 3.2^{-N} \qquad (2.50)$$

$$\frac{\sigma^2_{e_0}}{\sigma^2_P} = 2.2^{-2N} \qquad (2.51)$$

where the maximum output error to maximum product ratio is three times larger than that of digital multiplier, but in simulation it gives less than half of that which indicating that all errors due to addition, subtraction and rounding are very unlikely to have maximum possible error occuring at the same time. The output noise to product power ratio of floating point SSRM is twice that of digital multiplier.

The ROM storage requirements are :

a) $2[2^{N+1} \times M]$ bits for fixed point

b) $2[2^{N-1} \times M]$ bits for floating point (N=mantissa-wordlength)

where $M_{max} = 2(N+1)$ for fixed point and $M_{max} = 2(N-1)$ for floating point.

From Fig. 10 to Fig. 13, it is found that in order the fixed point and the floating point SSRM have comparable output noise to product power ratio, the latter requires one bit more in the mantissa. As for equal input signal (N) the required ROM storage for floating point is one fourth of that required for fixed point, then with one bit more the

floating point storage requirement is still one half of that of fixed point. Table 1 summarizes the ROM storage requirement for fixed point and for floating point arithmetic.

Since, the floating point arithmetic does not give a comparable accuracy to that of digital multiplier and also due to the fact that fixed point is the most widely used arithmetic mode for real time application, therefore, the rest of this thesis will concentrate on fixed point arithmetic only.

TABLE 1

THE ROM STORAGE REQUIREMENT FOR FIXED POINT AND FLOATING POINT SSRM WITH M=N+4

N	STORAGE SIZE (BITS)	
	FIXED POINT	FLOATING POINT
7	5.5K	1.375K
8	12K	3K
9	26K	6.5K
10	56K	14K
11	120K	30K
12	256K	64K
13	544K	136K
14	1.152M	288K
15	2.432M	608K

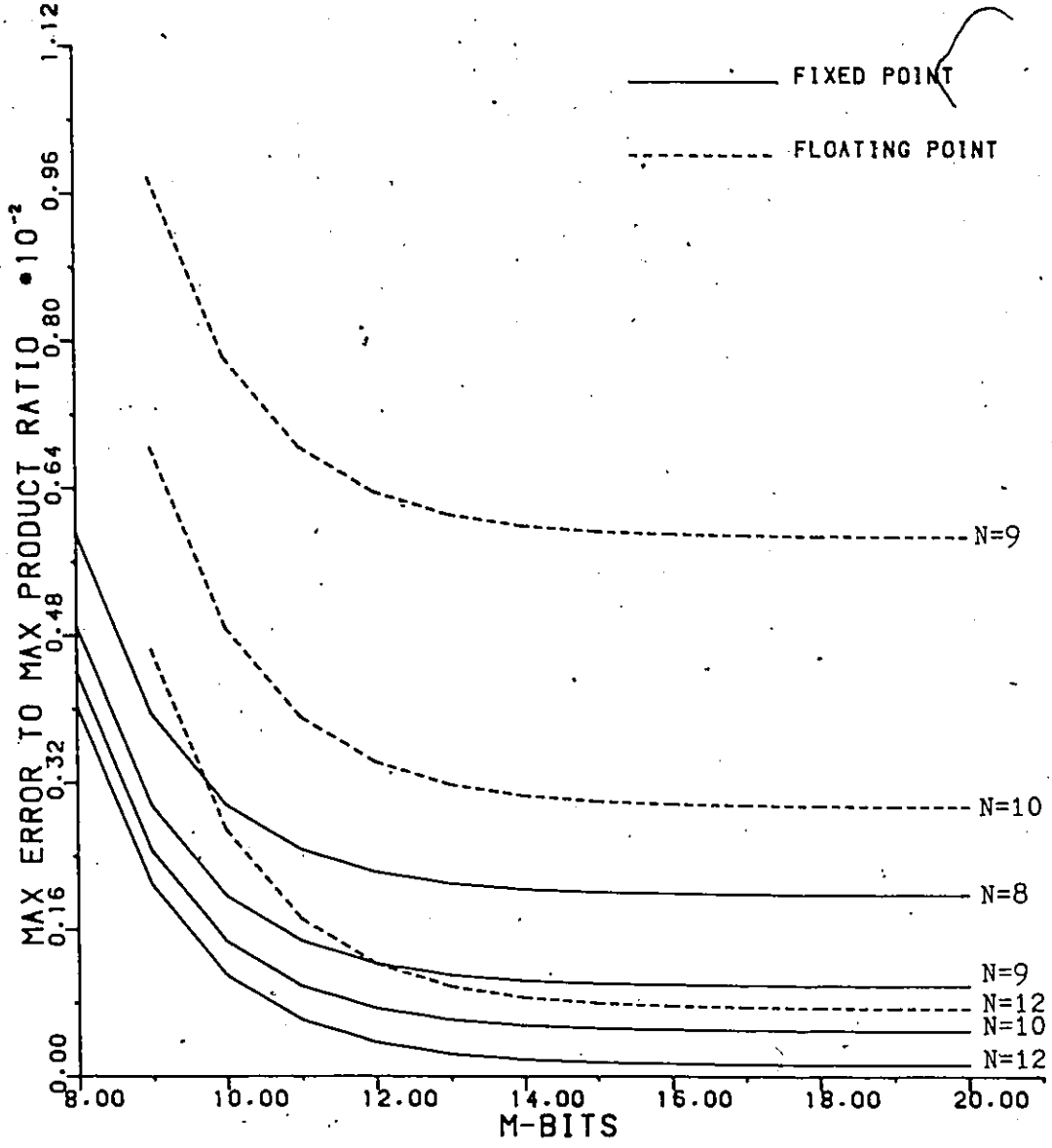


Figure 10: Max output error to max product ratio for fixed point and floating point plotted as a function M for various N.

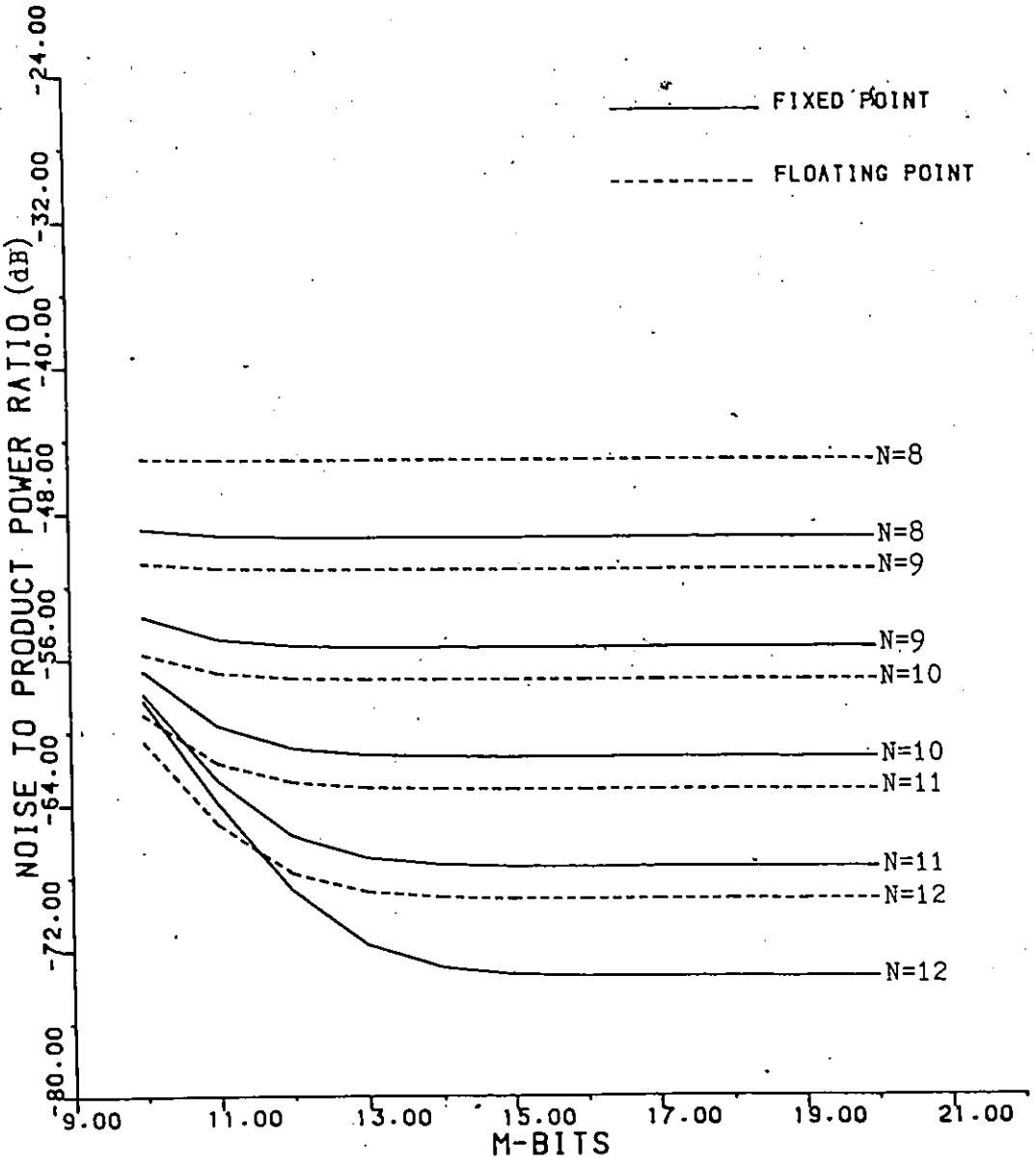


Figure 11: Output noise to product power ratio for fixed point and floating point plotted as a function M for various N.

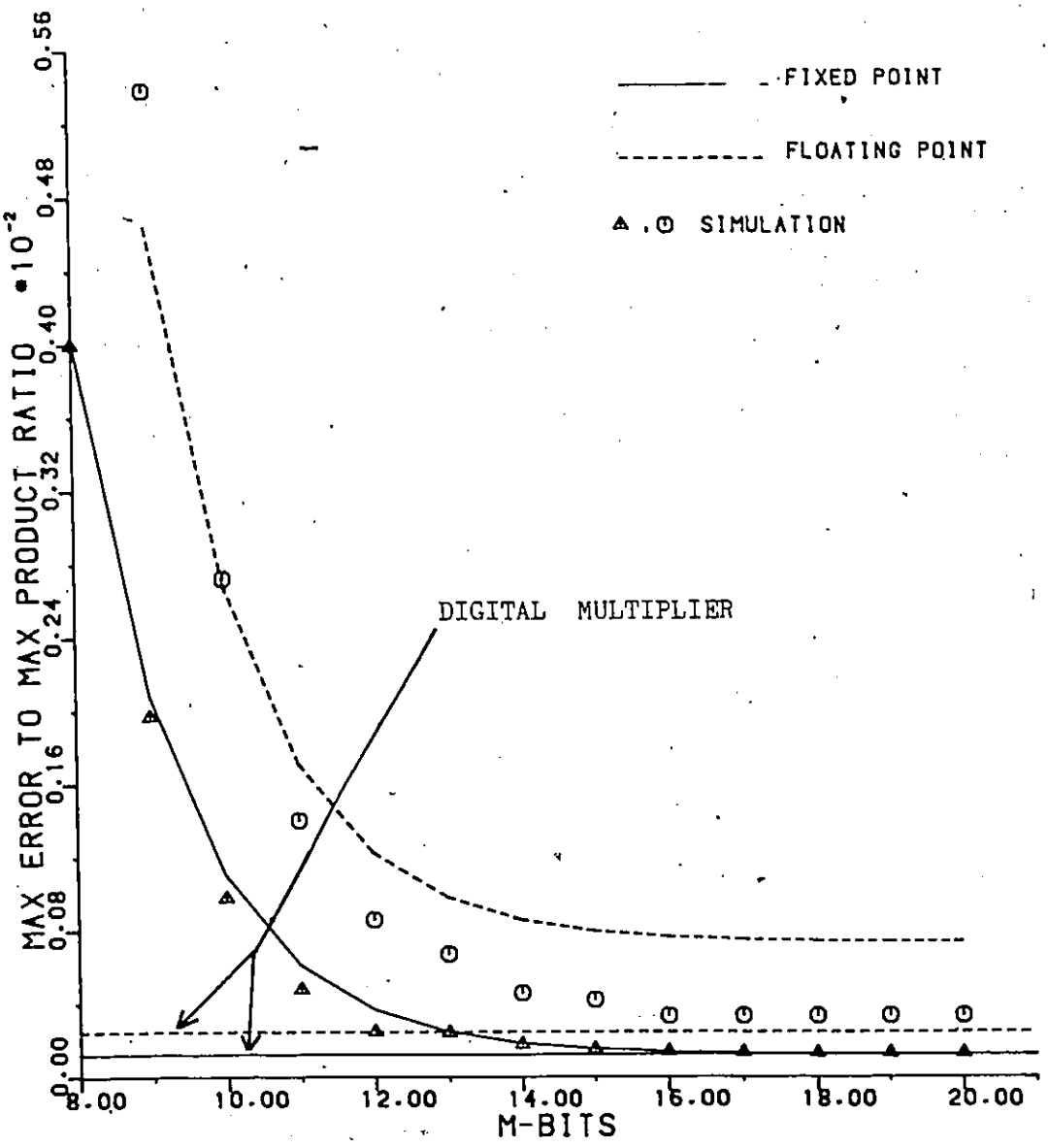


Figure 12: Maximum output error to maximum product ratio with N=12.

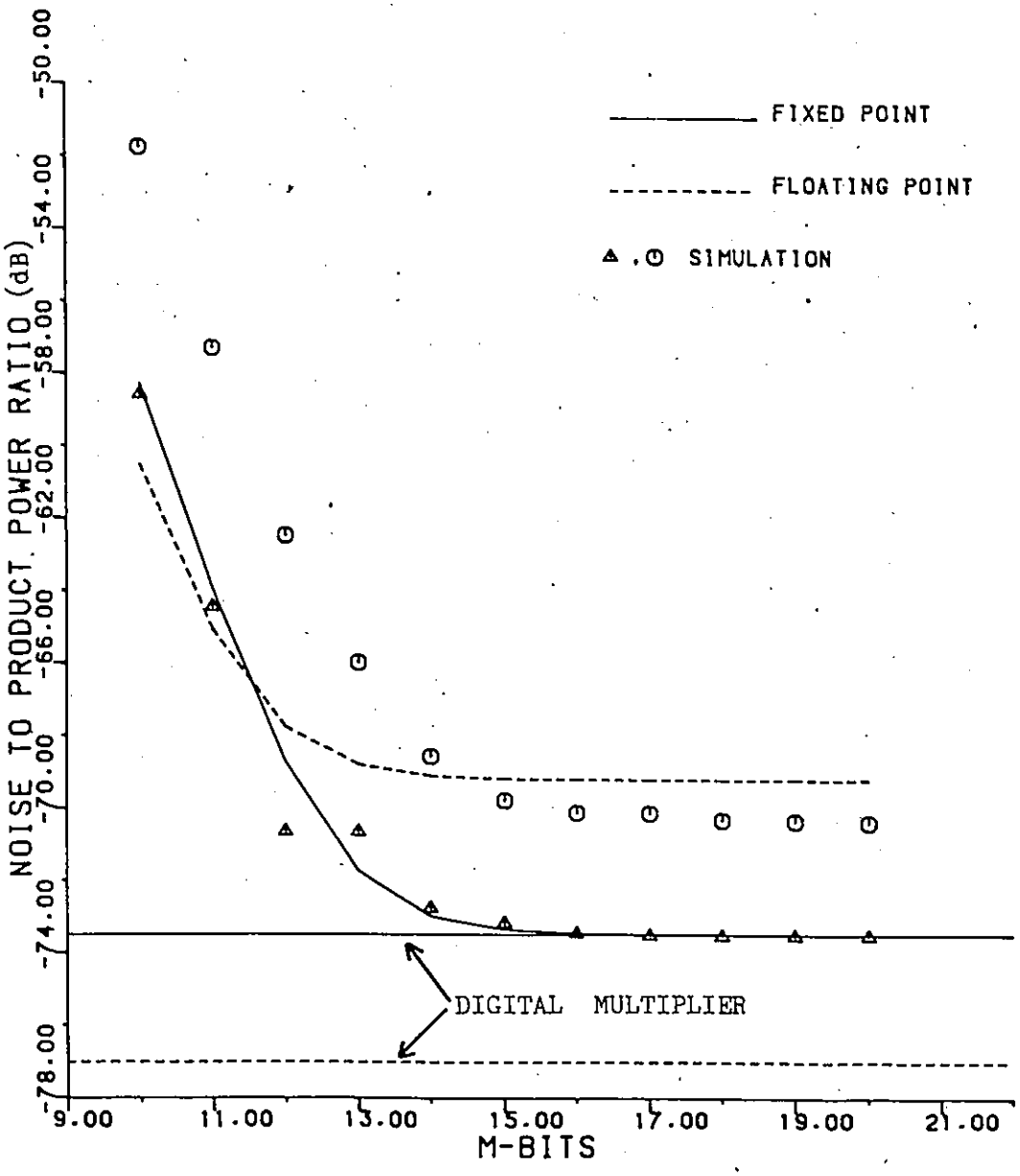


Figure 13: Output noise to product power ratio with N=12.

2.8 REDUCING THE SIZE OF ROM STORAGE REQUIREMENT

The stored squares technique presents a solution to the replacement of binary multipliers with ROMs. An analysis of how many bits are required to be stored shows that for address wordlength of up to 11 bits this leads to numbers presently realizable at reasonable cost.

For wordlengths of 12 bits and larger a possible reduction of the numbers of bits stored is mandatory, and this can be accomplished by using a variation of the split-address technique of stored product digital filtering [8][9].

For the stored square ROM multiplier this is done by replacing the ROM elements of figures 1 and 5 by the "Reduced ROM Squaring Circuit" of Fig. 14.

Consider an address wordlength of K bits, split into two parts of K_1 least significant bits and $K_2=K-K_1$ most significant bits. The magnitude of the address can be expressed as :

$$|K| = \sum_{i=1}^{K_2} b_i 2^{-i} + \sum_{i=K_2+1}^{K} b_i 2^{-i} \qquad (2.52)$$

where $b_i = [0,1]$ (binary number).

The square of (2.52) will be :

$$|K|^2 = \left(\sum_{i=1}^{K_2} b_i 2^{-i}\right)^2 + \left(\sum_{i=K_2+1}^{K} b_i 2^{-i}\right)^2 + 2\left(\sum_{i=1}^{K_2} b_i 2^{-i}\right)\left(\sum_{i=K_2+1}^{K} b_i 2^{-i}\right) \qquad (2.53)$$

Rewriting (2.53), we have :

$$|K|^2 = \left(\sum_{i=1}^{K_2} b_i 2^{-i}\right)^2 + \left(\sum_{i=K_2+1}^{K} b_i 2^{-i}\right)^2 + 2^{-(K_2-1)} \left(\sum_{j=1}^{K_1} b_{j+K_2} 2^{-j} \sum_{i=1}^{K_2} b_i 2^{-i}\right) \qquad (2.54)$$

Fig. 14 shows the realization of (2.54) in block diagram form.

In general, with split address technique the ROM storage requirement is :

$$2^{K_2} \times 2K_2 + 2^{K_1} \times 2K_1 + 2(2^{K'} \times 2K') \text{ bits per squaring circuit.}$$

For minimal storage requirement, it is found that K_1 and K_2 have to be of the same length or as close as possible. With $K_1 = K_2$, the storage requirement will be :

$$2[2^{K/2} \times K] + 4[2^{K/2+1}(\tfrac{K}{2} + 1)] \text{ bits per squaring circuit.}$$

A fixed point Stored Square ROM Multiplier with N=11 and M=24, if implemented using split address technique will require a ROM storage of :

$$2\ 2[2^{12/2} \times 12] + 4[2^7 \times 7] = 10 \text{ kbits.}$$

Comparing with direct implementation which requires 192 Kbits, there is a reduction of almost 20 times and further with N=15 and M=32 we would have a reduction of almost 79

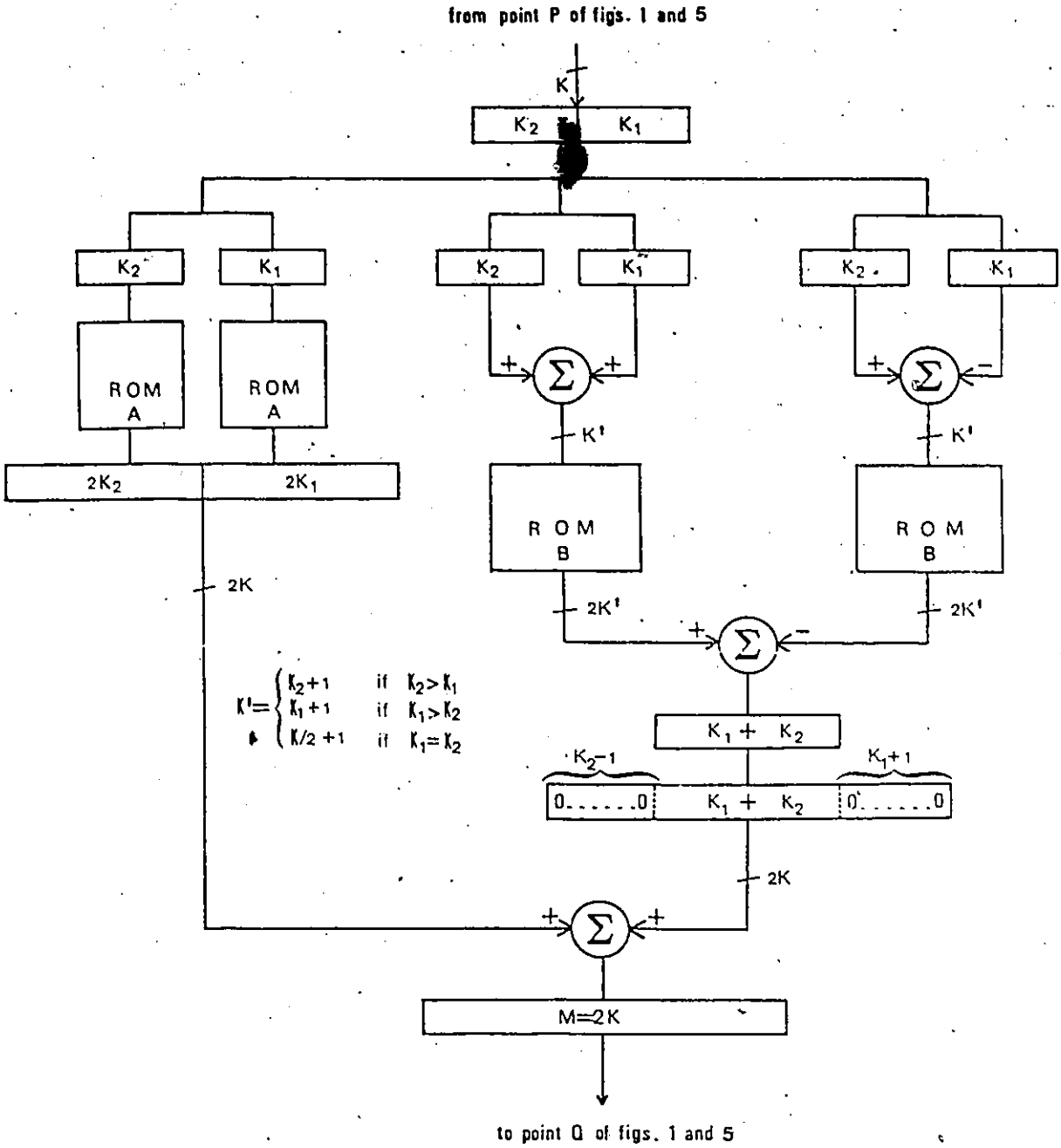


Figure 14: Block diagram of "Reduced ROM Squaring Circuit". ROMs A are squaring ROMs. ROMs B are squaring and divide by 4 ROMs.

times. Table 2 shows the comparison of the ROM's size requirement between direct and split address implementation for fixed point with $M=M_{max}$.

TABLE 2

COMPARISON OF ROM STORAGE REQUIREMENT

N	STORAGE SIZE (BITS)	
	DIRECT IMPL.	SPLIT ADDRESS IMPL.
7	8K	1.75K
8	18K	3.875K
9	40K	4.5K
10	88K	9.125K
11	192K	10K
12	416K	21K
13	896K	23K
14	1.92M	47.5K
15	4M	52K

In terms of cost, Table 3 shows the cost comparison between direct and split address implementation for wordlengths of 8, 12 and 16 bits. With direct implementation the memory cost takes 48%, 94% and 99.6% of the total cost respectively for 8, 12 and 16 bits. On the other hand, the split address implementation respectively takes 30%, 35% and 58% of the total cost. From the memory cost percentage information it reflects that as the price of ROMs is anticipated further to go down, the implementation cost of SSRM either with direct or split address implementation will be reduced significantly in the future.

TABLE 3

COST COMPARISON

WORDLENGTH (BITS)	ESTIMATED COST (CAN $)		MEMORY COST PERCENTAGE	
	DIRECT IMPL.	SPLIT ADDR. IMPL.	DIRECT IMPL.	SPLIT ADDR. IMPL.
8	$47	$50	48%	30%
12	$608	$85	94%	35%
16	$12550	$165	99.6%	58%

2.9 HARDWARE REALIZATION AND SPEED CONSIDERATIONS

So far the SSRM has been evaluated on its performance as a multiplier. In this section aspects of the realization and the speed will be discussed.

The hardware implementation of SSRM essentially consists of adders, absolute value circuits and ROMs. As adders and ROMs are readily available in IC (Integrated Circuit) form, it will be simple to realize custom built implementations.

The absolute value circuit can be realized based on the truth table [6] given in Fig. 15. A is the quantity for which the absolute value S is desired, SB is the sign bit and C_{n-1} is the carry from the previous stage.
In mathematical form the output for the n^{th} bit is

$$S_n = A_n \oplus SB \oplus C_{n-1} \qquad (2.55)$$

and the carry C_n is

$$C_n = \overline{A}_n \cdot C_{n-1} \qquad (2.56)$$

To increase the speed [6], the time required for carry propagation can be eliminated by obtaining the carry value needed at each stage as :

$$C_{n-1} = \overline{A}_{n-1} \cdot \overline{A}_{n-2} \cdots \overline{A}_1 \cdot \overline{A}_0 \cdot SB \qquad (2.57)$$

" $\oplus$ " and " $\cdot$ " are the notations for logical Exclusive OR and logical AND respectively.

Fig. 16 shows the hardware realization of the absolute value circuit for 8 bits with fast carry. The propagation delay is in the order of three logic gate delays.

A_n	SB	C_{n-1}	S_n	C_n
0	0	0	0	0
0	0	1	-	-
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	-	-
1	1	0	0	0
1	1	1	1	0

- DON'T CARE

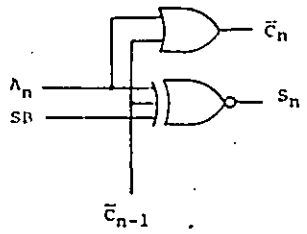


Figure 15: Truth table and circuit for bit n of absolute value circuit.

The function of the absolute value circuit is to detect any negative address number coming to the ROM and change it

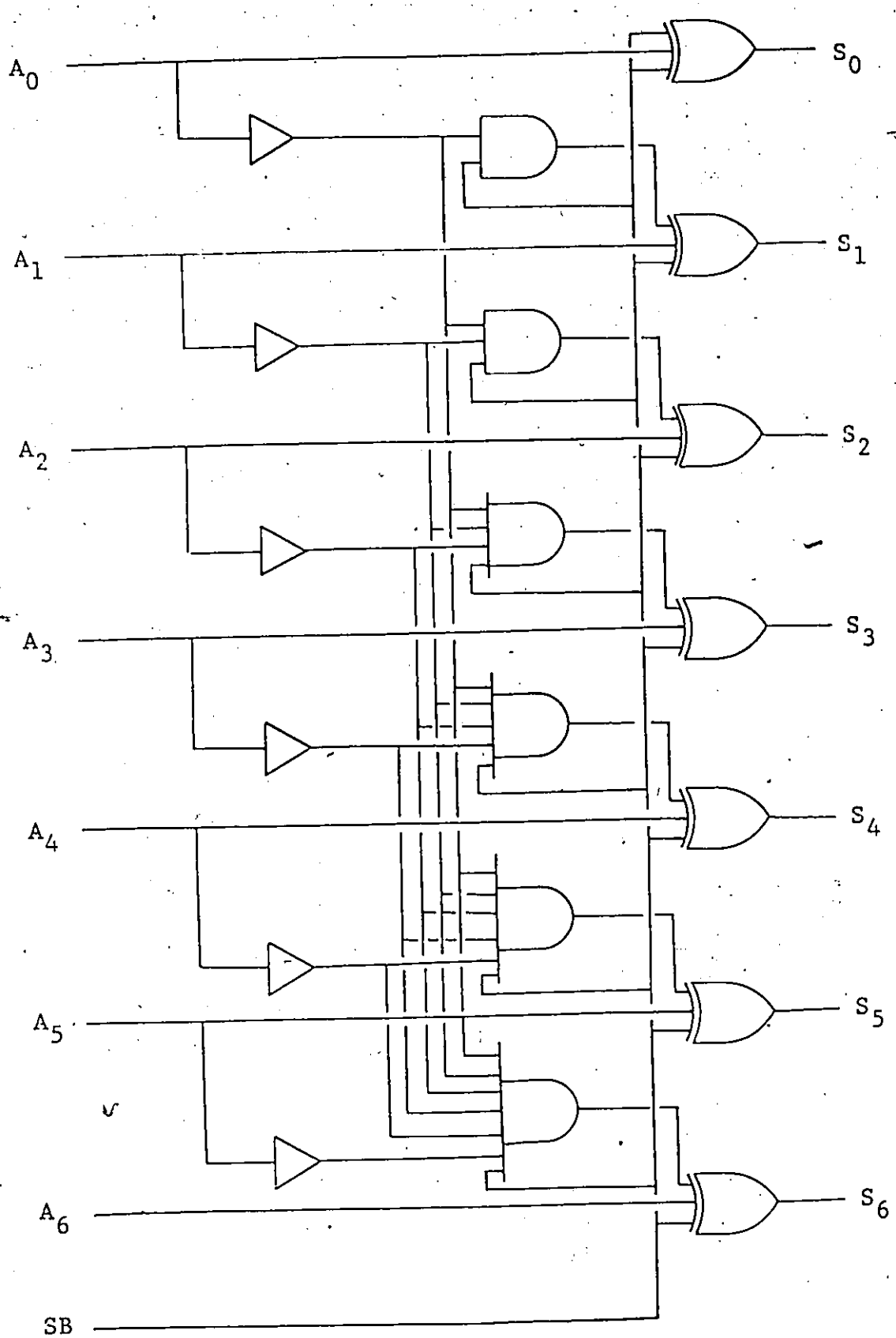


Figure 16: Circuit diagram of 8 bit absolute value circuit with fast carry.

into positive address number, hence, reducing the required ROM storage size by one-half. However, for short input signal and coefficient wordlengths, this circuit can be omitted, since, the cost of the ROMs are relatively the same for different input wordlength with short addresses, so it is not of much benefit in cost saving. In general, the decision on using the absolute value circuit lies on the matter of how long the address is, the availability and the cost of the ROMs.

The multiplication throughput rate of SSRM is very dependent on the architecture of the realization. Here three types of architecture will be considered, namely :

a) Sequential architecture.

b) Parallel architecture.

c) Pipelined architecture.

A. Sequential architecture

Since the ROM contents are equal they can be time-shared and this forms the basis for the sequential architecture shown in Fig. 17. There is only one squaring ROM used. The ADD/SUB circuit will function as an adder or a subtractor depending on the control signal (CS). When it is in the addition mode the ROM's output data will be subtracted by zero and the result will be saved in the latch. In the subtraction mode the ROM's output data will be subtracted by the previous output saved in the latch, hence, each multiplication requires two sequences of operation.

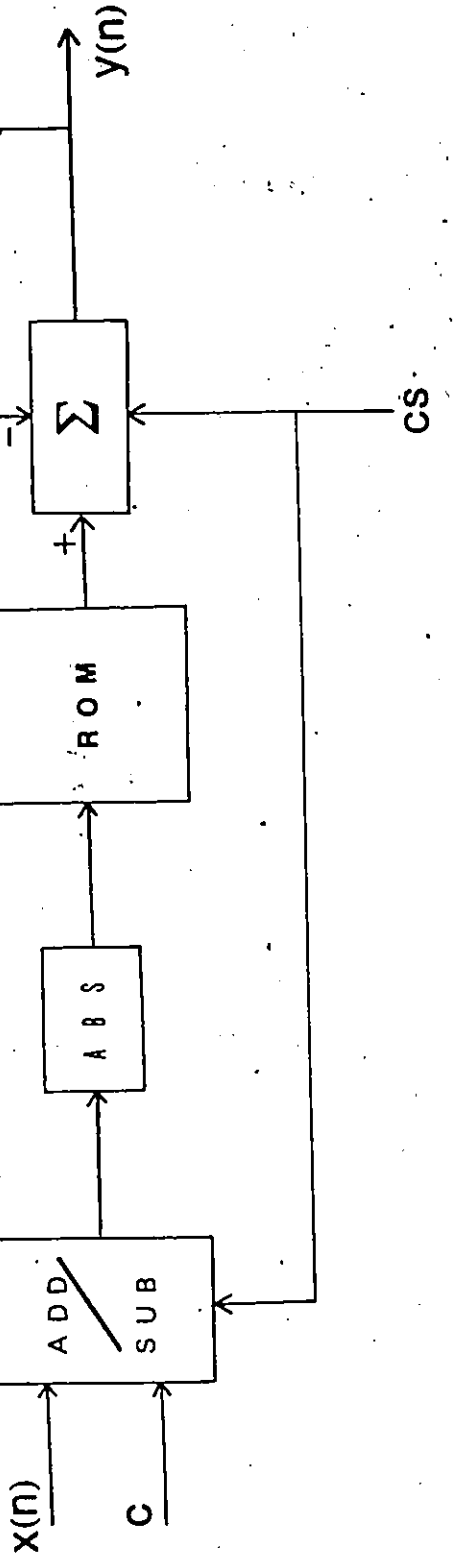


Figure 17: Block diagram of sequential SSRM architecture.

The multiplication speed of this architecture can be expressed as :

$$T_{mul} = 2[T_{A1} + T_{ABS} + T_{SQ} + T_{A2}] \qquad (2.58)$$

where, T_{A1} - time required to add or subtract the input signal with the coefficient.

T_{ABS} - propagation delay of the absolute value circuit.

T_{SQ} - time required to access the stored square in ROM.

T_{A2} - time required to subtract the squared difference from the squared sum.

T_{mul} - total time required to process one multiplication.

The typical speed of 8x8 multiplication using TTL technology is around 200 ns.

The disadvantages of this architecture as compared to the other two are slower multiplication rate and the requirement of a control signal which makes the design more complex, however, this architecture gives the minimum hardware cost.

B. Parallel architecture

A multiplication rate twice of that obtainable in sequential architecture can be accomplished through the parallel architecture which employs twice the amount of hardware. This configuration does not require a control signal and every operation within the multiplier is done in parallel, hence, it has simpler operation. The time required to process a multiplication with this architecture is :

$$T_{mul} = T_{A1} + T_{ABS} + T_{SQ} + T_{A2} \qquad (2.59)$$

C. Pipelined architecture

A multiplication with throughput rate in the order of 50 ns or better can be obtained with pipelined architecture. This implementation is based on the parallel architecture with the insertion of pipeline registers as shown in Fig. 18. The insertion of these registers (or latches) enables it to process the next input signal and coefficient in the first stage while the previous input signal is being processed in the second stage and the one before that in the third stage and so on (see Fig. 19). Hence, the throughput rate of this architecture is basically determined by the processing time of that particular stage which has the slowest execution time and it is independent of the number of stages.

. P.R - pipeline register

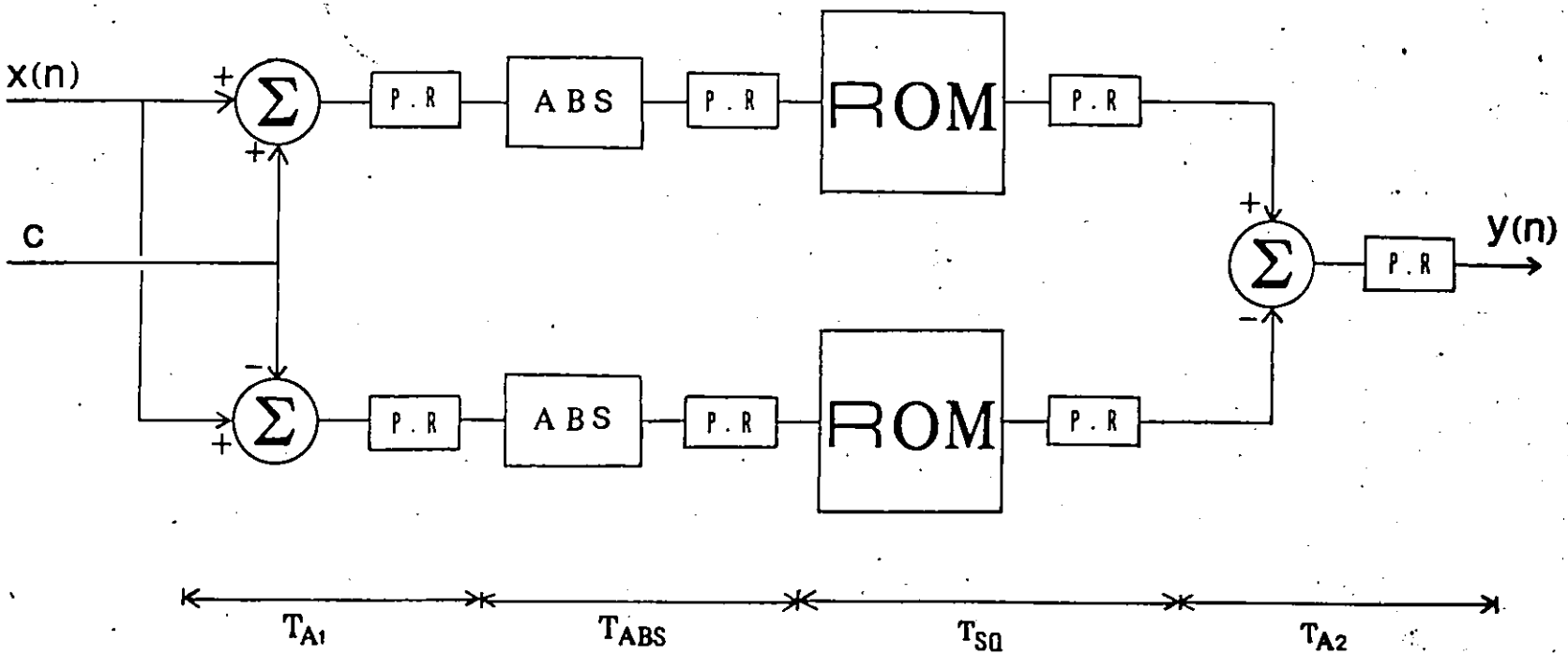


Figure 18: Block diagram of pipelined SSRM architecture.

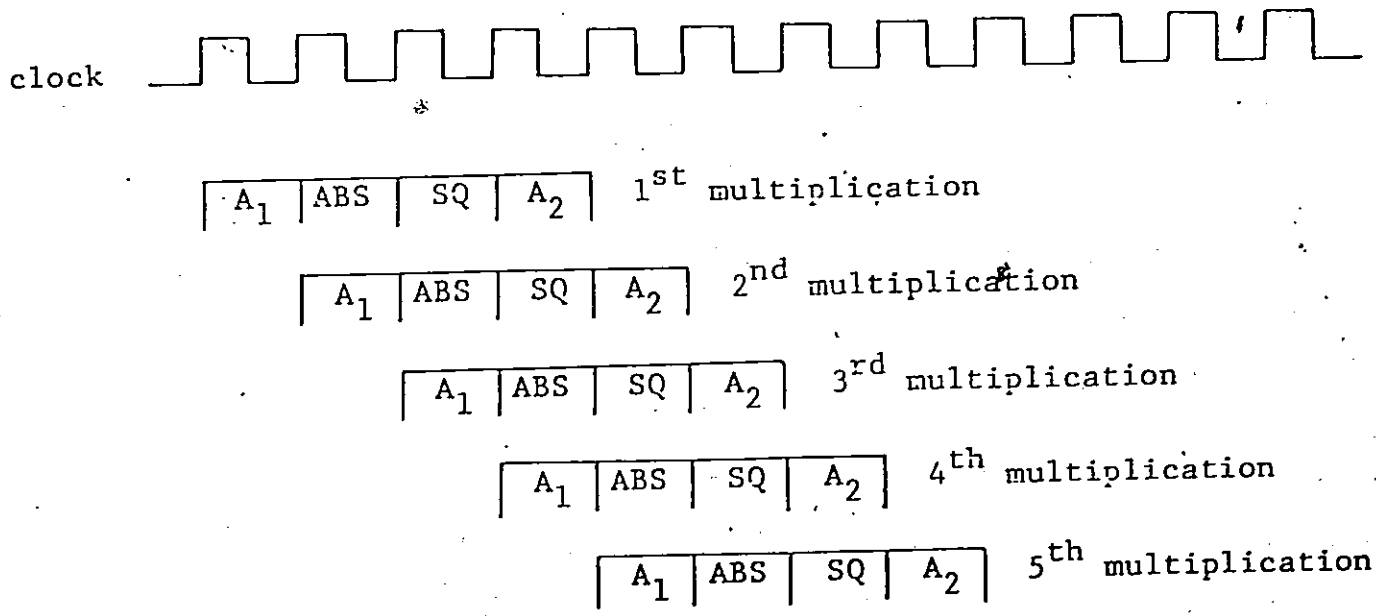


Figure 19: Timing diagram of pipelined SSRM.

Of all components used in the SSRM, the ROMs and the adders relatively posses the slowest execution time. Based on the fact that a 45 ns 16 bit Schottky carry look ahead adder and ROMs with access time in the order of 50 ns are readily available at lower cost, therefore, a pipelined SSRM with throughput rate in the order of 50 ns is warranted. In addition, with some extra cost an addition and ROM access time in the order of 35 ns can be formed.

As previously mentioned, the throughput rate of the pipelined SSRM is independent of the number of stages, however, the delay from output sample to input sample is dependent on the number of stages and refered to as initial delay. Therefore, the initial delay of a pipelined SSRM is equivalent to four sample times. If a split-address technique is used the initial delay will increase to seven sample times. Fig. 20 shows the realization of the pipelined Reduced ROM Squaring Circuit with the overall timing diagram of pipelined SSRM using split-address technique is shown in Fig. 21.

Table 4 gives the tabulation of the multiplication speed for both the parallel and pipelined architecture with input signal and coefficient wordlength up to N=11. For N=12 and longer the multiplication speed of both architectures using split-address technique are tabulated in Table 5. In both cases the stored squares wordlengths (M) are considered as M=N+4 and TTL devices are used.

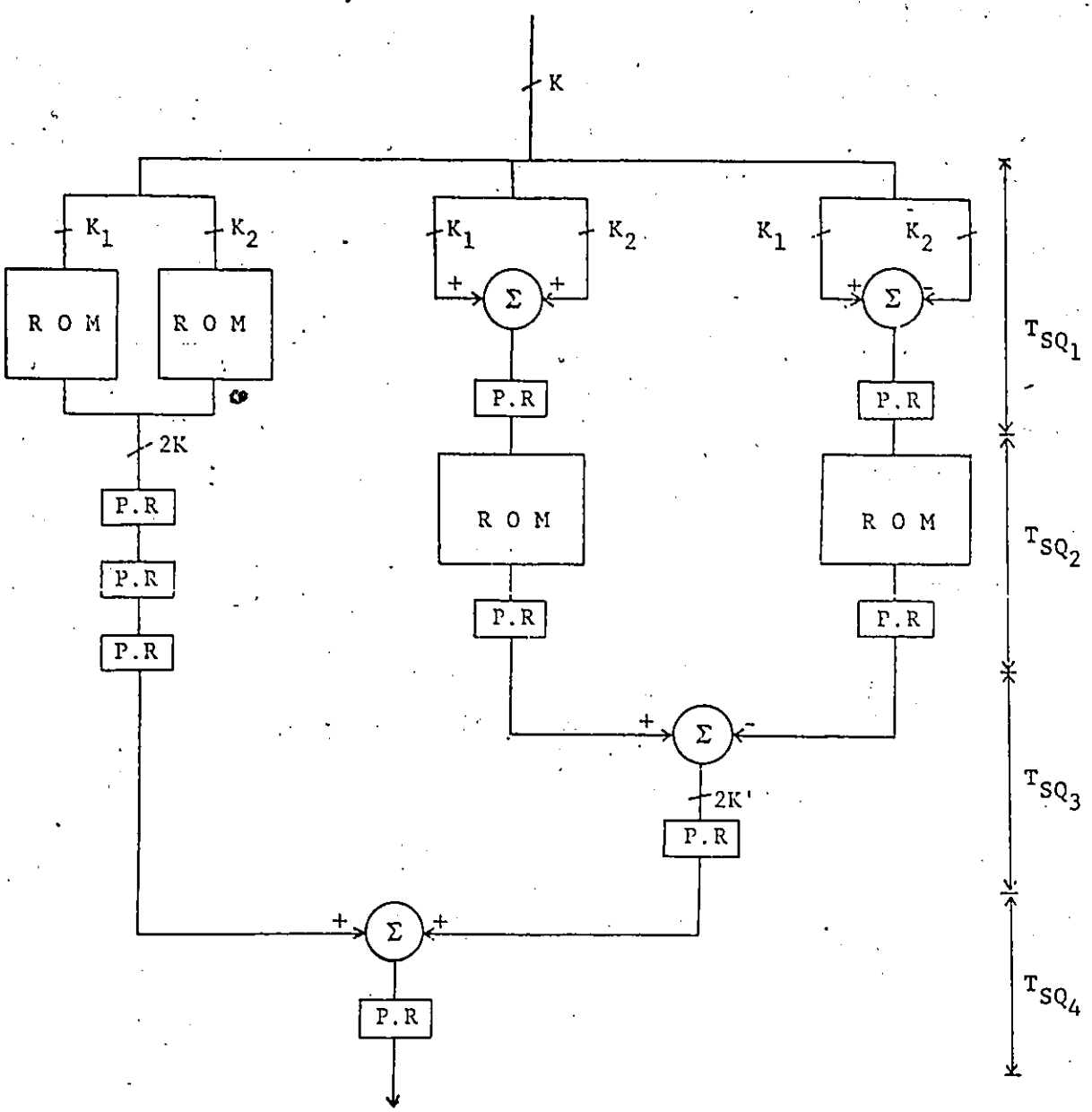


Figure 20: Reduced ROM Squaring Circuit with pipelined architecture.

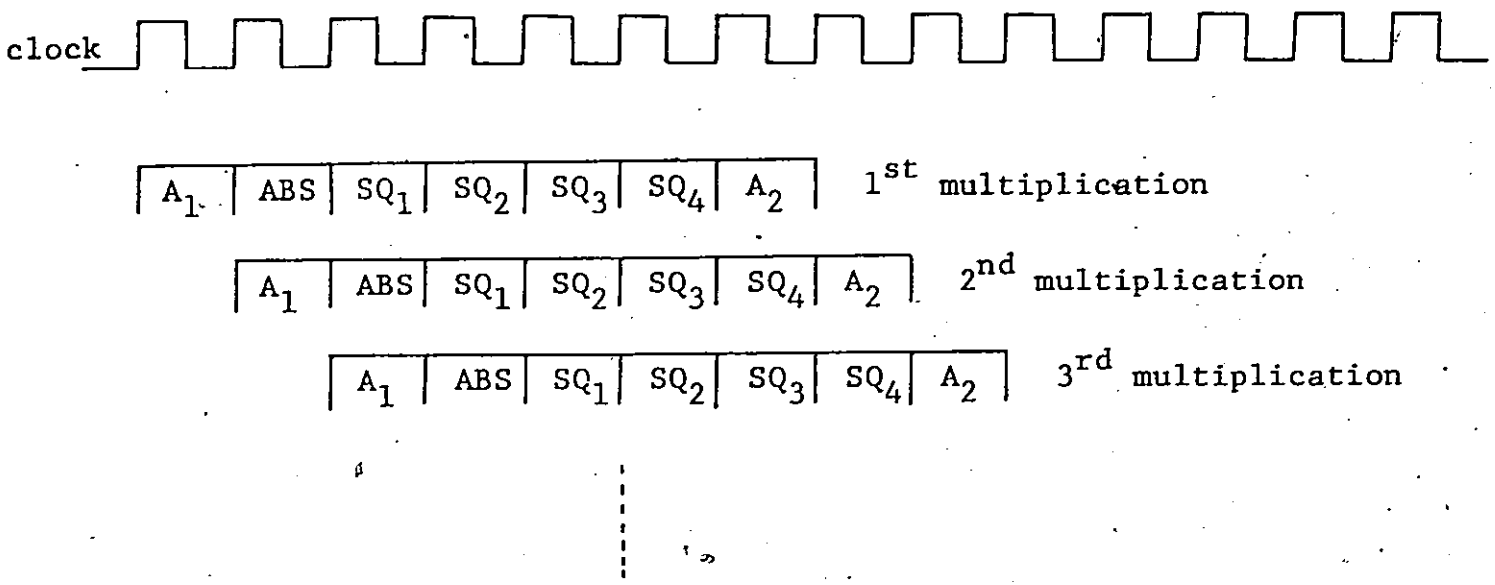


Figure 21: Timing diagram of pipelined SSRM with split-address technique.

TABLE 4

TYPICAL MULTIPLICATION SPEED OF PARALLEL AND PIPELINE ARCHITECTURE WITHOUT SPLIT-ADDRESS TECHNIQUE

N	T_{A1} (ns)	T_{ABS} (ns)	T_{SQ} (ns)	T_{A2} (ns)	EFFECTIVE MULT. TIME (ns)	
					PARALLEL	PIPELINE
7	23	10	40	25	98	45
8	25	10	45	25	105	50
9	25	10	45	30	110	50
10	25	10	45	30	110	50
11	25	10	45	30	110	50

TABLE 5

TYPICAL MULTIPLICATION SPEED OF PARALLEL AND PIPELINE ARCHITECTURE WITH SPLIT-ADDRESS TECHNIQUE

N	T_{A1} (ns)	T_{ABS} (ns)	T^{*}_{SQ} (ns)	T_{A2} (ns)	EFFECTIVE MULT. TIME (ns)	
					PARALLEL	PIPELINE
12	30	10	133	30	203	50
13	30	10	138	40	218	50
14	30	10	138	40	218	50
15	30	10	148	40	228	50

* The time required to do the squaring by "Reduced ROM Squaring Circuit".

The speed comparison between the parallel and the pipeline architecture with direct implementation and split address implementation are illustrated in Fig. 22 for word-

lengths of 8, 12 and 16 bits. The speed of commercially available digital multipliers are included in the figure as well. It shows that the pipelined SSRMs have a faster multiplication throughput rate than the digital multipliers. Fig. 23 gives the cost comparison between SSRMs and commercially available digital multipliers. With the split address technique the implementation cost is cheaper than the commercial digital multipliers.

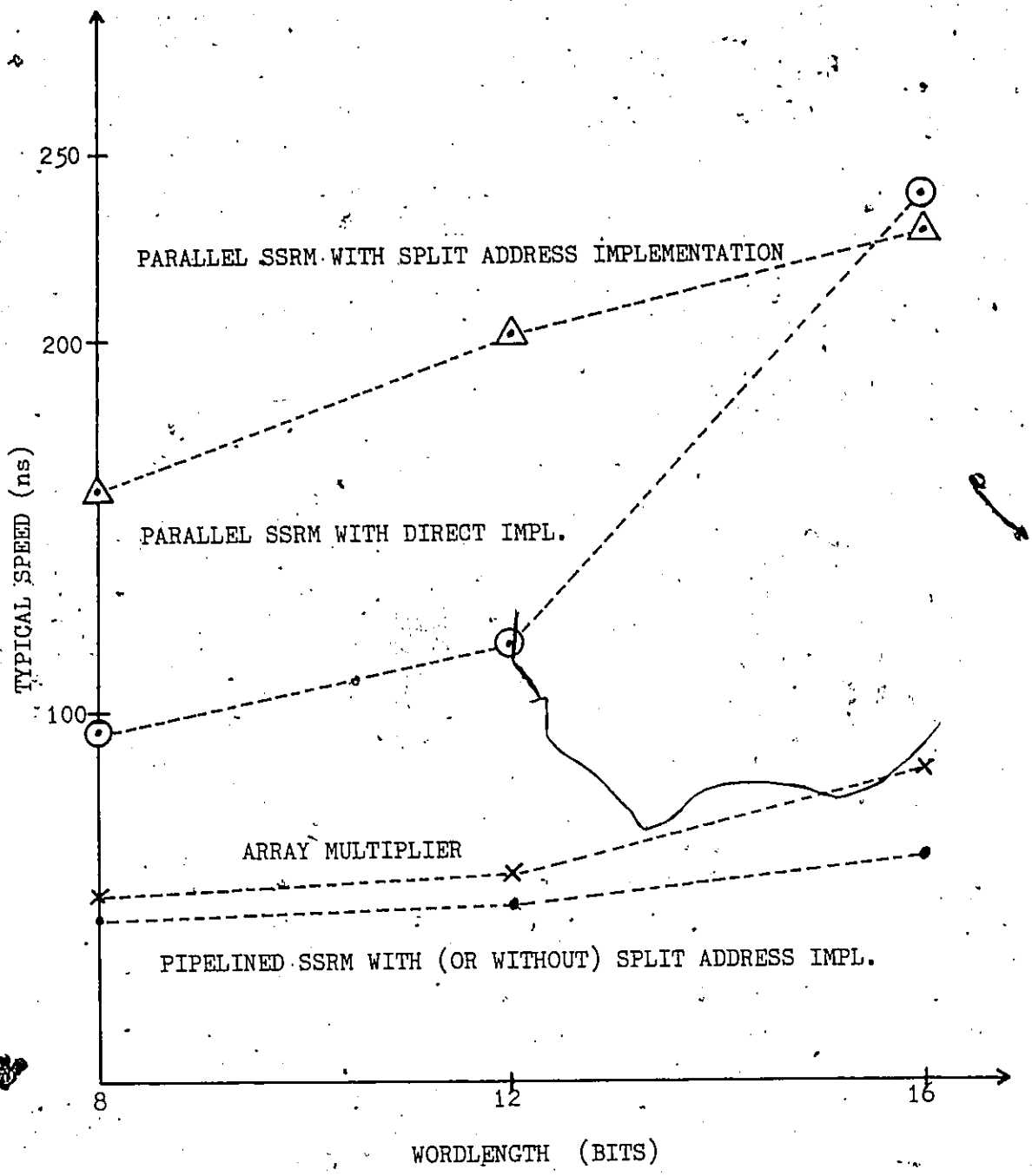


Figure 22: Speed comparison chart

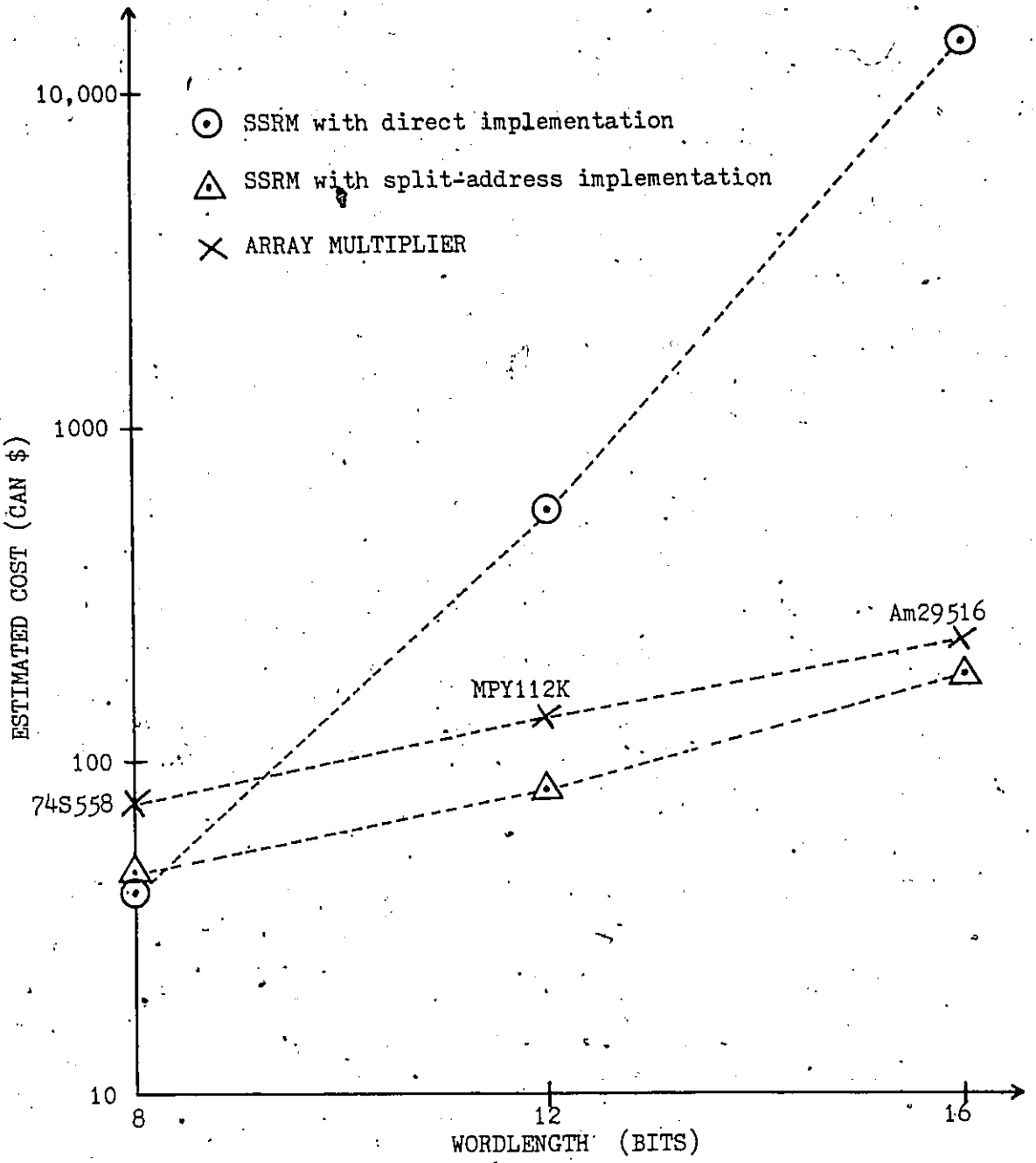


Figure 23: Cost comparison chart

Chapter III

PIPELINED STORED SQUARE DIGITAL FIR FILTER

3.1 INTRODUCTION

In this chapter the multiplication method using stored square technique described in the preceding chapter will be applied to the realization of non-recursive or FIR (Finite Impulse Response) digital filter. The implementation is based on the pipelined SSRM architecture which can have a sampling frequency of over 15 MHz. The main objective of this chapter is to evaluate the performance of such realization.

The filter is examined for several input signal, coefficient and product wordlengths for a number of different designs and filter lengths. The evaluations are performed by computer simulations with impulse and sampled sinusoidal input signals and compared with the performance based on the realization using conventional digital multipliers.

It is shown that in general, when M>N+3 the stored square digital filter has a comparable performance to the filter implemented using conventional digital multipliers. This result is similar to that obtained for the single SSRM in chapter II.

3.2 PROPOSED STRUCTURE FOR REALIZATION

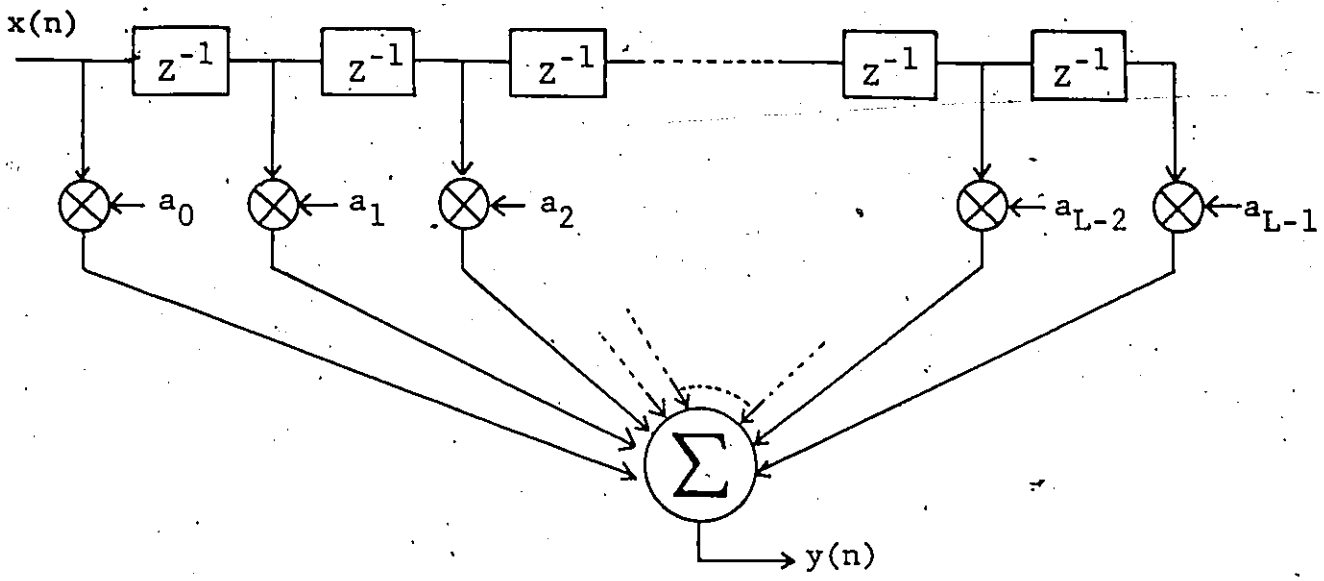
Digital filter implementation cannot begin until a decision is made as to what filter structure will be used. This section is devoted to arrive at that decision for implementation based on pipelined SSRM architecture.

In general, digital FIR filter can be expressed as :

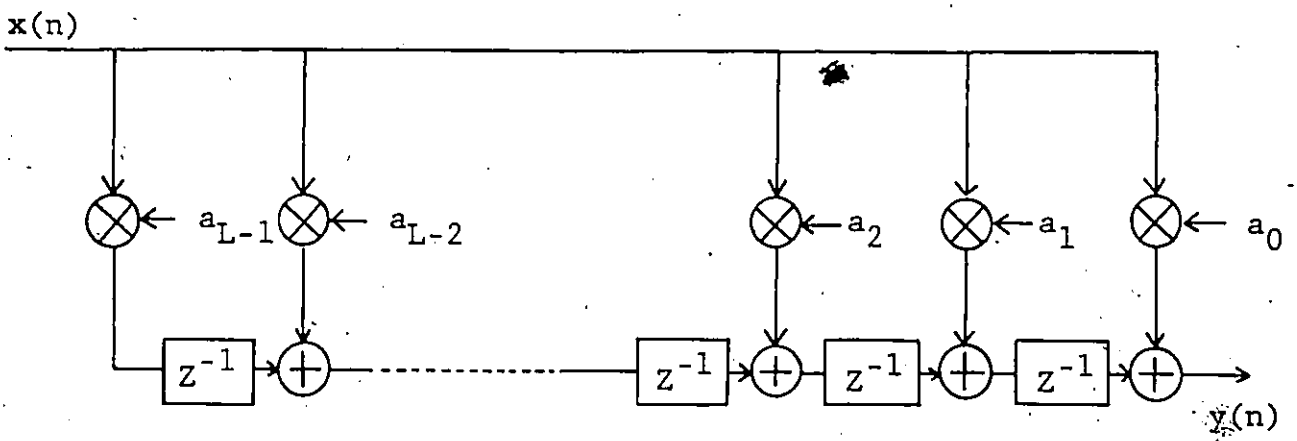
$$y(n) = \sum_{i=0}^{L-1} a_i \, x(n-i) \qquad (3.1)$$

where x(n), y(n) are the input and output signal samples respectively and a_i is the coefficient of the i^{th} tap with the total number of taps equal to L. Essentially, there are two structures of realization of (3.1) as shown in figures 24 a and 24 b.

For implementations using pipelined architecture, Fig. 24 b is the best suited structure for that applications, since, the input signal x(n) is multiplied by all coefficients $a_0, a_1, a_2, \ldots\ldots, a_{L-1}$ at each sampling time, where x(n) is a common line to all coefficients so makes it possible to process in parallel of all the partial sums with each other and the product of x(n) and the coefficients, hence, pipelined SSRMs can be directly applied to replace conventional digital multipliers with the only addition of a unit delay element at the output of the filter. In addition, this structure has an advantage of possible saving in the number of the required multipliers by one-half if a linear phase



(a)



(b)

Figure 24: The realizations of equation (3.1)

FIR filter is implemented, since, L tap linear phase FIR filter, magnitudewise, has L/2 (L=even) or L-1/2 + 1 (L=odd) distinct coefficients.

3.3 REALIZATION OF PIPELINED STORED SQUARE FIR FILTER

Based on the structure proposed in section 3.2, a digital FIR filter using stored square ROM multipliers with pipelined architecture can be implemented. The realization is shown in Fig. 25.

As described in section 2.9 of chapter II, the pipelined SSRM has an initial delay of 4 sample times, therefore, the output of the pipelined stored square FIR filter is practically delayed by 5 sample times and has the following difference equation :

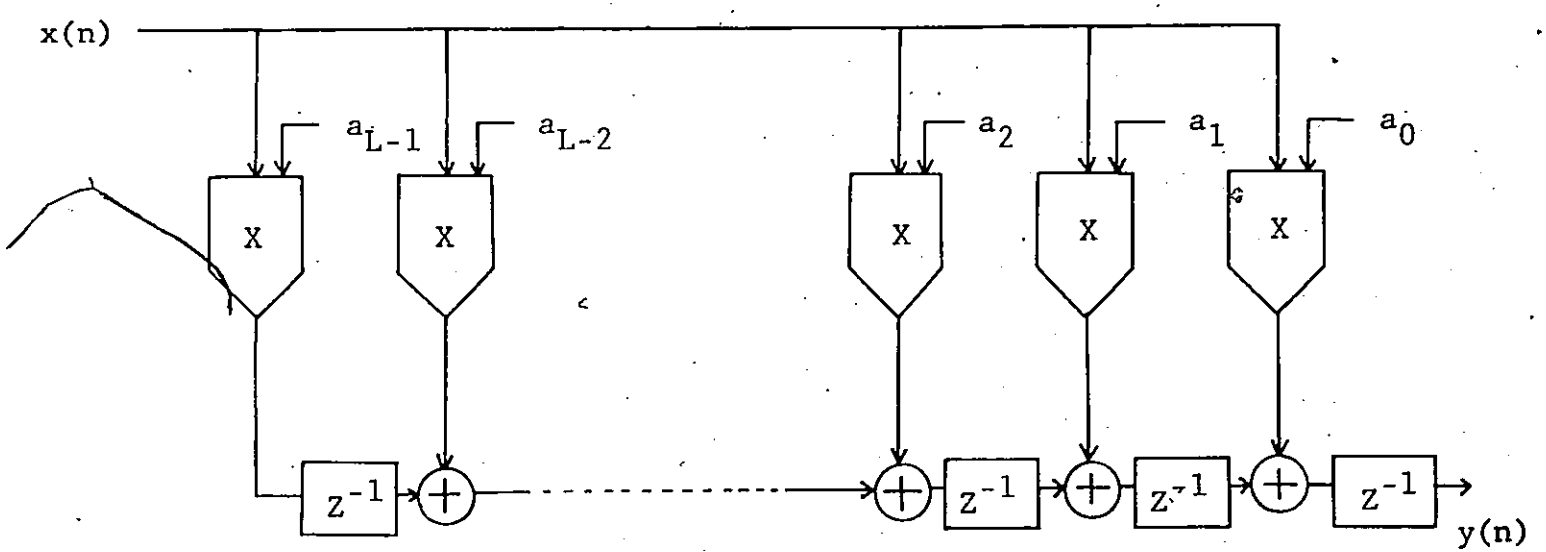
$$y(n-5) = \sum_{i=0}^{L-1} a_i \, x(n-i) \qquad (3.2)$$

or we can write as :

$$y(n) = \sum_{i=0}^{L-1} a_i \, x(n-5-i) \qquad (3.3)$$

If a split address technique is used, then the output of the filter will be delayed by 8 time samples and the difference equation becomes :

$$y(n) = \sum_{i=0}^{L-1} a_i \, x(n-8-i) \qquad (3.4)$$



X - pipelined stored square ROM multiplier

Figure 25: Pipelined stored square FIR filter realization

The sampling rate of this pipelined FIR filter is determined by the processing time of one particular stage within the filter which has the slowest execution time. Since the adders and the ROMs are the ones that relatively have the slowest execution time, so the sampling rate is limited by either the time required to do the addition or to do the ROM table look up (which is in the order of 50 ns), therefore, sampling rates in excess of 15 MHz can be easily achieved.

In practice, digital filters are implemented with finite wordlengths, therefore, it is necessary to quantize the coefficients and signal values. The coefficient quantization errors cause the change in pole and zero locations of the transfer function which in effect causes errors in the frequency response. On the other hand, the input signal and the product quantization errors (or the arithmetic roundoff errors) will produce errors in the output which is referred to as noise accumulation. These errors will be analyzed in the next two sections with all quantization done by rounding.

3.4 THE EFFECT OF QUANTIZATION AND ROUNDOFF ERRORS ON THE IMPULSE RESPONSE

The following analysis is based on the computational technique [13] defined as the square of the error in the energy of a filter output signal which can be written as :

$$E^2 = \sum_{n=0}^{L-1} (y_\infty(n) - y(n))^2 \qquad (3.5)$$

where $y(n)$ and $y_\infty(n)$ are the rounded and the ideal output responses respectively. The block diagram of this measurement technique is given in Fig. 26. A unit impulse is applied to the ideal and to the non-ideal filters. The ideal filter response is computed with signal and coefficient wordlengths of more than 32 bits, representing the ideal case, as compared to the non ideal filter with variable wordlengths of 16 bits or less.

Based on this technique, a computer simulation was carried out to evaluate the error E^2 for various input signal and coefficient wordlengths (N) and product wordlengths (M) in comparison with the performance of filters implemented with conventional digital multipliers. The filter's output signal wordlengths were taken to be equal to the product wordlengths (M).

Five examples of linear phase FIR filter designs [14] were used for the evaluation with the coefficients are tabulated in Tables 6 - 10.

From the results given in figures 27, 28, 29, 30 and 31 it shows that the pipelined stored square FIR filters have the same errors (E^2) as in conventional digital multiplier based filters when implemented with $M>N+3$. It is also observed that, in general, the errors start to converge at $M = N$.

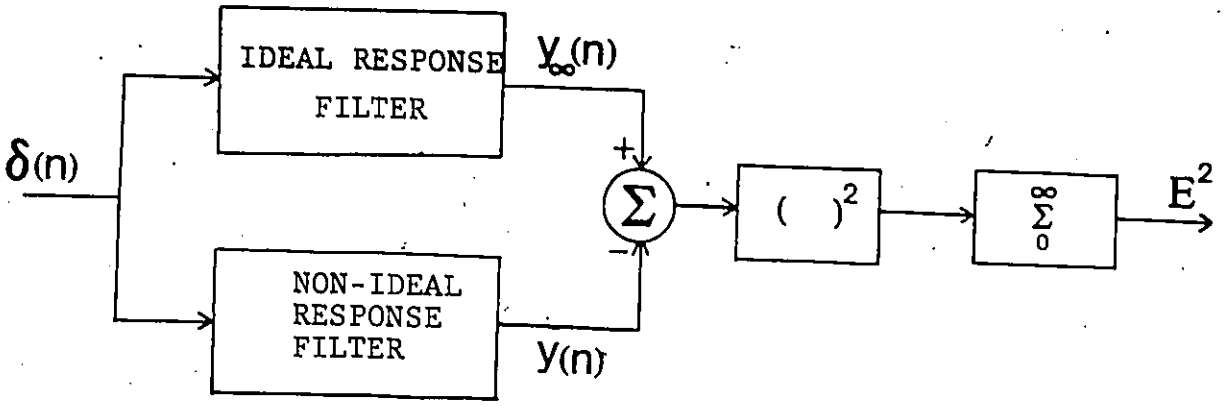


Figure 26: The technique of computing the error E^2 on the impulse response

TABLE 6

COEFFICIENTS OF THE 8 TAP LOWPASS FIR FILTER

a(1)	=	0.037489504
a(2)	=	-0.062394589
a(3)	=	-0.023215204
a(4)	=	0.555577160
a(5)	=	0.555577160
a(6)	=	-0.023215204
a(7)	=	-0.062394589
a(8)	=	0.037489504

TABLE 7

COEFFICIENTS OF THE 23 TAP LOWPASS FIR FILTER

a(1)	=	0.010359909
a(2)	=	0.014152668
a(3)	=	0.0068499334
a(4)	=	-0.0047560036
a(5)	=	-0.026848957
a(6)	=	-0.037932307
a(7)	=	-0.031249356
a(8)	=	0.0098157711
a(9)	=	0.075671554
a(10)	=	0.15457571
a(11)	=	0.21536756
a(12)	=	0.24031579
a(13)	=	0.21536756
a(14)	=	0.15457571
a(15)	=	0.075671554
a(16)	=	0.0098157711
a(17)	=	-0.031249356
a(18)	=	-0.037932307
a(19)	=	-0.026848957
a(20)	=	-0.0047560036
a(21)	=	0.0068499334
a(22)	=	0.014152668
a(23)	=	0.010359909

TABLE 8

COEFFICIENTS OF THE 23 TAP HIGHPASS FIR FILTER

a(1)	=	0.0055616572
a(2)	=	-0.00034328387
a(3)	=	-0.010401845
a(4)	=	-0.017904773
a(5)	=	-0.0089721791
a(6)	=	0.019517459
a(7)	=	0.047751546
a(8)	=	0.040013105
a(9)	=	-0.028421048
a(10)	=	-0.14438570
a(11)	=	-0.25551814
a(12)	=	0.69858849
a(13)	=	-0.25551814
a(14)	=	-0.14438570
a(15)	=	-0.028421048
a(16)	=	0.040013105
a(17)	=	0.047751546
a(18)	=	0.019517459
a(19)	=	-0.0089721791
a(20)	=	-0.017904773
a(21)	=	-0.010401845
a(22)	=	-0.00034328387
a(23)	=	0.0055616572

TABLE 9

COEFFICIENTS OF THE 32 TAP BANDPASS FIR FILTER

a(1)	=	-0.0057534091
a(2)	=	0.00099032000
a(3)	=	0.0075733364
a(4)	=	-0.0065141730
a(5)	=	0.013960522
a(6)	=	0.0022951365
a(7)	=	-0.019994050
a(8)	=	0.007136941
a(9)	=	-0.039657339
a(10)	=	0.011260167
a(11)	=	0.066233575
a(12)	=	-0.010497220
a(13)	=	0.085136056
a(14)	=	-0.12024987
a(15)	=	-0.29678535
a(16)	=	0.30410886
a(17)	=	0.30410886
a(18)	=	-0.29678535
a(19)	=	-0.12024987
a(20)	=	0.085136056
a(21)	=	-0.010497220
a(22)	=	0.066233575
a(23)	=	0.011260167
a(24)	=	-0.039657339
a(25)	=	0.007136941
a(26)	=	-0.019994050
a(27)	=	0.0022951365
a(28)	=	0.013960522
a(29)	=	-0.0065141730
a(30)	=	0.0075733364
a(31)	=	0.00099032000
a(32)	=	-0.0057534091

TABLE 10

COEFFICIENTS OF THE 64 TAP BANDPASS FIR FILTER

a(1)	=	0.00044571351
a(2)	=	0.0011982126
a(3)	=	-0.0020269575
a(4)	=	-0.0017503640
a(5)	=	0.0025727358
a(6)	=	0.00041575264
a(7)	=	0.0019333260
a(8)	=	-0.00040633697
a(9)	=	-0.0076069571
a(10)	=	0.0027127080
a(11)	=	0.0038588485
a(12)	=	0.0015340215
a(13)	=	0.0065284707
a(14)	=	-0.013639171
a(15)	=	-0.0070497543
a(16)	=	0.011264734
a(17)	=	0.00013792887
a(18)	=	0.016741514
a(19)	=	-0.0067519695
a(20)	=	-0.033000246
a(21)	=	0.013678338
a(22)	=	0.0034060925
a(23)	=	0.020837426
a(24)	=	0.030845255
a(25)	=	-0.068832994
a(26)	=	-0.015662715
a(27)	=	0.021438628
a(28)	=	0.0040430576
a(29)	=	0.14673573
a(30)	=	-0.095474064
a(31)	=	-0.28499079
a(32)	=	0.24706584
a(33)	=	0.24706584
a(34)	=	-0.28499079
a(35)	=	-0.095474064
a(36)	=	0.14673573
a(37)	=	0.0040430576
a(38)	=	0.021438628
a(39)	=	-0.015662715
a(40)	=	-0.068832994
a(41)	=	0.030845255
a(42)	=	0.020837426
a(43)	=	0.0034060925
a(44)	=	0.013678338
a(45)	=	-0.033000246
a(46)	=	-0.0067519695
a(47)	=	0.016741514
a(48)	=	0.00013792887
a(49)	=	0.011264734

```
a(50) =  -0.0070497543
a(51) =  -0.013639171
a(52) =   0.0065284707
a(53) =   0.0015340215
a(54) =   0.0038588485
a(55) =   0.0027127080
a(56) =  -0.0076069571
a(57) =  -0.00040633697
a(58) =   0.0019333260
a(59) =   0.00041575264
a(60) =   0.0025727358
a(61) =  -0.0017503640
a(62) =  -0.0020269575
a(63) =   0.0011982126
a(64) =   0.00044571351
```

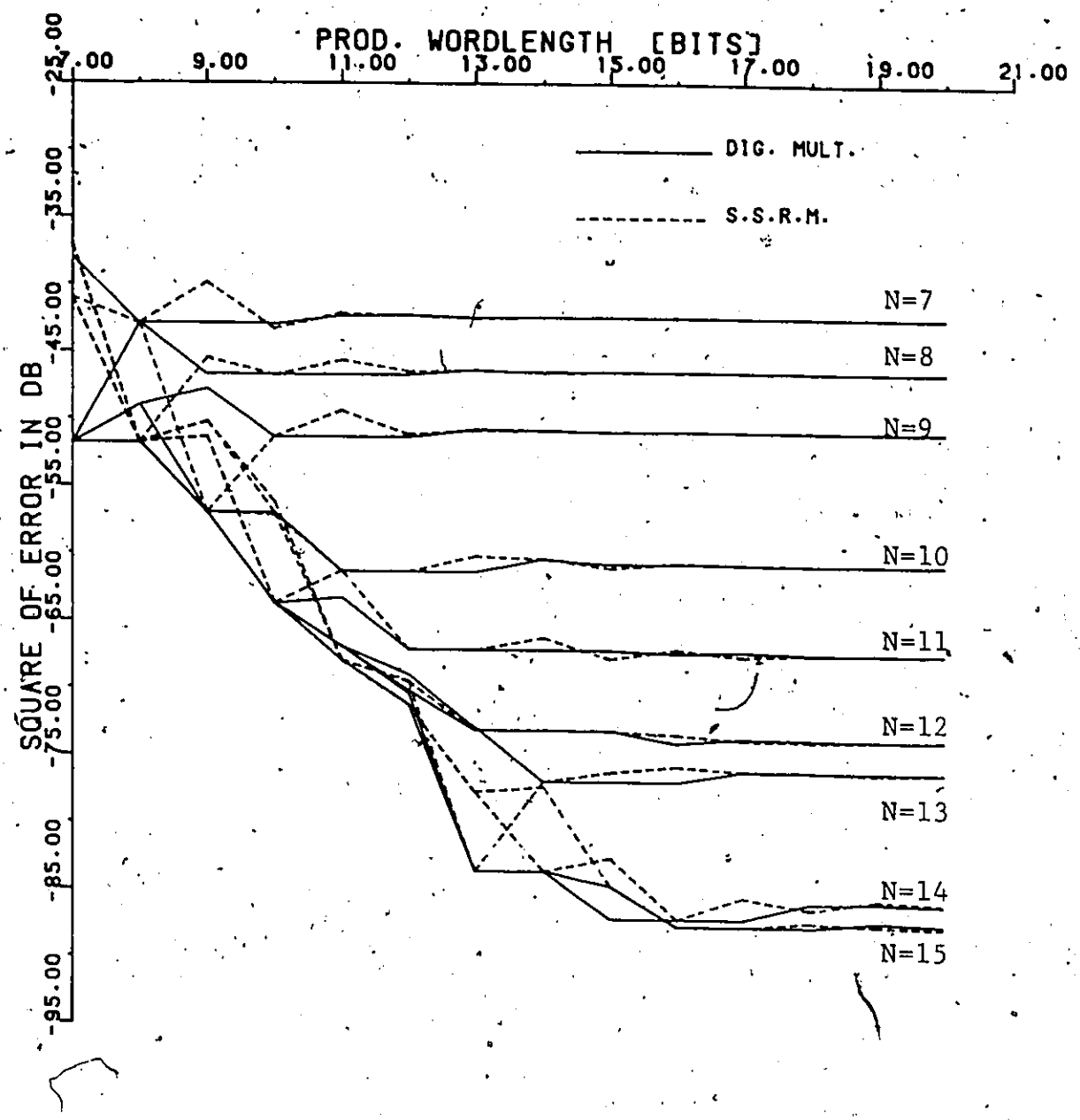


Figure 27: The errors E^2 on the impulse response of 8 tap lowpass FIR filter.

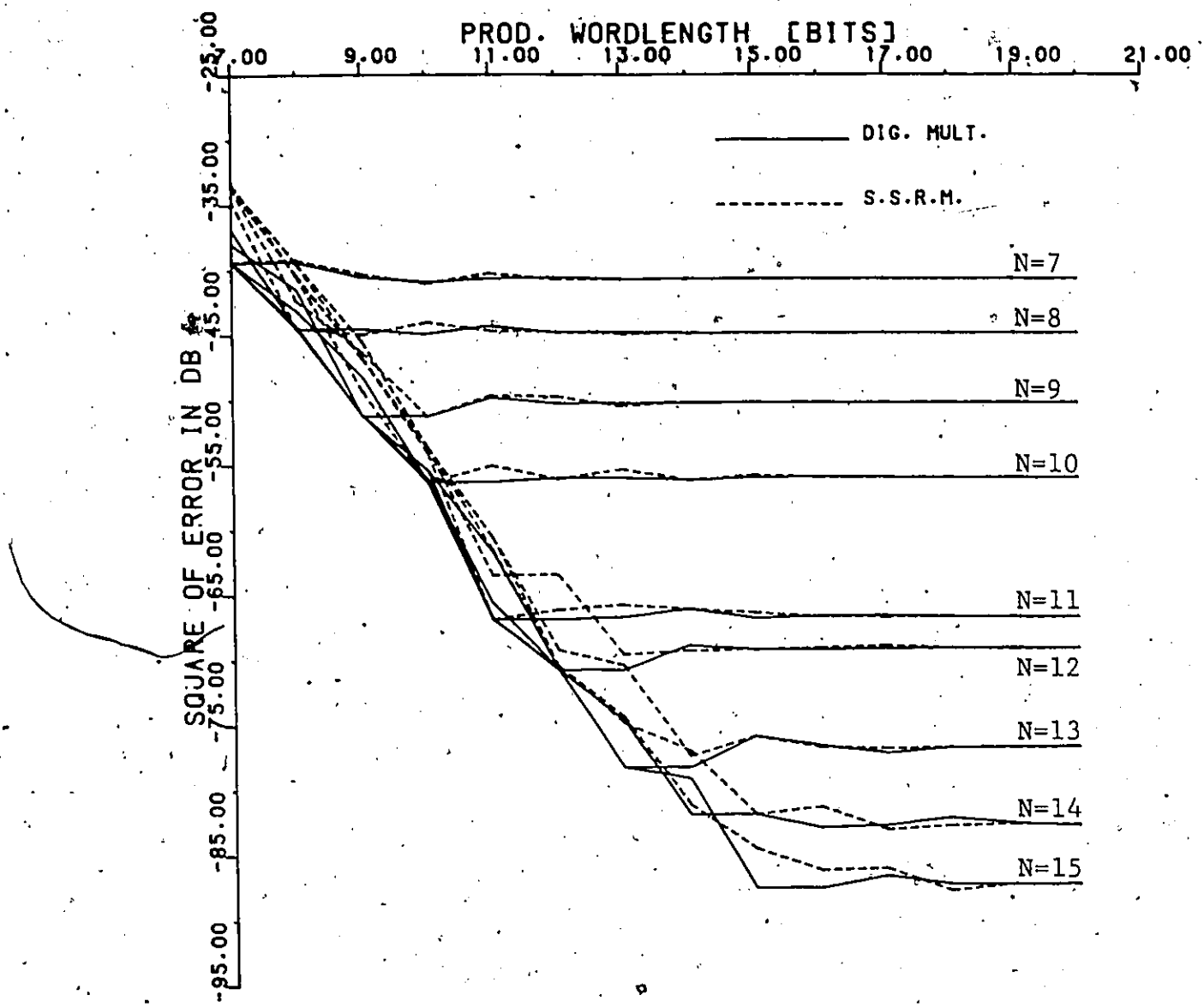


Figure 28: The errors E^2 on the impulse response of 23 tap lowpass FIR filter.

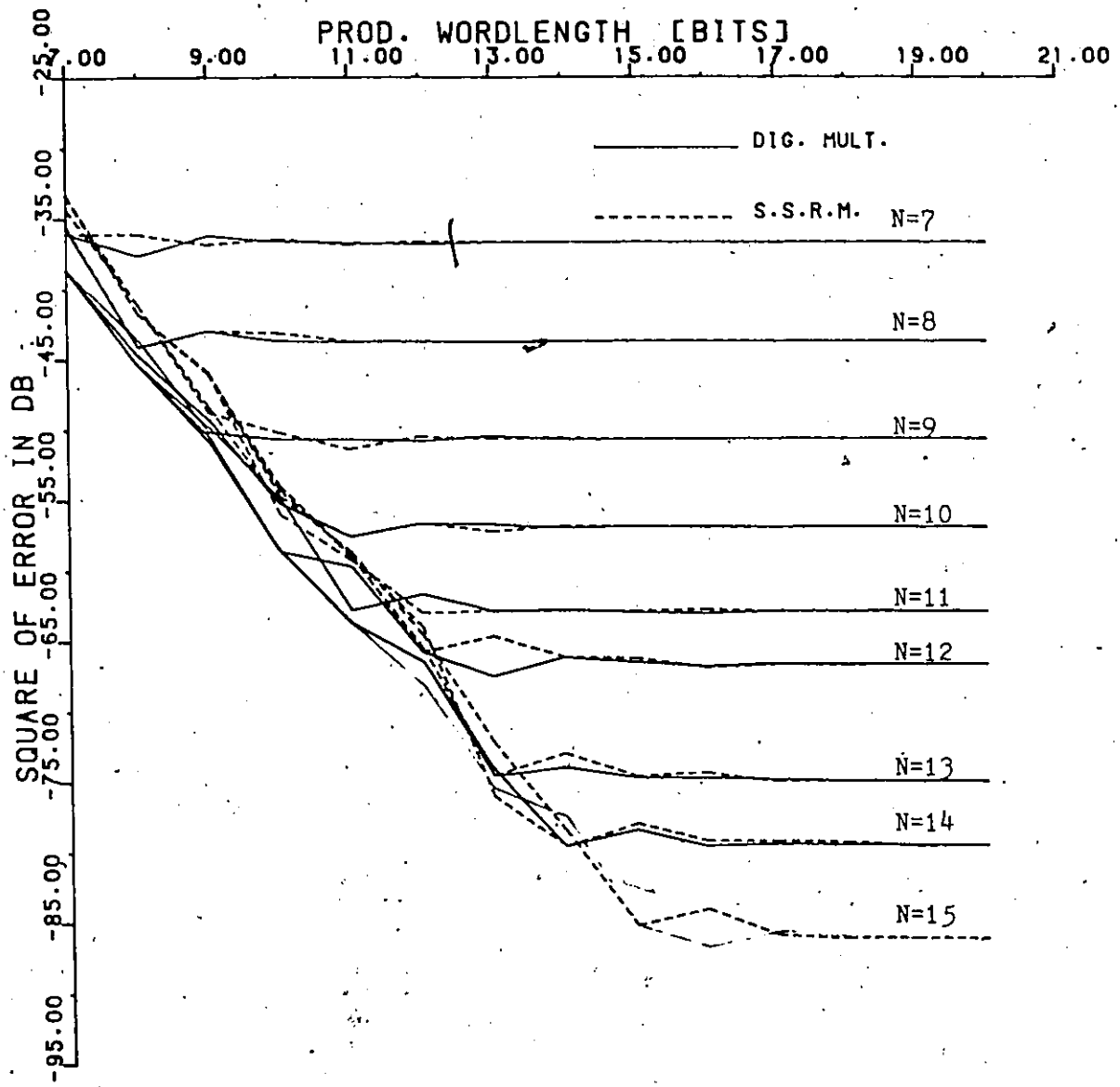


Figure 29: The errore E^2 on the impulse response of 23 tap highpass FIR filter.

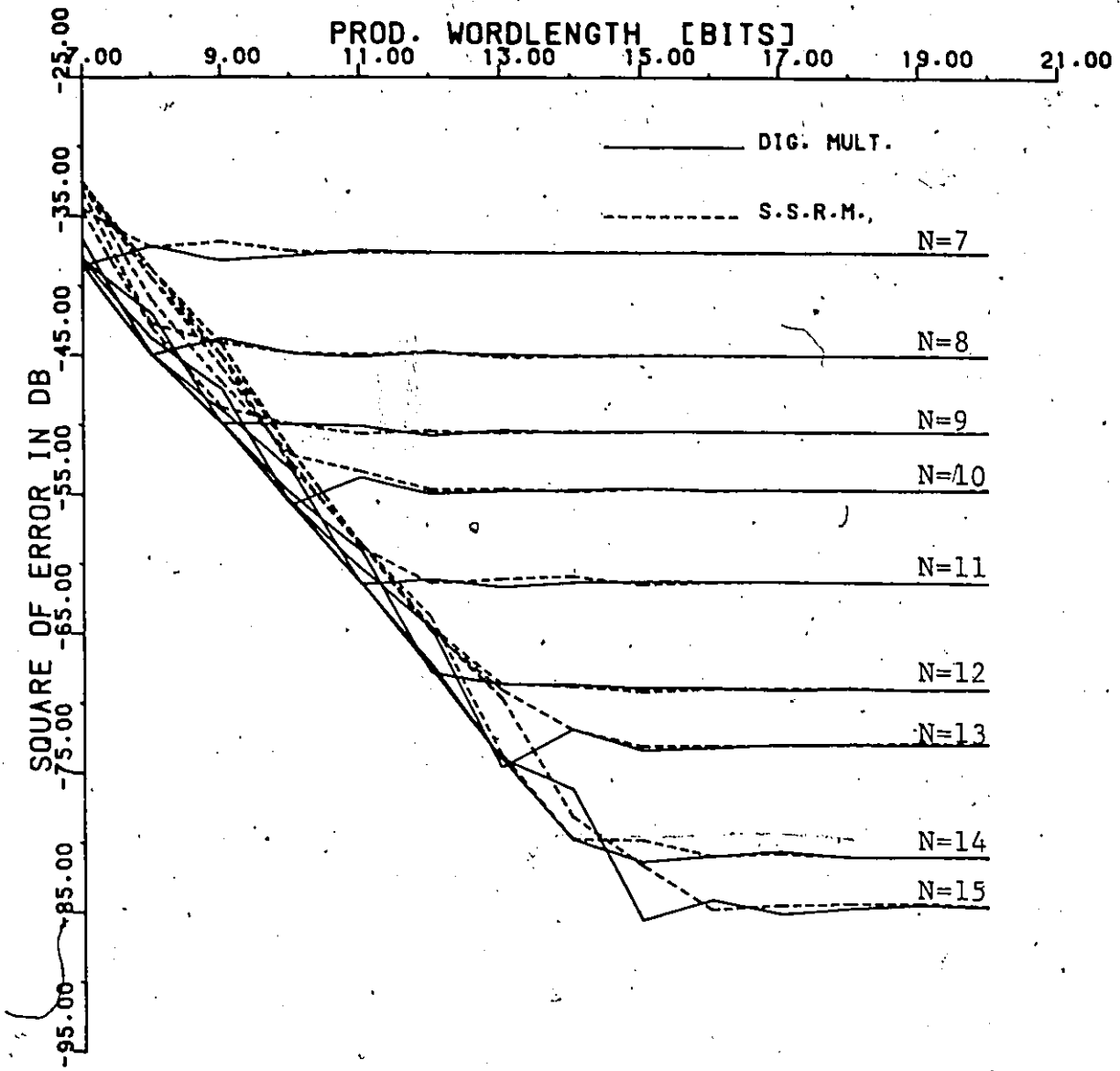


Figure 30: The errors E^2 on the impulse response of 32 tap bandpass FIR filter.

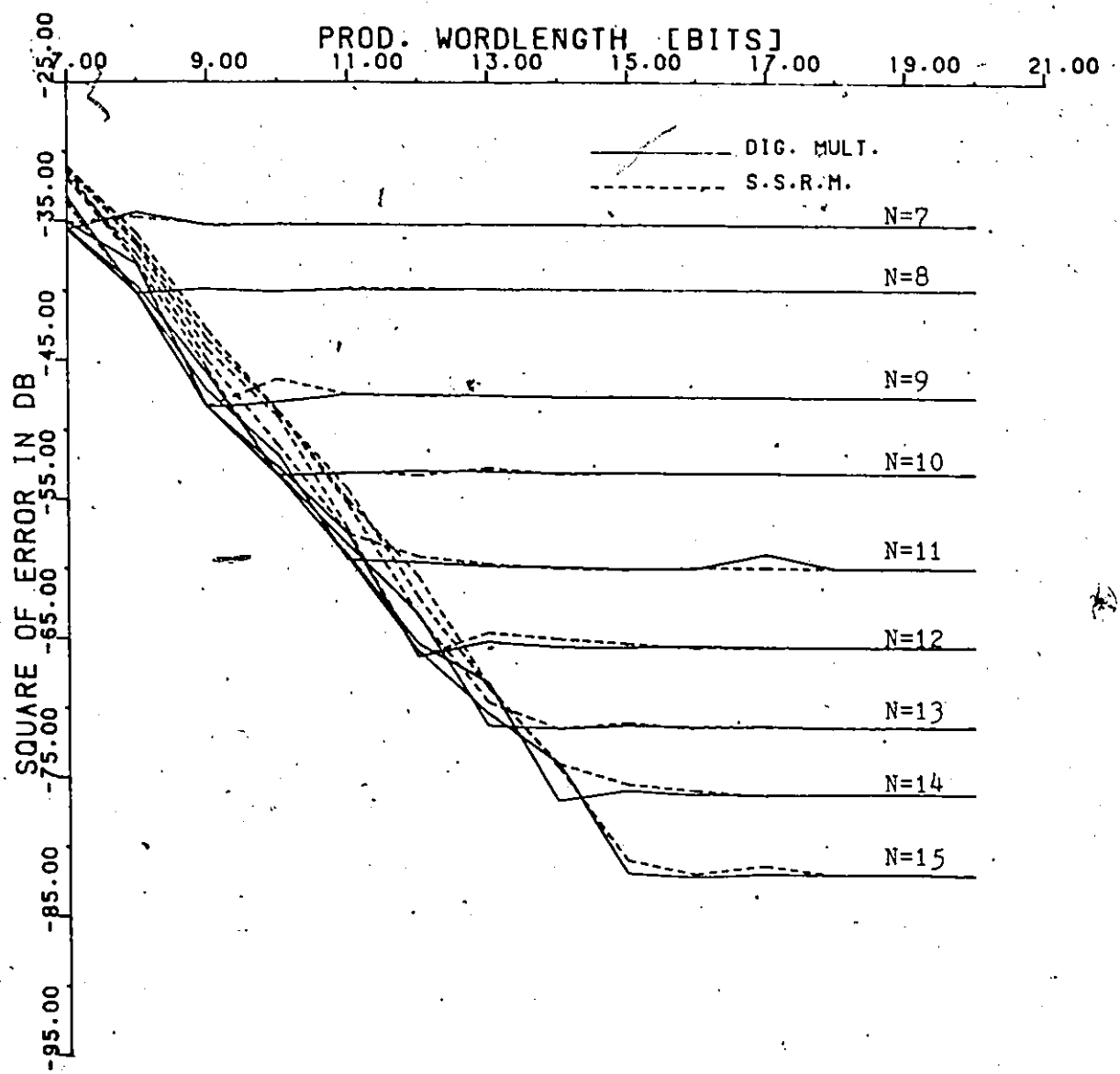


Figure 31: The errors E^2 on the impulse response of 64 tap bandpass FIR filter.

3.5 THE EFFECT OF QUANTIZATION AND ROUNDOFF ERRORS ON THE FREQUENCY RESPONSE

In this section the effect of quantization and roundoff noise on the frequency response of the filter design examples of section 3.4 will be examined.

The magnitude response of the filters are computed for frequencies between zero and half the sampling frequency. The rms (root mean square) deviations of the response from the ideal case are then calculated using the following expression :

$$\text{rms dev} = \sqrt{\frac{\sum_{f=0}^{f_s/2} (|H_I(f)| - |H(f)|)^2}{N_s}} \qquad (3.6)$$

where, $|H_I(f)|$ = magnitude response of the ideal filter at frequency f.

$|H(f)|$ = magnitude response of the non-ideal filter at frequency f.

f_s = sampling frequency.

N_s = number of computations taken at different frequencies between zero and half the sampling frequency.

Table 11 gives the computer simulation printout sample of the 8 tap lowpass pipelined stored square FIR filter with input signal and coefficient wordlength (N) of 7 bits, and

output signal and product wordlength (M) of 11 bits. A sampling frequency of 16 KHz was used. Several cycles of input signal were processed but not included in the computation of the magnitude response in order to establish the steady state condition.

Figures 32, 33, 34 and 35 show the plot of the rms deviations of the filters implemented with SSRMs and digital multipliers for input signal and coefficient wordlengths of 7, 11 and 15 bits. The figures show that a same accuracy can be obtained from the stored square digital filters as in the conventional digital multiplier filters when M>N+3.

With input signal and coefficient wordlengths (N) of 7 bits and M>N+3, a 32 tap bandpass FIR filter has an rms deviation of 14 dB, if input signal and coefficient wordlengths of 11 bits are used the rms deviation will be improved by 8.3 dB and a further improvement of 11.8 db can be obtained with N = 15.

The 8 tap lowpass FIR filter has a minimumn rms deviation of 0.33 dB with N = 7. For 23 tap highpass filter, it requires input signal and coefficient wordlengths of 11 bits to have an rms deviation of 0.77 dB, while, for 32 tap and 64 tap bandpass filter will have rms deviations of 2.2 dB and 3.5 dB respectively with N = 15.

The frequency response of the 8 tap and 23 tap lowpass filters in Fig. 36 and 37 show the closeness of the stored

TABLE 11

COMPUTER SIMULATION OUTPUT OF PIPELINED STORED SQUARE FIR FILTER

********** FILTER LENGTH = 8 **********

INPUT SIGNAL IS 7 BITS
PROD. WORDLENGTH IS 11 BITS
SAMPLING FREQ IS 16000 Hz

FREQ	IDEAL RES.	NON-IDEAL RES.	DEVIATION
0.01	0.357 DB	0.332 DB	0.025 DB
0.02	0.108 DB	-0.003 DB	0.111 DB
0.03	0.283 DB	0.106 DB	0.177 DB
0.04	0.052 DB	-0.155 DB	0.207 DB
0.05	0.157 DB	0.011 DB	0.146 DB
0.06	-0.021 DB	-0.090 DB	0.069 DB
0.07	0.017 DB	0.023 DB	-0.006 DB
0.08	-0.089 DB	-0.061 DB	-0.028 DB
0.09	-0.095 DB	-0.089 DB	-0.006 DB
0.10	-0.128 DB	-0.158 DB	0.031 DB
0.11	-0.142 DB	-0.201 DB	0.060 DB
0.12	-0.121 DB	-0.177 DB	0.056 DB
0.13	-0.111 DB	-0.157 DB	0.046 DB
0.14	-0.066 DB	-0.106 DB	0.039 DB
0.15	-0.017 DB	-0.067 DB	0.050 DB
0.16	0.018 DB	-0.050 DB	0.067 DB
0.17	0.101 DB	0.030 DB	0.071 DB
0.18	0.098 DB	0.040 DB	0.058 DB
0.19	0.186 DB	0.153 DB	0.033 DB
0.20	0.129 DB	0.109 DB	0.020 DB
0.21	0.179 DB	0.159 DB	0.021 DB
0.22	0.057 DB	0.016 DB	0.042 DB
0.23	0.025 DB	-0.029 DB	0.055 DB
0.24	-0.169 DB	-0.220 DB	0.051 DB
0.25	-0.322 DB	-0.344 DB	0.022 DB
0.26	-0.602 DB	-0.600 DB	-0.002 DB
0.27	-0.900 DB	-0.879 DB	-0.021 DB
0.28	-1.291 DB	-1.267 DB	-0.023 DB
0.29	-1.748 DB	-1.725 DB	-0.023 DB
0.30	-2.286 DB	-2.268 DB	-0.018 DB
0.31	-2.909 DB	-2.877 DB	-0.032 DB
0.32	-3.641 DB	-3.605 DB	-0.036 DB
0.33	-4.443 DB	-4.419 DB	-0.024 DB
0.34	-5.418 DB	-5.439 DB	0.020 DB
0.35	-6.428 DB	-6.495 DB	0.068 DB
0.36	-7.693 DB	-7.800 DB	0.106 DB
0.37	-8.966 DB	-9.080 DB	0.114 DB
0.38	-10.567 DB	-10.675 DB	0.108 DB
0.39	-12.189 DB	-12.297 DB	0.108 DB

0.40	-14.178 DB	-14.328 DB	0.150 DB
0.41	-16.268 DB	-16.475 DB	0.207 DB
0.42	-18.727 DB	-19.012 DB	0.285 DB
0.43	-21.445 DB	-21.826 DB	0.381 DB
0.44	-24.508 DB	-24.965 DB	0.457 DB
0.45	-28.063 DB	-28.667 DB	0.604 DB
0.46	-31.960 DB	-32.810 DB	0.851 DB
0.47	-36.664 DB	-37.907 DB	1.243 DB
0.48	-41.955 DB	-43.271 DB	1.316 DB
0.49	-49.557 DB	-48.483 DB	-1.075 DB

THE RMS DEVIATION IS 0.3565604091E 00 DB

square digital filters to the conventional digital multiplier filters response when implemented with M=N+4. The frequency response of the other stored square FIR filter designs are plotted in figures 38 - 40.

The simulation results obtained in this chapter demonstrate that digital FIR filters implemented based on the stored square multiplication technique will definitely have an accuracy comparable to those filters with conventional digital multipliers when the stored squares (and the products) are more than 3 bits longer than the input signal and coefficient wordlengths.

Advantages in cost and speed of these filters, in integrated form are anticipated as all ROM contents are to be identical, and pipeline operation can be achieved.

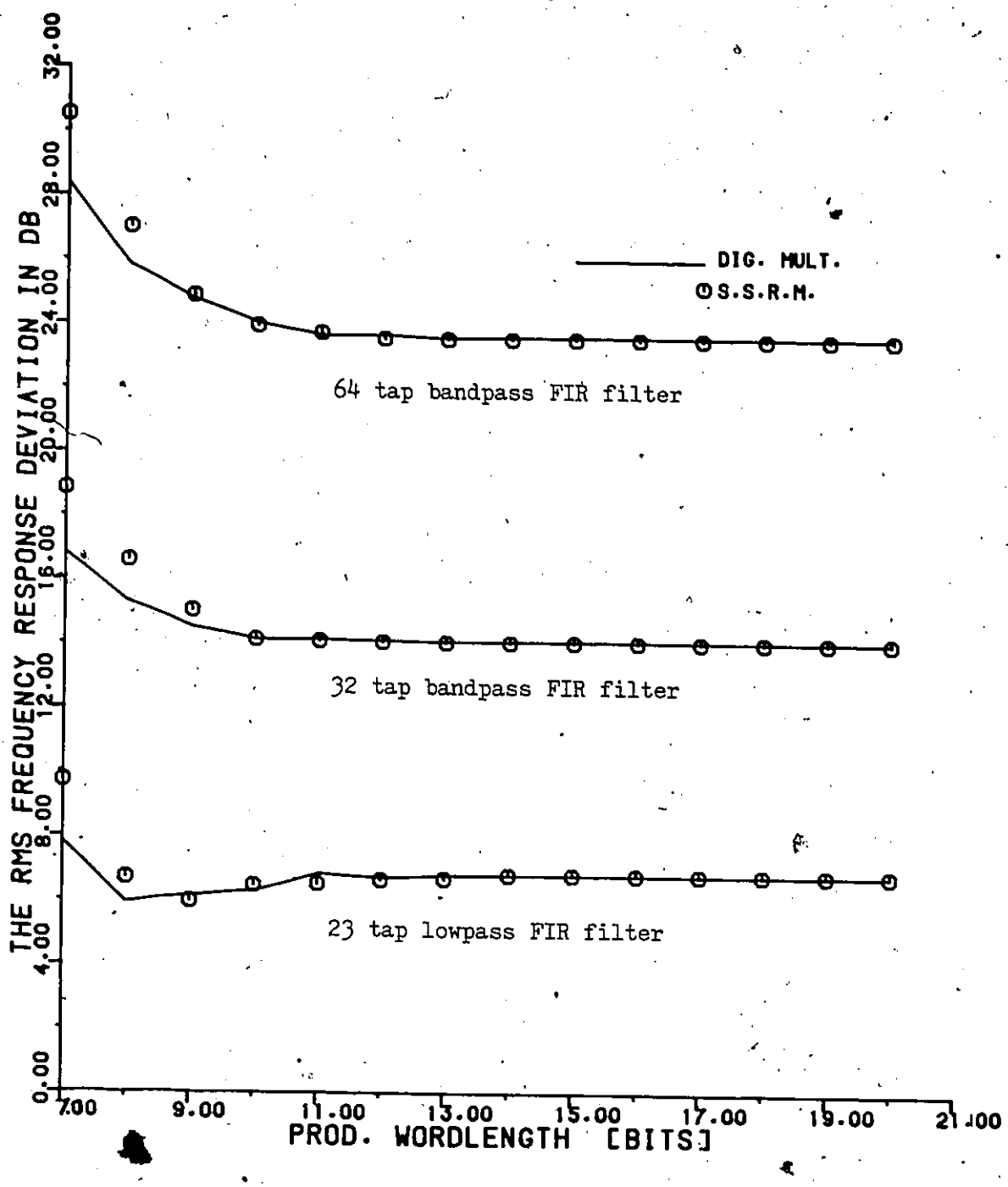


Figure 32: The rms frequency response deviation of the 32 and 64 tap bandpass filters and the 23 tap lowpass filter with N = 7.

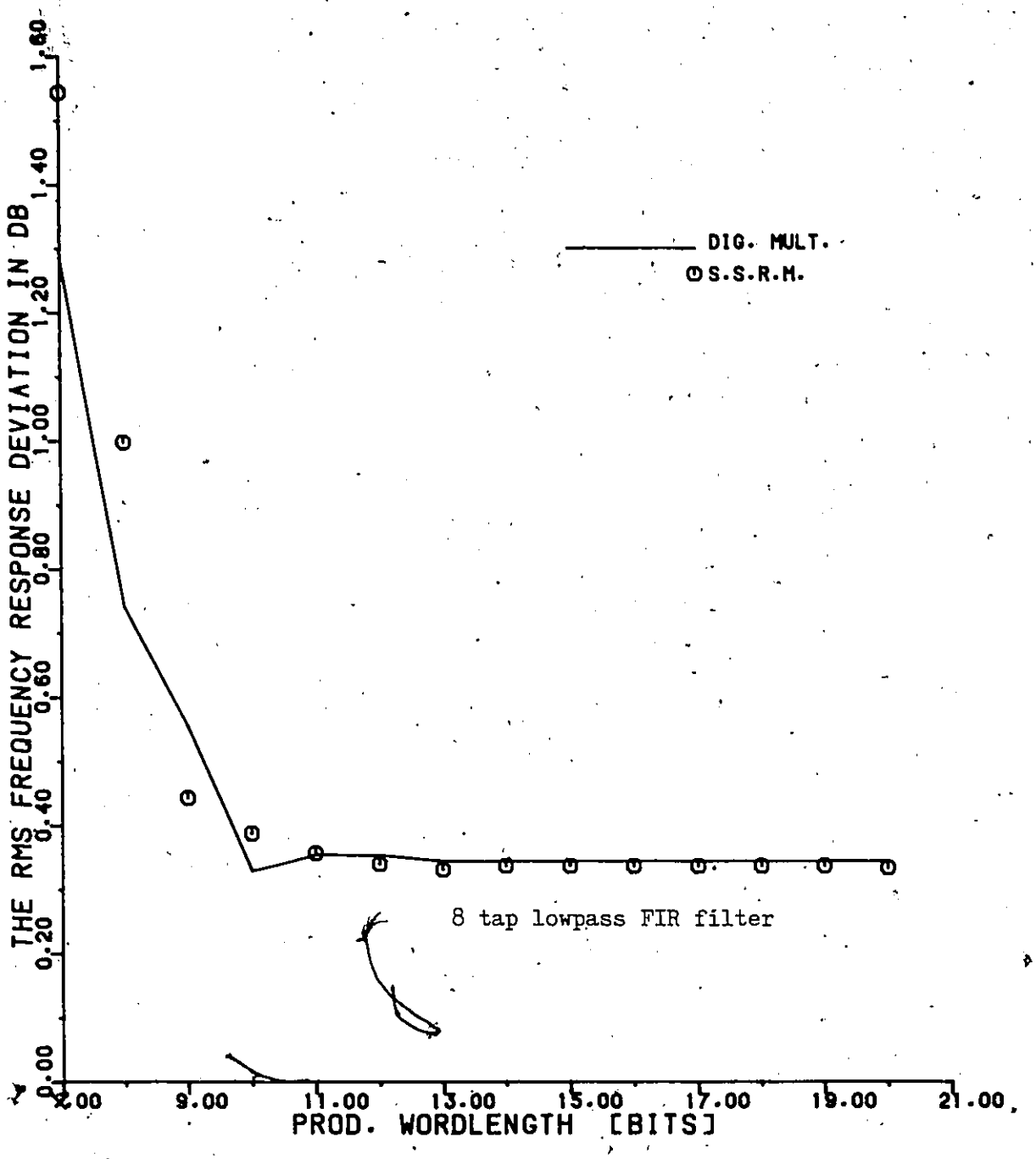


Figure 33: The rms frequency response deviation of the 8 tap lowpass filter with N = 7.

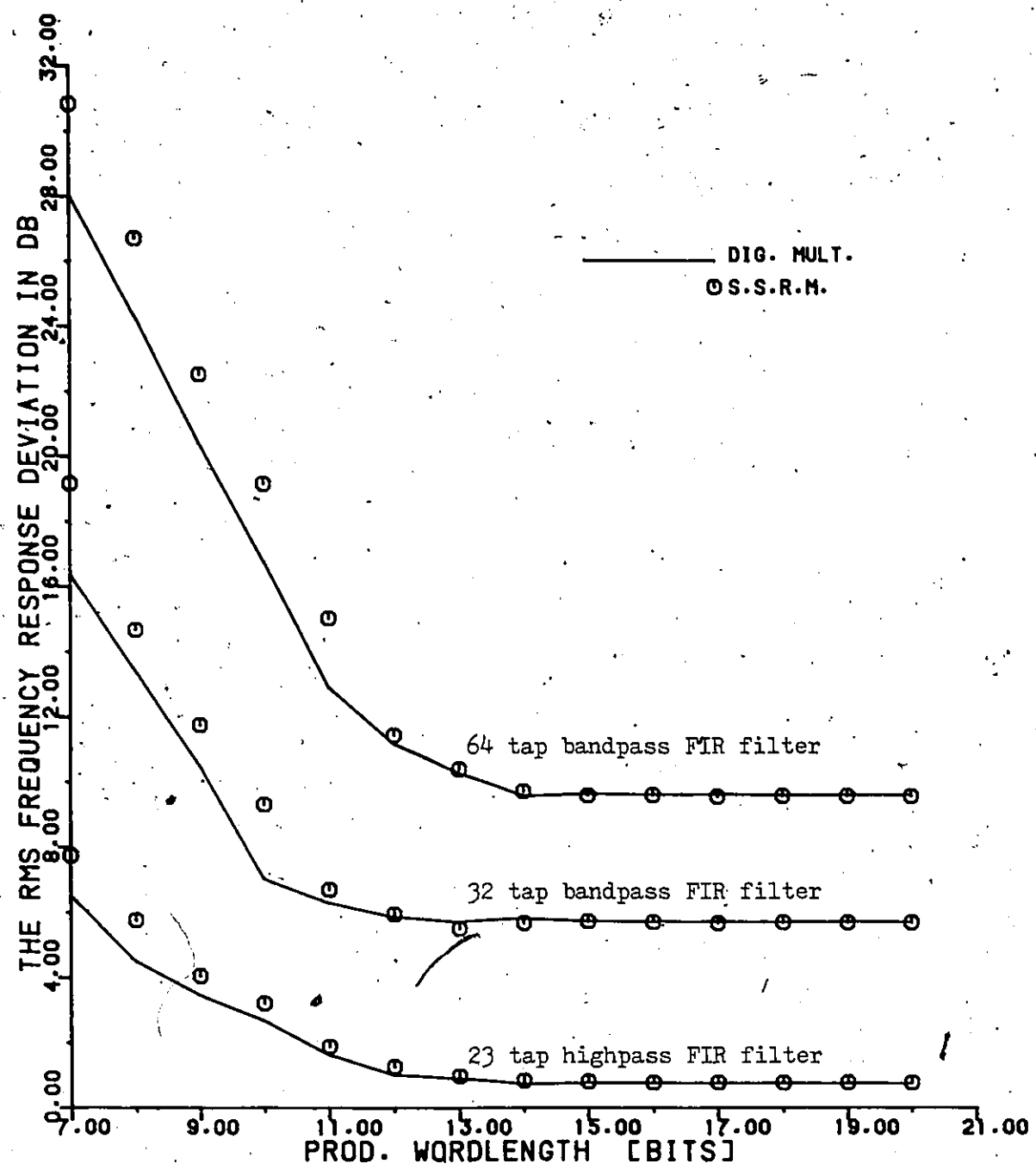


Figure 34: The rms frequency response deviation of the 32 and 64 tap bandpass filters and the 23 tap highpass filter with N = 11.

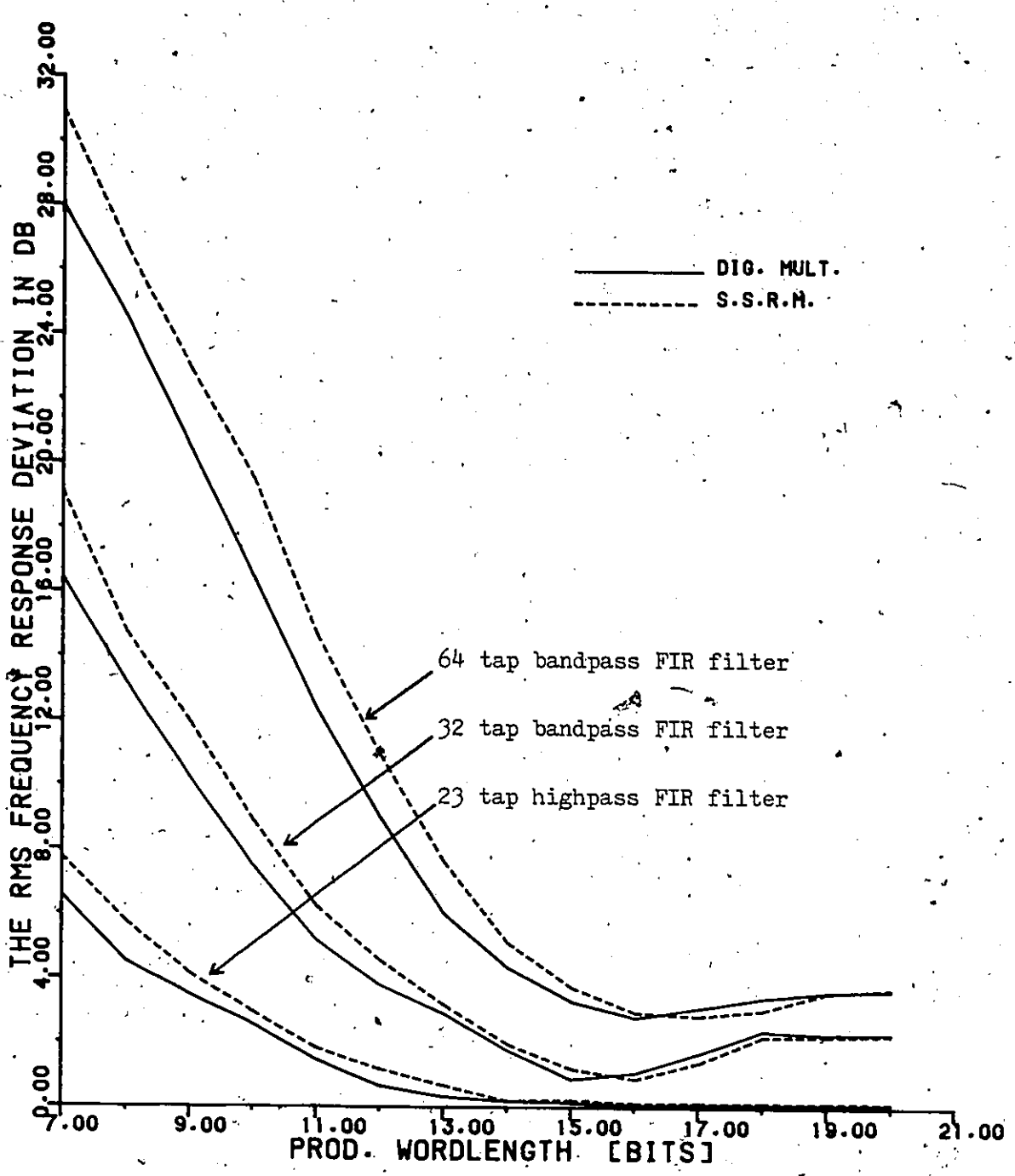


Figure 35: The rms frequency response deviation of the 32 and 64 tap bandpass filters and the 23 tap highpass filter with N=15.

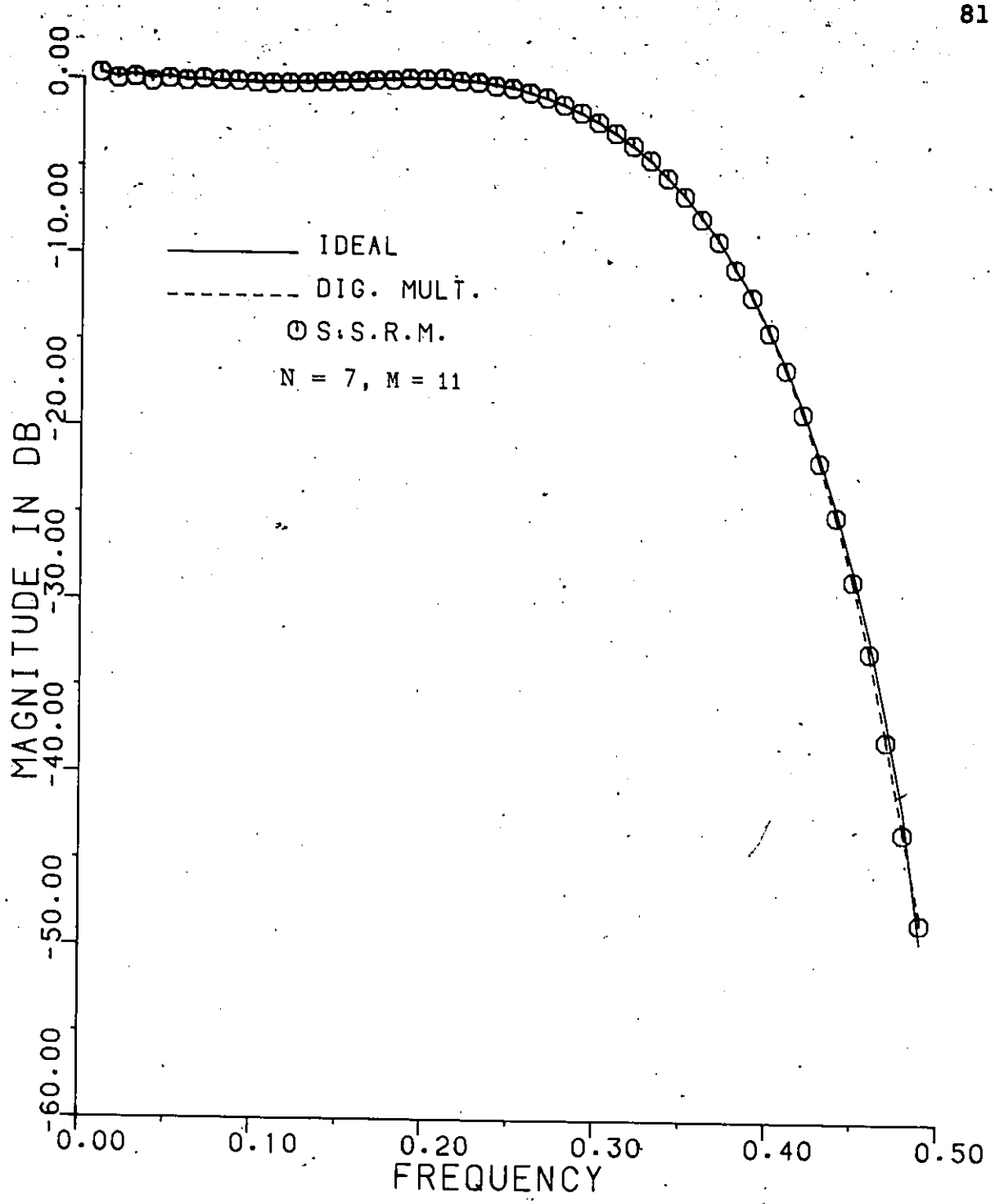


Figure 36: Comparison between the SSRM and the digital multiplier on the frequency response of the 8 tap lowpass FIR filter.

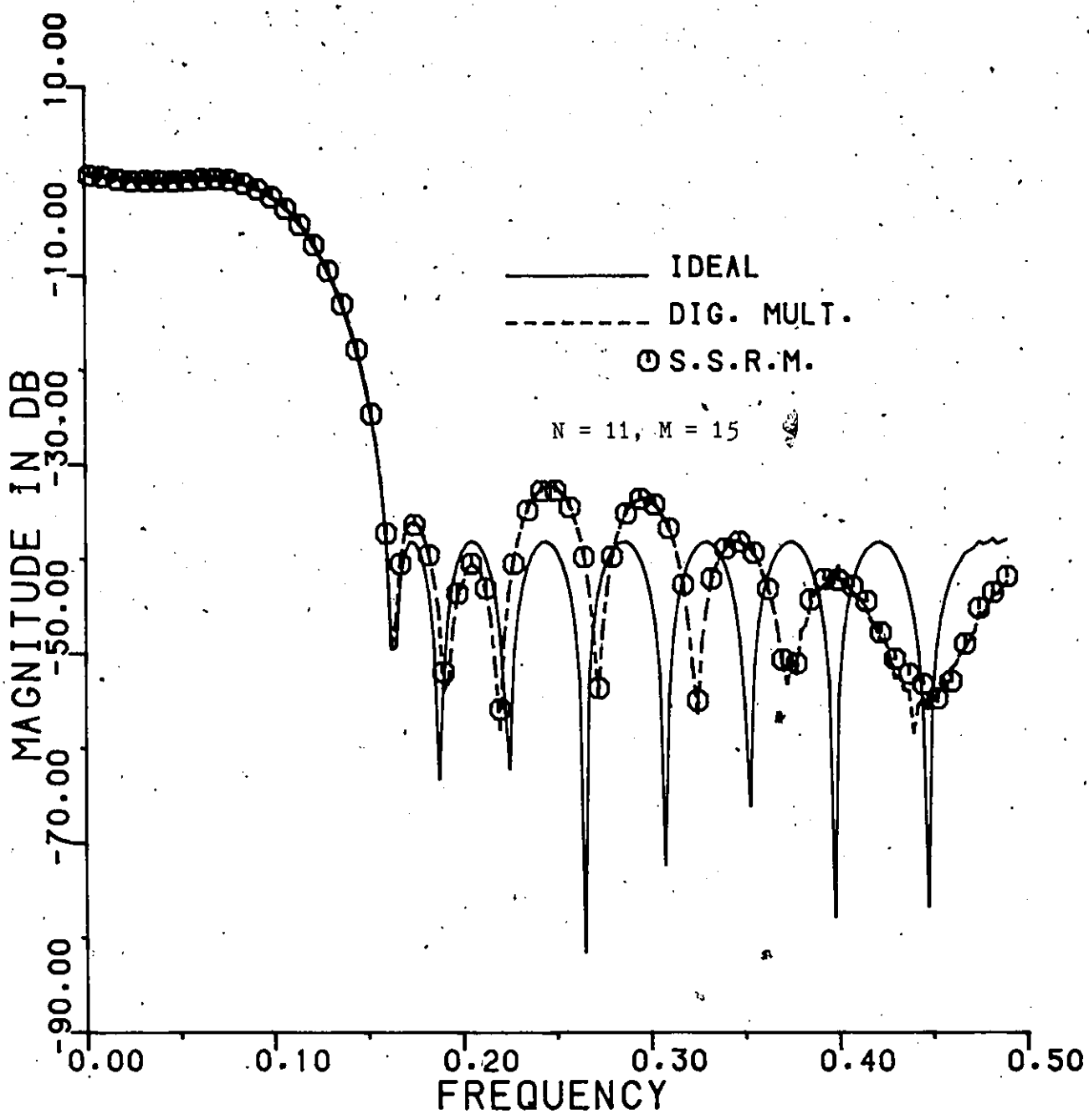


Figure 37: Comparison between the SSRM and the digital multiplier on the frequency response of the 23 tap lowpass FIR filter.

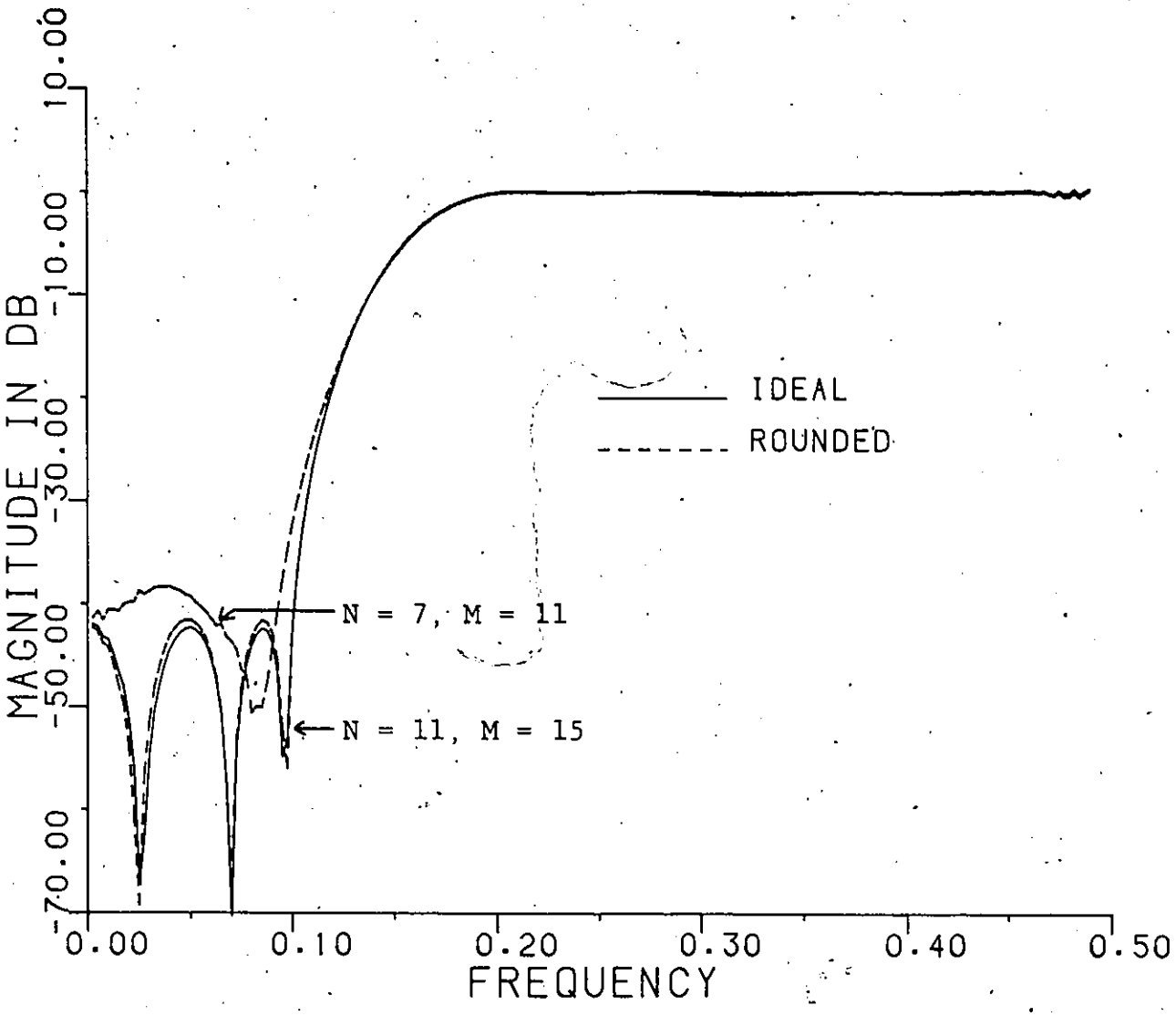


Figure 38: Frequency response of the 23 tap highpass FIR filter.

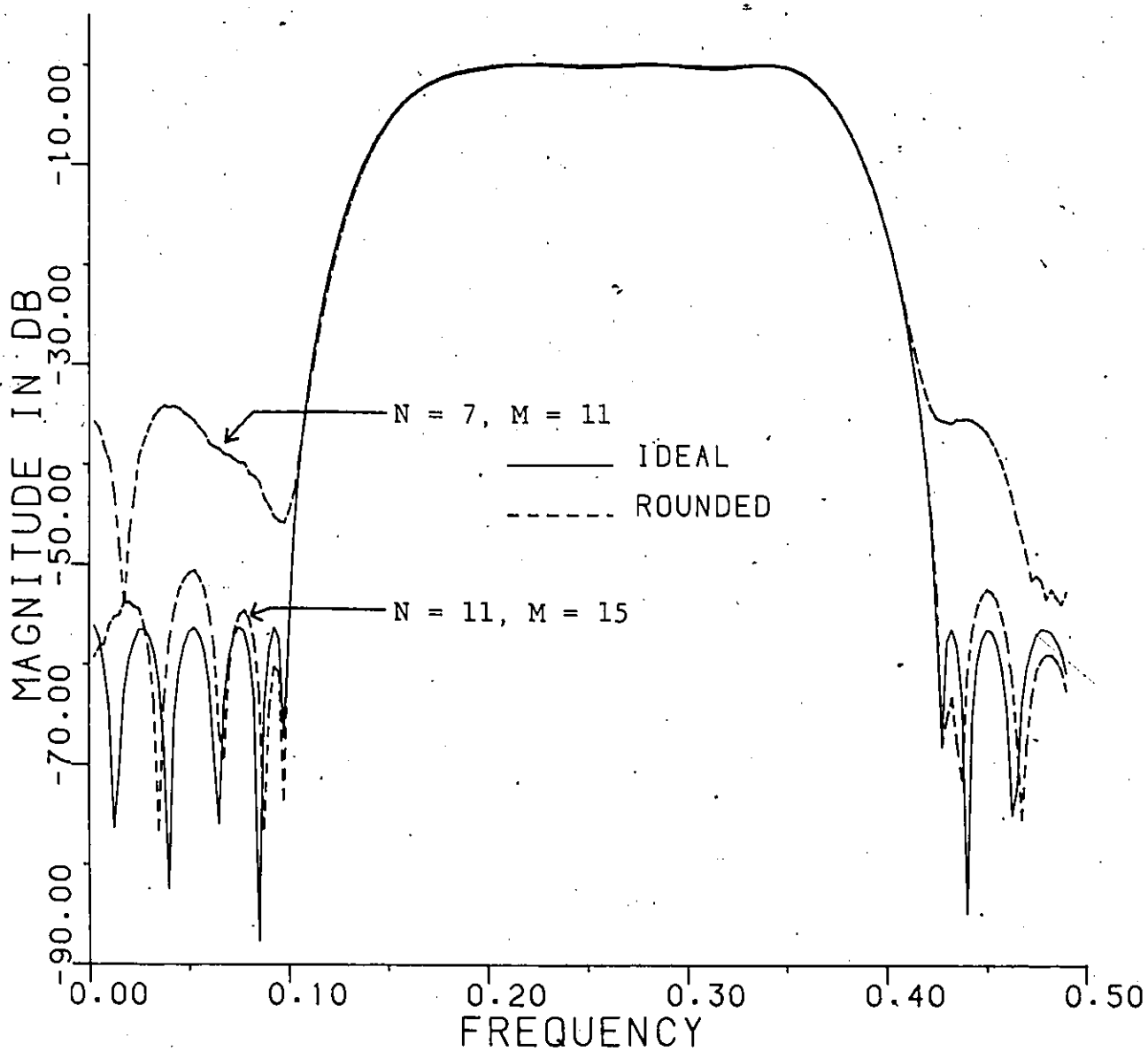


Figure 39: Frequency response of the 32 tap bandpass FIR filter.

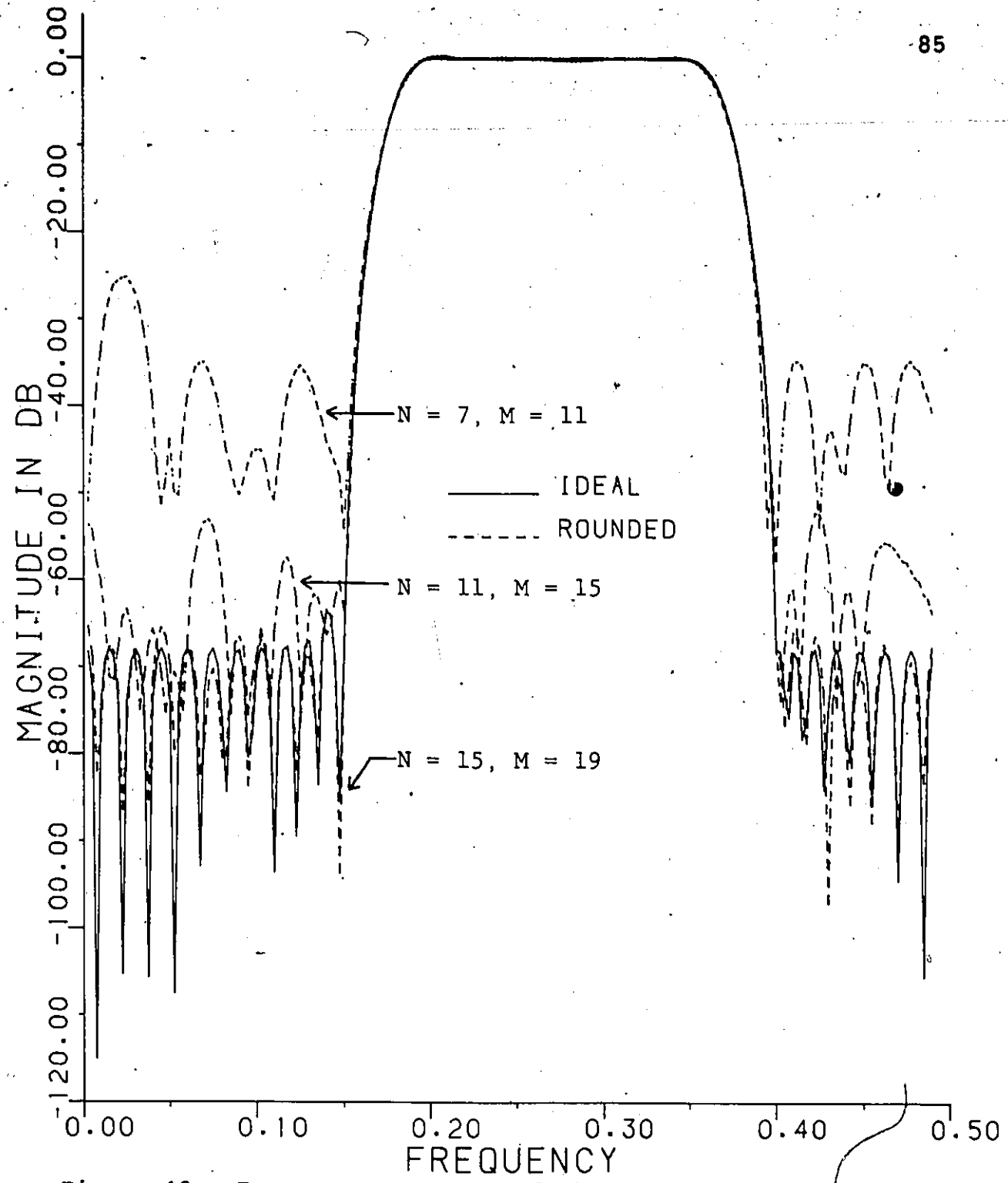


Figure 40: Frequency response of the 64 tap bandpass FIR filter.

Chapter IV

APPLICATION OF PIPELINED STORED SQUARE FIR FILTER TO ADAPTIVE FILTERING

4.1 INTRODUCTION

One of the advantages of the stored square/multiplication technique over the distributed arithmetic and the stored product technique is the independence of the ROM contents from the multiplier coefficients. Such an advantage makes the use of stored square technique potentially attractive to adaptive filtering applications where variability of the coefficients is necessary.

In this chapter the implementation of an adaptive digital filter using the stored square ROM multipliers will be described. The implementation will be based on the pipelined digital filter of chapter III, with high sampling rates, in excess of 15 MHz, obtainable. The coefficients adaptation is based on the Least Mean Square (LMS) algorithm. A review of this algorithm is given in section 4.2.

An evaluation of such implementation is performed by computer simulation, with the filter applied as a parameter estimator to model the impulse response of an unknown filter and subsequently as an adaptive noise canceller.

4.2 A REVIEW OF ADAPTIVE FILTER THEORY

In this section a review of the adaptive filter concept and of the least mean square (LMS) algorithm as a technique of minimizing the mean square error is given. Much of the introductory material in this section can be found in [15].

4.2.1 Concept of The Adaptive Filter

An adaptive filter is a self optimizing filter, basing its internal adjustment settings upon estimated (measured) statistical characteristics of input and output signals. The coefficient values are varied by a recursive algorithm that automatically updates the system adjustments with the arrival of each new data sample. Here, the filter to be considered for realization is the non-recursive digital filter.

Fig. 41 illustrates the adaptive linear combiner. A set of L measurements $x_i(t)$, with i=0,1,....,L-1, is sampled to form L sampled signals $x_i(n)$, where n is the time index. Each of the sampled signals is multiplied by a corresponding coefficient (weight) a_i and the products are summed to form an output y(n). The output y(n) is then subtracted with a desired response d(n) to obtain the error $\varepsilon(n)$. The objective is to adjust the coefficients in such a way that the error is minimized in a least mean squared sense.

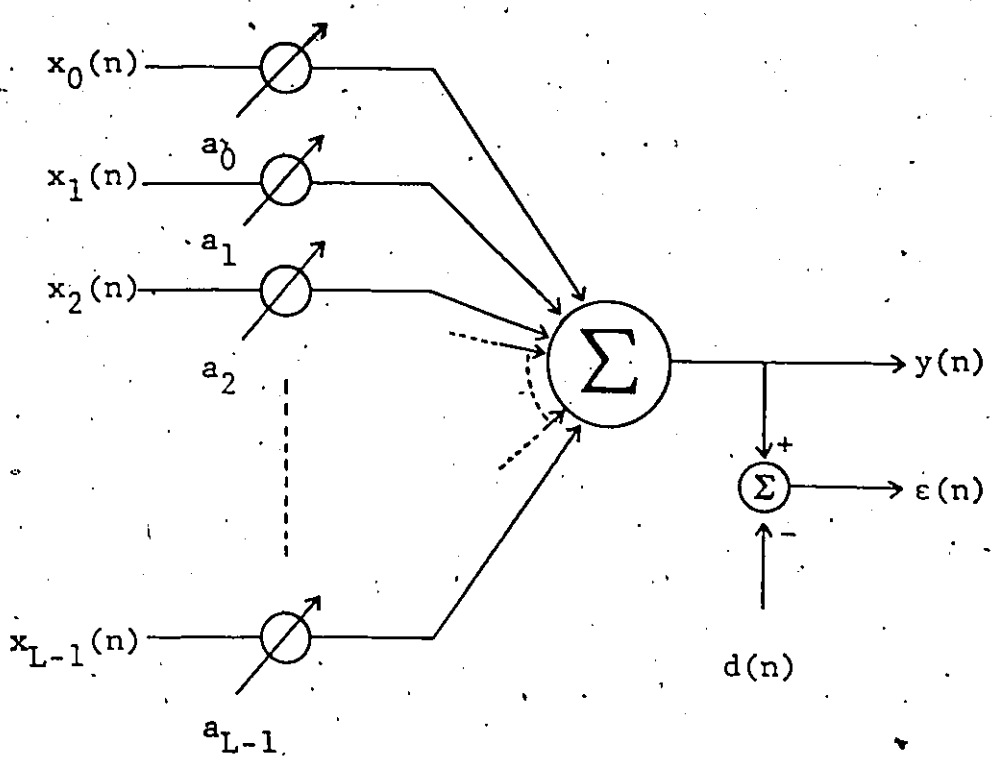


Figure 41: Adaptive linear combiner

The n^{th} output of the linear combiner can be represented as

$$y(n) = \sum_{i=0}^{L-1} a_i \, x_i(n) \qquad (4.1)$$

or in the matrix form as :

$$y(n) = [x(n)]^T[A] = [A]^T[x(n)] \qquad (4.2)$$

where [x(n)] and [A] are respectively the input sampled signal and the coefficient vectors.

The error is given by :

$$\varepsilon(n) = d(n) - y(n) = d(n) - [A]^T[x(n)] \qquad (4.3)$$

The square of the error is :

$$\varepsilon^2(n) = d^2(n)+[A]^T[x(n)][x(n)]^T[A]-2d(n)[x(n)]^T[A] \qquad (4.4)$$

The mean square (expected value) of (4.4) is :

$$E\{\varepsilon^2(n)\} = E\{d^2(n)\} + [A]^T E\{[x(n)][x(n)]^T\}[A] - 2E\{d(n)[x(n)]^T\}[A] \qquad (4.5)$$

or rewritten :

$$E\{\varepsilon^2(n)\} = E\{d^2(n)\} + [A]^T[R][A] - 2[P]^T[A] \qquad (4.6)$$

where [P] is the cross-correlation vector between the desired response and the vector [x(n)] which is defined as :

$$[P] \triangleq E\{d(n)[x(n)]\} = E\{d(n)x_1(n), d(n)x_2(n), \ldots, d(n)x_{L-1}(n)\}^T \qquad (4.7)$$

and [R] is the input correlation matrix defined as :

$$[R] \triangleq E\{[x(n)][x(n)]^T\} = E\begin{Bmatrix} x_0(n)x_0(n) & x_0(n)x_1(n)\ldots\ldots\ldots \\ x_1(n)x_0(n) & \\ \vdots & \\ & x_{L-1}(n)x_{L-1}(n) \end{Bmatrix} \qquad (4.8)$$

It may be observed from (4.5) and (4.6), that, for stationary input signals the mean square error is a second order function of the coefficients and can be pictured as a

concave hyperparaboloidal surface, a function with a unique minimum. The minimum error is found by descending along this surface which is accomplished by adjusting the coefficients. For this purpose the gradient methods are commonly used.

The gradient at any point on the error surface may be obtained by differentiating the mean square error function of (4.6) with respect to the coefficient vector.
The gradient is :

$$\nabla[E\{\varepsilon^2(n)\}] \triangleq \left\{\frac{\partial E\{\varepsilon^2(n)\}}{\partial a_1}, \ldots\ldots, \frac{\partial E\{\varepsilon^2(n)\}}{\partial a_{L-1}}\right\}^T \qquad (4.9)$$

The optimal (or Wiener) coefficient vector $[A]^*$ is obtained by setting the gradient to zero. Accordingly,

$$[A]^* = [R]^{-1}[P] \qquad (4.10)$$

Equation (4.10) is the Wiener-Hopf equation in matrix form. Substituting (4.10) into (4.6), we obtain the minimum mean square error (mse) as :

$$\text{minimum mse} = E\left\{\varepsilon^2(n)\right\}_{min} = E\left\{d^2(n)\right\} - [A]^{*T}[P] \qquad (4.11)$$

4.2.2 The Least Mean Square (LMS) Algorithm

One way of finding the optimum coefficient vector is simply to solve equation (4.10), but it presents serious computational problems when the number of coefficients (L) is large and when input data rates are high. It requires the inversion of LxL matrix and in addition, it requires as many as [L(L+1)]/2 auto-correlation and cross-correlation measurements to be made for the elements of vectors [R] and [P].

Another method is to find the approximate solutions to equation (4.10). The accuracy of this method is limited by the statistical sample size, since the coefficient values found are based on finite time measurements. This method does not require the matrix inversion nor the explicit measurements of correlation functions.

It is called the LMS (least mean square) algorithm which is based on the method of steepest descent and is given by :

$$[A(n+1)] = [A(n)] - \mu \nabla [E\{\varepsilon^2(n)\}] \qquad (4.12)$$

where $[A(n+1)]$ = coefficient vector after adaptation

$[A(n)]$ = coefficient vector before adaptation

$\nabla [E\{\varepsilon^2(n)\}]$ = instantaneous gradient of the mean square error

μ = the step size which controls the convergence rate and stability.

A crude but efficient way of estimating the instantaneous gradient $\nabla[E\{\varepsilon^2(n)\}]$ is by assuming that the square of a single error sample $\varepsilon^2(n)$ is an estimate of the mean square error, then,

$$\nabla[E\{\varepsilon^2(n)\}] = \left[\frac{\partial\varepsilon^2(n)}{\partial a_1}, \ldots\ldots, \frac{\partial\varepsilon^2(n)}{\partial a_{L-1}}\right]^T_{[A]=[A(n)]}$$

$$= 2\varepsilon(n)\left[\frac{\partial\varepsilon(n)}{\partial a_1}, \ldots., \frac{\partial\varepsilon(n)}{\partial a_{L-1}}\right]^T_{[A]=[A(n)]} \qquad (4.13)$$

From equation (4.3),

$$\nabla[\varepsilon(n)] = \nabla[d(n) - [A(n)]^T[x(n)]] = -[x(n)] \qquad (4.14)$$

Thus,

$$\nabla[E\{\varepsilon^2(n)\}] = 2\varepsilon(n)\nabla[\varepsilon(n)]$$

$$= -2\varepsilon(n)[x(n)] \qquad (4.15)$$

Therefore, the LMS algorithm of (4.12) becomes :

$$[A(n+1)] = [A(n)] + 2\mu\varepsilon(n)[x(n)] \qquad (4.16)$$

This algorithm will converge in the mean and will remain stable as long as μ is bounded by :

$$1/\lambda_{max} > \mu > 0$$

where λ_{max} is the largest eigenvalue of [R].

4.3 ADAPTIVE TRANSVERSAL FILTER

One of the most common methods of applying the LMS algorithm is to connect the combiner to a tapped delay line to form the adaptive transversal filter of Fig. 42. The impulse response of this filter is equivalent to its coefficient vector, consequently, it can approximate any impulse response and hence any frequency response if the number of taps is large enough. The LMS algorithm of this adaptive transversal filter can be written as :

$$a_i(n+1) = a_i(n) + 2\mu\varepsilon(n)x(n-i) \; ; \; i=0,1,\ldots,L-1 \qquad (4.17)$$

where $a_i(n+1)$ = the coefficient of the i^{th} tap after adaptation

$a_i(n)$ = the coefficient of the i^{th} tap before adaptation

$x(n-i)$ = the input signal at the i^{th} tap.

The L weighted adaptive transversal filter output can be expressed as :

$$y(n) = a_0(n)x(n)+a_1(n)x(n-1)+\ldots\ldots+a_{L-1}(n)x(n-L+1) \qquad (4.18)$$

or, after adaptation :

$$y(n+1) = a_0(n+1)x(n+1)+a_1(n+1)x(n)+\ldots+a_{L-1}(n+1)x(n-L+2) \qquad (4.19)$$

Substituting (4.17) into (4.19), we have

$$y(n+1) = [a_0(n)+2\mu\varepsilon(n)x(n)]x(n+1)+[a_1(n)+2\mu\varepsilon(n)x(n-1)]x(n)$$

$$+\ldots\ldots+[a_{L-1}(n)+2\mu\varepsilon(n)x(n-L+1)]x(n-L+2) \qquad (4.20)$$

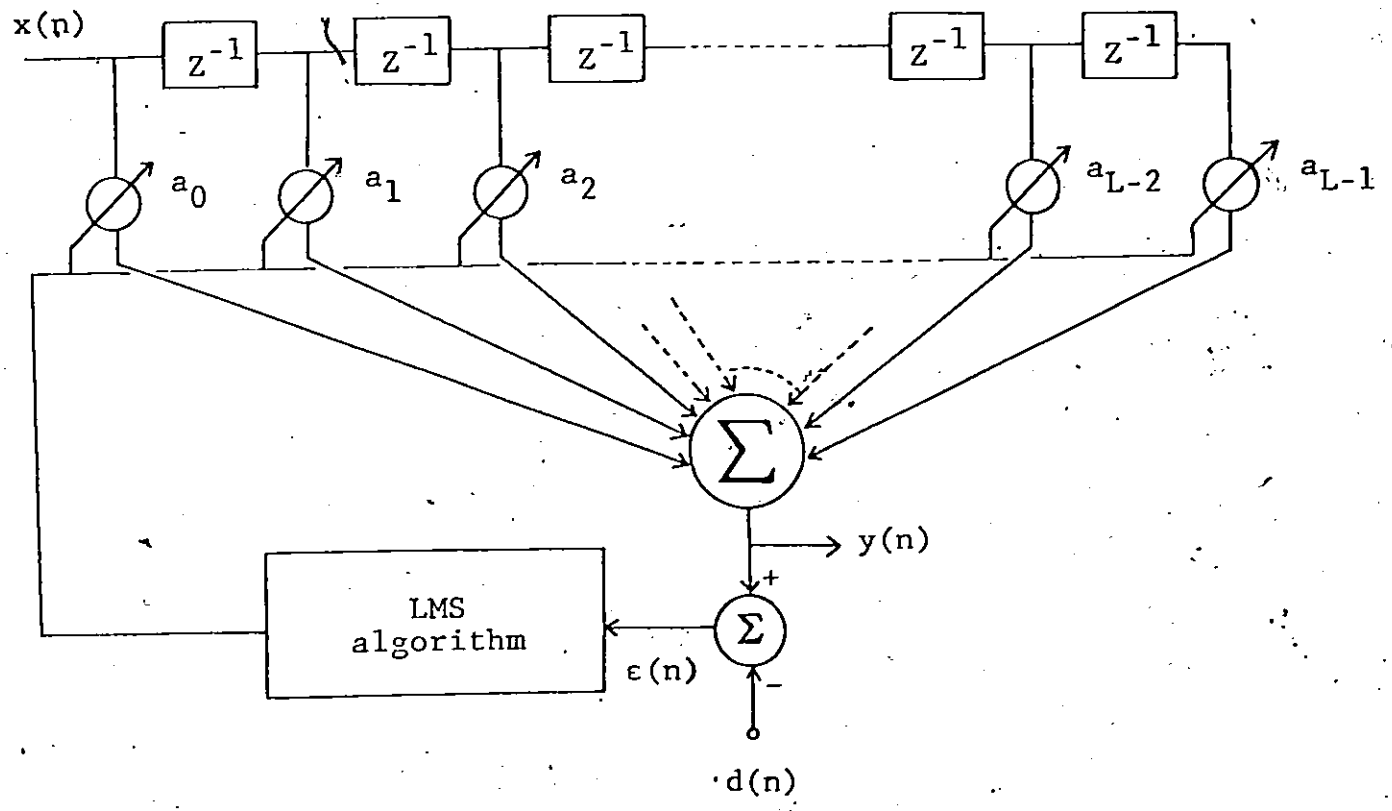


Figure 42: L - weighted adaptive transversal filter.

It has been shown [18] that the update of the filter coefficient need not necessarily take place in the time period directly following n, but may be delayed by some arbitrary time lag n', so equation (4.17) can be rewritten as :

$$a_i(n+n') = a_i(n) + 2\mu\varepsilon(n)x(n-i) \; ; \; i=0,1,\ldots,L-1 \qquad (4.21)$$

From this fact, an adaptive filter with high sampling rate and with minimal hardware complexity can be implemented, since, it is not necessary to update all the coefficients at each sampling time interval. This technique will be applied in the next section for the realization of an adaptive filter using the pipelined stored square FIR filter technique.

4.4 ADAPTIVE FILTER USING PIPELINED STORED SQUARE FIR FILTER

Equation (4.20) shows that each coefficient is updated based on the error $\varepsilon(n)$ and its corresponding delayed input signal $x(n-i)$. A direct implementation of the LMS algorithm based on (4.20) will require a large amount of hardware, since, for each coefficient update it requires two multipliers and an adder. This parallel updating implementation can be avoided by an alternative approach of burst updating implementation where two multipliers and an adder are multiplexed to update all the coefficients at each sampling time interval, however, this method will slow down the sampling rate by a factor of L for an L weighted filter. Another ap-

proach which is a compromise between the two is the serial updating implementation, where at each sampling time interval only one coefficient is updated, so maintaining high sampling rate and at the same time keeping the hardware requirements to a minimum, however, the penalty of this technique is a slower convergence rate, which is however still acceptable for many applications.

The implementation of an adaptive stored square digital filter with pipeline configuration and serial updating technique is shown in Fig. 43. The error $\varepsilon(n)$ is latched and held for at least L samples where in between that time one coefficient is updated at each sampling time interval.

Since the output $y(n)$ is delayed by 5 time samples due to the pipelining process, the output error $\varepsilon(n)$ that has to be multiplied by 2μ will arrive at the multiplier after 7 sample times, therefore, the input signal $x(n)$ has to be deliberately delayed by 7 unit samples before coming to the LMS update circuit. The multiplication between the output error $\varepsilon(n)$ and the convergence factor 2μ is done with the stored product technique [4] [9]. Notice that, since a pipelined SSRM is used as one of the multiplier in the update circuit, the output product will be delayed by 4 time samples, and hence, this delay has to be inserted between latching time L_1 and L_{a_0} in addition to delays by addition and multiplication of 2μ as shown in the timing diagram of Fig. 44.

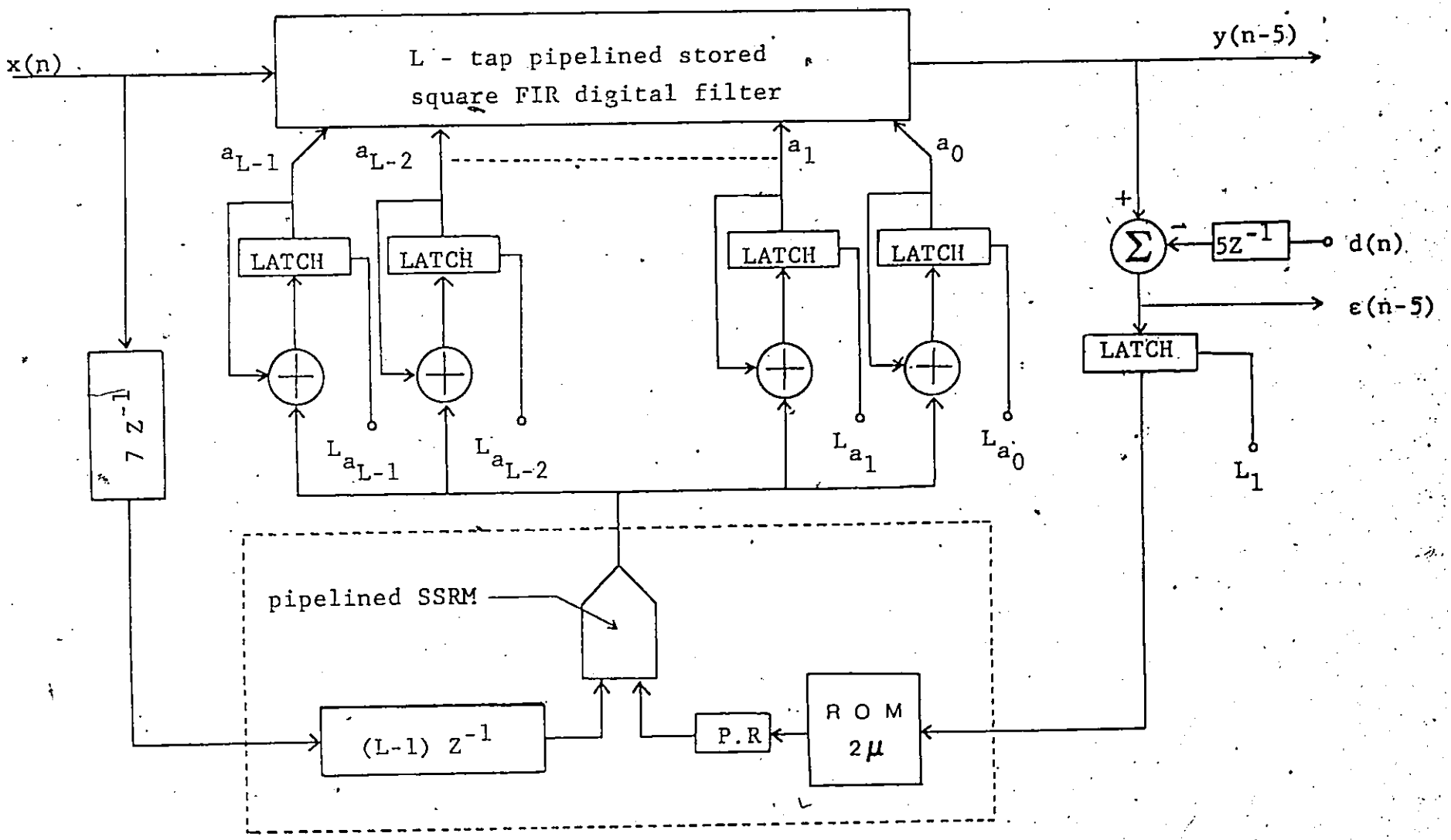


Figure 43: Adaptive stored square digital filter.

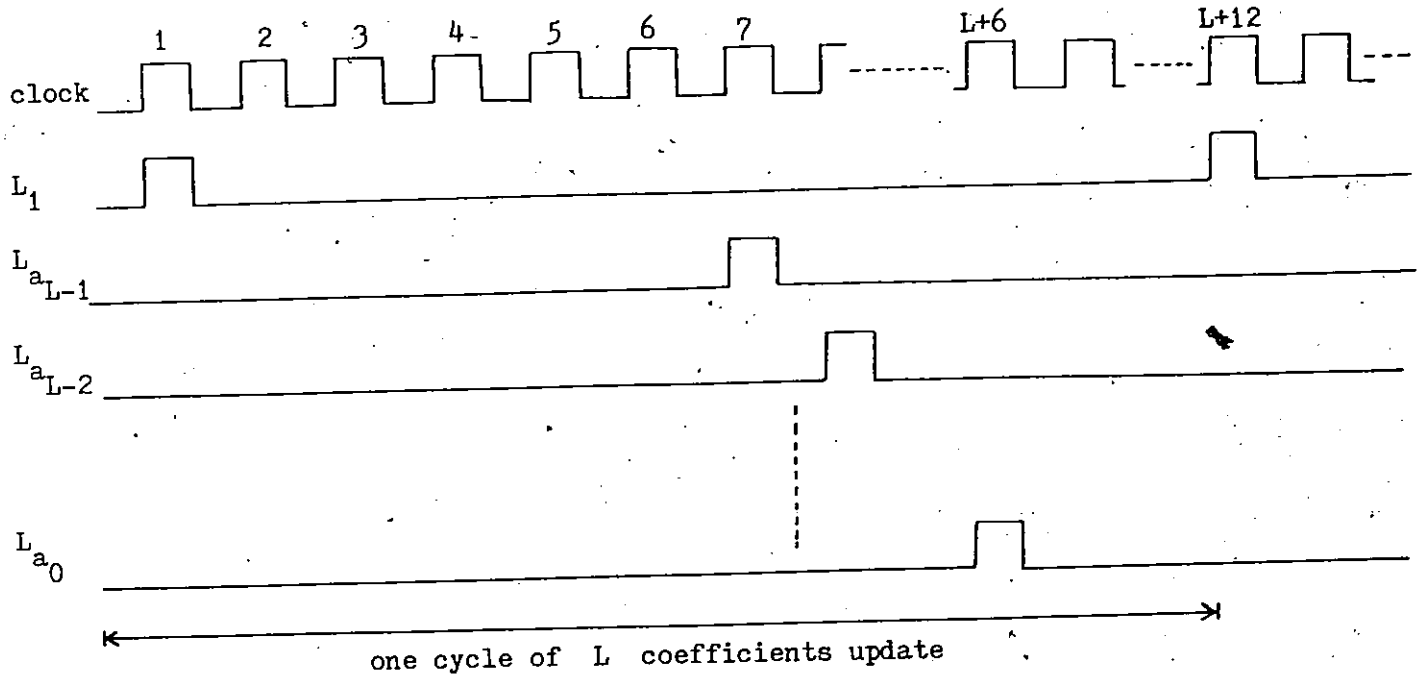


Figure 44: Timing diagram of latching sequence.

4.5 COMPUTER SIMULATION RESULTS

Two applications of the adaptive stored square digital filter were conducted in the simulations, namely the adaptive parameter estimator and the adaptive noise canceller.

The objectives of the simulations are to show that a definite convergence is indeed achievable with the realizations using the stored square technique and to demonstrate that the adaptive noise canceller is applicable to high frequency operation.

4.5.1 Adaptive Parameter Estimator

The adaptive filter can be applied as a parameter estimator with the system model shown in Fig. 45. The sampled input signal x(n) is applied to the adaptive filter with response y(n), as well as, to the unknown filter with response d(n). The difference between y(n) and d(n) forms the error $\varepsilon(n)$ which is minimized by the LMS algorithm to produce the model of the unknown filter (or system). The model of Fig. 45 was simulated by a computer. A gaussian random input sampled signal x(n) was used in the simulation with all the coefficients were initially set to zero.

Figure 46 shows the computer simulation results of adapting a 23 tap FIR type system using a 16 tap adaptive filter with input signal and product wordlengths of 11 bits and 15

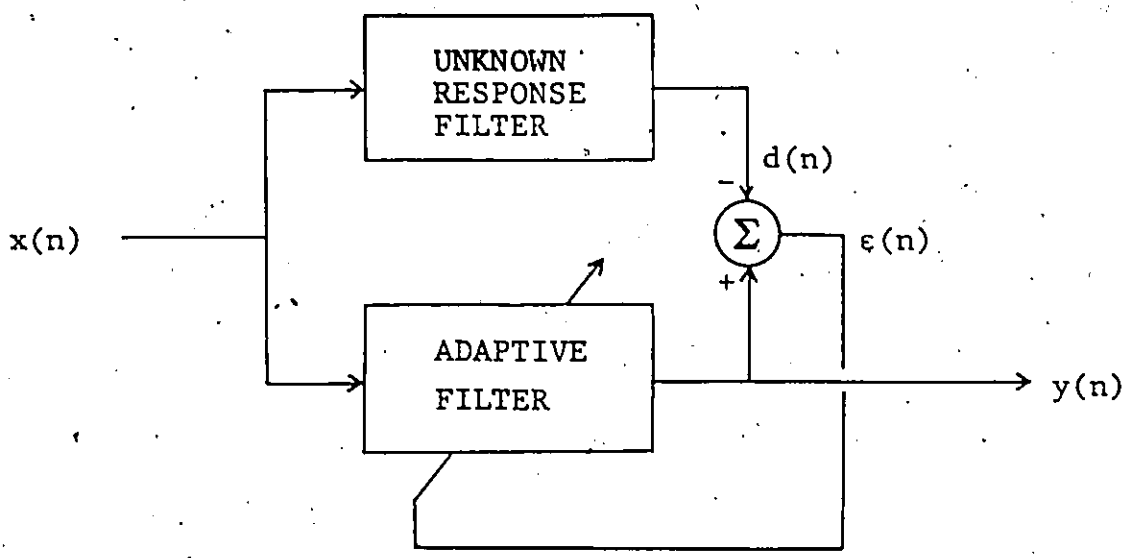


Figure 45: Adaptive parameter estimator to model the output response of an unknown filter.

bits respectively. It can be seen that the mse reached its minimum value after approximately 3000 iterations[1], with a sampling frequency of 15 MHz the mse has a convergence time of 200 μs. If a 32 tap adaptive filter was used, the mse will reach its minimum value after approximately 6000 iterations[1] or has a convergence time of 400 μs as shown in figure 47.

The 32 tap adaptive filter has a slower convergence rate than the 16 tap adaptive filter which is expected due to the higher tap number. However, the 32 tap adaptive filter has a smaller minimum mse and from the frequency response after adaptation shown in figures 48 and 49 it is obvious that it has better output response than the 16 tap adaptive filter.

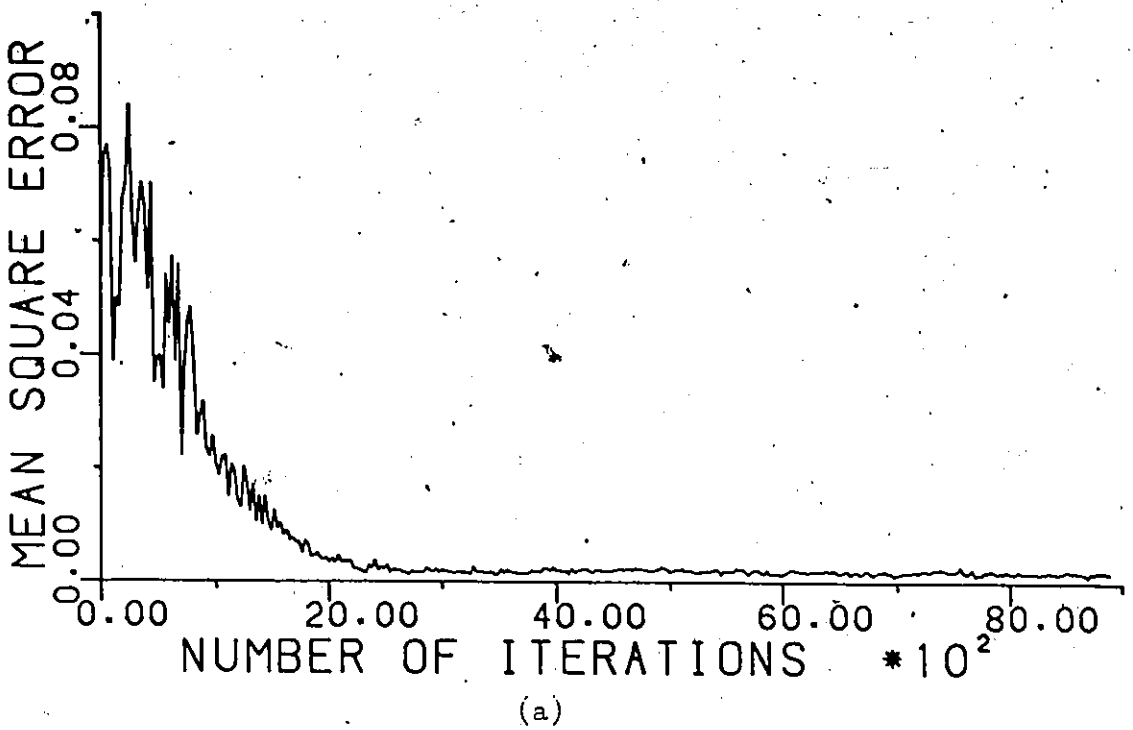
The results of adapting a sixth order Butterworth IIR (infinite impulse response) system are shown in figures 50- 52.

Although the convergence is somewhat slower with the serial updating technique compare to the fully implemented (parallel) updating technique, however, the computer simulation results show that it is not a serious problem, since, in most cases the convergence times are less than 10000 number of iterations and with a pipeline configuration where

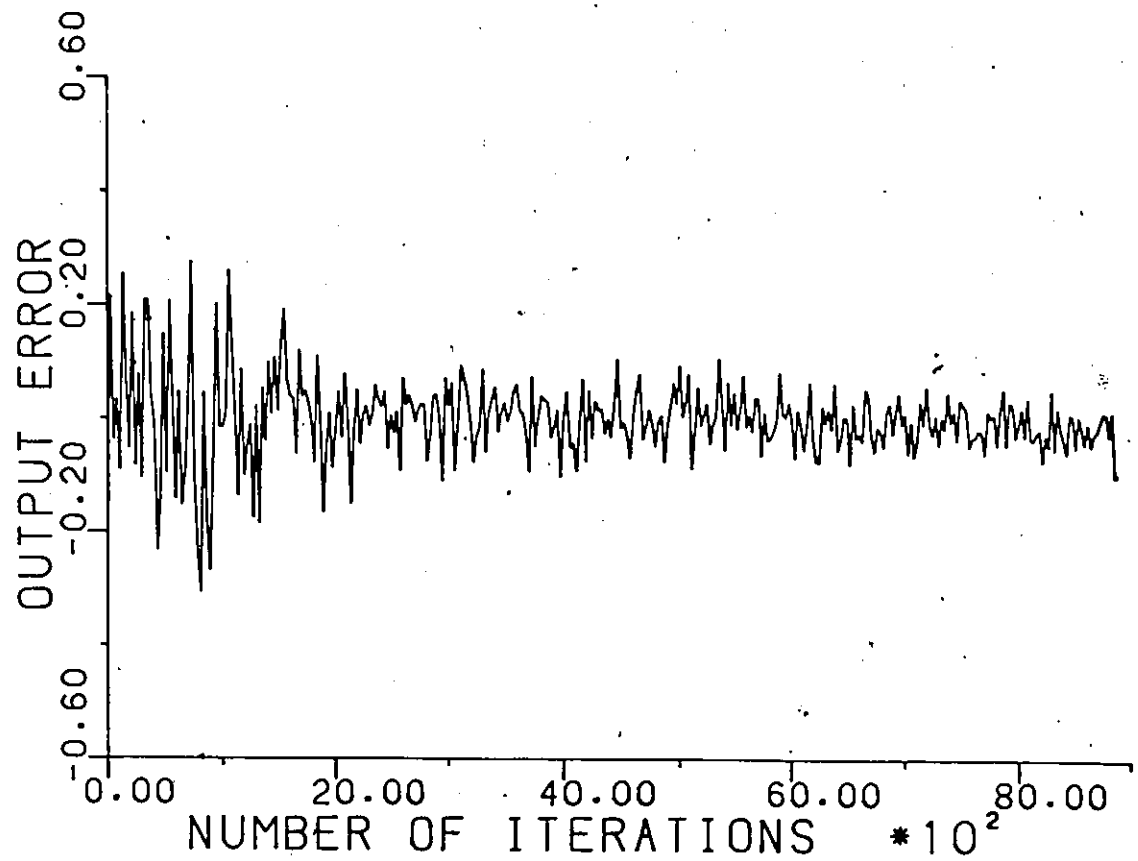
---

[1]Note that the convergence factors (μ) used in the simulations are not necessary the optimal ones, therefore, the convergence times shown do not represent the fastest possible times.

sampling frequencies in excess of 15 MHz are obtainable, then, the convergence times in most cases will be in the order of less than 700 μs which are acceptable for most applications.

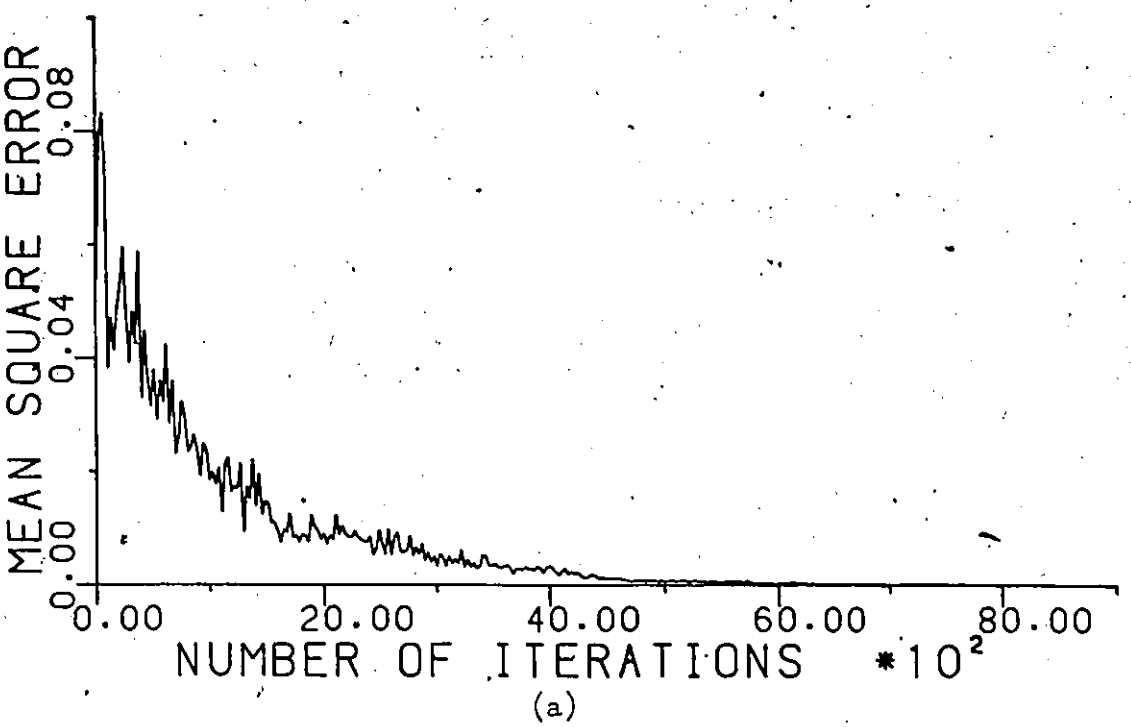


(a)

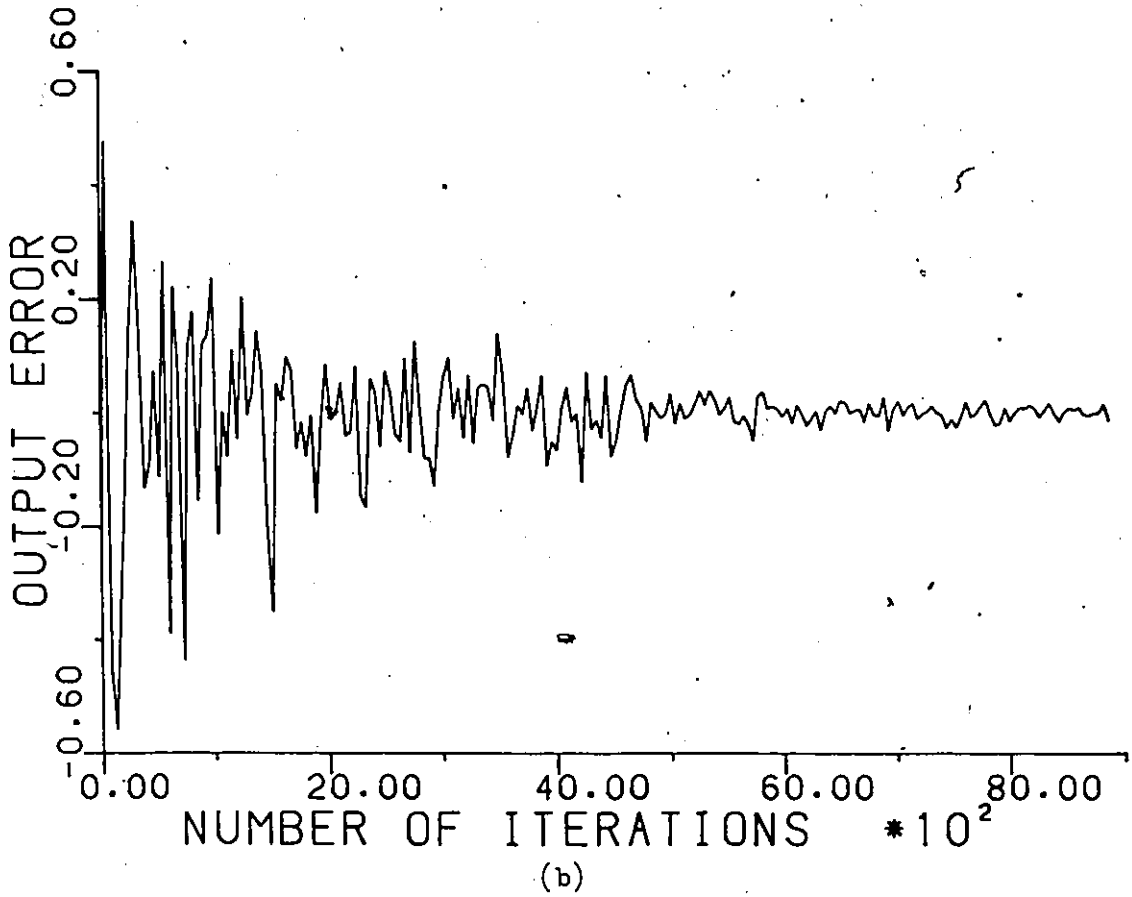


(b)

Figure 46: Results of adapting an FIR system using a 16 tap filter with N=11 and M=15, a) Mean square error, b) Output error.



(a)



(b)

Figure 47: Results of adapting an FIR system using a 32 tap filter with N=11 and M=15, a) Mean square error, b) Output error.

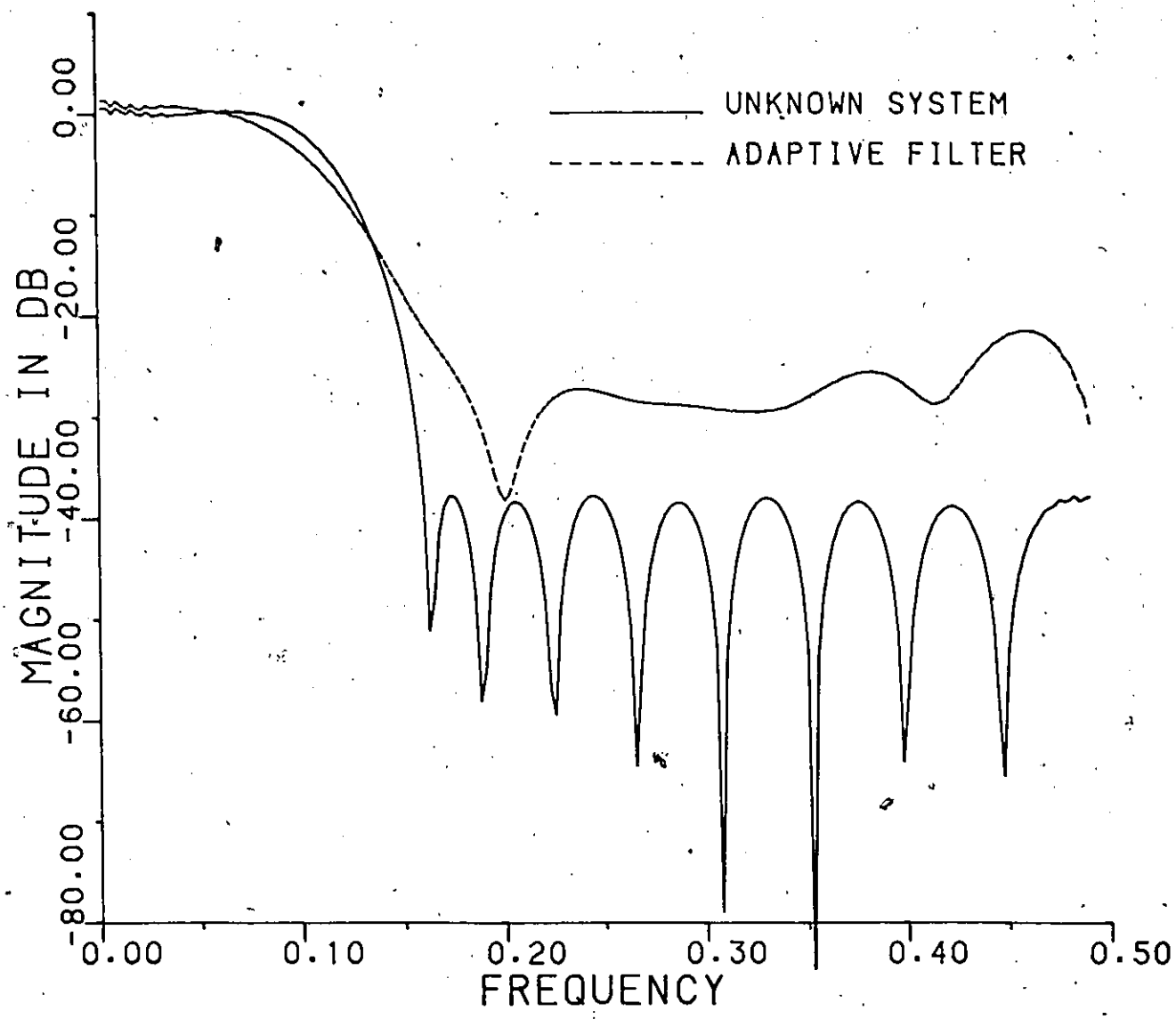


Figure 48: Output response of the 16 tap adaptive filter and the FIR type unknown system after adaptation with N=11 and M=15.

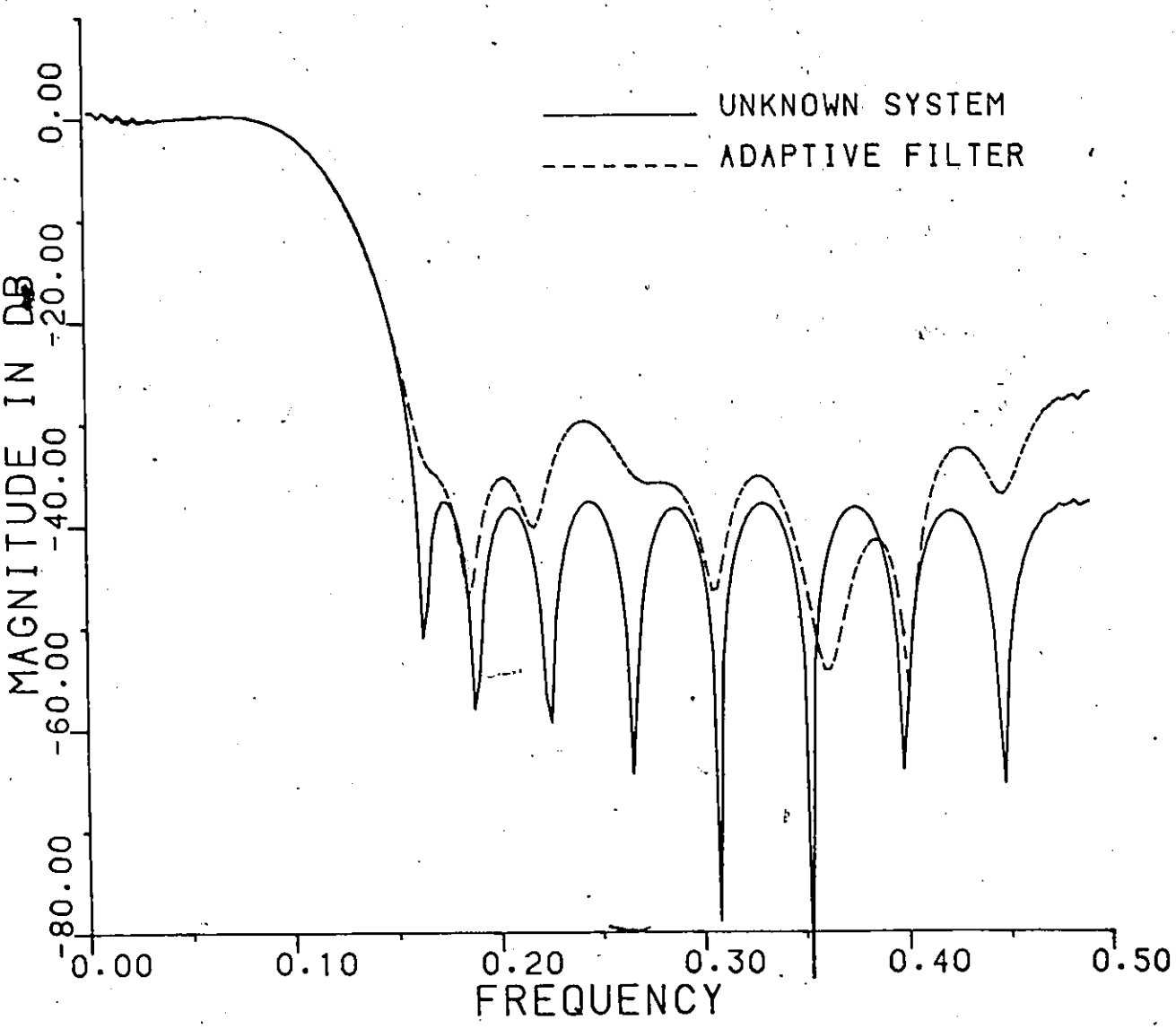
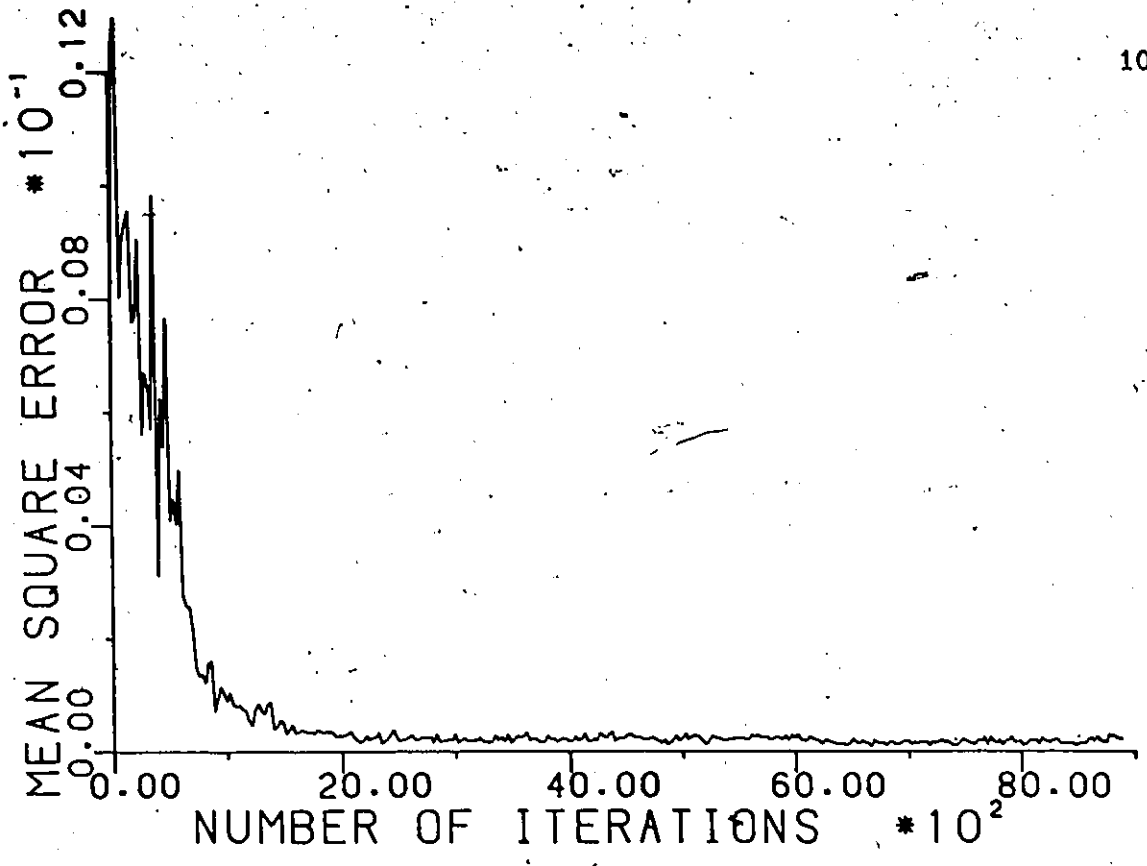
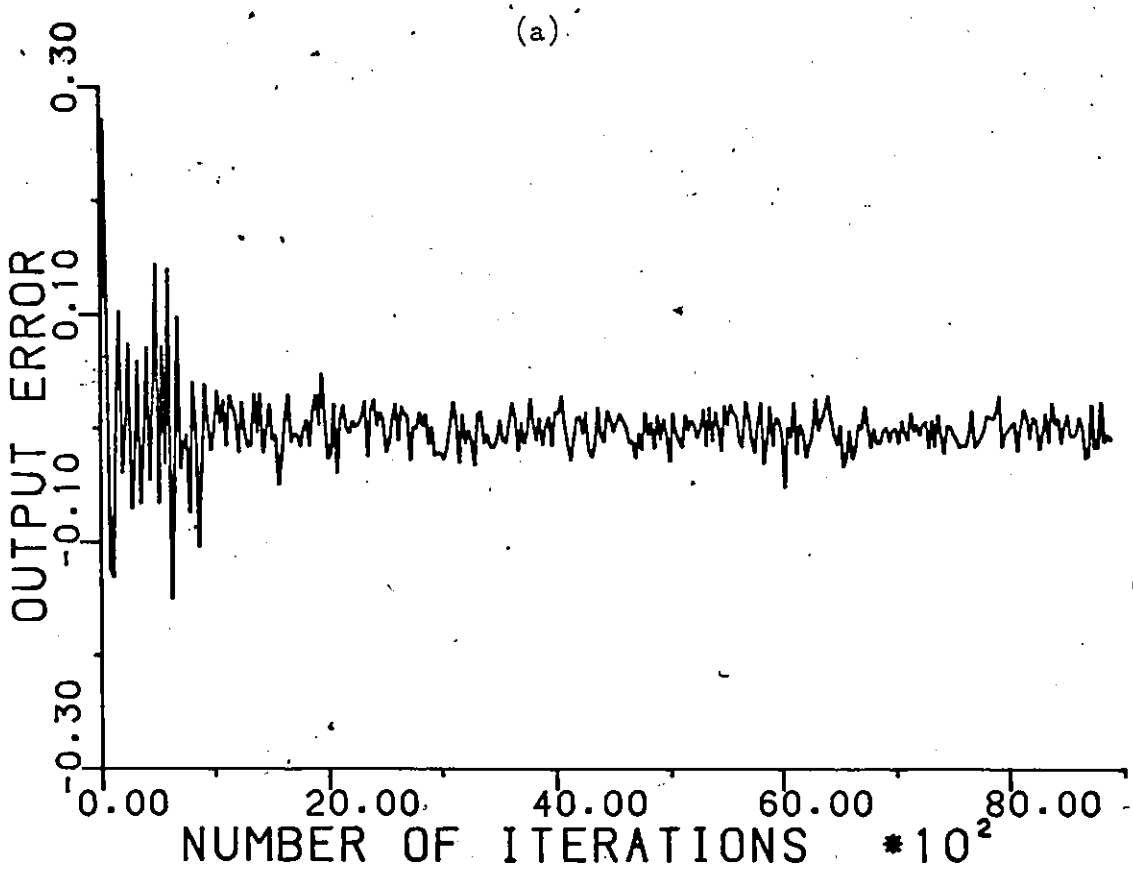


Figure 49: Output response of the 32 tap adaptive filter and the FIR type unknown system after adaptation with N=11 and M=15.



(a)



(b)

Figure 50: Results of adapting an IIR system using a 16 tap filter with N=11 and M=15, a) Mean square error, b) Output error.

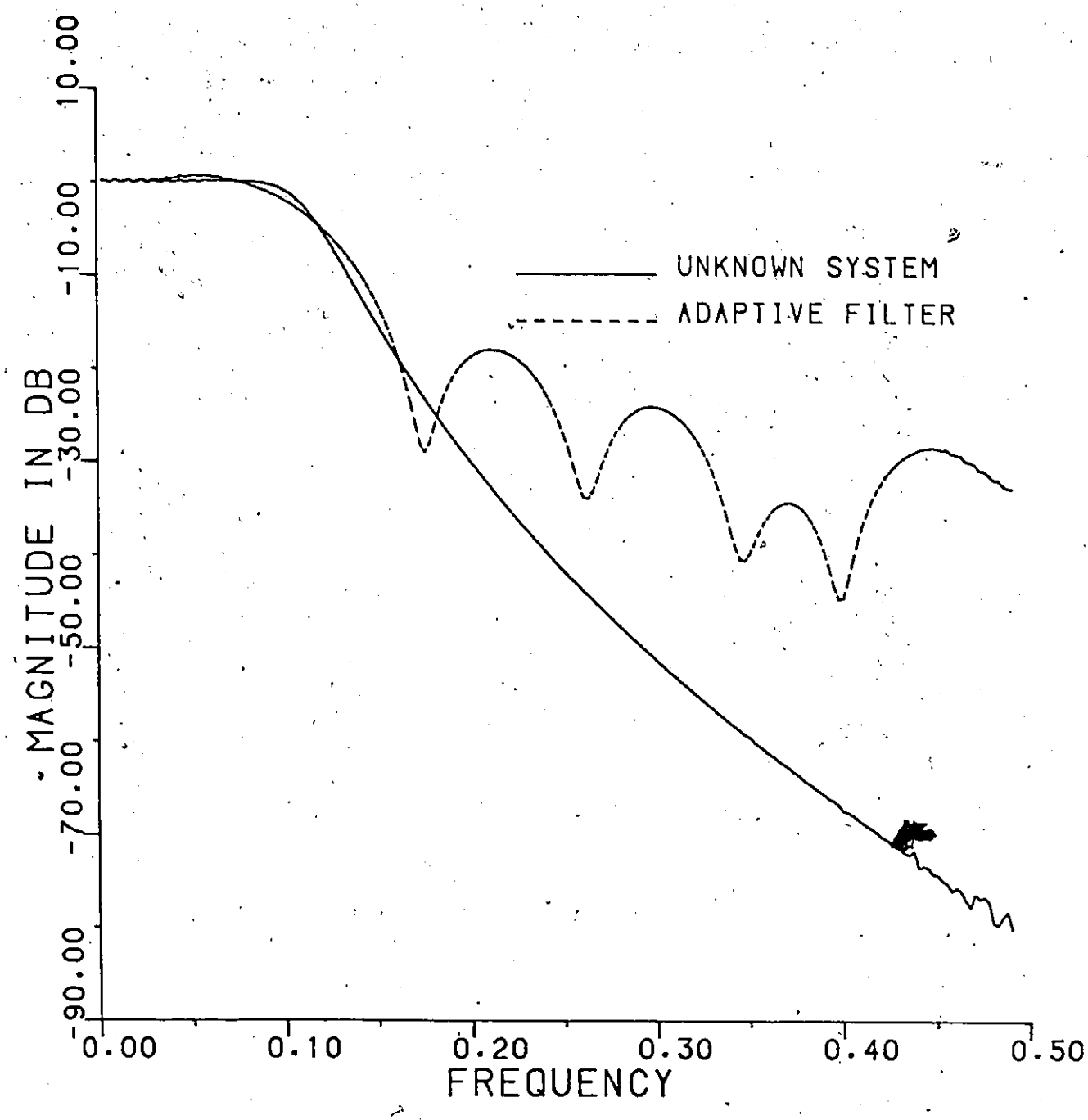


Figure 51: Output response of the 16 tap adaptive filter and the IIR type unknown system after adaptation with N=11 and M=15.

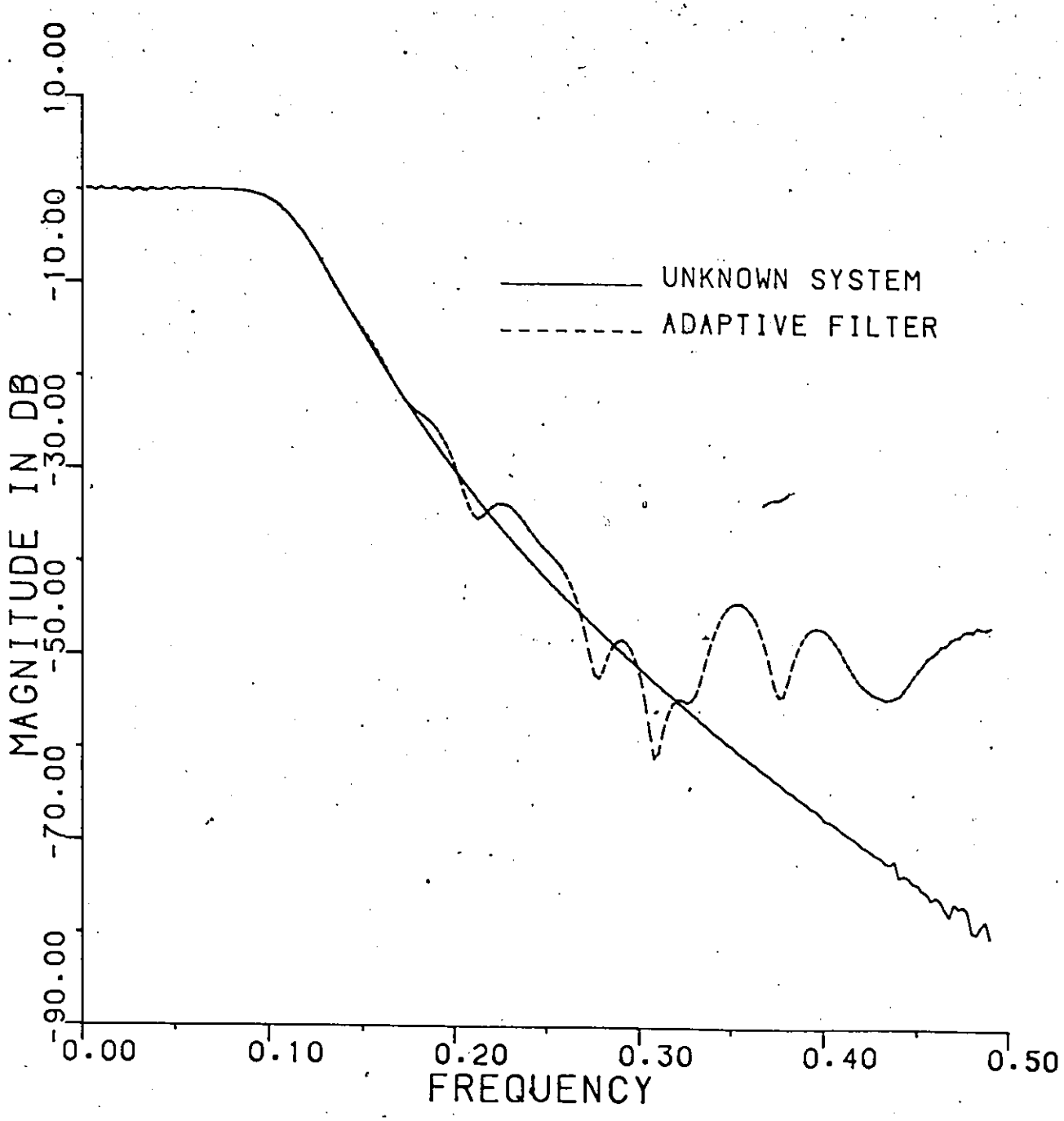


Figure 52: Result of adapting an IIR type system using a 32 tap filter with N=11 and M=15.

4.5.2 Adaptive Noise Canceller

Another commonly used application of the adaptive filter is noise cancellation. The basic principles of adaptive noise cancelling are illustrated in Fig. 53. The input to the adaptive filter is a noise signal (n_1) which is correlated with the additive noise (n_0) of the primary input but is uncorrelated with the input signal s(n).
The noise signal n_1 in one way can be derived from a sensor located at a point in a noisy field where the input signal s(n) is undetectable. The objective of the filter is to produce $\hat{n}_0$ which is an estimate of n_0 in a cause to cancel the additive noise n_0. It has been proven [15] that, providing s(n) is uncorrelated with both n_0 and n_1, adapting the filter to produce a system with the least output energy of z(n) is equivalent to causing the output to be a best least squares estimates of the input signal s(n).

In the simulation, the adaptive stored square digital FIR filter is applied as a noise canceller for a voice frequency signal in a noisy channel. Input signal s(n) and output signal z(n) are of 15 bit wordlength and the reference noise signal (n_1) is of 11 bit wordlength. The performance of the adaptive noise canceller is measured in terms of signal to noise ratio improvement. Fig. 54 shows the results of the noise cancellation of a noise corrupted 1 kHz sine wave input signal with an 8 tap filter. The sampling frequency used

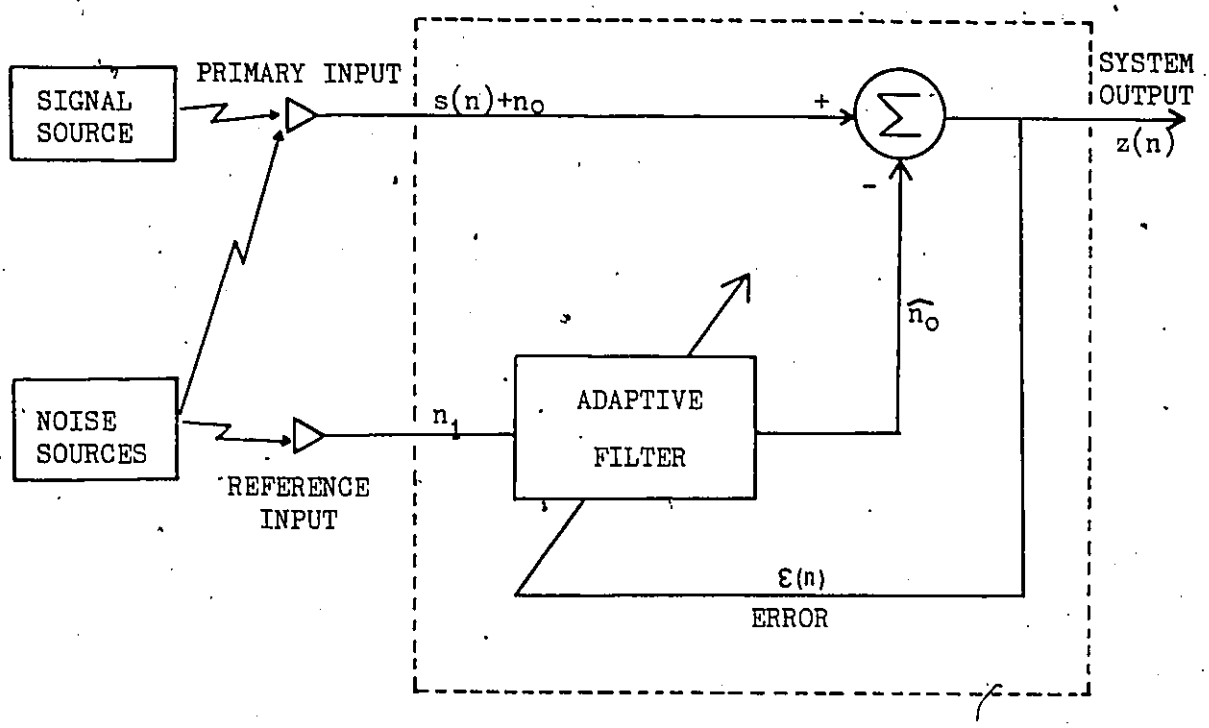
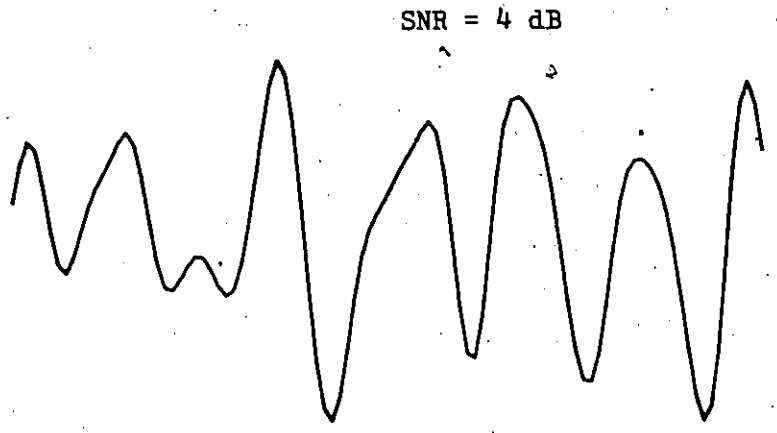


Figure 53: Adaptive noise cancelling model

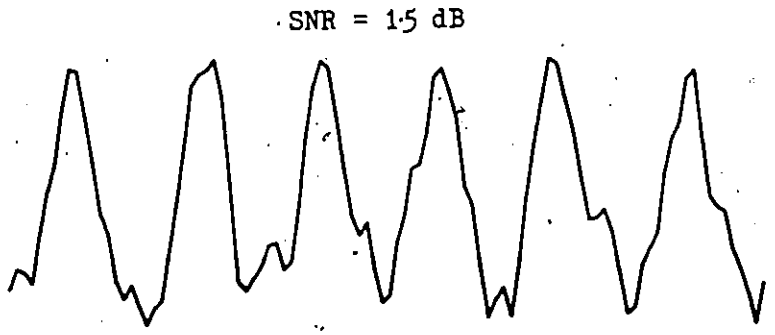
was 16 kHz with an 8 kHz bandlimited gaussian noise as the reference signal and the channel noise was simulated by filtering the reference noise with a sixth order Butterworth lowpass filter which has a 3 dB cut-off frequency of 1.6 kHz. The input signal was kept at 4 dB SNR (signal to noise ratio). After adaptation the output signal is about 15 dB above the noise level.

A higher frequency operation was simulated using a 40 kHz bandlimited gaussian noise as the reference signal with a sampling frequency of 100 kHz. The channel noise was again simulated by filtering the reference noise with a sixth order Butterworth lowpass filter of 3 dB cut-off frequency at 10 kHz. To represent a wide band signal, a 1 kHz square wave input signal was used. Fig. 55 shows the results of noise cancelling a -18 dB SNR input signal. After adaptation the output SNR is improved to 15.7 dB using a 16 tap filter.

The performance of the adaptive noise canceller with input signal to noise ratio values ranging from -18 dB to 14 dB are tabulated in Table 12. The table shows that at -18 dB the output signal demonstrates about a 30 dB improvement in SNR for both the 16 tap and the 32 tap filters, while on the other hand, the 8 tap filter shows an improvement of only about 15 dB. At 14 dB input SNR almost the same improvements of about 9 dB are obtained from all the three filters.

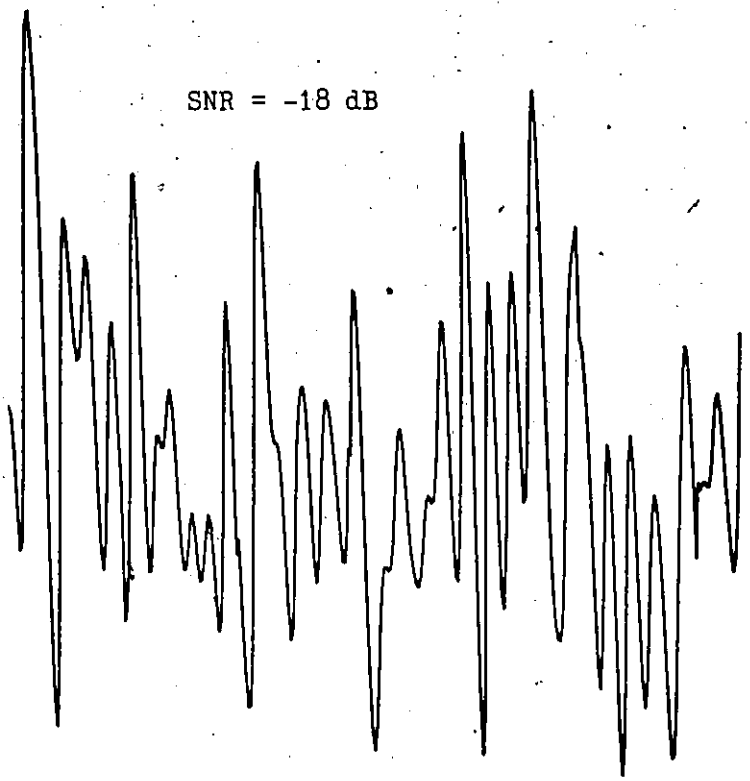


(a)

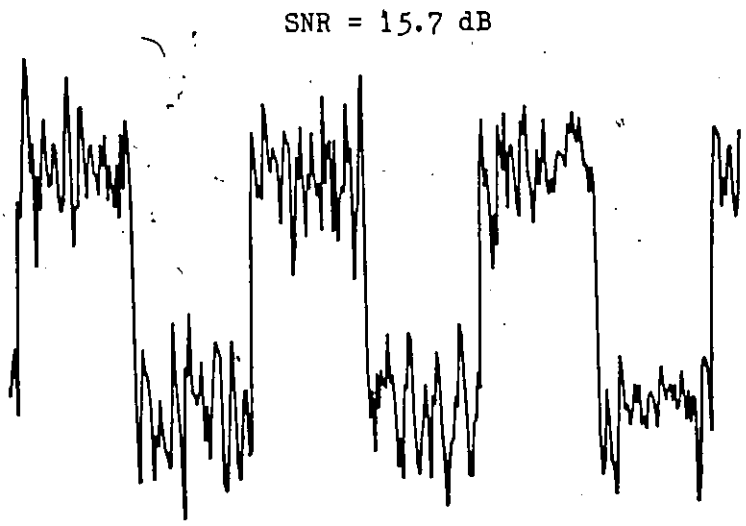


(b)

Figure 54: Results of noise cancelling 1 kHz sine wave at 16 kHz sampling frequency. a) Input signal. b) Output signal.



(a)



(b)

Figure 55: Results of noise cancelling 1 kHz square wave at 100 kHz sampling frequency. a) Input signal. b) Output signal.

TABLE 12

RESULTS OF ADAPTIVE NOISE CANCELLING 1 KHZ SQUARE WAVE INPUT SIGNAL AT 100 KHZ SAMPLING FREQUENCY

INPUT-SNR	OUTPUT-SNR		
	8 TAP	16 TAP	32 TAP
-18.0 dB	-3.3 dB	15.7 dB	17.9 dB
-11.0 dB	3.5 dB	17.2 dB	19.2 dB
- 8.0 dB	6.3 dB	18.4 dB	20.7 dB
0.0 dB	13.3 dB	20.2 dB	22.3 dB
2.7 dB	15.5 dB	21.8 dB	22.3 dB
5.8 dB	17.7 dB	22.0 dB	23.6 dB
9.4 dB	20.1 dB	22.6 dB	24.0 dB
14.0 dB	22.9 dB	23.4 dB	24.4 dB

The results show that the more severe the noise, the more dramatic the improvement in SNR. This adaptive noise cancelling scheme is very suitable for signal detection applications where the input signal is severely corrupted by noise and the statistical property of the noise is unknown which makes it impossible to simply use a fixed coefficient filter.

Chapter V

CONCLUSIONS

In this thesis the use of a stored square multiplication technique to replace conventional binary digital multipliers in filtering has been studied. Advantages in cost and speed of this technique, in integrated form, are anticipated as all ROM contents are to be identical and a multiplication throughput rate in the order of 20 MHz can be achieved with a pipeline implementation. In the construction of VLSI the identical ROM contents are very practical.

The error analysis has been evaluated for fixed point and floating point arithmetic. The stored squares are to be of the length M>N (M=stored squares wordlength and N=input signal and coefficient wordlength) in order to approach the error obtainable from direct binary multiplication. In general, M>N+3 for the product rounding to be comparable, however, for the floating point case the output noise to product power ratio does not approach the one obtainable by direct multiplication and it is found to remain larger even for the case when M>>N. Nevertheless, as for equal input signal wordlength (N) the required ROM storage for floating point is one-fourth of that required for fixed point and with one bit more in the mantissa, the floating point SSRM

will have an output noise to product power ratio comparable to the fixed point SSRM.

For address wordlengths of 12 bits and longer the ROM storage requirement can be reduced by means of "reduced ROM squaring circuit". A large reduction of the storage size can be obtained with this technique.

The applications of the stored square ROM multiplier to digital FIR filters and adaptive filters have been evaluated. With pipeline structure sampling frequencies in the order of 15 MHz or more can be obtained. It is shown that, in general, when the stored squares wordlength is more than 3 bits longer than the address wordlength the stored square digital FIR filters have an accuracy comparable to that of filters implemented using conventional digital multipliers.

The computer simulation results show that the convergence of the adaptive stored square digital FIR filter using the serial updating technique is somewhat slower than the parallel updating technique. This is not a serious problem since with high sampling frequencies of 15 MHz or more the convergence times in most cases will be in the order of less than 700 μs which are fast enough for many applications. The operation of the adaptive filter as a noise canceller has been simulated as well and satisfactory results have been demonstrated.

It is clear that the potential applications of the stored square ROM multiplication technique are in the area of digital signal processing where variability of the coefficients is necessary such as the case of adaptive filtering. This property makes the use of the stored square multiplication technique very attractive for digital signal processing applications. In terms of flexibility, this technique has a definite advantage over the distributed arithmetic and the stored product technique.

Since, the use of a stored square ROM multiplication technique in filtering is relatively new, therefore, there are still many areas remain to be studied. Here are some listing of suggested topics for further studies:

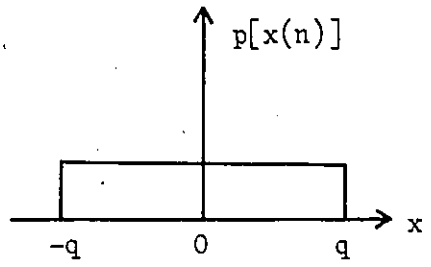
- Application of the stored square ROM multiplier in recursive digital filters.
- Adaptive filtering using recursive stored square digital filters.
- Microprocessor based digital filter implementation associated with the stored square ROM multiplier.
- LSI implementation of the stored square ROM multiplier.

Appendix A

DERIVATION OF THE MEAN AND THE MEAN SQUARE OF THE SQUARED SUM AND THE SQUARED DIFFERENCE OF TWO VARIABLES

Assume that $x(n)$ and c are each uniformly distributed between $-q$ and $+q$ with the probability density function (pdf) shown in Fig. A.1, then the pdf of s where $s=x(n)\pm c$ is

$$p(s) = -\frac{1}{(2q)^2}|s| + \frac{1}{2q} \quad ; \quad -2q \leqslant s \leqslant 2q \qquad (A.1)$$



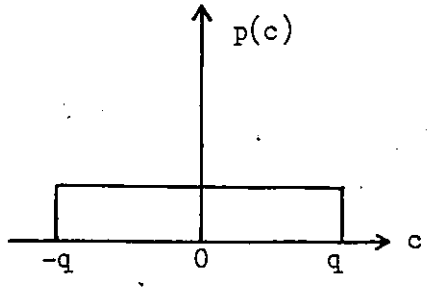


Figure A.1: Probability density function of x(n) and c.

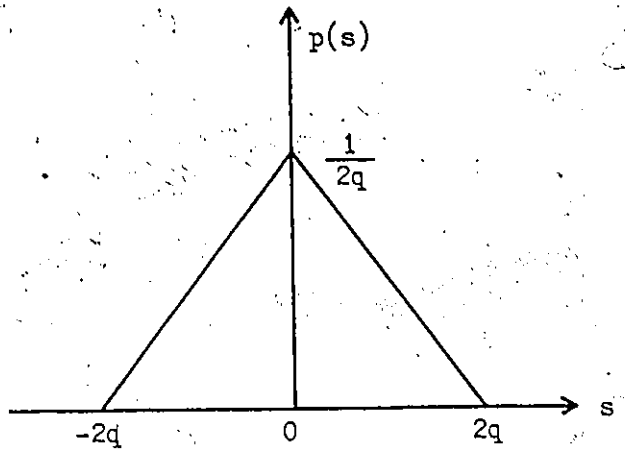


Figure A.2: Probability density function of $s=x(n)\pm c$.

Now, if $y=s^2$, then [7]

$$\overline{y} = \int_0^{(2q)^2} y\, g(y)\, dy \qquad (A.2)$$

where

$$g(y) = \frac{f(-s)+f(s)}{2\sqrt{y}} = \frac{-1/2q\,|s|+1}{2q\sqrt{y}} \qquad (A.3)$$

so,

$$\overline{y} = \frac{1}{2q} \int_0^{(2q)^2} y\, \frac{(-1/2q\,|s|+1)}{\sqrt{y}}\, dy$$

$$= \frac{1}{2q} \left\{ -\frac{1}{2q}\left[\frac{y^2}{2}\right]_0^{(2q)^2} + \int_0^{(2q)^2} \sqrt{y}\, dy \right\} \qquad (A.4)$$

Let $z = \sqrt{y}$ so $z^2 = y$ and $dy = 2z.dz$ therefore,

$$\overline{y} = \frac{1}{2q}\left\{-\frac{(2q)^3}{2} + 2[z^3/3]_0^{2q}\right\}$$

$$= \frac{2}{3}q^2 \qquad (A.5)$$

and

$$\overline{y^2} = \int_0^{(2q)^2} y^2 g(y)\,dy = \frac{1}{2q}\int_0^{(2q)^2} y^2\,\frac{(-1/2q|s|+1)}{\sqrt{y}}\,dy$$

$$= \frac{1}{2q}\int_0^{(2q)^2}\left(-\frac{y^2}{2q} + \frac{y^2}{\sqrt{y}}\right)dy$$

$$= \frac{16}{15}q^4 \qquad (A.6)$$

With $\alpha = \frac{(x(n)+c)^2}{4}$ and $\beta = \frac{(x(n)-c)^2}{4}$

then

$$\overline{\alpha} = \overline{\beta} = \tfrac{1}{4}(\overline{y}) = \frac{q^2}{6} \qquad (A.7)$$

and

$$\overline{\alpha^2} = \overline{\beta^2} = (\tfrac{1}{4})^2(\overline{y^2}) = \frac{q^4}{15} \qquad (A.8)$$

REFERENCES

[1] A. Peled and B. Liu, "A New Hardware Realization of Digital Filters", IEEE Trans. Acoust., Speech Signal Process., vol. ASSP-22, pp. 456-462, Dec. 1974.

[2] A. Croisier, D.J. Esteban, M.E. Levilion and V. Rizo, "Digital Filter for PCM Encoded Signals", U.S. Patent 3 777 130, Dec. 3, 1973.

[3] O. Monkewich and W. Steenaart, "Companding for Digital Filters", Proc. 1975, IEEE ISCAS, pp. 68-71.

[4] O. Monkewich and W. Steenaart, "Stored Product Digital Filtering With Non-linear Quantization", Proc. 1976, IEEE ISCAS, pp. 157-160..

[5] M. A. Soderstrand and E. L. Fields, "Multipliers for Residue Number Arithmetic Digital Filters," Electronics Letters, vol. 13, pp.164-166, March 17,1977.

[6] E.L. Johnson, "A Digital Quarter Square Multiplier", IEEE Trans. on Comp. vol-C29, No. 3, pp. 258-261, March 1980.

[7] J.B. Thomas, "An Introduction to Statistical Communication Theory", John Wiley and Sons Inc., pp. 30-31, 1968.

[8] W. Steenaart, D. Dubois and O. Monkewich, "Digital Filtering, Structure, Potential and Applications", Proc. 1981 European Conference on Circuit Theory and Design, pp. 118-126.

[9] D. Dubois and W. Steenaart, "High Speed Stored Product Recursive Digital Filters", IEEE Trans. on Circuit and System, vol. CAS-29, No. 6, pp. 390-393, June 1982.

[10] T.Tjahjadi and W. Steenaart, "On The Accuracy of ROM Stored Square Multipliers", IEEE Trans. on Circuit and System, vol. CAS-29, No. 7, pp.441-447, July 1982.

[11] L. R. Rabiner and B. Gold, "Theory and Application of Digital Signal Processing", Prentice Hall, Inc., New Jersey, 1975.

[12] B. Liu, "Effect of Finite Wordlength on the Accuracy of Digital Filters - A Review", IEEE Trans. on circuit Theory, vol. CT-18, Nov 1971.

[13] J. B. Knowles and E. M. Olcayto, "Coefficient Accuracy and Digital Filter Response", IEEE Trans. on Circuit Theory, vol. CT-15, pp. 31-41, March 1968.

[14] IEEE, ASSP Society, "Programs for Digital Signal Processing", IEEE, Inc., New York, 1979.

[15] J. M. McCool and B. Widrow, "Principles and Applications of Adaptive Filters : A Tutorial Review", Proc. 1980, IEEE ISCAS, pp. 1143-1157.

[16] M. R. Sambur, "Adaptive Noise Cancelling for Speech Signals", IEEE Trans. on Acoust., Speech Signal Process., vol. ASSP-26, No. 5, pp. 419-423, Oct. 1978.

[17] M. A. Soderstrand and J. K. Kelley, "Design of a Low-Cost adaptive FIR Filter with Sampling Rate in Excess of 10 MHz", Proc. 1981 IEEE ISCAS, vol. 2, pp.432-434.

[18] C. F. N. Cowan, J. W. Arthur and J. Mavor, "CCD Based Adaptive Filters : Realization and Analysis", IEEE Trans. Acoust., Speech Signal Process., vol. ASSP-29, pp.220-229, April 1981.

[19] B. K. Ahuja, M. A. Copeland and C. H. Chan, "A Sampled Analog MOS LSI Adaptive Filter", IEEE Journal of Solid State Ciruits, vol. SC-14, pp.148-154, Feb. 1979.

[20] B. K. Ahuja, M. A. Copeland and C. H. Chan, "A New Adaptive Algorithm and Its Implementation in MOS LSI", IEEE Journal of Solid State Circuits, vol. SC-14, pp.747-753, Aug. 1979.

[21] M. A. Soderstrand, "Cost and Performance Comparisons of Several Implementations of Adaptive Recursive Filters, Proc. 1980 Asilomar Conf., pp.416-420.

[22] T. Tjahjadi and W. Steenaart, "Stored Square ROM Multiplier", presented at the Eleventh Biennial Symposium on Communications, Kingston, Ontario, May 31, June 1-2, 1982.