A VLSI-nMOS HARDWARE IMPLEMENTATION OF AN IIR BANDPASS ORTHOGONAL DIGITAL FILTER

A Thesis Presented to

The Faculty of the College of Engineering and Technology

Ohio University

In Partial Fullfillment

Of The Requirements for the Degree

Master of Science

by

Fadi M. Kaake

March, 1986

Acknowledgements

The author would like to express his sincere gratitude to Dr. Janus Starzyk, the thesis advisor, for his patient assistance and guidance throughout this work.

Special thanks to Robert Nobles for his help and valuable suggestions.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

A Very Large Scale Integration (VLSI) program has started recently at Ohio University. The program was first implemented by installing computer aided design tools on the VAX-11/750. The CAD tools were developed at the University of Berkeley in California, Stanford University, Carnegie Mellon University and the University of Washington at Seattle. Then, a SCALDstar computer aided design system was procured by the department of ECE. The CAD system supports two graphic CRT displays, a monochromatic display for logic designs and a color display for physical layout of VLSI circuits.

Since some of the software packages for circuits simulation are not available on the SCALDstar system, a communication link between the VAX and the CAD system had to be established. As a result, circuit layouts can be generated on the graphic displays, then files that describe the geometry of the layout can be easily produced and transferred to the VAX for simulation.

In this thesis a VLSI implementation of a model filter will be presented and the steps needed to test the design using both systems will be discussed.

The recent development of Very Large Scale Integrated (VLSI) MOS technology has led to an economical solution for digital signal processors operating at frequencies higher than 1MHz. A major area that underlies the field of signal processing is filtering, since the purpose of filtering is to modify the spectrum of a received signal in some desired manner.

The digital filter to be implemented is a bandpass orthogonal filter. This type of filters have very low sensitivity in the passband and stopband, and they are easily realizable and well suited for custom VLSI implementation. A multiplierless structure based on the use of cannonical sign-digit coding was adopted for the hardware implementation of the digital filter. The basic building blocks of the hardwired filter are adders/subtractors and shift registers; which are realized as basic cells then replicated to obtain the global structure.

Organization of the thesis.

In Chapter 2, digital filters representation, network realization, and the transformation of direct form coefficients to orthogonal structure coefficients will be discussed. A VLSI realization of a bandpass digital filter will be the subject of Chapter 3, which covers the coefficients representation of the filter, the structure of the basic cells of the design and a description of how the layout structure was configurated. Chapter 4 is the core of the thesis, since it reflects the usage of the design tools and the CAD system in testing the validity of VLSI circuits. In this chapter, design rule checking, power consumption, timing analysis and logic simulation of the filter is discussed.

Finally, Chapter 5 illustrates different stages of the design including design rules checking, circuit extraction and simulation on a four stage shift register example. An additional information about plotting the designs layout and schematic is included. Chapter 5 is completely related to simulation sessions listed in Appendix B.

Chapter 2

DIGITAL FILTERS

2.1 Definition and representation

A digital filter is described as a discrete time signal processor for which both inputs and outputs are discrete time signals. The task of such a processor is to operate on the input signal, either by enhancing or supressing some of its features. Depending on the type of the digital filter, the output can either be represented as a function of present and past input values described by the following equations :

Difference equation : $y(n) = \sum_{k=0}^{M} a_k x(n-k)$ (2.1.1a)

or equivalently by

Z transform equation : $Y(z) = \sum_{k=0}^{M} a_k X(z) z^{-k}$ (2.1.1b)

or in addition to its dependence on input signal, output can be related to the past output values which leads to other equations namely :

$$y(n) = \sum_{k=0}^{M} a_k x(n-k) - \sum_{k=1}^{M} b_k y(n-k) \quad \text{where } b_0 = 1 \qquad (2.1.2a)$$

or

$$Y(z) = \sum_{k=0}^{M} a_k X(z) z^{-k} - \sum_{k=1}^{M} b_k Y(z) z^{-k} \qquad (2.1.2b)$$

□ X(z) and Y(z) represent the z transform of the input and output sequence respectively .

□ $a_0 \ldots\ldots a_M$ and $b_0 \ldots\ldots b_M$ are the coefficients to be derived according to the specifications of the digital filter .

□ z^{-1} is a delay operator

The input sequence may be obtained by sampling and quantizing a continuous time signal by means of analog to digital converters, while the output signal can be reconstructed using digital to analog converters.

A hardware realization to perform a computational algorithm on data sequences represented by the above equations requires the following basic units :

- Adders and multipliers to weight and add the sampled signals at different nodes of the digital filter structure .
- Shift registers to latch the past input or input and output values .

2.2 FIR and IIR digital filters .

Before we discuss the orthogonal structure of digital filters it is important to give a brief description of different types and structures of digital filters.

Digital filters are subdivided into two classes , infinite and finite impulse response (recursive and nonrecursive) filters.

Outputs and inputs of FIR and IIR digital filters are respectively related by equations (2.1.1 and 2.1.2) or by their transfer functions which are given by :

$$H(z) = \sum_{k=0}^{M} a_k z^{-k} \qquad \text{for FIR filters} \qquad (2.2.1)$$

and by

$$H(z) = [\sum_{k=0}^{M} a_k z^{-k}] / [\sum_{k=0}^{M} b_k z^{-k}] \quad \text{for IIR filters} \qquad (2.2.2)$$

As can be observed from the above equations , H(z) is a rational function of z^{-1} for recursive filters (2.2.2) while it is a polynomial of z^{-1} for nonrecursive filters. IIR filters design is based on analog (active or passive) filter design for which the transfer function is a rational function of complex variables, for instance

$$H(s) = [s^2 + a]/[s^2 + bs + c] \qquad (2.2.3)$$

In order to obtain a transfer function (2.2.2) that meets the same specifications and characteristics of an analog filter with a transfer function H(s) , a procedure should be followed which maps the s - plane into the z - plane domaine.

Different procedures have been developed and described in many literature about digital filters, two of them will be mentioned:

1 - The bilinear mapping procedure based on the transformation

$$f(z) = [2(1 - z^{-1})]/[T(1+z^{-1})] \qquad (2.2.4)$$

2 - The impulse invariant mapping procedure which changes a given transfer function $H(s) = \sum_{k=1}^{M} A_k /(s-p_k)$, to

$$H(z) = \sum_{k=1}^{M} A_k/(1-z^{-k}b_k), \text{ where } b_k = \exp(-p_kT) \qquad (2.2.5)$$

(T is the sampling period).

FIR filters are designed using different techniques namely, the method of windowing and the frequency sampling method. Both techniques will result in a transfer function in the form (2.2.1) .

The choice between both types depends basically on input/output characteristics we are seeking. If a linear phase filter is sought, an IIR filter is not the choice, since with such type of filters we cannot achieve linear phase characteristics possible with FIR filters. However; in general, IIR filters are more efficient and less hardware is needed to perform computations required. For example, given an FIR filter transfer function defined by

$$H(z) = 1 + z^{-1} + z^{-2} + z^{-3} + \ldots\ldots\ldots + z^{-M} \qquad (2.2.6)$$

The sum of the M terms of the above geometric progression is given by

$$S = [1-z^{-(M+1)}]/[1-z^{-1}] = H(z) \qquad \{\text{IIR function}\} \quad (2.2.7)$$

Therefore, it can be seen that the same FIR function of (2.2.7) can be realized using IIR filters with less hardware and computation but with tradeoff in phase characteristics [13,14].

2.3 Network realization

After obtaining the transfer function that meets the specification of the desired digital filter, the next step is to implement it. Therefore, a network realization of the transfer function should be obtained.

Different realizations are possible, some of the well known are the direct form, cascaded, parallel, lattice and ladder structures. The performance, properties and amount of hardware needed vary with each structure. As its name implies, the direct form is the simplest structure to realize, a diagram that realizes the structure of a direct form filter is shown in Fig. 2.1, where network 1 realizes the numerator

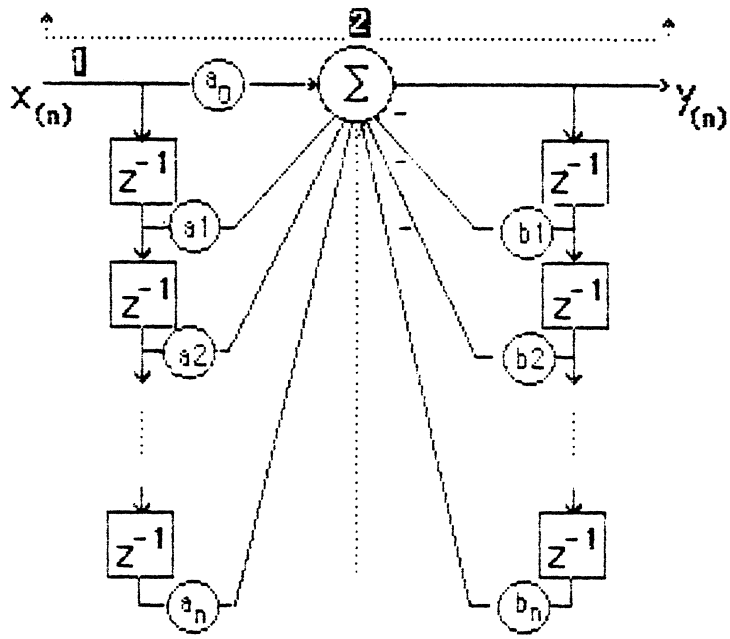


Fig. 2.1 Network realization of IIR and FIR digital filters.

polynomials of equations (2.1.1b, 2.2.1) for a FIR filter and the cascade of both networks 1 and 2 realizes (2.1.2b , 2.2.2) for an IIR filter.

Similarly, the other structures are obtained by manipulating the transfer function, which yields to different network configurations. For example, in order to obtain a parallel realization, the transfer function H(z) should be expanded into partial fractions in the form of

$$H(z) = \sum_{i=1}^{M} H_i(z) \qquad (2.3.1)$$

or by factoring the transfer function to obtain

$$H(z) = \prod_{i=1}^{M} H_i(z) \qquad (2.3.2)$$

which results in cascaded networks.

In the next section, a procedure developed by Gray and Markel [2,6] will be adopted in order to transform a direct form structure into an orthogonal structure.

2.4 Orthogonal filters

2.4.1 Definition : An orthogonal filter is a digital filter that performs matrix operation on input and present state samples in order to generate new output and state samples [5]. The matrix is a unitary matrix and is given by

$$A = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \qquad (2.4.1)$$

or by

$$A = \begin{bmatrix} (1-R^2)^{1/2} & -R \\ R & (1-R^2)^{1/2} \end{bmatrix} \qquad (2.4.2)$$

where R is a new coefficient which will be determined according to the coefficients of the direct form structure. Then, the input/output relation will be represented by

$$\begin{bmatrix} \text{next state} \\ \text{output} \end{bmatrix} = \begin{bmatrix} (1-R^2)^{1/2} & -R \\ R & (1-R^2)^{1/2} \end{bmatrix} \begin{bmatrix} \text{present state} \\ \text{input} \end{bmatrix} \quad (2.4.3)$$

2.4.2 system sensitivity : Hardware implementation of digital filters have serious effects on system function specifications such as frequency response.

Given a transfer function for an IIR filter with a set of coefficients $a_0 \ldots\ldots a_M$ and $b_0 \ldots\ldots b_M$, a limited wordlength should be chosen for binary representation of the coefficients that should be quantized (by rounding or truncation) to fit the wordlength.

Meanwhile, a change in coefficients values will cause a change in zeros and poles values. Equivalently, their locations will differ from the original ones, and the filter will fail to meet the desired specifications.

Different network realizations have been developed; some of them were mentioned in Section 2.3, and depending on the adopted structure, the sensitivity of the system to wordlength limits varies.

It has been shown that a structure based upon orthogonal polynomial structures, has nice properties such as very low sensitivity in the passband and stopband, especially in the case of digital filters

with narrow bandwidth where the poles and zeros are tightly close to each other [4,9]. In the same time, this type of filters is easily implemented in VLSI technology. Digital filters realized with such a structure are called orthogonal filters.

2.4.3 Orthogonal transformation : The block shown in Fig. 2.2 is

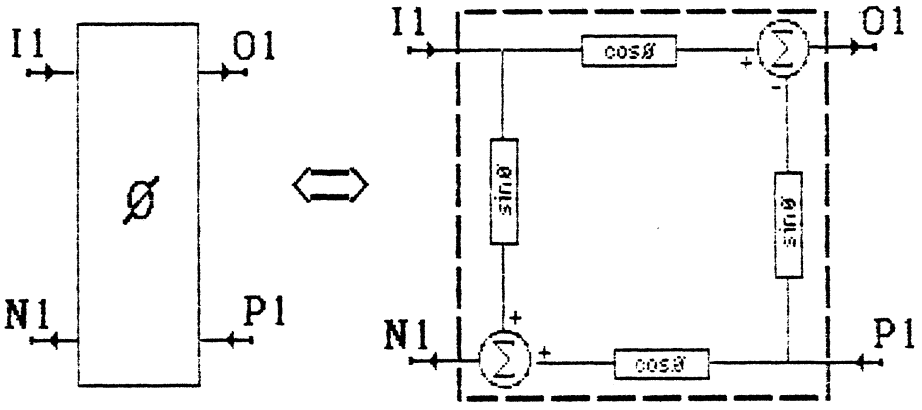


Fig. 2.2 Input/Output relation using orthogonal transformation.

considered to be the basic element that defines the structure of orthogonal filters, with P1 and I1 as inputs, O1 and N1 as outputs and ø as the angle around which the orthogonal transformation will be performed.

As illustrated in Fig. 2.3, the output values O1 and N1 are obtained geometrically by projecting the input coordinate values I1 and

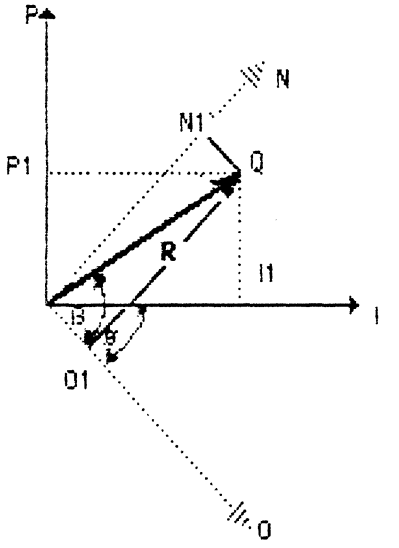


Fig. 2.3 Transformation of point Q due to the rotation of the (I,P) axis.

P1 on the new axis (O,N) obtained by rotation of the (I,P) coordinates system around the origin by an angle ø.

Given a point Q with the coordinates (I1,P1) with respect to (I,P) axis and with coordinates (O1,N1) with respect to (O,N) axis, and a vector R from the origin to Q, having an angle ß with the O axis, we can obtain an input/output relationship by solving the following set of equations

$$O1=R\cos\beta \quad , \quad N1 = R\sin\beta \qquad (2.4.4a)$$

$$I1 = R\cos(\beta-\phi) \quad , \quad P1 = R\sin(\beta-\phi) \qquad (2.4.4b)$$

I1 and P1 can be rewritten as

$$I1 = R\,(\cos\beta\cos\phi + \sin\phi\sin\beta) \qquad (2.4.5a)$$

$$P1 = R\,(\sin\beta\cos\phi - \cos\beta\sin\phi) \qquad (2.4.5b)$$

By substituting (2.4.4) in (2.4.5) we obtain

$$I1 = O1\cos\phi + N1\sin\phi \qquad (2.4.6a)$$

$$P1 = N1\cos\phi - O1\sin\phi \qquad (2.4.6b)$$

or equivalently in matrix form

$$\begin{bmatrix} I1 \\ P1 \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} O1 \\ N1 \end{bmatrix} \qquad (2.4.7a)$$

$$\begin{bmatrix} O1 \\ N1 \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}^{-1} \begin{bmatrix} I1 \\ P1 \end{bmatrix} \qquad (2.4.7b)$$

$$= \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} I1 \\ P1 \end{bmatrix} \qquad (2.4.7c)$$

$$\begin{bmatrix} O1 \\ N1 \end{bmatrix} = \begin{bmatrix} (1-R^2)^{1/2} & R \\ R & (1-R^2)^{1/2} \end{bmatrix} \begin{bmatrix} I1 \\ P1 \end{bmatrix} \qquad (2.4.8a)$$

$$= \begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} I1 \\ \\ P1 \end{bmatrix} \qquad (2.4.8b)$$

2.4.4 A global pipelined orthogonal structure : Based on the block diagram implementation of (2.4.7) shown in Fig. 2.2, a digital filter of the order M can be realized by cascading M similar blocks with different rotating coefficients (R_0 . . . . R_{M-1}). The output is obtained by weighting the next output states using another set of coefficients (W_0 . . . . W_M) and summing without introducing any extra delay elements. The derivation of coefficients (R_1 . . . . R_{M-1}) and (W_0 . . . . W_M) is discussed in the next section.

In order to examine the behavior of such a structure the cascaded blocks are shown in Fig. 2.4 for M = 5; and for the sake of clarity the flow of data throughout the network is illustrated in Fig. 2.5, where the input and output samples are shown at different instants. By inspecting Fig. 2.5, it can be seen that the next output state of each block is valid after two levels of block operations, and we notice that pipelining of

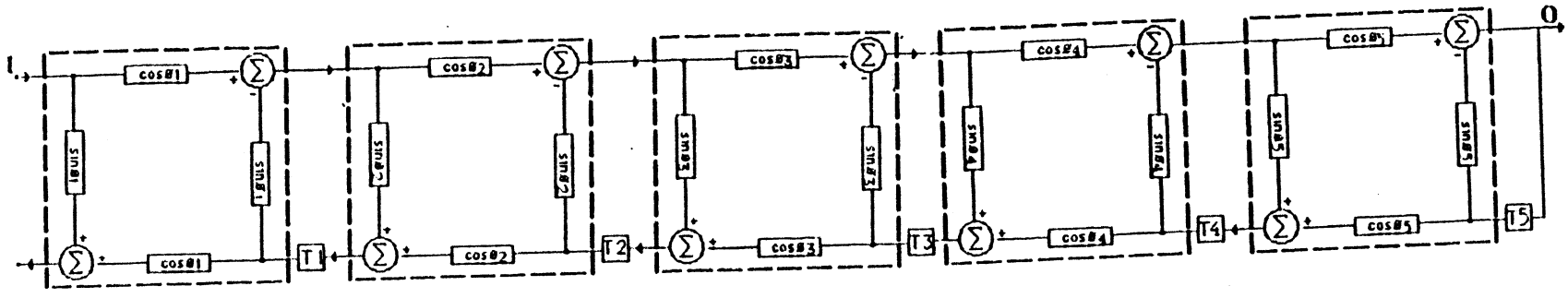


Fig 2.4. An orthogonal structure to the 5th order

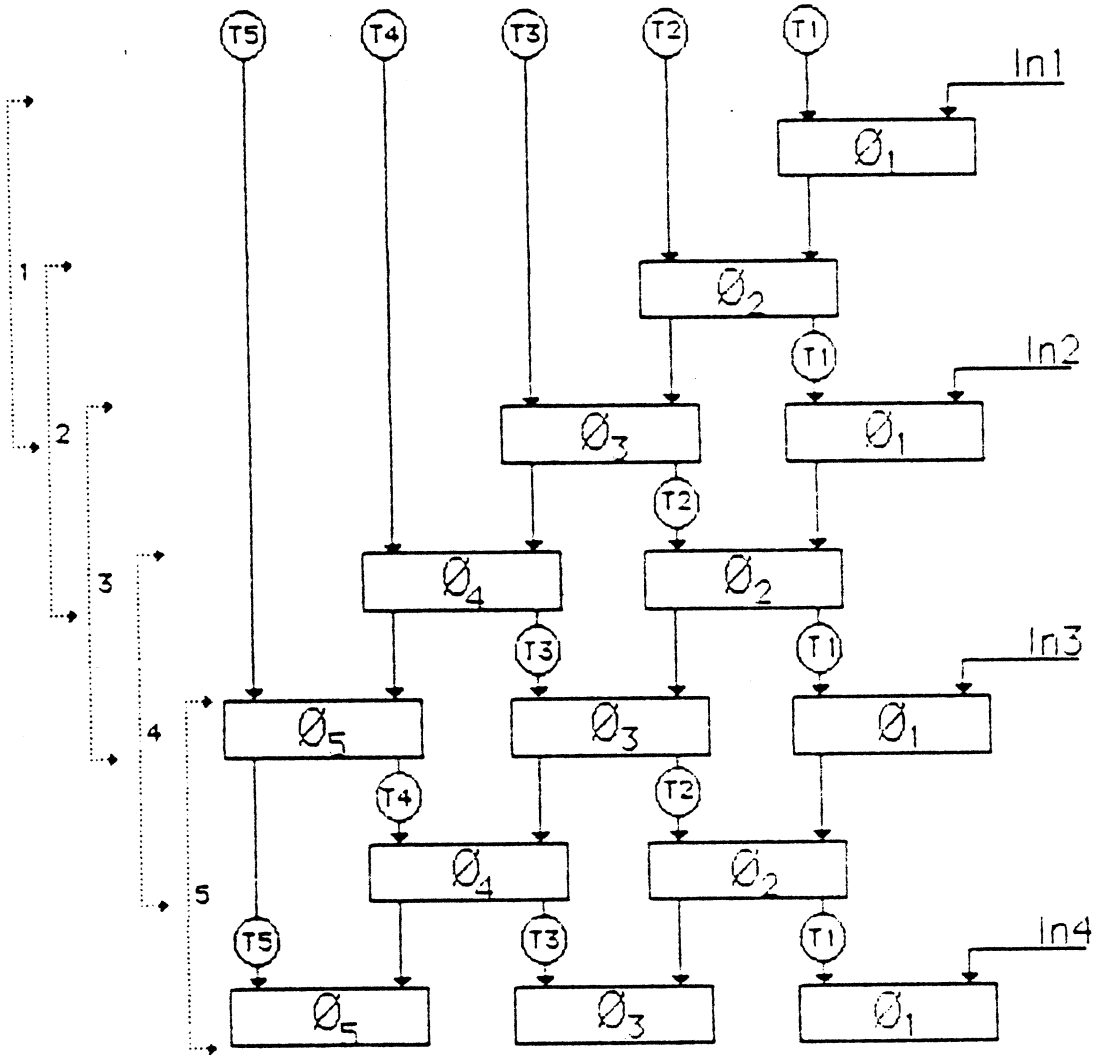


Fig. 2.5 An equivalent network of Fig2.4 showing the input and output samples at different instants.

(In1, In2, In3, In4, In5) Sampled inputs

($\emptyset_1$, $\emptyset_2$, $\emptyset_3$, $\emptyset_4$, $\emptyset_5$) Rotation angles

(T1, T2, T3, T4, T5) Current states

* Critical path = two blocks

such structure is possible [3]. Pipelining is a well known technique that is used wherever an overlapped processing is possible, allowing several pieces of the hardware to be used concurrently, increasing the through put rate of the network and improving its performance .

Each block of the critical path shown in Fig. 2.4 will produce a next output state. Since this path goes through two levels of blocks, a pipelining could be performed over a period of 2T. Therefore, achieving a network sampling rate of 2T period . "T" is the time taken by a processor to perform a unitary matrix operations on a set of data.

2.4.5 Direct form to orthogonal structure transformation:

A recursive digital filter defined by a direct form transform function

$$H(z) = A(z) / B(z) \qquad (2.4.9)$$

$$= [\sum_{i=0}^{M} a_i z^{-i}] / [\sum_{i=0}^{M} b_i z^{-i}] \quad \text{where } b_0 = 1$$

can be transformed into an orthogonal filter described in the previous section by obtaining a new set of coefficients ($R_0 \ldots R_{M-1}$) and ($W_0 \ldots W_M$).

A technique developed by Gray and Markel [2,6] based upon inner

product formulation to obtain the new set of coefficients will be presented in this section.

Let H(z) be the transfer function of an inverse filter of the form

$$H_1(z) = G / B(z) \qquad (2.4.10)$$

$$= G / [1 + \sum_{i=1}^{M} b_i z^{-i}]$$

where G is the gain of the filter, M is the order of the filter and B(z) is the denominator polynomial from (2.4.9). The difference equation of $H_1(z)$ can be written as

$$G\,\partial(k) = \sum_{i=0}^{M} b_i h(k-i) \qquad \text{Difference equation} \qquad (2.4.11)$$

where $b_0 = 1$ and $\{ \partial(k)|_{k=0} \} = 1$

Let us define $\{ r(n) \}$ as the autocorrelation sequence of the given filter which is given by

$$r(n) = r(-n) = \sum_{k=-\infty}^{\infty} h(k)h(k+n) \qquad (2.4.12)$$

and for $n = i - j$ the above relation can be rewritten as

$$r(i-j) = \sum_{k=-\infty}^{\infty} h(k)h(k+i-j) \qquad (2.4.13a)$$

$$= \sum_{k=-\infty}^{\infty} h(k-i)h(k-j) \qquad (2.4.13b)$$

By multiplying both sides of (2.4.13b) by b_j an summing over the values of j we obtain

$$\sum_{i=0}^{M} b_i r(i-j) = \sum_{i=0}^{M} \sum_{k=-\infty}^{\infty} b_i h(k-i)h(k-j) \qquad (2.4.14)$$

and the same relation will result if we multiply both sides of (2.4.11) by h(k-j) and sum them over the values of k, from which we obtain

$$\sum_{k=-\infty}^{\infty} G\partial(k)h(k-j) = \sum_{i=0}^{M} \sum_{k=-\infty}^{\infty} b_i h(k-i)h(k-j) \qquad (2.4.15)$$

and since $\partial(k) = 0$ for $k \neq 0$, (2.4.15) can be rewritten as

$$\sum_{i=0}^{M} \sum_{k=-\infty}^{\infty} b_i h(k-i)h(k-j) = Gh(-j) \qquad (2.4.16)$$

Now assuming the causality of the filter [$h(k) = 0$ for $k < 0$] and by evaluating (2.4.11) for $k = 0$ we will obtain

$$G\,\partial(k)\Big|_{k=0} = G \qquad (2.14.17)$$

$$= \sum_{i=0}^{M} b_i\, h(-i)$$

$$= h(0)$$

Based upon the above conditions and using (2.4.11) and (2.4.14) we can write

$$\sum_{i=0}^{M} b_i r(i-j) = \sum_{k=-\infty}^{\infty} G\partial(k)h(k-j) \qquad (2.4.18a)$$

$$= \quad Gh(-j) \qquad (2.4.18b)$$

$$= 0 \qquad \text{for } j = [1 \ldots M] \qquad (2.4.18c)$$

$$= G^2 \qquad \text{for } j = 0 \qquad (2.4.18d)$$

Let R(z) be the z transform of the autocorrelation sequence r(n) where

$$R(z) = \sum_{n=-\infty}^{\infty} r(n) z^{-n} \qquad (2.4.19)$$

and

$$r(n) = r(-n) = (1/2\pi j) \int_f R(z) z^{-n-1} dz \qquad (2.4.20)$$

where f is a closed contour that encircles the origin of the z plane.

With $n = i-j$ the above equation becomes

$$r(i-j) = r(j-i) = (1/2\pi j) \int_f R(z)\, z^{i-j-1} dz \qquad (2.4.21)$$

Using the above results and applying inner product, we obtain an orthogonal relationship between the polynomial B(z) and powers of z. If the inner product of functions $F_1(z)$ and $F_2(z)$ is defined as

$$\langle F_1(z), F_2(z) \rangle = (1/2\pi j) \int_f R(z)\, F^*_1(z) F_2(z)\, z^{-1} dz \qquad (2.4.22)$$

where F^* represents complex conjugated value of F.

By letting $F_1(z) = z^{-i}$ and $F_2(z) = z^{-j}$ we obtain

$$\langle z^{-i}, z^{-j} \rangle = (1/2\pi j) \int_f R(z)\, z^{i}\, z^{-j}\, z^{-1} dz \qquad (2.4.23)$$

$$= (1/2\pi j) \int_f R(z)\, z^{i-j-1} dz$$

By comparing the above equation with (2.4.21) we can write

$$r(i-j) = \langle z^{-i}, z^{-j} \rangle \qquad (2.4.24)$$

Multiplying (2.4.24) by b_i and summing over the values of i it becomes

$$\sum_{i=0}^{M} b_i\, r(i-j) = \sum_{i=0}^{M} b_i \langle z^{-i}, z^{-j} \rangle \qquad (2.4.25)$$

$$= \langle \sum_{i=0}^{M} b_i z^{-i}, z^{-j} \rangle$$

$$= \langle B(z), z^{-j} \rangle$$

and by recalling (2.4.18), $[\sum_{i=0}^{M} b_i r(i-j) = 0$ for $j = 1 \ldots M$, and G^2 for $j = 0]$

we deduce that B(z) is orthogonal to z^{-j} for all values of $j = 1 \ldots M$

(two functions are said to be orthogonal if their inner product is zero)

and (2.4.25) can be rewritten as

$$\langle B(z), z^{-j} \rangle = 0 \quad \text{for } j = 1 \ldots M \qquad (2.4.26a)$$

and

$$\langle B(z), 1 \rangle = G^2 \quad \text{for } j = 0 \qquad (2.4.26b)$$

Now let us define a new polynomial $B_m(z)$ in the form

$$B_m(z) = \sum_{i=0}^{m} b_{m,i}\, z^{-i} \qquad (2.4.27)$$

where $B_m(z) = B(z)$ for $m = M$

Again, we can say that $B_m(z)$ is orthogonal to z^{-i} for $i = 1, \ldots, m$ and we can write

$$\langle B_m(z), z^{-i} \rangle = 0 \quad \text{for } i = 1, \ldots, m \qquad (2.4.28)$$

Using inner product properties

$$1.\ \langle F_1(z), F_2(z) \rangle = \langle F_2(z), F_1(z) \rangle^* \qquad (2.4.29a)$$

$$2.\ \langle F_1(z), 1 \rangle = \langle F_1(z), F_1(z) \rangle \qquad (2.4.29b)$$

$$3.\ \langle z^{-k} F_1(z), z^{-k} F_2(z) \rangle = \langle F_1(z), F_2(z) \rangle \qquad (2.4.29c)$$

we can write

$$\langle B_m(z), z^{-i} \rangle = \langle z^{i}, B_m(1/z) \qquad (2.4.30)$$

$$= \langle z^{i-m-1}, B_m(1/z)\, z^{-m-1} \rangle$$

$$= 0 \quad \text{for } i = 1, \ldots, m$$

and by letting $(1+m-i) = p$ and defining another polynomial function $P_m(z)$ given by

$$P_m(z) = z^{-(m+1)} B_m(1/z) \qquad (2.4.31)$$

another polynomial orthogonality will arise and the new inner product formulation will be written as

$$\langle z^{-p}, P_m(z) \rangle = 0 \quad \text{for } p = 1 \ldots . m \qquad (2.4.32)$$

Note that the coefficients of $P_m(z)$ are the same as those of $B_m(z)$ and comparing the following equations

$$B_m(z) = \sum_{i=0}^{m} b_{m,i} z^i \qquad (2.4.33)$$

$$= b_{m,0} + b_{m,1} z^{-1} + b_{m,2} z^{-2} + \ldots . b_{m,m} z^{-m}$$

and

$$P_m(z) = z^{-(m+1)} B_m(1/z) \qquad (2.4.34)$$

$$= \sum_{i=0}^{m} z^{-(m+1)} b_{m,i} z^i$$

$$= b_{m,0} z^{-(m+1)} + b_{m,1} z^{-m} + \ldots . b_{m,m} z^{-1}$$

$$= p_{m,m+1} z^{-(m+1)} + p_{m,m} z^{-m} + \ldots . p_{m,1} z^{-1}$$

we can see that

$$p_{m,-i+m+1} = b_{m,i} \qquad (2.4.35)$$

with $p_{m,m+1} = b_{m,0} = 1$ and $p_{m,0} = 0$

Since $P_m(z)$ and $B_m(z)$ are of the order m+1 and m respectively, both polynomials can be linearily related using new coefficients $R_0 \ldots R_{m-1}$

$$B_m(z) = B_{m-1}(z) + R_{m-1}P_{m-1}(z). \qquad (2.4.36)$$

Using (2.4.31), the conjugate of both sides gives

$$B_m(z) = z^{-(m+1)}P_m(1/z). \qquad (2.4.37)$$

Substituting (2.4.37) and (2.4.31) in (2.4.36) we obtain

$$z^{-(m+1)}P_m(1/z) = z^{-m}P_{m-1}(1/z) + R_{m-1}z^{-m}B_{m-1}(1/z) \quad (2.4.38a)$$

and

$$P_m(z) = z^{-1}P_{m-1}(z) + z^{-1}R_{m-1}B_{m-1}(z) \qquad (2.4.38b)$$

or equivalently

$$P_{m-1}(z) = zP_m(z) - R_{m-1}B_{m-1}(z) \qquad (2.4.38c)$$

Using(2.4.36) and (2.4.38c) to solve for $B_{m-1}(z)$ results in

$$B_{m-1}(z) = [B_m(z) - R_{m-1}zP_m(z)]/(1-R^2_{m-1}) \quad (2.4.39)$$

By inspection from (2.4.35), $b_{m,0} = p_{m,m+1} = 1$ and recalling

that the order of $P_m(z)$ is higher than the order of $B_m(z)$ by one, (2.4.39) becomes true if

$$R_{m-1} = b_{m,m} \qquad (2.4.40)$$

As a result, given a polynomial B(z) of the order M, the coefficients $R_0 \ldots R_{M-1}$ can be evaluated by solving the recursive equations (2.4.39) and (2.4.38c) starting with $R_{M-1} = b_{M,M}$ and based upon the results given by equations (2.4.35) and (2.4.40).

Now considering another polynomial equation $A_m(z)$ given by

$$A_m(z) = \sum_{i=0}^{m} a_{m,i} z^{-1} \quad \text{for } m = 0 \ldots\ldots M \qquad (2.4.41)$$

where for m = M, $A_m(z)$ is equal to A(z), the numerator of (2.4.9). Another recursive relation between $A_m(z)$ and $P_m(z)$ can be obtained by

$$A_{m-1}(z) = A_m(z) - zP_m(z)W_m \qquad m=0 \ldots\ldots M \quad (2.4.42)$$

where $W_0 \ldots\ldots W_M$ are the new set of coefficients described in Section 2.4.3.

Since $P_m(z)$ is of the order m+1, $zP_m(z)$ is a polynomial of the

order m, with the coefficient $p_{m,m+1}$ equal to 1 on the basis of (2.4.35), therefore, we obtain

$$W_m p_{m,m+1} z^{-(m+1)} z = W_m z^{-m} \qquad (2.4.43)$$

Setting $W_m = (a_{m,m})$, (2.4.42) results in a polynomial of the order m-1, since both terms ($a_{m,m}z^{-m}$) and ($w_m p_{m,m+1} z^{-(m+1)}$) will cancel each other.

Starting with $W_M = a_{M,M}$ and using the results of equations (2.4.38) and (2.4.43), a solution for $W_0 \ldots . W_M$ can be obtained by solving the recursive equation (2.4.42) and as a result, $P_m(z)$ can be expressed in terms of W_m and $P_m(z)$ by

$$P_M(z) = \sum_{m=0}^{M} W_m z P_m(z) \qquad (2.4.44)$$

since

$$A_{M-1}(z) = A_M(z) - zP_M(z)W_M$$

$$A_{M-2}(z) = A_{M-1}(z) - zP_{M-1}(z)W_{(M-1)} \qquad (2.4.45)$$

$$A_0(z) = A_1(z) - zP_1(z)W_1$$

$$0 = A_0(z) - zP_0(z)W_0$$

which yields

$$A_M(z) = A_{M-1}(z) + zP_M(z)W_M \qquad (2.4.46)$$

$$= A_{M-2}(z) + zP_{M-1}(z)W_{M-1}zP_M(z)W_M + zP_{M-1}(z)W_M$$

.

.

$$= zW_0P_0(z) + zP_1(z)W_1 + zP_2(z)W_2 + \ldots + zP_M(z)W_M$$

$$= \sum_{m=0}^{M} z\, W_m P_m(z)$$

The results obtained can be summarized by the following equations

$$P_m(z) = z^{-(m+1)}B_m(1/z) \qquad (2.4.47)$$

$$B_{m-1}(z) = B_m(z) - R_{m-1}P_{m-1}(z) \qquad (2.4.48a)$$

$$B_{m-1}(z) = [B_m(z) - R_{m-1}zP_m(z)]/(1-R^2_{m-1}) \qquad (2.4.48b)$$

$$zP_m(z) = P_{m-1}(z) + R_{m-1}B_{m-1}(z) \qquad (2.4.49)$$

$$P_M(z) = \Sigma\, W_m zP_m(z) \qquad (2.4.50)$$

$$H(z) = P_M(z)/B_M(z) \qquad (2.4.51)$$

$$= [\sum_{m=0}^{M} W_m z P_m(z)\ ]/B_M(z)$$

where $P_m(z)$ is the new orthogonal polynomial and ($R_0 \ldots R_{M-1}$, $W_0 \ldots .$ W_{M-1}) are the parameters to be obtained by recursively solving the above equations for m = 0 . . . . M .

Further manipulations of these equations will lead to a normalized structure. First, by defining a set of parameters $\beta_0 \ldots . \beta_M$ given by

$$\beta_m = \prod_{j=m}^{M} (1+\Omega_j R_j) \qquad \text{for } m = 0 \ldots . M \qquad (2.4.52)$$

where $\Omega_0 \ldots \Omega_M$ are called sign parameters. Each of them is equal to plus one or minus one, and they are used to optimize scaling which is discussed in [9] .

* Note that $\beta_M = 1$ since R_M is not valid and considered null .

Relations can be established between β_m, the polynomials $P_m(z)$, $B_m(z)$ and the coefficients { W_m }

$$P'_m(z) = \beta_m P_m(z) \qquad (2.4.53a)$$

$$B'_m(z) = \beta_m B_m(z) \qquad (2.4.53b)$$

$$W'_m = W_m / \beta_m \qquad (2.4.53c)$$

from which we obtain

$$H(z) = \sum_{m=0}^{M} W'_m z P'_m(z)/B_M(z) \qquad (2.4.53d)$$

Multiplying both sides of equation (2.4.48) by β_{m-1} we obtain

$$\beta_{m-1} B_{m-1}(z) = \beta_{m-1} B_m(z) - \beta_{m-1} R_{m-1} P_{m-1}(z) \qquad (2.4.54)$$

$$= [\beta_{m-1}/\beta_m][\beta_m B_m(z)] - \beta_{m-1} R_{m-1} P_{m-1}(z)$$

or equivalently

$$B'_{m-1}(z) = [\beta_{m-1}/\beta_m] B'_m(z) - R_{m-1} P'_{m-1}(z) \qquad (2.4.55)$$

Similarly from (2.4.49) we can obtain

$$z[\beta_{m-1}/\beta_m] P'_m(z) = P'_{m-1}(z) + R_{m-1} B'_{m-1}(z) \qquad (2.4.56)$$

and by substituting (2.4.56) in (2.4.55) we obtain

$$z P'_m(z) = R_{m-1} B'_m(z) + (1-R^2_{m-1})[\beta_m/\beta_{m-1}] P'_{m-1}(z) \qquad (2.4.57)$$

Now, in order to obtain an orthonormal function $B'_m(z)$, the

polynomial $B_m(z)$ should be divided by its norm and hence the value of β_m should be chosen equal to the inverse of that norm. Denoting the norm square of $B_m(z)$ by N_m, the term β_m/β_{m-1} of (2.4.57) can be replaced by $(N_{m-1}/N_m)^{1/2}$. Then by recalling (2.4.36) using the inner product formulation, both sides of the equation can be rewritten as

$$\langle z^{-m} , B_m(z)\rangle = \langle z^{-m} , B_{m-1}(z)\rangle + R_{m-1}\langle z^{-m} , P_{m-1}(z)\rangle \qquad (2.4.58)$$

$$= 0 \text{ (since } B_m(z) \text{ is orthogonal to } z^{-m})$$

and solving for R_{m-1}, we obtain

$$R_{m-1}(z) = -[\langle z^{-m} , B_{m-1}(z)\rangle]/[\langle z^{-m} , P_{m-1}(z)\rangle] \qquad (2.4.59)$$

Applying (2.4.31) we can write

$$\langle z^{-m} , P_{m-1}(z)\rangle = \langle z^{-m} , z^{-m}B_{m-1}(1/z)\rangle \qquad (2.4.60)$$

and

$$R_{m-1} = -[\langle z^{-m} , B_{m-1}(z)\rangle] / [\langle z^{-m} , z^{-m}B_{m-1}(1/z)\rangle] \qquad (2.4.61)$$

$$= -[\langle z^{-m}, B_{m-1}(z)\rangle]/[\langle B_{m-1}(z), 1\rangle]$$

It should be noted that $\langle B_m(z) , 1\rangle$ is the normsquare of $B_m(z)$,

therefore, (N_m/N_{m-1}) can be obtained as a function of R_{m-1} by

$$N_m - N_{m-1} = \langle B_m(z) , 1 \rangle - \langle B_{m-1}(z) , 1\rangle \qquad (2.4.62)$$

$$= \langle B_{m-1}(z) + R_{m-1}P_{m-1}(z) , 1\rangle - \langle B_{m-1}(z) , 1\rangle$$

$$= R_{m-1}\langle z^{-m}B_{m-1}(1/z) ,1\rangle \qquad \text{from (2.4.31)}$$

$$= R_{m-1}\langle B_{m-1}(1/z) , z^m\rangle \qquad \text{from (2.4.29c)}$$

$$= R_{m-1}\langle z^{-m},B_{m-1}(z)\rangle \qquad \text{from (2.4.29a)}$$

$$= R_{m-1}(-R_{m-1}N_{m-1}) \qquad \text{from (24.61)}$$

and finally we obtain

$$\beta_{m-1}/\beta_m = N_m/N_{m-1} \qquad (2.4.63)$$

$$= 1-R^2_{m-1}$$

Using the above results and multiplying both sides of (2.4.55) andd (2.4.57) by $X(z)/B_M(z)$ we obtain

$$[B'_{m-1}(z)X(z)]/B_M(z) =$$

$$(X(z)/B_M(z))[\, B'_m(z)\,(1 - R^2_{m-1})^{1/2} - R_{m-1}P'_{m-1}(z)] \qquad (2.4.64)$$

and

$$[zP'_m(z)X(z)/B_M(z)] =$$

$$(X(z))/B_M(z))\{ R_{m-1}B'_m(z) + (1-R^2_{m-1})^{1/2}P'_{m-1}(z)] \quad (2.4.65)$$

or equivalently the matrix from

$$\frac{X(z)}{B_M(z)}\begin{bmatrix} B'_{m-1}(z) \\ zP'_m(z) \end{bmatrix} = \begin{bmatrix} (1-R^2_{m-1})^{1/2} & -R_{m-1} \\ R_{m-1} & (1-R^2_{m-1})^{1/2} \end{bmatrix}\begin{bmatrix} B'_{m-1}(z) \\ P'_{m-1}(z) \end{bmatrix}\frac{X(z)}{B_M(z)} \quad (2.4.66)$$

which is similar to the unitary matrix of equation (2.4.8).

A network realization of a normalized orthogonal filter with the transfer function H(z), described by equation (2.4.45) is shown in Fig. 2.6. The cascaded sections of the structure are realized using (2.4.61).

Two computer programs are listed in Appendix A. Program# 1 is used for transforming the direct form coefficients ($a_0 \ldots a_M, b_0 \ldots b_M$) of a digital filter to the orthogonal polynomial structure coefficients ($R_0 \ldots R_{M-1}, W'_0 \ldots W'_M$). Program# 2 will transform the quantized values of the ($R_0 \ldots R_{M-1}, W'_0 \ldots W'_M$) coefficients to their corresponding ($a_0 \ldots a_M, b_0 \ldots b_M$) coefficients which are needed

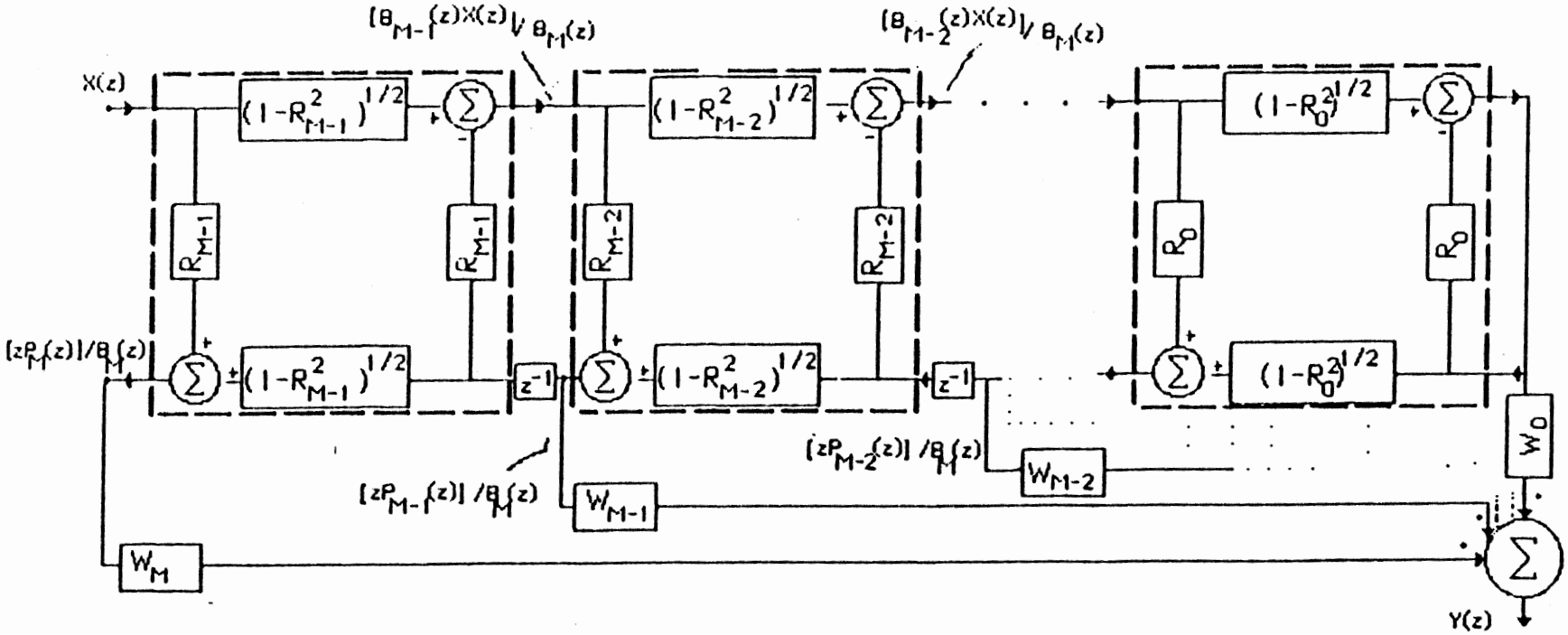


Fig. 2.6. An orthohogonal filter structure to the order M

in order to compare the frequency response characteristics with and without coefficient quantizations. More details will follow in the next chapter.

Chapter 3

VLSI realization of a bandpass digital filter

3.1 Introduction

In many digital systems, where real time processing is required, components with high processing speed are needed. In the same time, lower cost, less energy consumption, minimum chip area and simple arithmetic operations are preferable.

With the presence of VLSI components, some of the above requirements can be met. In the case of the hardware implementation of digital filters, weighting the signals with filter coefficients is implemented using multipliers. It would be preferable to eliminate general multipliers, since they are considered to be the slowest components of the digital filter. By applying cannonical sign digit coding technique on the quantized coefficients and with the use of multiplierless structure [10], full-fledged multipliers are replaced by hardwired shifters followed by adders/subtractors.

A VLSI-nMOS realization of the orthogonal structure of a

bandpass IIR digital filter will be presented. A program to transform the direct form coefficients into the orthogonal structure coefficients is applied, then a ripple adder was used as a basic element to perform the two's complement arithmetic operations needed to realize each of the obtained coefficients. The result is an array of adders/subtractors synchronized with clocked registers. The layout of the whole structure is presented starting from the primitive cells. A different layout of cells with the same function is needed in order to optimize the chip area and make the routing of the signal paths possible.

3.2 coefficients representation of a bandpass IIR digital filter.

A 10th order bandpass direct form digital filter is designed with the following parameters:

1 - Passband ripple: 1dB

2 - Passband edges: 0.2 MHz and 0.8 MHz

3 - Stopband edge: 0.16 MHz

4 - Peak response: 0 dB

5 - Sampling frequency: 2 MHz

with the direct form coefficients listed in Table 3.1 [5].

i	a_i	b_i
0	0.118108100	1.000000000
1	0.000000000	0.000000000
2	-0.430313607	-0.267878458
3	0.000000000	0.000000000
4	0.749821211	1.246797710
5	0.000000000	0.000000000
6	-0.749821211	0.139827617
7	0.000000000	0.000000000
8	0.430313607	0.363467696
9	0.000000000	0.000000000
10	-0.118108100	0.161830500

Table 3.1

The frequency response characterisitics of the digital filter are given by

$$|H(e^{j\phi})| = \left|\sum_{n=0}^{m} a_i e^{-jn\phi}\right| / \left|\sum_{n=0}^{m} b_i e^{-jn\phi}\right| \qquad (3.1)$$

where " ø " is the " digital frequency " and is given by

$$\phi = wT_s \qquad (3.2)$$

with " w " being the analog frequency and T_s being the sampling period. A plot for the frequency characteristics of the given bandpass filter is

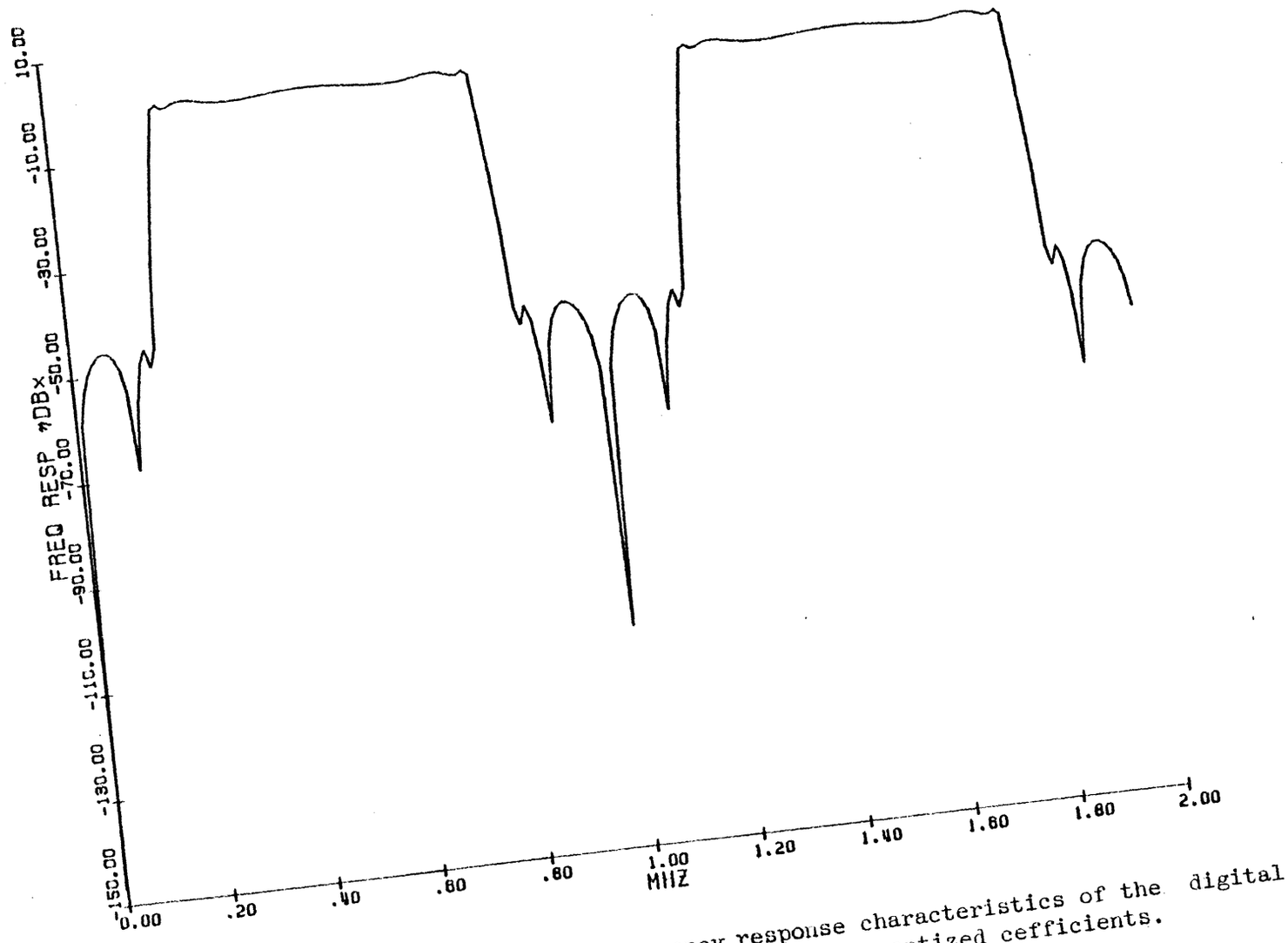


Fig 3.1. Frequency response characteristics of the digital filter with the nonquantized cefficients.

shown in Fig. 3.1 and is obtained by using program#3 listed in Appendix A.

	R_i	i	W_i
0	0.000000	0	-0.590350
1	-0.257363	1	0.000000
2	0.000000	2	0.104179
3	0.936896	3	0.000000
4	0.000000	4	0.192151
5	0.104908	5	0.000000
6	0.000000	6	-0.951028
7	0.421083	7	0.000000
8	0.000000	8	0.472831
9	0.161830	9	0.000000
		10	-0.118108

Table 3.2

By applying program #1 (Appendix A), we can obtain the coefficients for the orthogonal structure shown in Table 3.2.

Since the arithmetic operations in most digital systems are based on the binary number representation, a binary equivalent with a finite wordlength should be chosen for each of the above coefficients. Therefore, the coefficients should be quantized in a way that the resultant error will not alter the frequency response characteristics of the digital filter.

$*R_i$

i	Decimal representation	Binary representation	Canonical Signed_Digit Code
0	0.0000000	0.000000000	0.000000000
1	0.2578125	0.010000100	0.010000100
2	0.0000000	0.000000000	0.000000000
3	0.9375000	0.111100000	1.000$\bar{1}$00000
4	0.0000000	0.000000000	0.000000000
5	0.1093750	0.000111000	0.00100$\bar{1}$000
6	0.0000000	0.000000000	0.000000000
7	0.4375000	0.011100000	0.100$\bar{1}$00000
8	0.0000000	0.000000000	0.000000000
9	0.1562500	0.001010000	0.001010000

Table3.3a

$*(1-Ri)^{1/2}$

i	Decimal representation	Binary representation	Canonical Signed_Digit Code
0	1.0000000	1.000000000	1.000000000
1	0.9687500	0.111110000	1.0000$\bar{1}$0000
2	1.0000000	1.000000000	1.000000000
3	0.3750000	0.011000000	0.10$\bar{1}$000000
4	1.0000000	1.000000000	1.000000000
5	0.9921875	0.111111100	1.000000$\bar{1}$00
6	1.0000000	1.000000000	1.000000000
7	0.9375000	0.111100000	1.000$\bar{1}$00000
8	1.0000000	1.000000000	1.000000000
9	0.9843750	0.111111000	1.00000$\bar{1}$000

Table3.3b

*Wi

i	Decimal representation	Binary representation	Canonical Signed_Digit Code
0	0.0546875	0.000011100	$0.000100\bar{1}00$
1	0.0000000	0.000000000	0.000000000
2	0.1093750	0.000111000	$0.001001\bar{0}00$
3	0.0000000	0.000000000	0.000000000
4	0.1875000	0.001100000	$0.010\bar{1}00000$
5	0.0000000	0.000000000	0.000000000
6	0.9375000	0.111100000	$1.000\bar{1}00000$
7	0.0000000	0.000000000	0.000000000
8	0.4687500	0.011110000	$0.1000\bar{1}0000$
9	0.0000000	0.000000000	0.000000000
10	0.1171875	0.000111100	$0.001000\bar{1}00$

Table3.3c

In multiplierless structured digital filters, the number of add/subtract operations required, is determined by the number of the nonzero bits of the binary equivalent of each of the coefficients. Therefore, a minimum number of nonzero bits is sought for each of the coefficients. In our design, a trial and error approach has been adopted in order to select the binary equivalents of the coefficients, while introducing a minimum error in frequency characteristics.

The absolute values of the quantized coefifcients in decimal and binary representation are listed in Table 3.3a, b and c.

A technique called cannonical sign digit coding is used to reduce the number of nonzero bits in the obtained results. CSD coding [1,7] of a binary-number can described by the relation.

$$N = 2^{k+j+1} - 2^{k} \qquad (3.3)$$

$$= 2^{k+j} + 2^{k+j-1} + 2^{k+j-2} + \ldots\ldots 2^{k}$$

which indicates that a multiplication of any number by the number N, needs (j-1) number of add operations that can be reduced to one subtraction.

By applying the CSD coding on each of the coefficients, we obtained the results listed in the rightmost column of Tables 3a, b and c.

The bit represented by " $\bar{1}$ " is equal to " -1 ", which means that the shifted value of the signal will be subtracted instead of being added to the previous results.

A 10 bits wordlength was chosen in order to compensate for the truncation error resulting from the signals added. In order to obtain the effects of the modified coefficients on the frenquency characteristics, the modified coefficients of the orthogonal filter were transformed into their direct form equivalents by applying program#2 listed in Appendix A.

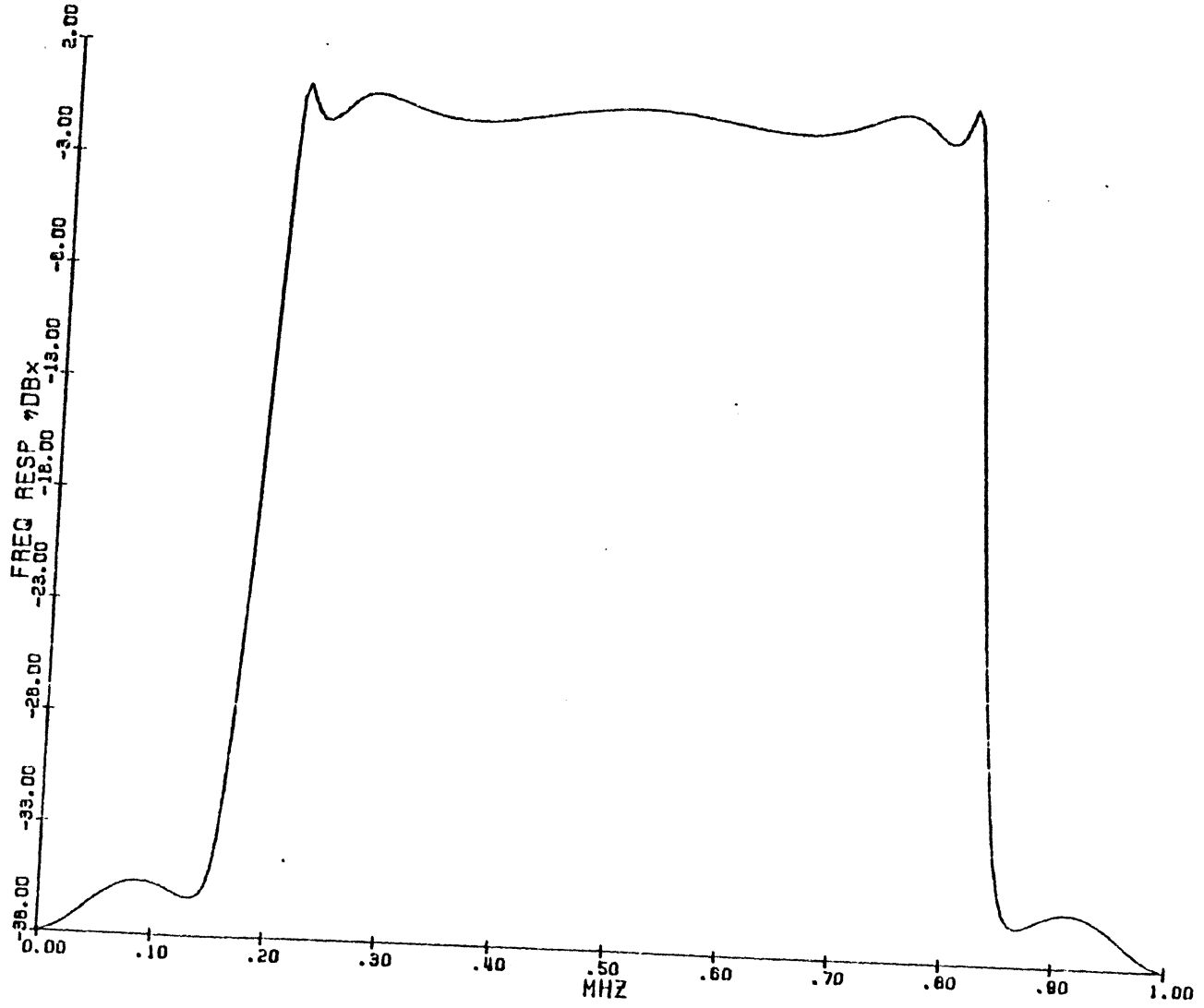


Fig. 3.2 Frequency response characterestics of the digital filter with the quantized coefficients.

By using the results of program #2 and applying program #3, it can be seen that the effect of the modified coefficients on the frequency characteristics shown in Fig. 3.2 is unnoticeable.

3.3 A VLSI realization of the bandpass filter.

3.3.1 Structure configuration. Based upon the CSD coefficients representation of the bandpass filter listed in Table 3.3a, b and c, and using the structure shown in Fig. 2.6 with M = 10 (the order of the filter), a Very Large Scale Integrated Circuit realization of the digital filter is performed using a 2.5 μm nMOS technology. The multiplication of the input signals by the coefficients is implemented by the shifting operations followed by two's complement arithmetic operations on the shifted signals.

An example of signals weighting is illustrated in Fig. 3.3. The value of the coefficient in this example is equal to $0.100\bar{1}$. Since the second and fifth most significant bits of the coefficient are nonzero bits, the 10-bits input signal with the most significant bit as the sign bit should first be shifted once then four times to the right. The unfilled inputs of the subtractor should be set to the sign bit value of the input

signal. In order to add the shifted signals, parallel subtractors are used

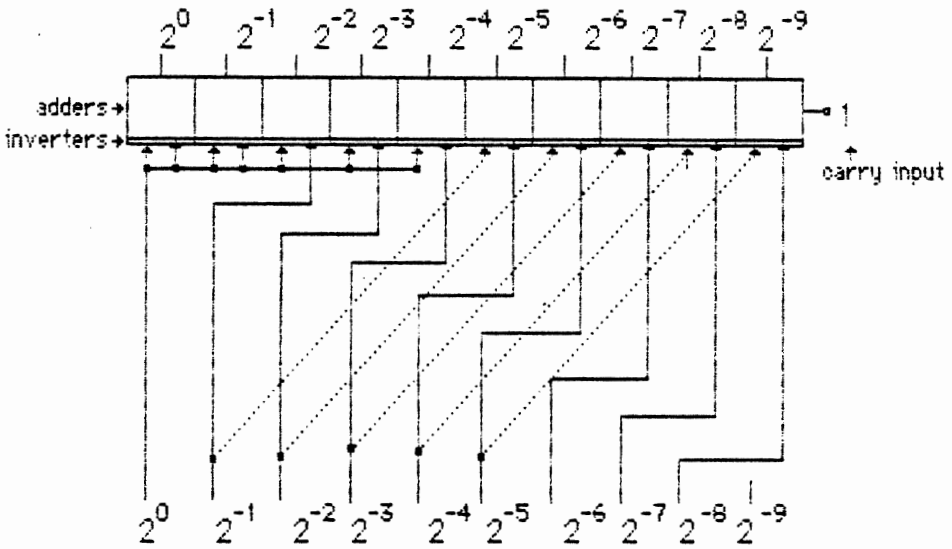


Fig. 3.3

instead of parallel adders since the fifth MSB of the coefficient is "1".

The complexity of any logic circuit design using nMOS technology, like hardware, delay and power dissipation involved, depends mainly on the logic level structure. Therefore, a manipulation of the logic equations is needed in order to obtain a good implementation with the available technology.

The basic cell of the design is the full adder cell, with the logic equations of the outputs given by

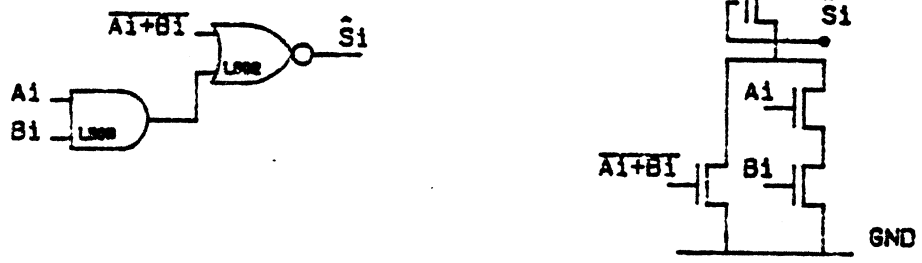


Fig 3.4a

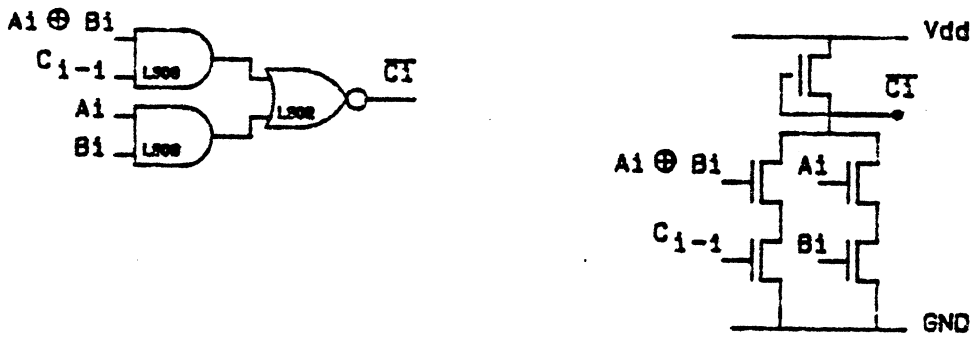


Fig 3.4b

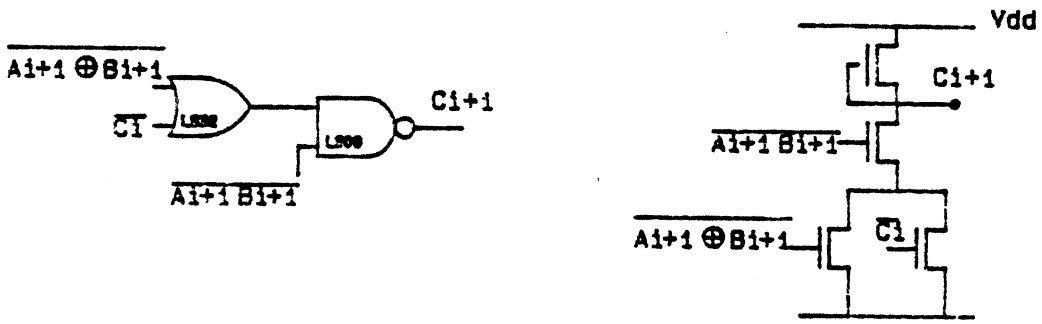


Fig 3.4c

$$S_i = A_i \oplus B_i \oplus C_{i-1} \qquad (3.3.1)$$

$$C_i = (A_i \oplus B_i)C_{i-1} + A_iB_i$$

where A_i and B_i are the input bits, C_{i-1} is the input carry bit, S_i and C_i are the sum and the output carry bits. The whole combination circuit can be thought of as two stages of half adders, which are equivalent to two XOR gates in addition to an OR gate needed in order to obtain the output carry. The XOR logic equation is manipulated in order to obtain

$$\hat{S}_i = A_i \oplus B_i = \overline{\overline{A}_iB_i + \overline{B}_iA_i} \qquad (3.3.2)$$
$$= A_iB_i + \overline{(A_i + B_i)}$$

which is easily realized as shown in Fig. 3.4a. The NAND and NOR logic circuits are not shown since they can be easily realized. The NAND gate can be constructed with a depletion mode pull-up transistor in series with two enhancement mode transistors while the NOR gate is a combination of two parallel enhancement transistors in series with a depletion mode pull-up transistor.

Now, in order to add n-bit numbers, we need n full adders which are commonly known as carry-ripple adders since the carry ripples through stages of the adder. The addition time can be obtained as a function of carry propagation time and is given by

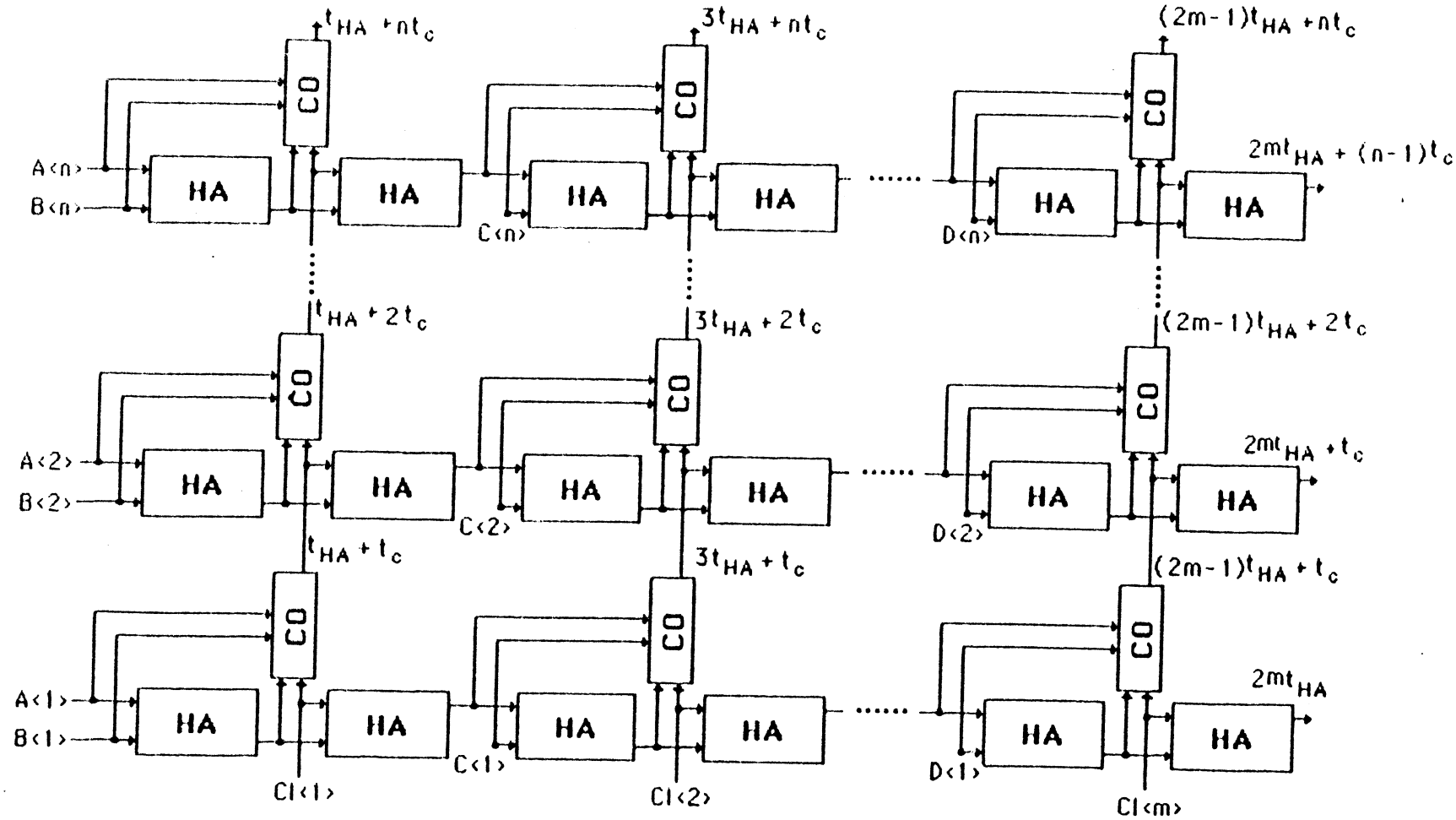

$t_{HA} + nt_c$
$3t_{HA} + nt_c$
$(2m-1)t_{HA} + nt_c$
$2mt_{HA} + (n-1)t_c$
$t_{HA} + 2t_c$
$3t_{HA} + 2t_c$
$(2m-1)t_{HA} + 2t_c$
$2mt_{HA} + t_c$
$t_{HA} + t_c$
$3t_{HA} + t_c$
$(2m-1)t_{HA} + t_c$
$2mt_{HA}$
CO
HA
A<n>
B<n>
C<n>
D<n>
A<2>
B<2>
C<2>
D<2>
A<1>
B<1>
C<1>
D<1>
Cl<1>
Cl<2>
Cl<m>

Fig 3.5

$$T_{add} = (n-1)t_c + 2mt_{HA}$$

which is illustrated in Fig. 3.5, where t_c is the carry propagation time, t_{HA} is the half adder or XOR gate delay and m is the number of cascaded parallel adders. By examining Fig. 3.4b which realizes the output carry C_i by adding an inverter at the output, we can deduce that a realization of the next adder in function of the negated output carry can save us one inverter delay [8,11]. The logic equation that produce the next carry and sum bits in function of C_i are given by

$$\begin{aligned} S_{i+1} &= A_{i+1} \oplus B_{i+1} \oplus C_i = \overline{\overline{A_{i+1} \oplus B_{i+1} \oplus C_i}} \\ &= \overline{\overline{(A_{i+1} \oplus B_{i+1})\,\bar{C}_i} + \overline{(\overline{A_{i+1} \oplus B_{i+1}})\,\bar{C}_i}} \end{aligned} \qquad (3.3.3)$$

$$\begin{aligned} C_{i+1} &= \overline{\overline{(A_{i+1} \oplus B_{i+1})C_i + A_{i+1}B_{i+1}}} \\ &= \overline{\overline{(\overline{A_{i+1} \oplus B_{i+1}}) + \bar{C}_i}\;\overline{(\overline{A_{i+1}B_{i+1}})}} \end{aligned}$$

Again, the XNOR logic equation is manipulated to obtain

$$\overline{(A_{i+1} \oplus B_{i+1})} = \overline{\overline{(A_{i+1}B_{i+1})(A_{i+1} + B_{i+1})}} \qquad (3.3.4)$$

with its logic diagram and circuit realization shown in Fig. 3.4c. As a

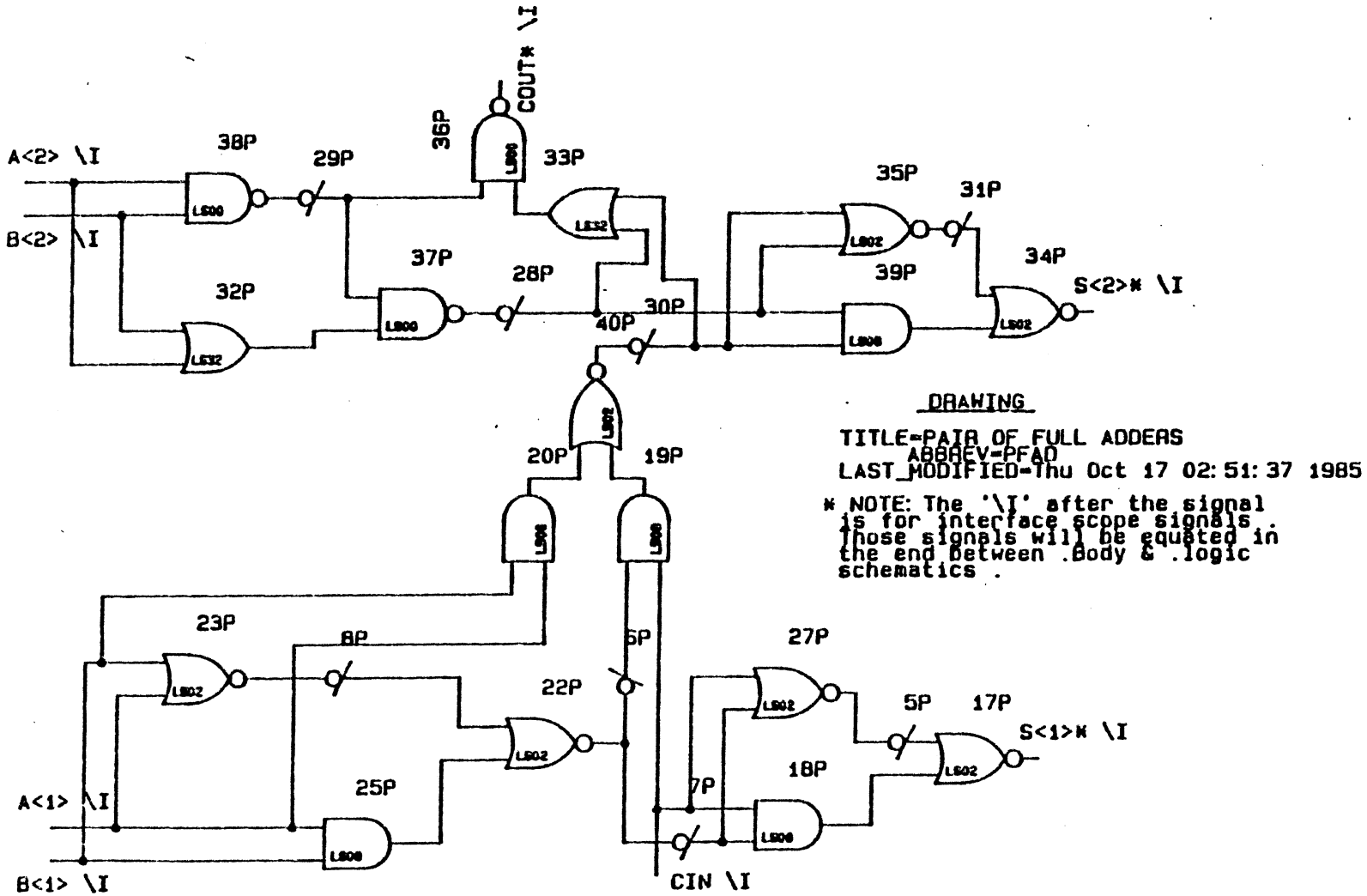


Fig. 3.6 A logic diagram of the pair of full adders.

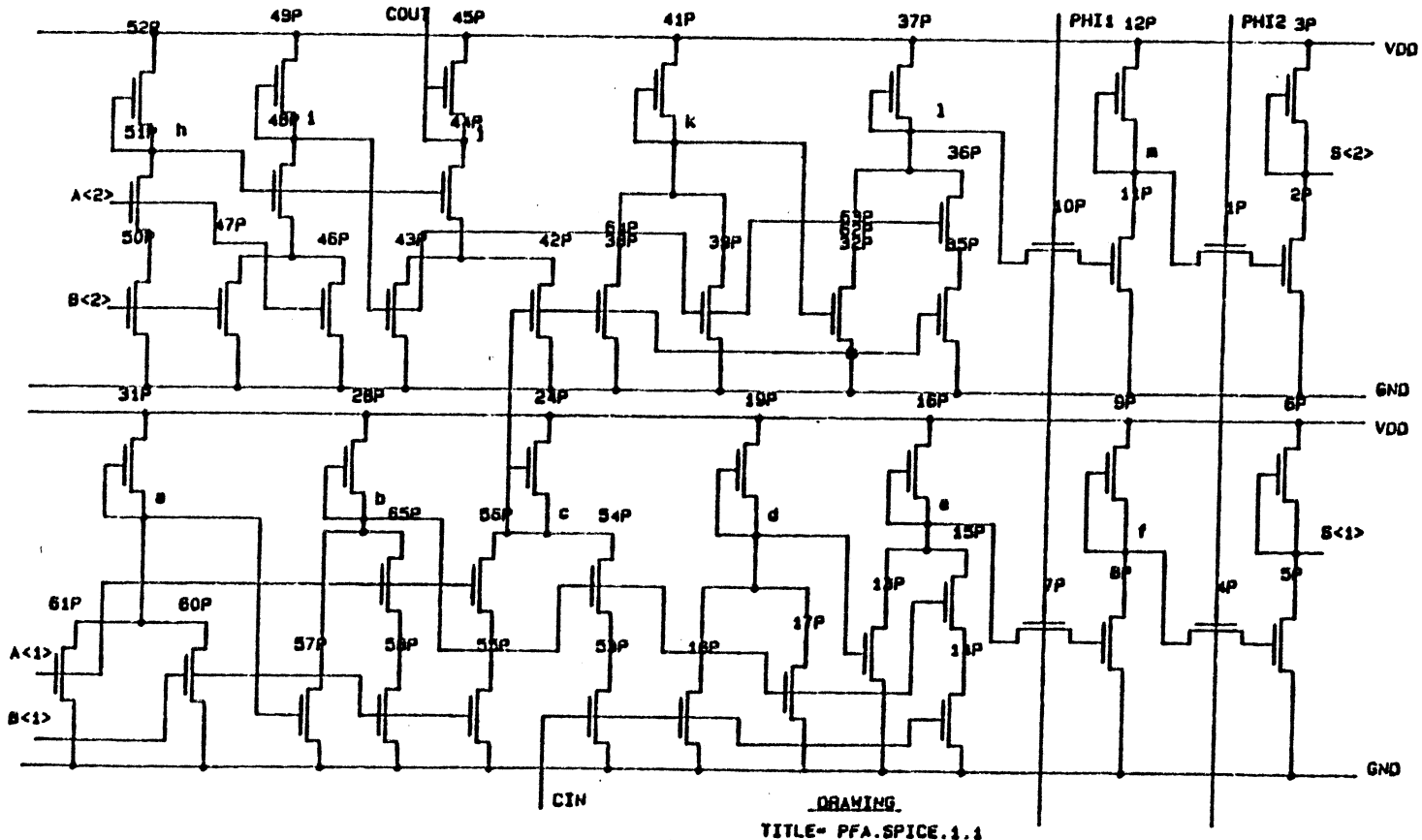


Fig. 3.7. A circuit diagram of the PFA.

result another inverter delay is saved at the output of C_{i+1}.

From the above results a pair of full adders (PFA) structure is obtained with the logic diagram and circuit realization shown in Figs. 3.6 and 3.7 respectively. Again the delay time can be expressed by

$$T_{add} = (n-1)t_c/2 + 2mt_{HA} \qquad (3.3.5)$$

where t_c is the carry propagation time through a pair of full adders.

The other two basic cells of the design are the subtractor and the shift register. The subtractor is implemented by including an inverter at the input nodes of the adders and by setting the input carry of the first PFA to "1" in order to perform two's complement operations. The shift register which is used to implement the delay units of the digital filter is realized using two inverters coupled by pass transistors. Then, with the output of the first inverter coupled to the input of the second pass transistor, the input data will be transferred from left to the right of the register by applying two nonoverlapping clock signals to the gates of the pass transistors.

3.3.2 Layout design. The integrated system layout and logic

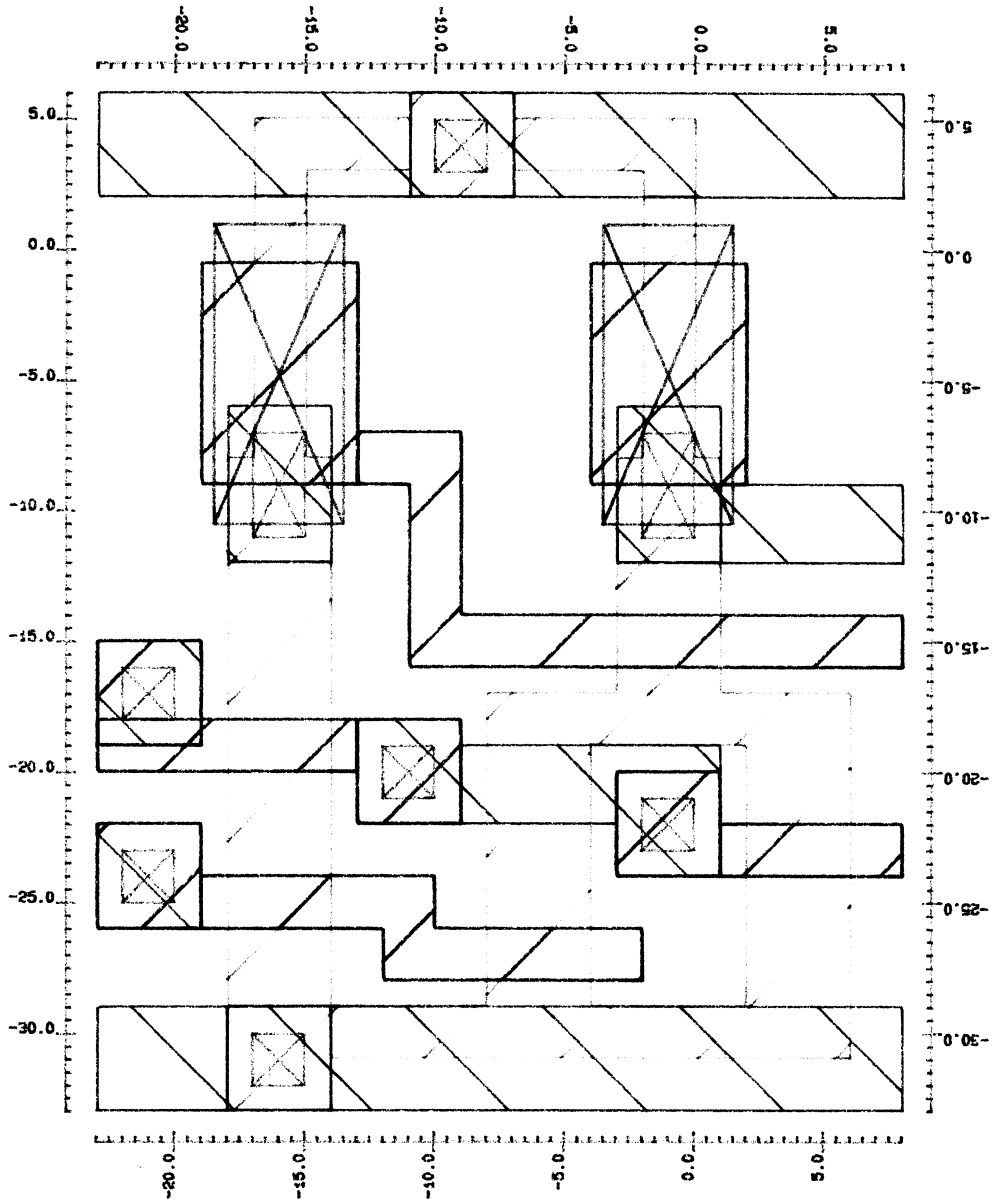


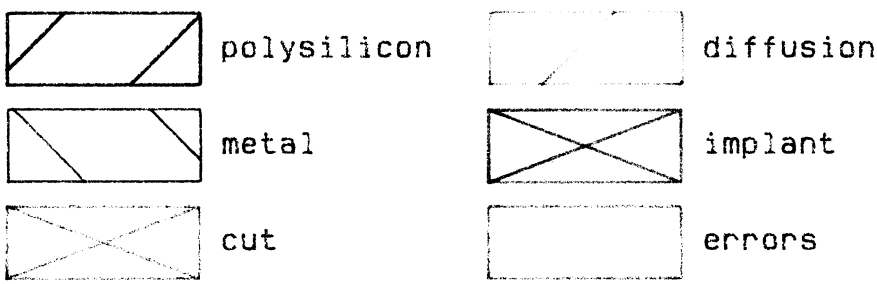
Fig 3.8

Cell: CELL1 Mirror: OFF

Scale: 3420.53 Rotation: 0

Sun Dec 8 08:11:09 1985

Valid Logic Systems Inc. SCALDstar Plot Ver. 8.0



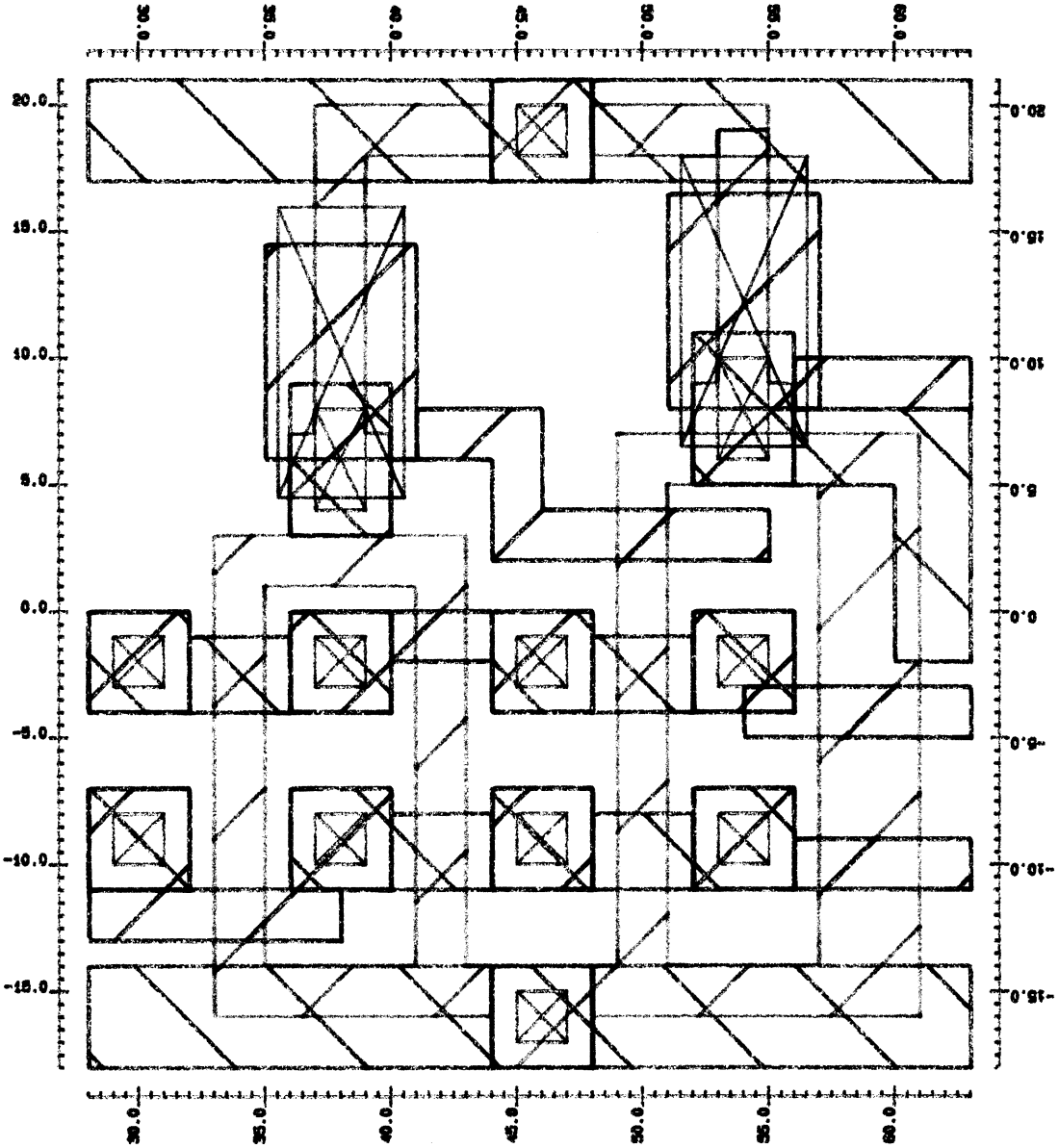


Fig 3.9 cell 3

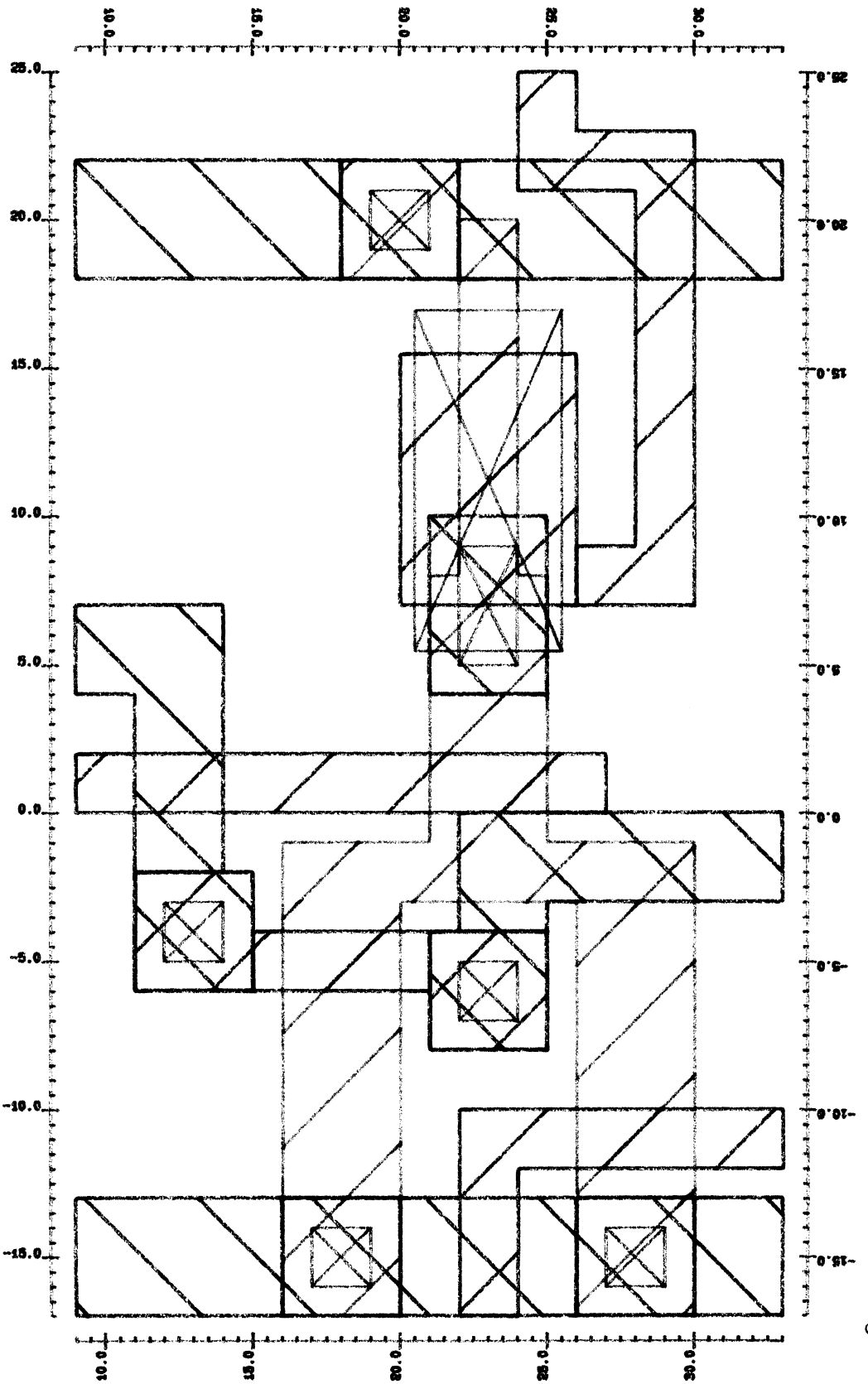


Fig3.10
cell 2

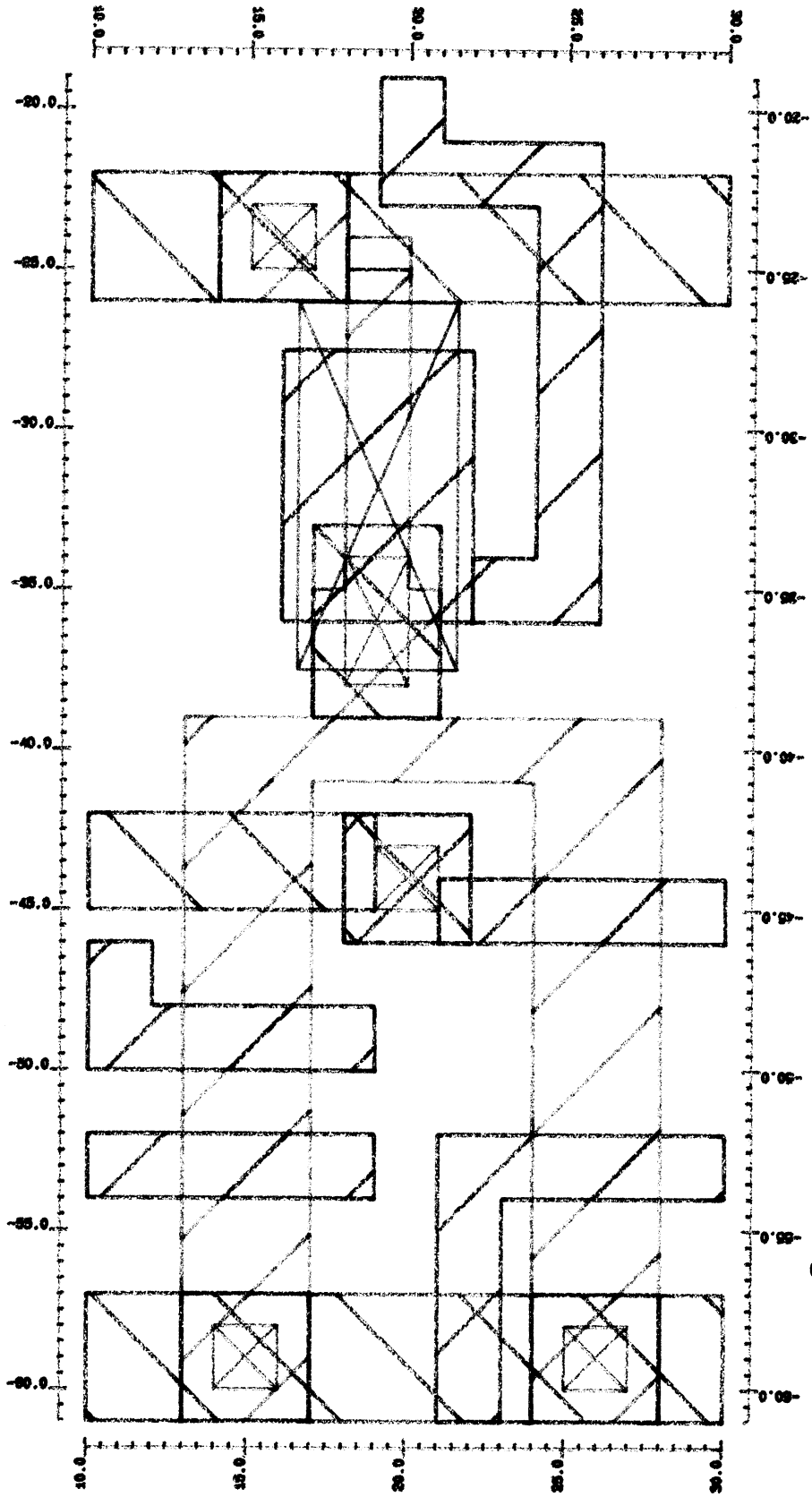


Fig3.11
Cell 4

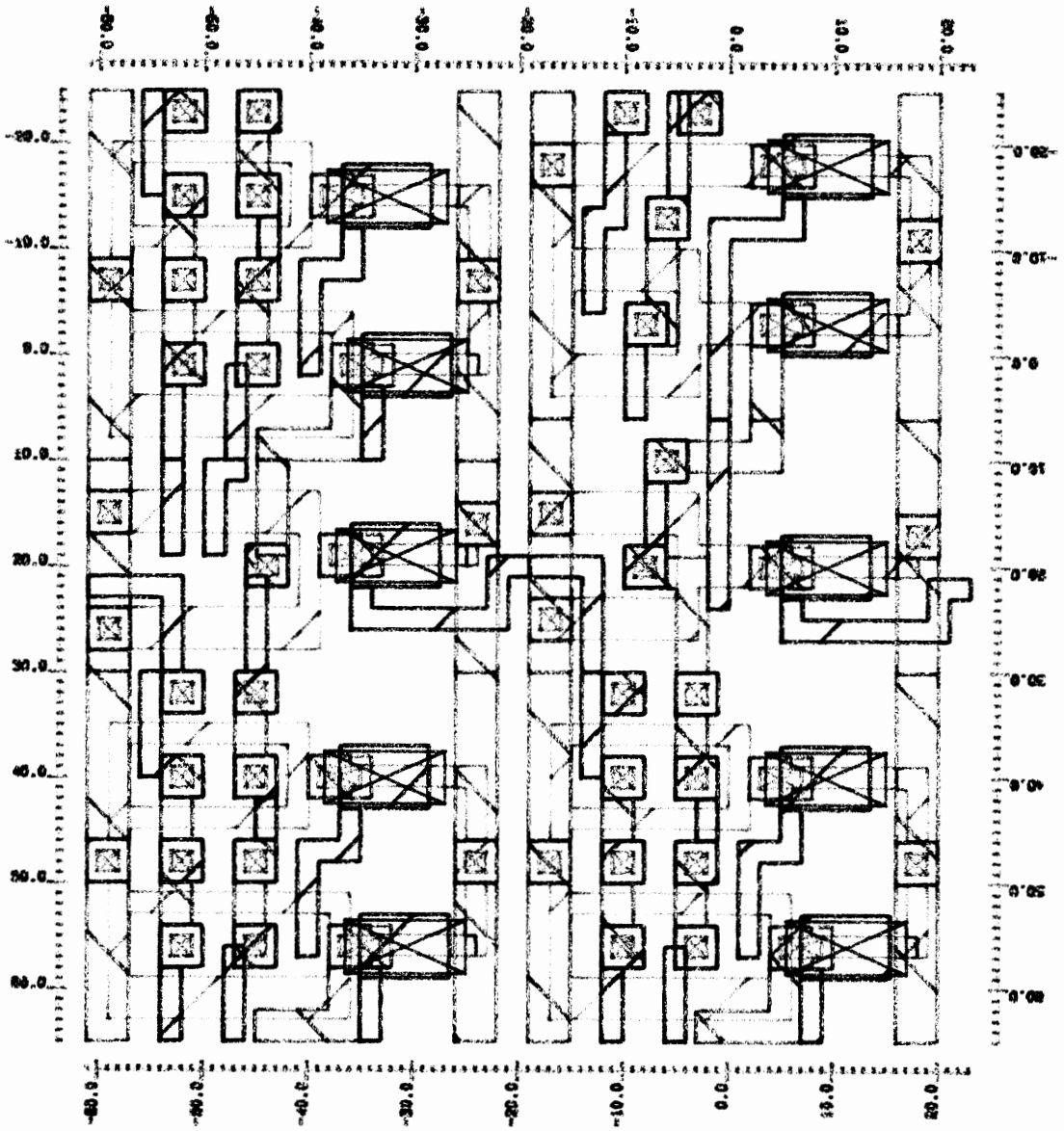


Fig 3.12 The PFA layout

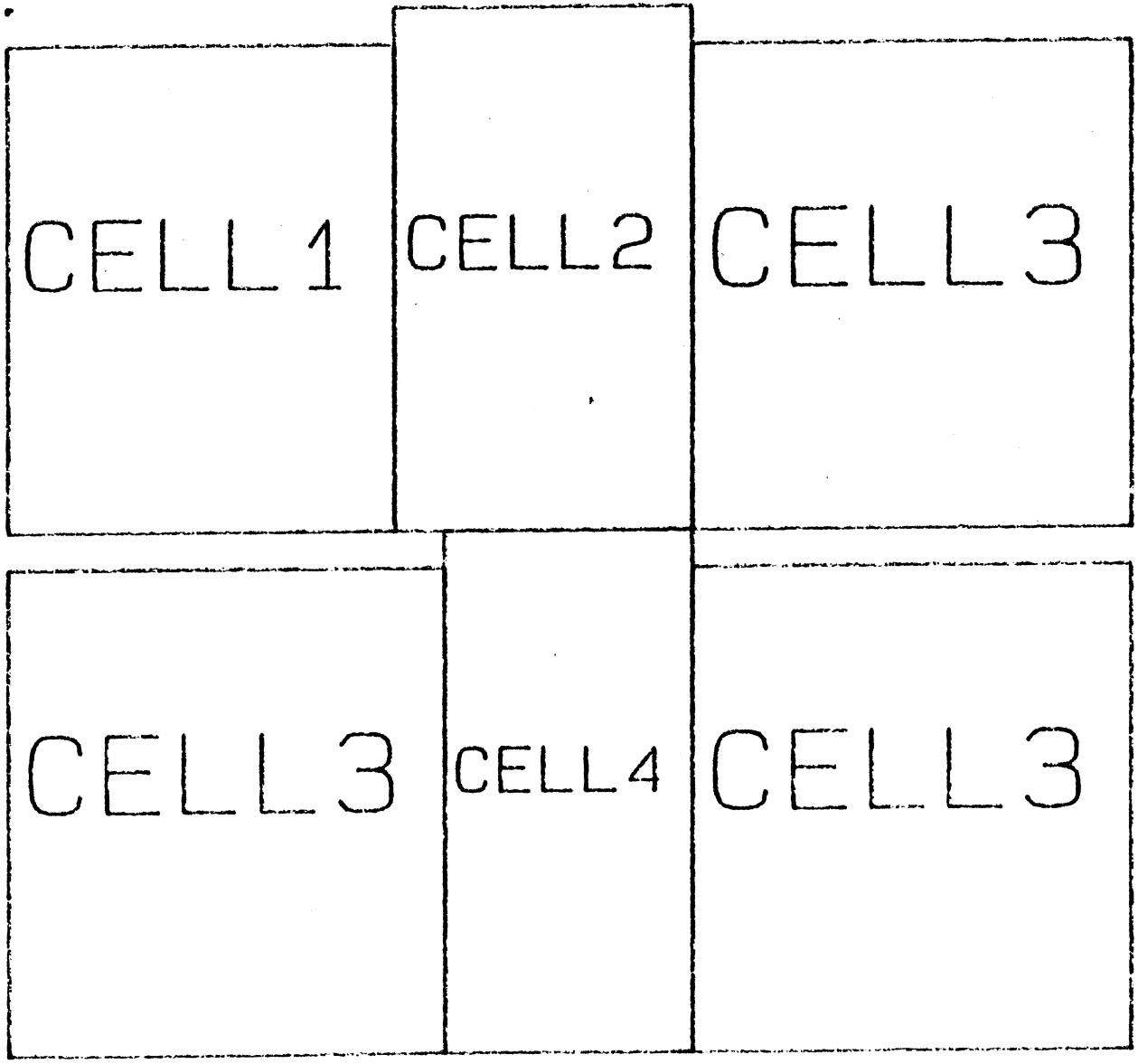


Fig. 3.13 The subcells arrangement of the pair of full adders layout.

designs of the digital filter were created directly on two graphic CRT displays, a monochromatic display for logic designs and a color display for physical layouts which are available on the SCALDstar computer aided design system for schematic design and layout of VLSI circuits.

Starting with the subcells shown in Figs. 3.8, 3.9, 3.10 and 3.11 which were created using the layout editor program (LED) and the color graphic display, a hierarchical organization was followed throughout the layout implementation of the digital filter. Cell1, cell3, cell4 and cell2 realize the XNOR gate, the XOR gate, the negated output carry of the first full adder and the output carry of the PFA respectively. The PFA layout shown in Fig. 3.12 can be obtained by using some of the (LED) commands that allow us to recall and duplicate the subcells listed above and add them to each other. The PFA structure with less details showing the arrangement of the subcells is shown in Fig. 3.13. Again a subtractor can be obtained by adding an inverter cell to one of the inputs of each adder. The four stage shift registers layout shown in Fig. 3.14 is obtained by cascading four cells containing a pass transistor followed by an inverter. The 10-bits adder and subtractor are shown in Fig. 3.15 ans 3.16, where

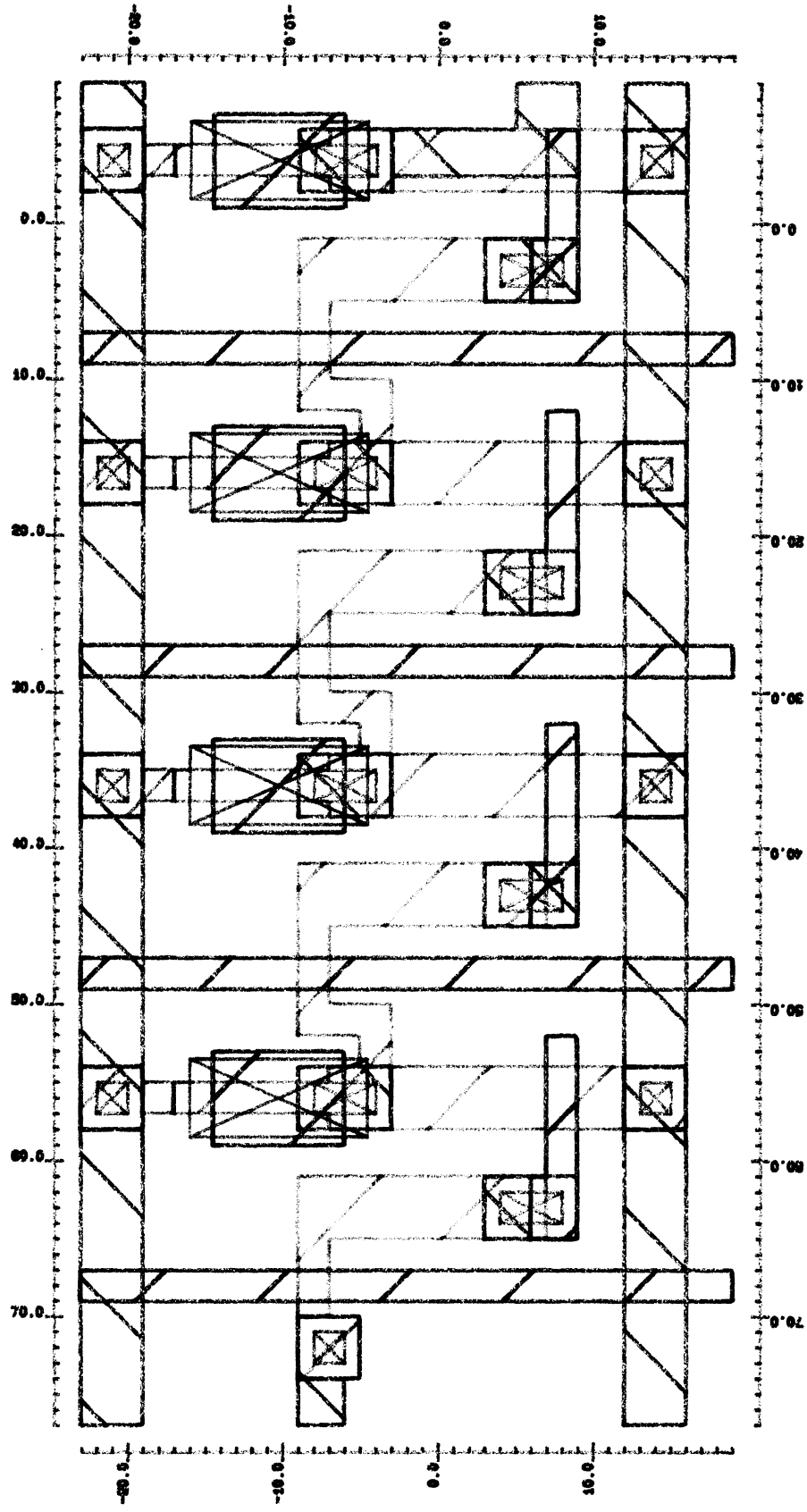


Fig. 3.14 Four stage shift register

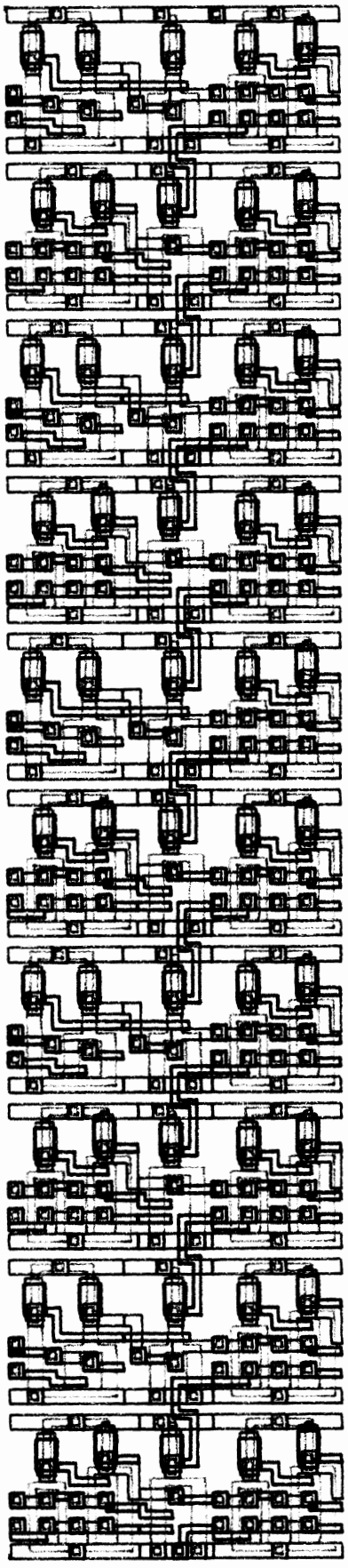


Fig.3.15 10-bits adder

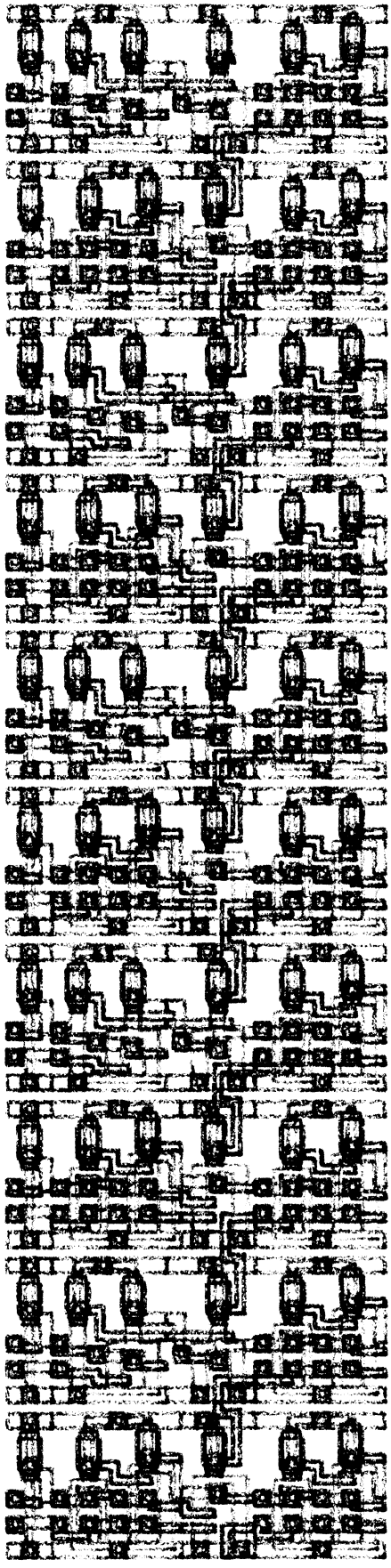


Fig. 3.16 10-bits subtractor

the input carry of the 10-bits adder is connected to ground while the input carry of the subtractor will be set high when included in the structure. The layout of each of the blocks shown in Fig. 2.6 with its weighting coefficient W, was obtained by duplicating the 10-bits adder, subtractor and shift register cells, and running the necessary connections between them. The blocks needed for the layout implementation of the digital filter are shown in Fig. C.1, C.2, C.3, C.4 and C.5 of Appendix C. The whole structure can be obtained by connecting the five blocks starting from G9 through G7, G5, G3 to G1. The 10-bits input of the digital filter can be applied to the upper left 10-bits subtractor of G9 and the output can be taken from the lower right 10-bits subtractor of G1, with the upper node as the most significant bit for both inputs and outputs. The wide metal paths running along G9 and G1 are respectively the VDD and GND wires. The two nonoverlapping clocks are applied to the shift registers through metal lines and through polysilicon when crossing the power supply paths for small distance as shown in Figs. C.1 to C.5.

Chapter 4

COMPUTER SIMULATION AND ANALYSIS

4.1 Introduction

Once the layout is created on the CAD system, some tools are needed to test the validity of the design. Three sorts of tools are available and shared between the VAX-11/750 and the SCALDsystem: (a) A design rule checker to verify the correctness of the physical design rules, (b) an electrical extractor to produce the electrical connectivity files, and (c) a simulator to test the behavior of the design. Most of the tools are applied to the design, errors are reported then corrected till we obtain the intended results.

In this chapter, the simulation results will be presented and analyzed, and the use of the analysis tools will be discussed in the next chapter.

4.2 Design rule checking.

The design rules of the whole structure were checked using the

design rules checker (DRC) program on the CAD system, which examines the layout for all violated rules. The design rules define the geometrical aspects of the patterns used in fabrication such as widths, separations and extensions of the layers with respect to each other. Since hierarchy was used in the design (i.e. the cells used were made up of subcells which are themselves made up of another subcells and so on till we reach the basic cells) the DRC program checks each cell only once no matter how many times it is repeated, which results in a faster checking and in allocating less memory.

Caltech Intermediate Form (CIF) file that contains a graphic description of the layout created on the system was used as an input file to some of the VLSI design tools installed on the VAX-11/750 for simulation purposes. The VLSI tools used are mextra (Manhatan circuit extractor for VLSI simulation), esim (a logic level simulator), powest (a power estimator) and spice2g6 (a device level circuit simulator). Esim and powest were applied to the whole circuit while spice2g6 was used for simulating one pair of full adders, since using this simulator for large circuits requires a large amount of time and memory space.

An equivalent logic circuit of the layout design was created and simulated with the graphic editor program (GED) available on the CAD system.

4.3 An estimate power consumption of the digital filter.

Since neither polysilicon nor diffusion wires can carry heavy currents, power and ground wires were routed to the cells using metal lines, but again there are some contraints imposed on using metal wires. Due to a physical process called metal migration, that causes open circuits in the metal layer carrying heavy currents, a limited amount of current should flow through a metal wire with a cross Sectional area ($A = W*T$), where "W" and "T" are the width and thickness of the metal layer. For metals like aluminium, the limit is few milliamperes per square micron of cross Section with a 1µm thickness [12]. By simple calculations we deduce that a metal path with a 4 lambda (10µm) width can supply 10mA. The unit lambda is a length unit which depends on the available commercial processes and on the technology we are using, its value in our design was chosen to be 2.5µm. An approximation of the maximum current needed to supply the

circuit was obtained by assuming that the resistivity of the pulldown transistors is negligible and the pull-ups are in saturation, with the saturation current given by

$$I_{ds} = [\mu \pounds W/2LD](v_{gs} - v_{th})^2 \qquad (4.1)$$

where v_{th} for the depletion transistor was chosen to be -4 volts, v_{gs} is zero, the oxide thickness "D" is 0.1µm, the surface mobility " µ " is $600cm^2/V*s$, the premitivity "£" of the SiO_2 is ($3.9\pounds_0 = 34.32*10^{-12}$ F/m) and the width to length ratio (W/L) of the pull-up area of all the depletion transistors used in the design is 1/4. By substituting the above parameters in the above equation we obtained I_{ds} equal to 0.0412mA.

By testing the pair of full adders for different combinations of input values, the results obtained show that a maximum of three depletion transistors are on at a time for the upper half of PFA and a maximum of four are on for the lower half, which yields to a maximum of 0.1236mA flowing in the metal path supplying the upper half of the PFA and a maximum of 0.1648mA flowing in the lower path, therefore,

in the worst case a metal line with 10µm width can supply (10/0.1648) 60 cascaded half of PFA or (10/0.2884) 34 cascaded PFA. In our case the maximum number of cascaded adders do not exceed the above Figures, therefore, a number of metal paths with (4 lambda) 10µm width are able to supply the structure without being damaged. The width of the bus supplying those parallel metal paths can be computed by approximating the maximum needed by the whole structure.

The number of adders, subtractors and shift registers with the number of transistors are listed below:

{75 pair of full adders <=> (750 pull-up + 2025 pull-down) transistors}

{130 pair of subtractors <=> (1560 pull-up + 3770pull-down) transistors}

{100 shift registers <=> (200 pull-up + 400 pull-down) transistors}

{A total of (2510 pull-up + 6195 pull-down) transistors}

Based upon the results, that a maximum of 7 pull-ups of each PFA are conducting at a time and assuming that the inverters of the shift registers and subtractors are on, we deduce that a maximum

current of 79 mA is drawn by the circuit. And in order to compensate for any nonlinear behavior of the components, 2mA were added to the obtained results, therefore, the main Vdd and GND paths require a widths of 81µm or equivalently 33 lambda. The active chip area obtained from the layout editor is $4.38*6.56$ mm^2 and contains 8705 transistors. The power consumed by the circuit is 405mW (14.09mW/mm^2).

Using the powest program which estimates the power consumption of MOS circuits, and by applying it on G9 . . . G1, we obtained the results listed in Appendix C (Section C.1). Powest produces a count of the number of pull-up transistors with the average and maximum power estimate consumed by the circuit. The maximum power estimate is based on the assumption that all the devices are in all the time. The overall average DC power estimate is 350mW and the maximum power is 529mW. Our results fall between those values and they are more accurate since we checked for the maximum number of pull-up transistors of one PFA that they are on at a time, then we computed the maximum power needed by the whole circuit.

INPUT NODES VALUES					At t = (ns)	DELAY TIME AT OUTPUT NODES			
A1	B1	A2	B2	CIN		S1	S2	b	COUT
1	1	1	1	1	0	15	16	_	3
0	1	1	1	1	70	18	_	22	_
1	1	1	1	1	140	16	_	8	_
0	0	1	1	1	210	_	18	17	_
1	0	1	1	1	280	19	28	21	_
0	0	1	1	1	350	26	27	17	_
1	1	0	1	1	420	_	21	_	16
0	1	0	1	1	490	18	_	22	_
1	1	0	1	1	560	17	_	8	_
0	0	0	1	1	630	_	29	15	15
1	0	0	1	1	700	18	28	23	23
0	0	0	1	1	770	25	37	17	24
1	1	1	0	1	840	_	16	_	11
0	1	1	0	1	910	18	_	24	_
1	1	1	0	1	980	19	_	9	_
0	0	1	0	1	1050	_	28	15	15
1	0	1	0	1	1120	18	28	23	23
0	0	1	0	1	1190	25	37	17	24
1	1	0	0	1	1260	_	23	_	16
0	1	0	0	1	1330	18	_	21	_
1	1	0	0	1	1400	15	_	9	_
0	0	0	0	1	1470	_	19	17	_
1	0	0	0	1	1540	19	28	21	_
0	0	0	0	1	1610	26	27	17	_
1	1	1	1	0	1680	15	16	_	3

*Delay time at output nodes of the PFA for different input values

Table 4.1

INPUT NODES VALUES					At t = (ns)	DELAY TIME AT OUTPUT NODES			
A1	B1	A2	B2	CIN		S1	S2	b	COUT
0	1	1	1	0	1750	28	20	22	–
1	1	1	1	0	1820	16	16	8	–
0	0	1	1	0	1890	–	18	17	–
1	0	1	1	0	1960	27	–	21	–
0	0	1	1	0	2030	24	–	18	–
1	1	0	1	0	2100	–	21	–	17
0	1	0	1	0	2170	26	27	22	15
1	1	0	1	0	2240	16	16	9	11
0	0	0	1	0	2310	–	27	16	16
1	0	0	1	0	2380	27	–	22	–
0	0	0	1	0	2450	24	–	17	–
1	1	1	0	0	2520	–	16	–	12
0	1	1	0	0	2590	26	27	22	14
1	1	1	0	0	2660	16	16	8	10
0	0	1	0	0	2730	–	26	16	16
1	0	1	0	0	2800	26	–	22	–
0	0	1	0	0	2870	24	–	17	–
1	1	0	0	0	2940	27	22	–	15
0	1	0	0	0	3110	28	18	22	–
1	1	0	0	0	3080	17	19	8	–
0	0	0	0	0	3150	–	16	18	–
1	0	0	0	0	3220	28	–	22	–
0	0	0	0	0	3290	24	–	18	–
1	1	1	1	1	3360	18	18	–	3

*Delay time at output nodes of the PFA for different input values

Table 4.1 (continue)

4.4 Timing analysis.

A timing simulation of one PFA was performed using the transient analysis portion of SPICE2G6 (a general-purpose circuit simulation program). By assigning pulses with different widths and periods to the input nodes (A1, A2, B1, B2 and CIN), we obtained the transient response output at nodes (S1, S2, COUT and b) which are shown in Fig. 3.7. The results of the simulation are summarized in Table 4.1, where the number listed below each input node name corresponds to the input value and the one listed below the outputs is the delay time taken by the signal at that node to assume a stable state. The delay was checked for every 70ns since the combination of input values changes during this period. Note that the pulses periods and widths were chosen in a way to obtain all possible combinations. The minimum pulsed input width is at node A1 and has a value of 70ns while the maximum is at node CIN with a value of 1680ns. A portion of the spice output is listed in Appendix C (Section C.2), showing the input file and the transient response at different instants.

By inspecting Table 4.1 we deduce that the worst delay at node b

is 24ns and occurs at t = 910ns for the input values (A1 = 0, B1 = 1, A2 = 1, B2 = 0 and CI = 1). From Fig. 3.5 we can see that the carry propagation delay can be obtained from the difference between the delay time at COUT and at the half adder (XOR gate) output. Note that node b shown in Fig. 3.6 is the XOR gate output of the lower half of the PFA. Again, from Table 4.1 we can see that the maximum difference between the delay at nodes COUT and b is 17ns and occurs at t = 2100 for the input values (A1 =1, B1 = 1, A2 = 0, B2 = 1 and CIN = 0). By recalling Section 2.4.3, in order to obtain the sampling rate of the digital filter, we have to compute the time delay " T " taken by the processor to perform a unitary matrix operation on a set of data Fig. 2.2. By observing Figs C.1 . . . C.5 we deduce that " T " is the time delay of a maximum of three gates 10-bits subtractors. Using (3.3.5) with n = 10, m = 3, t_c = 17ns and t_{HA} = 24ns we obtain

$$T_{add} = (n-1)t_c/2 + 2mt_{HA} \qquad (4.2)$$

$$= 220.5ns$$

which is the maximum time delay of 3 cascaded 10 bits adders. Then, in

order to obtain the time delay for the subtractors, 2ns inverter delay should be added to each adder delay, since the subtractor is implemented by preceeding the adder with an inverter which results in T ≈ 232.5ns.

Again, from Section 2.4.3 the sampling rate is 2T = 465ns or equivalently 2.15MHz. Therefore, the sampling period of the digital filter should be equal or greater than 465ns so that the signal would reach a stable state at the input of each delay element, which is the case since the designed digital filter sampling period is 500ns (2MHz).

4.5 Logic simulation.

From the layout design created on the SCALDstar system a CIF file that contains the geometric description of the layout is obtained, then sent to the VAX-11/750 over the RS232 communications link [14]. A sim file that contains the circuit description of the layout is obtained by applying the circuit extractor program (Mextra) to the CIF file. The generated file will be an input for an event driven switch level simulator (Esim), which will read input commands given interactively by the user or from an already existing commands file.

The simulation process was performed on each of the blocks by giving a set of values for the input nodes, then after one simulation step, the data propagates throughout the circuit and the state of the output nodes will be updated. Results of the simulation with their input command files are listed in Appendix C (Section C.3).

An equivalent logic circuit to the layout design was created using the graphic editor program available on the SCALDsystem. The GED allows us to use libraries containing a wide range of electrical components and make the necessary connections between them in order to obtain the desired circuit. A hierarchy was followed throughout the design starting with the following components:

{ LS283 a " 4-bit adder "} and {LS04 " an inverter "}

The adders and subtractors used are shown in Figs. 4.1, 4.2, 4.3 and 4.4, and the equivalent logic circuits of G9, G7 and G3 are shown in Figs. 4.5, 4.8 and 4.11.

Hierarchical structures require less memory space, and changes made on the components will be reflected throughout the hierarchy.

The SCALDsystem supports interconnections in two ways:

a) Explicitly, by connecting the points with wires.

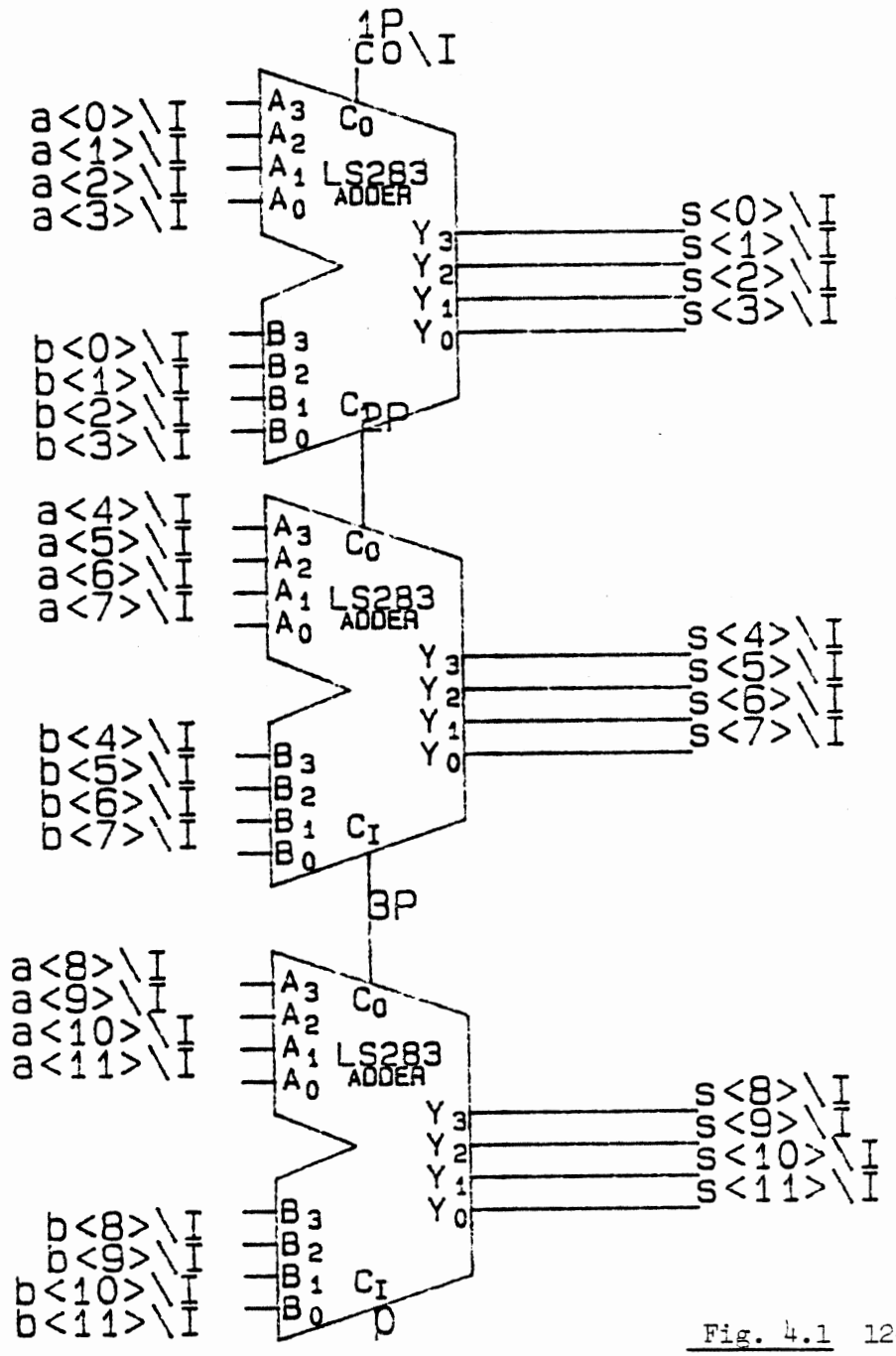
b) Implicitly, by giving the points to be connected the same signal name.

The second choice was selected, since connections by signal name will make the design more readable. Moreover, mergers/demergers were used to combine several signals into one signal which results in a concatenation of those signals. For instance, the four signal names A1, A2, A3 and A4 are equivalent to A<4..1> at the output of a 4 merger body. Once the drawings are completed, the GED creates a graphical and an electrical description of the design. The electrical description will be read by the compiler which performs an error checking and a hierarchical expansion on the design, then creates a file which will be used for logic simulation. The simulation results of G9, G7 and G3 are shown in Figs. (4.6,4.7), (4.9,4.10) and (4.12,4.13) respectively. By comparing the results obtained from " ESIM " and the ones obtained on the CAD system we conclude that the electrical connectivety of the design on the layout level is correct.

* The following is a listing of the input and output signal names used

for the layout circuit and their equivalent for the logic circuit:

EE (e10 . . . . e1)	<=>	S3<10 . . 1>	OUTPUT
BB(b10 . . . . b1)	<=>	S8<10 . . 1>	OUTPUT
DD(d10 . . . . d1)	<=>	S7<10 . . 1>	OUTPUT
CC(c10 . . . . c1)	<=>	B2<10 . . 1>	INPUT
AA(a10 . . . . a1)	<=>	B1<10 . . 1>	INPUT



DRAWING

TITLE= PF01A
ABBREV= PF01A

AST_MODIFIED=Thu Oct 17 04:55:51 1985

Fig. 4.1 12-bits adder
The structure is obtained on the graphic editor of the CAD system.

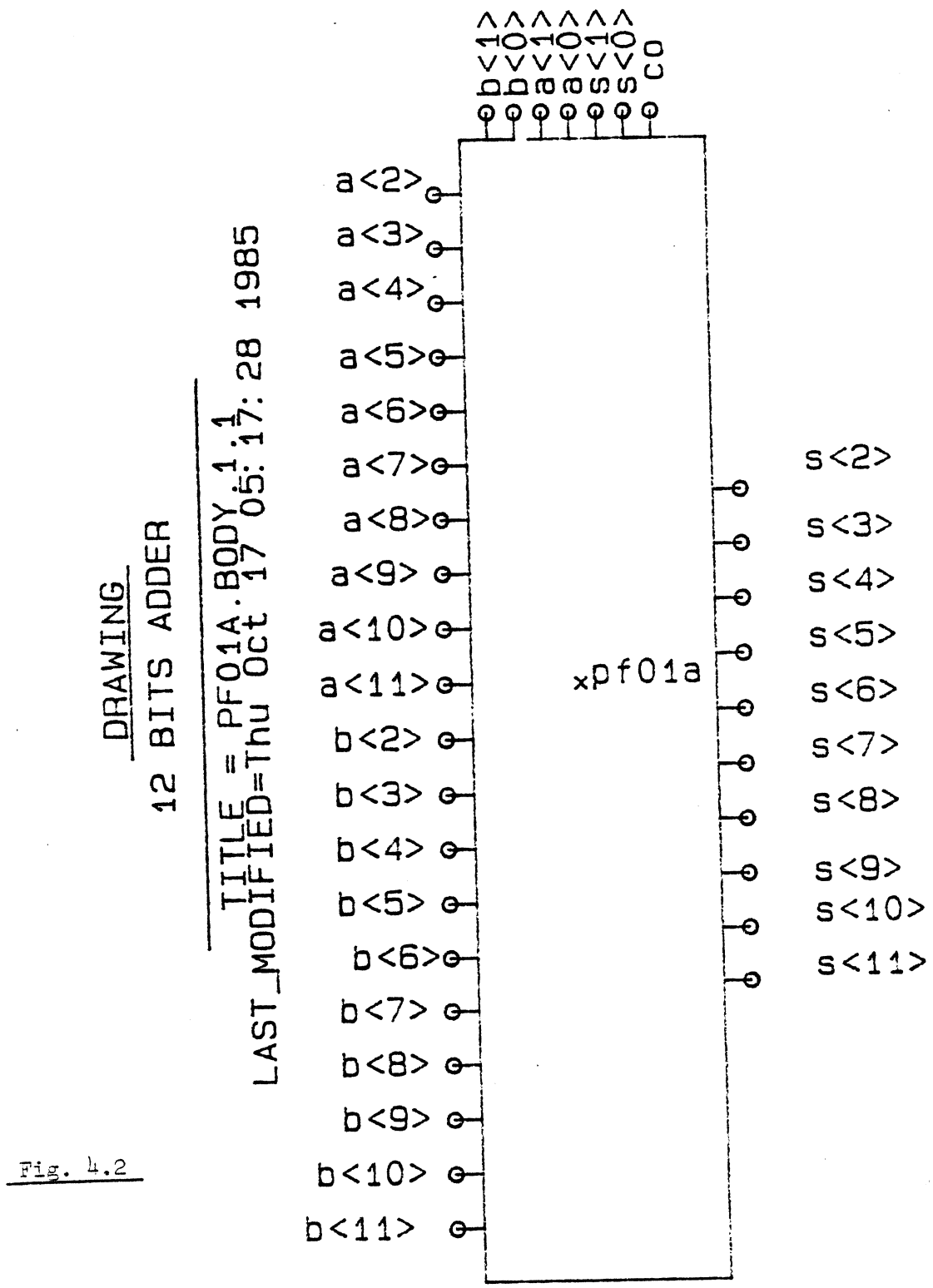

DRAWING
12 BITS ADDER
TITLE = PF01A.BODY.1.1
LAST_MODIFIED=Thu Oct 17 05:17:28 1985
b<1>^
b<0>^
a<1>^
a<0>^
s<1>^
s<0>^
co
a<2>
a<3>
a<4>
a<5>
a<6>
a<7>
a<8>
a<9>
a<10>
a<11>
b<2>
b<3>
b<4>
b<5>
b<6>
b<7>
b<8>
b<9>
b<10>
b<11>
xpf01a
s<2>
s<3>
s<4>
s<5>
s<6>
s<7>
s<8>
s<9>
s<10>
s<11>

Fig. 4.2

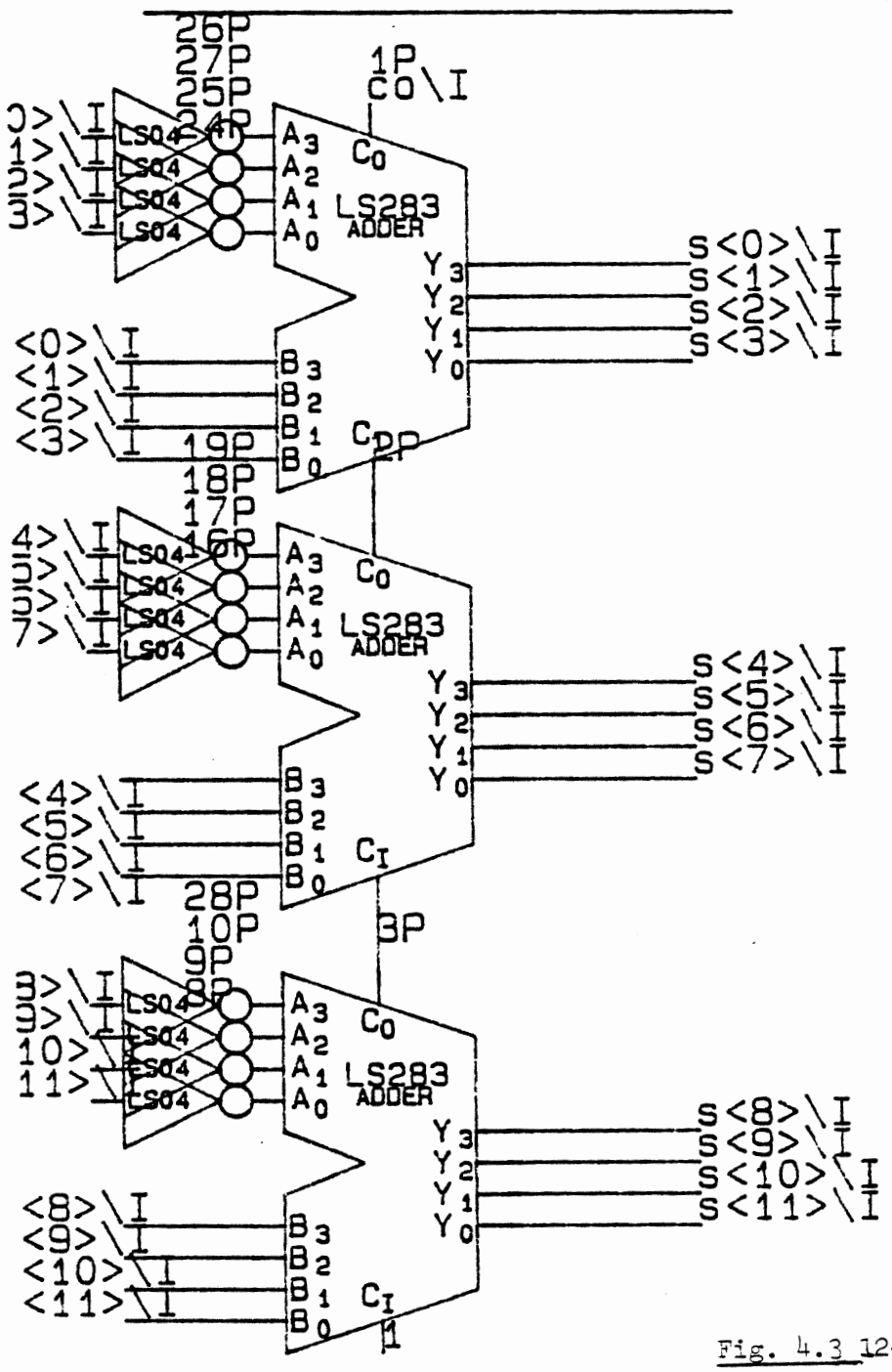


Fig. 4.3 12-bits subtractor

DRAWING
TITLE= SUB
ABBREV= SUB
LAST_MODIFIED=Thu Oct 17 05:26:42 1985

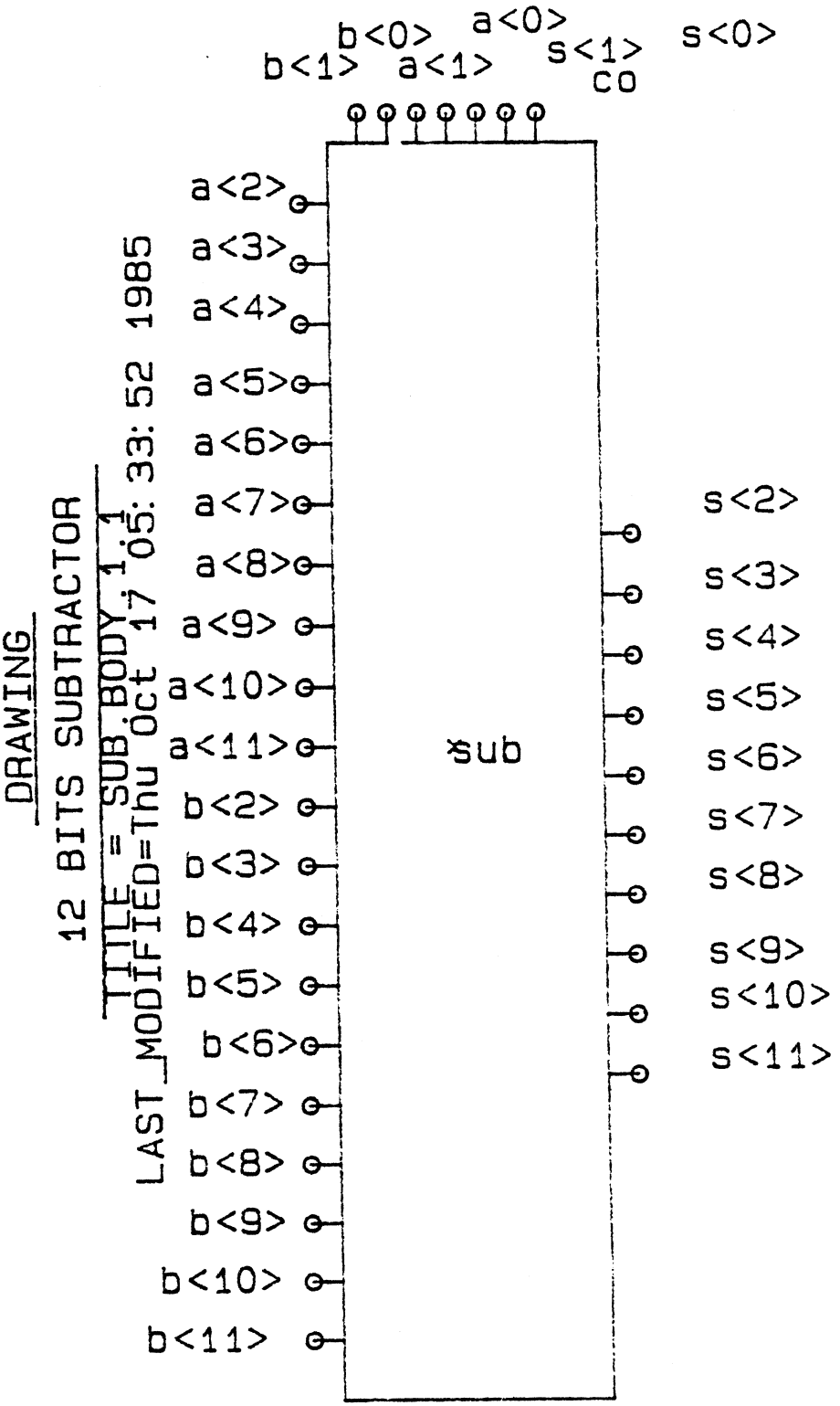
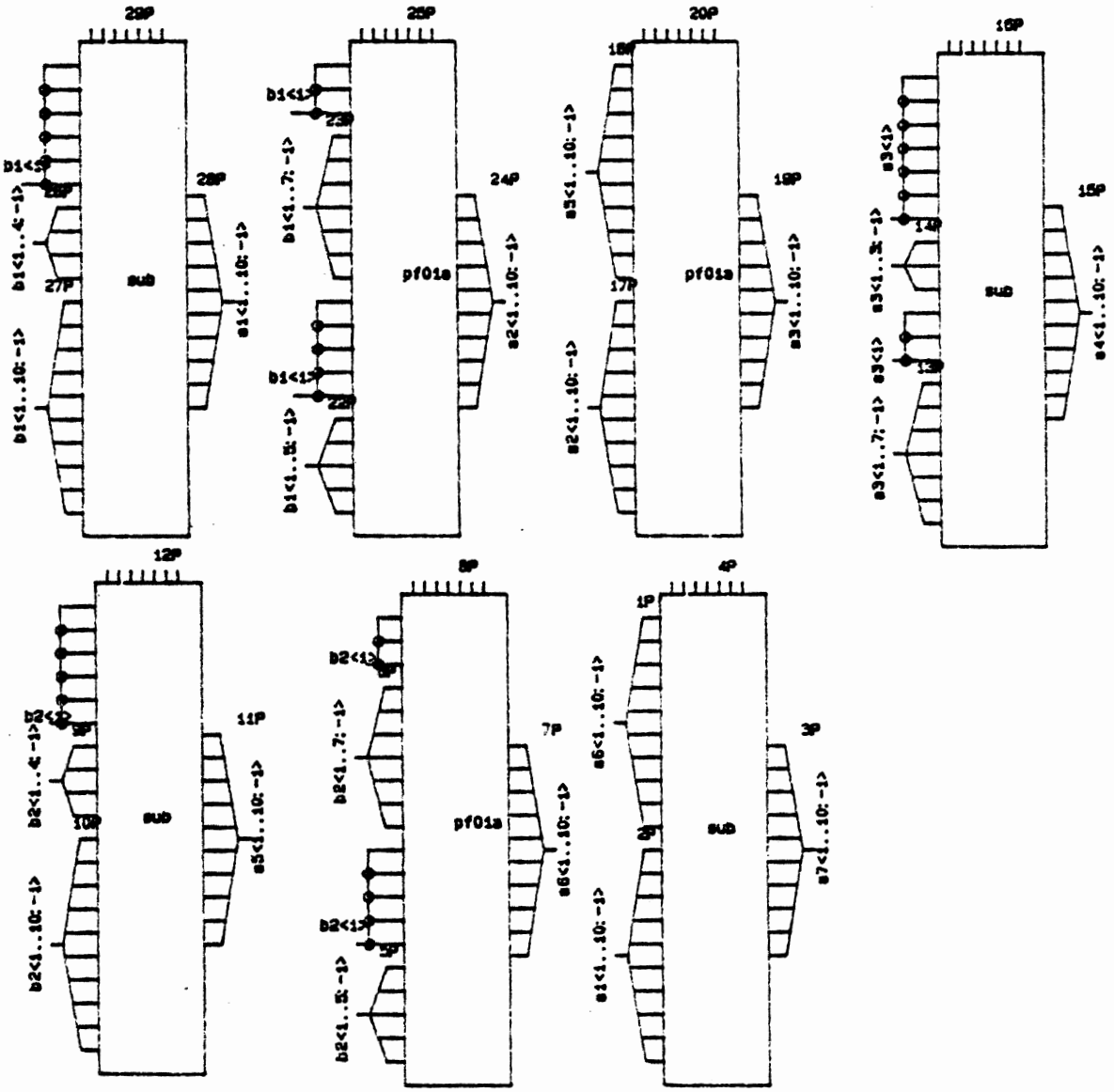

DRAWING
12 BITS SUBTRACTOR
TITLE = SUB_BODY 1.1
LAST_MODIFIED=Thu Oct 17 05: 33: 52 1985
b<0>
a<0>
s<1>
s<0>
b<1>
a<1>
co
a<2>
a<3>
a<4>
a<5>
a<6>
a<7>
a<8>
a<9>
a<10>
a<11>
b<2>
b<3>
b<4>
b<5>
b<6>
b<7>
b<8>
b<9>
b<10>
b<11>
sub
s<2>
s<3>
s<4>
s<5>
s<6>
s<7>
s<8>
s<9>
s<10>
s<11>

Fig. 4.4.



DRAWING
TITLE= G9
ABBREV= G9
LAST_MODIFIED=Thu Oct 17 04:31:56 1985

Fig. 4.5 Block structure of G9 using 10-bits adders/subtractors.

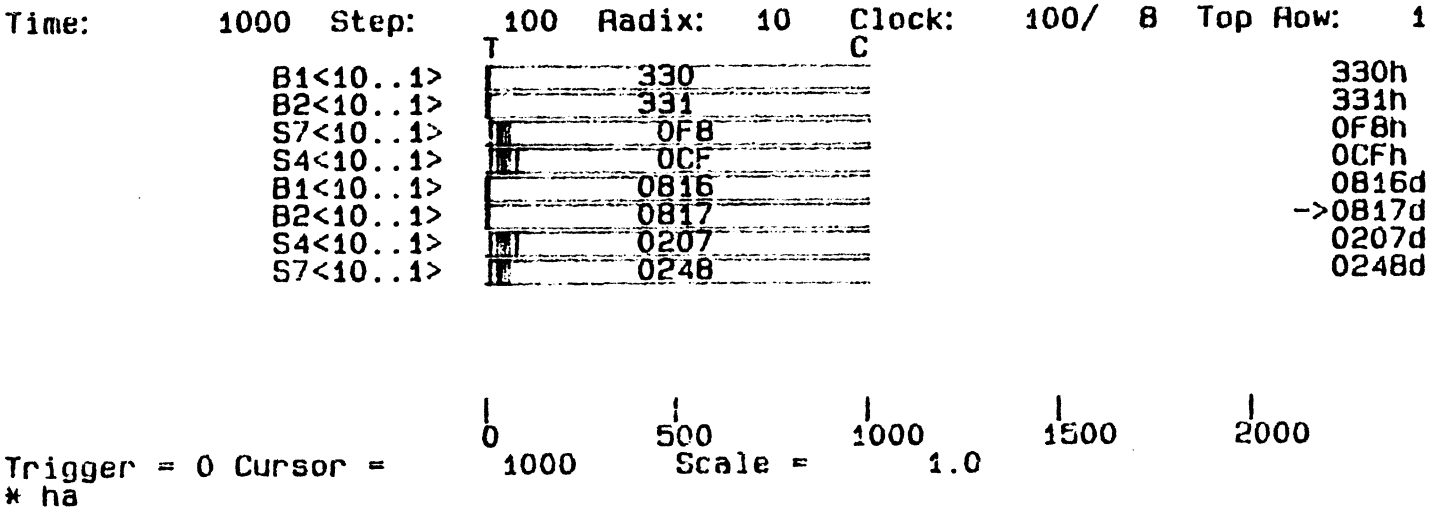


Fig. 4.6 Logic simulation results of G9.

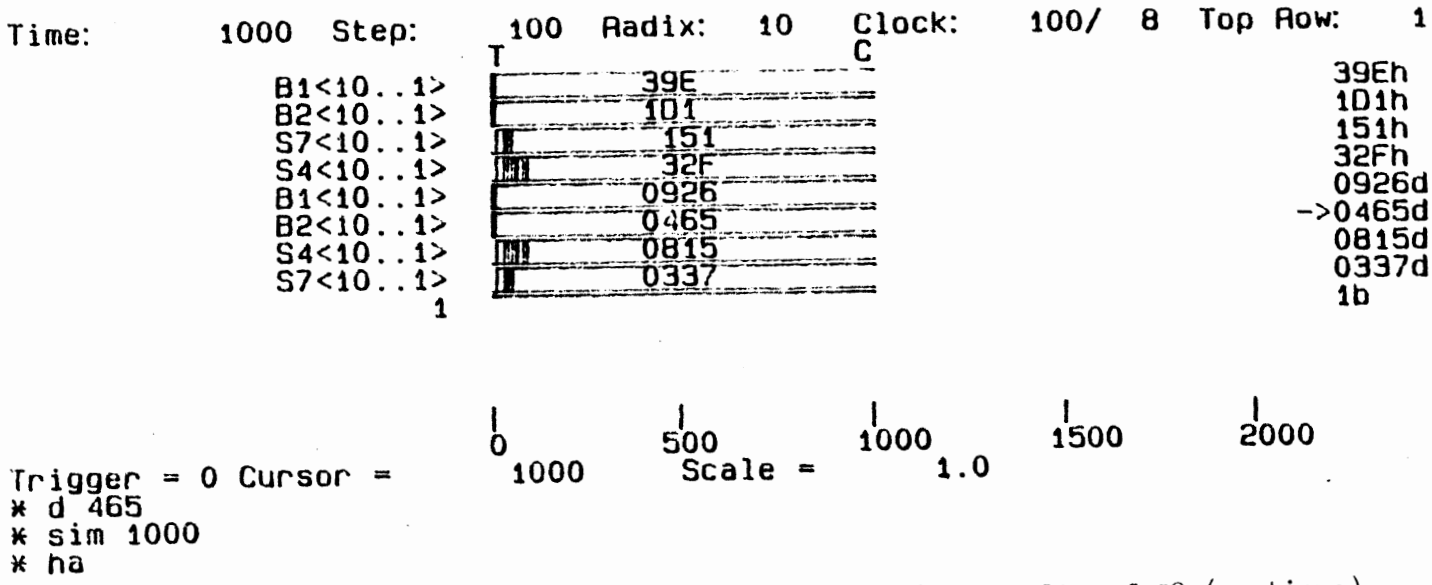


Fig. 4.7 Logic simulation results of G9 (continue)

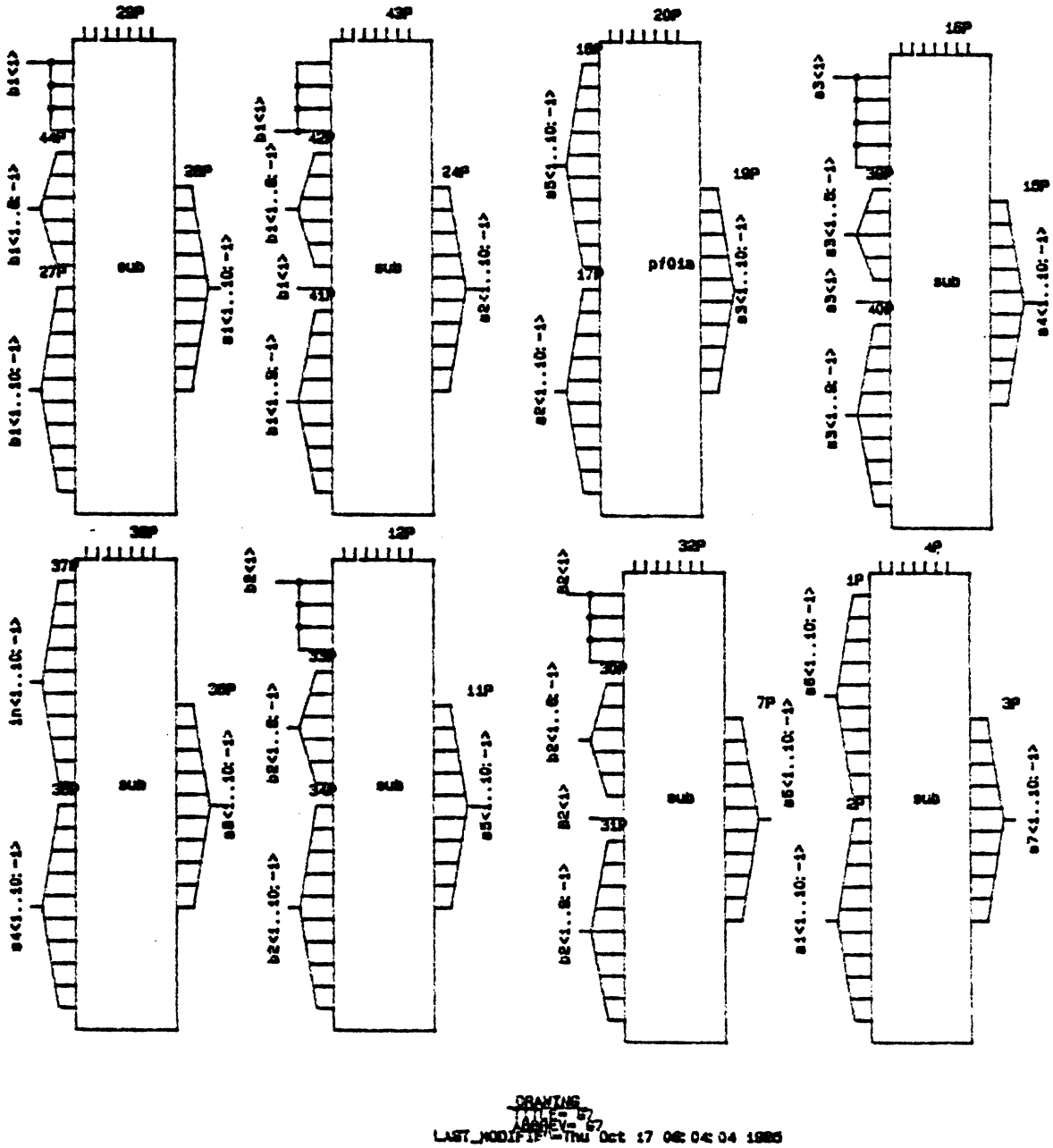


Fig. 4.8 Block structure of G7 using 10-bits adders/subtractors.

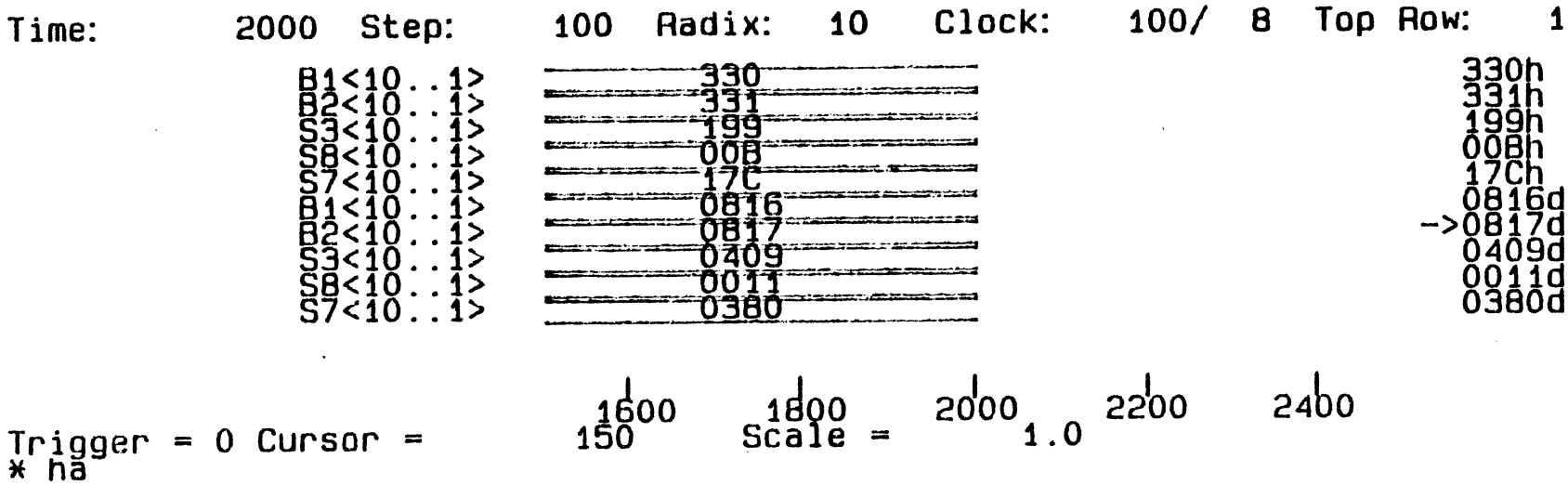


Fig. 4.9 Logic simulation results of G7

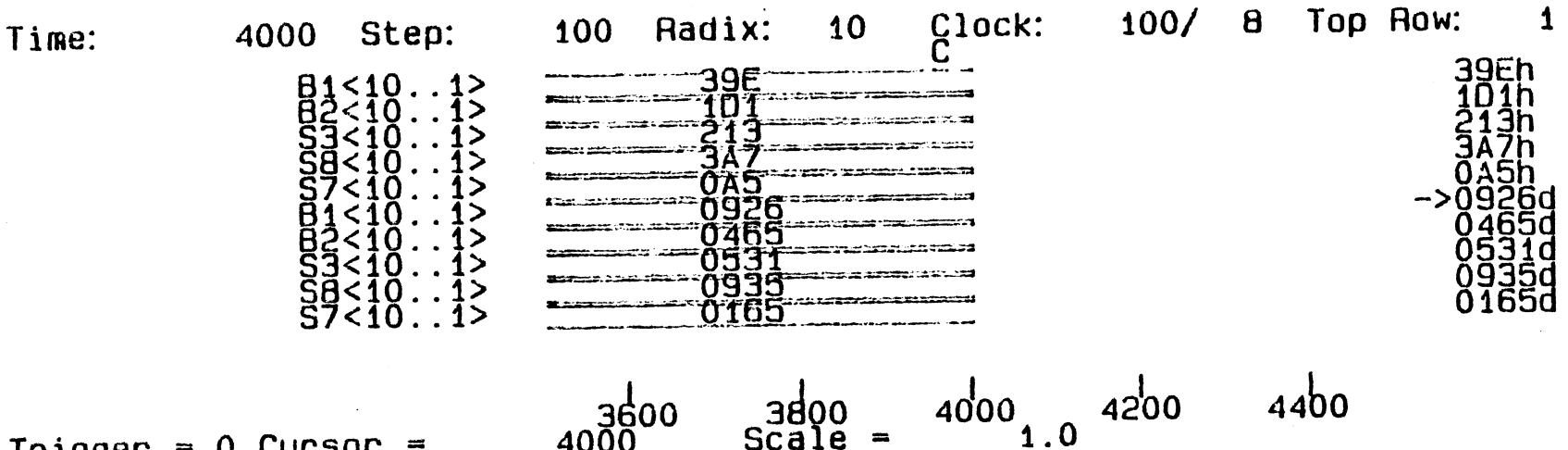


Fig. 4.10 Logic simulation results of G7(continue).

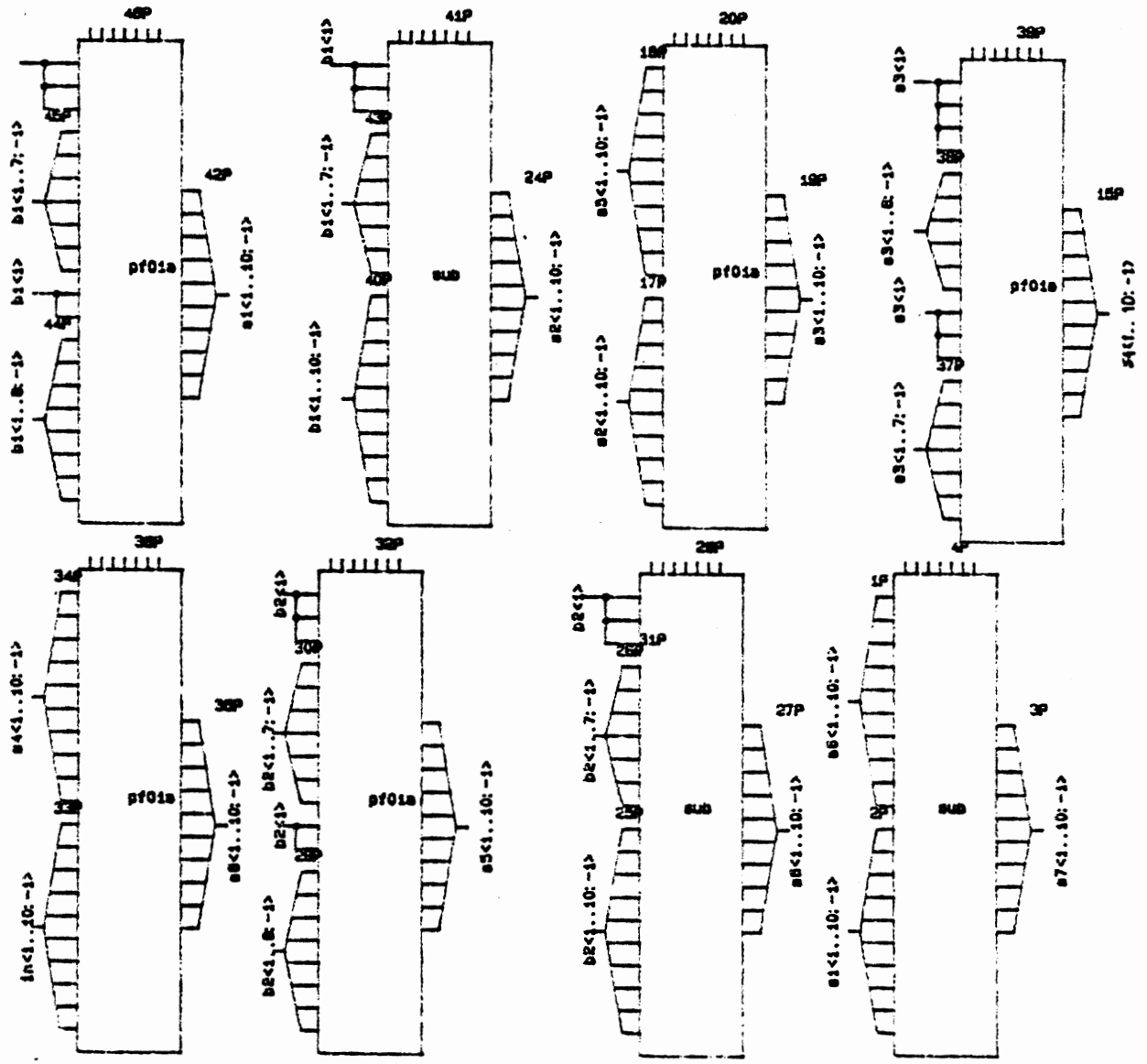


Fig. 4.11. Block structure of G3 using 10-bits adders/subtractors.

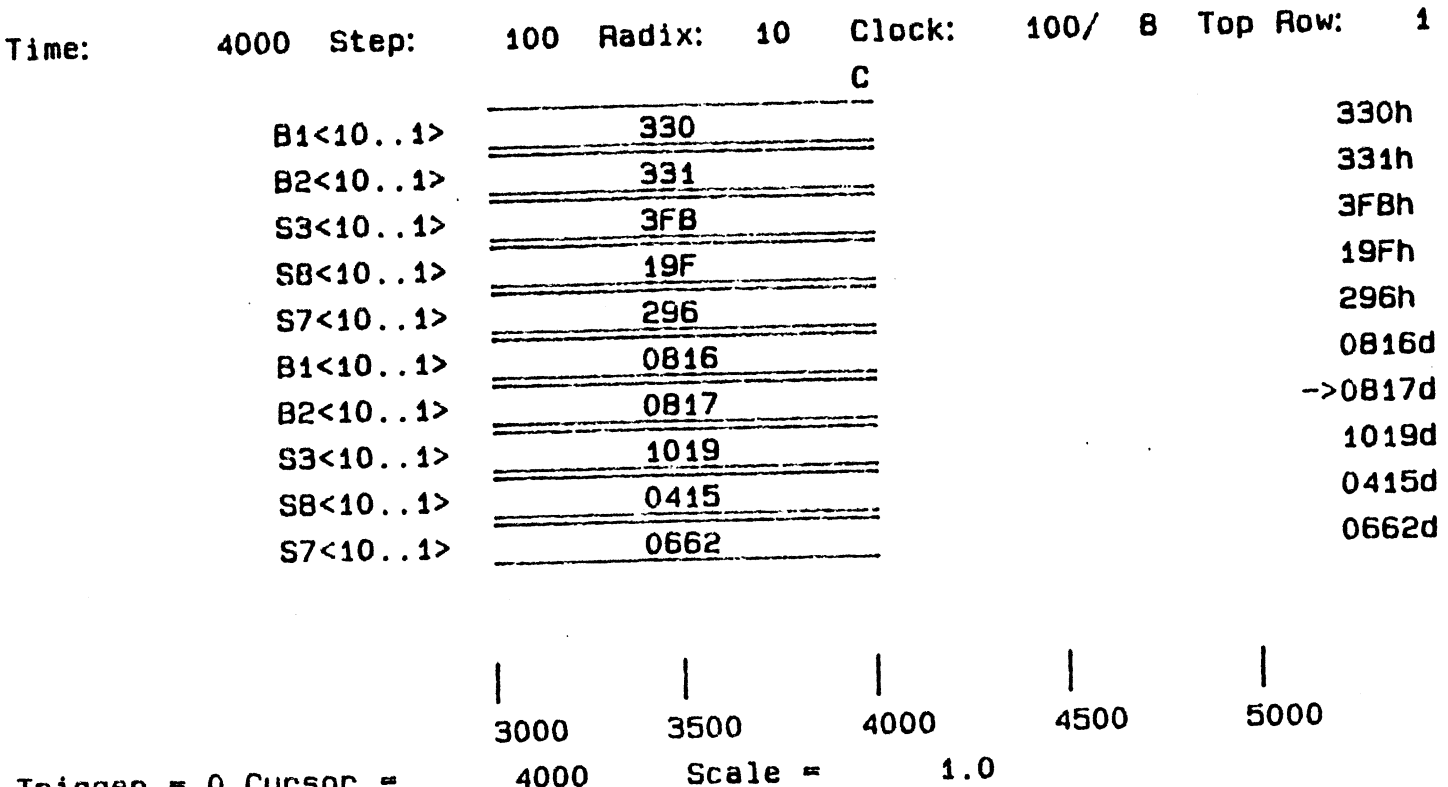


Fig. 4.12. Logic simulation results of G3.

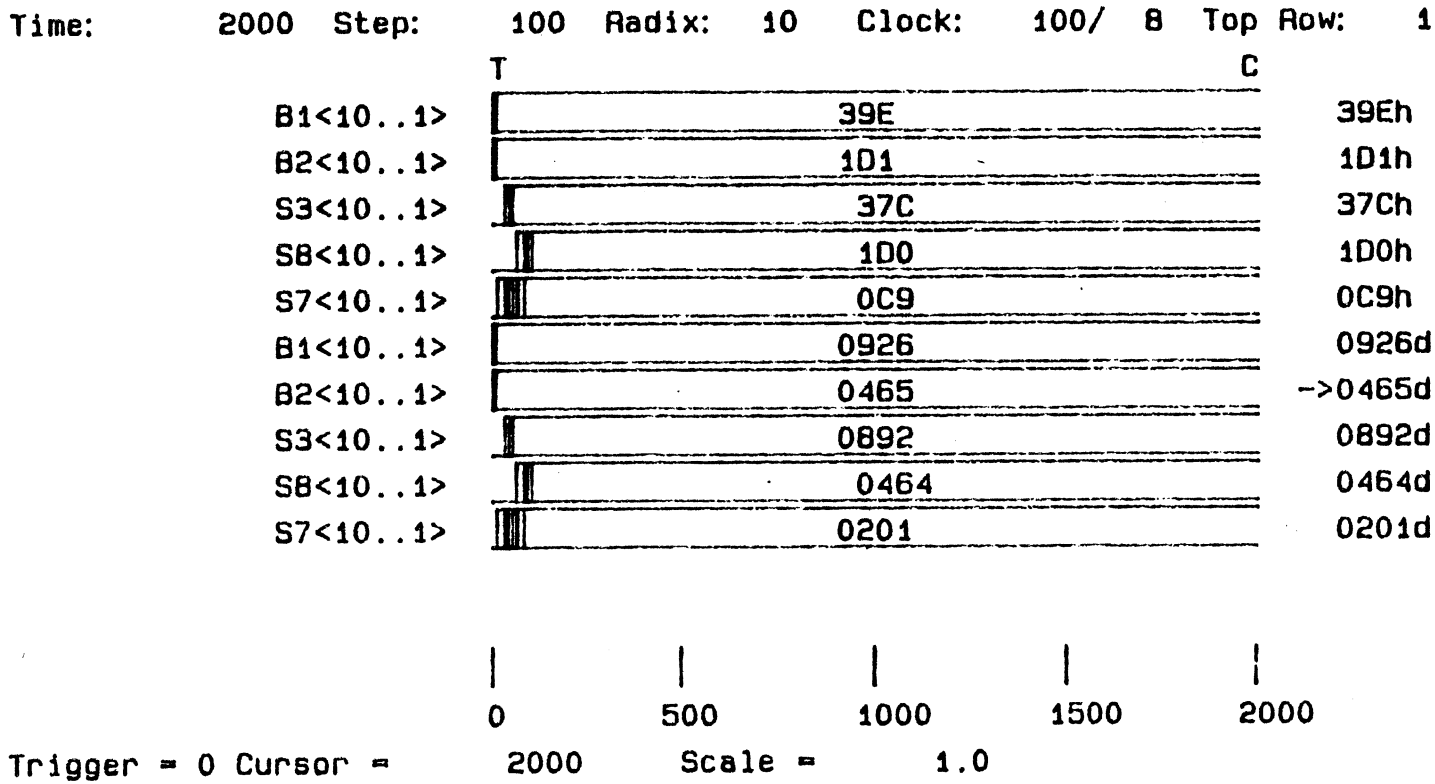


Fig. 4.13. Logic simulation results of G3 (continue).

```
* d 465
* sim 2000
* HARDCOPY
* ha
```

Chapter 5

DESIGN STAGES

5.1 Introduction.

The purpose of this chapter is to illustrate design rules checking, circuit extraction and simulation of an integrated circuit starting from the physical layout design. Additional information about the input files needed for plotting layout and schematic designs will be presented. The reader should be familiar with the SCALDstar CAD system and at least one text editor under UNIX.

A four stage shift register called " **shreg** " will be taken as an example cell throughout the chapter. The layout structure of the cell is shown in Fig.3.14.

5.2 Design rules checking.

Once the layout structure is created on the layout editor, the design rules should be checked using the DRC program which can be run from either UNIX or LED. The DRC program examines the layout for all violated rules and generates an error file which is displayed graphicaly after the program run is completed. The error messages can be

examined by zooming in the highlighted area or by reading the error file that resides in the cell directory. The DRC program results are based on information contained in a command file and on the database that describes the geometry of the layout. The command file must be created by someone who is familiar with the process rules for the technology used. The syntax to run the DRC program is:

drc command_filename in LED

or

drc cell_name command_filename in UNIX

example: **drc shreg drc_1.0.cmd**

The command_filename is by default drc.cmd and should be included in the user directory. (An example of the DRC output is listed in Appendix B).

5.3 SCALDstar layout database .

If the design is free of errors, the next step will be to create a file that describes the geometry of the layout in (Caltech Intermediate Format) CIF. The command to create the CIF file is

cif -p in LED

as a result a file named **cif** is created in the current directory. Different switches can follow the cif command to generate different kinds of information. The **-p** switch provides the (94) label construct which gives information about the signal names and their locations. The description of the CIF file is well covered in chapter 4.5 of Mead & Conway [12] and in chapter 7.2 of [15].

Every layout in the SCALDstar database is related to a technology file " technology_name.tech " which is described in the LED manual. The first number of the third line of this file specifies the number of centimicrons in one user unit. If the number is 100, this means that each grid will be interpreted as 1µm, therefore when the layout is written, an ASCII (lyout.1.1)/BINARY (lyout_bn.1.1) file will be created in the drawing directory with the size parameter lambda = 1µm which is similarly interpreted by the CIF file.

Note: If the command " **set write_ascii** " is not issued or included in the startup.led file, a binary data file which is machine readable will be created only. It is preferrable to write the equivalent BINARY files only since they are more compact than the ASCII files, but it is necessary sometimes to read the ASCII files which are similar to the

CAESAR type layout files.

Scaling the database of the design according to the technology used is sometimes necessary in order to generate files suitable for fabrication. This can be achieved by applying the following steps:

1- Edit the cell to be scaled in LED

2- Type : **set write_ascii** then write the cell in LED

This will produce a file called "**lyout.1.1**" in the cell directory.

3-The following commands should be typed while in UNIX:

/u0/layout/rotate cell/lyout.1.1 cell/temp "T sc 0.0 0.0 0.0 sc 0.0 " 0.0

(where sc is the scale factor which should be a real number and by which the x and y coordinates of the layout will be multiplied)

cd cell change to the cell directory

mv temp lyout.1.1 move the temporary file created during the transformation into the actual database file.

cd back to your home directory

Go back to the LED and edit the cell than write it in order to overwrite the old file. It will be noticed on the screen that the dimensions of the cell have been changed due to the scaling process.

An example of the CIF file for the " shreg " cell is listed in

Appendix B. The best way to understand the structure of the CIF file is to go through the comments statements included in the file listing.

5.4 VALID TO VAX.

In order to simulate the integrated circuit, the CIF file should be sent to the VAX-11/750 through the RS232 communication link available between both systems. At this point all commands will be used in the VAX environment while under UNIX.

The steps to follow in order to invoke the communication process from VAX are:

Type: **validcom**

Hit the return key

(login as a VALID user)

Type: **~ t fnl filename.cif**

Example: **~ t cif shreg.cif**

(when you type " ~ " it should not be echoed (hit both the shift and the " ~ " keys hard) but when you type " **t** " you will be prompted with " **~ take** ". Note that the file " **cif** " is the one to be copied from VALID and " **shreg.cif** " is the new file which will be created in the user home directory on VAX.).

To exit back to VAX, logout from VALID then type

~.

(again " ~ " should not be echoed)

For more information about the VAX to VALID communication refer to [14].

5.5 Circuit extraction.

Before we proceed in the circuit extraction, it is important to make the following changes on the cif file:

(The changes can be made while in VALID before sending the CIF file or on the VAX after receiving it)

1- Edit the cif file

2- Use the following commands to modify some of the statements that will affect the circuit simulation results and the performance of mextra.

a*	**1,$ s/94 .*:/94 /**	To remove prefixes of the signames
b*	**1,$ s/R 1 0 //**	Remove the rotate commands.

(Note: none of the subcells used in the design should be rotated, if any of them was rotated, it should be smashed by pointing to the subcell using the puck and typing the command **smash.** The " R 1 0" statement means that the cell is rotated by zero degrees.)

c* Add the " **!** " mark to all signal names repeated on more than one node, for example if the signal name " **PHI1** " is used on two nodes or more, the following command should be used:

1,$ s/PHI1/PHI1!/

(Note: Power supply and ground wires should be labeled by **Vdd** and **GND** respectively)

d* Since the layout is created with lambda = 1.0µm, the user can modify the value of lambda by changing the values of the second and third integers of the DS " definition start symbol ". For example, if the DS statement is

DS 1 1 4;

and a 2.5µm value of lambda is sought, the following command should be used

1,$ s/ 1 4;/ 25 40;/

* For information about shell scripts to do the above changes refer to

[14].

By applying the program " mextra " to filename.cif (filename is **shreg** in our case), four files will be created:

1- filename.log which contains a count of the number of transistors and their types, and messages about possible errors such as a list of signal names which are repeated on more than one node

2- filename.nodes which is a listing of node names and their CIF locations. Numbers will be assigned to nonlabeled nodes.

3- filename.al

4- filename.sim which is the most important file since it is compatible with most of the simulation tools. The format of the first line of the sim file is:

| units: scalefactor tech: technology name

where scalefactor is an integer by which all the coordinates listed in the sim file will be multiplied. The technology name in our case is nMOS. The other lines list the transistors with their types, locations and information about capacitance in the circuit. An example of each is listed below:

e 25 GND OUT1 500 1000 13500 1750

where

* e is the type of the transistor which is enhancement in this case and it could be (d for depletion or u for unusual transistors).

* 25, GND and OUT1 are respectively, the node names of the gate, source and drain of the transistor.

* 500 and 1500 are the channel length and width in centimicrons.

* 13500 1750 are the lower left coordinates of the transistor.

while the capacitance description is of the form

C 21 GND 61

which means that the capacitance between nodes 21 and GND is 61 femto-farads.

The syntax of the mextra command is

mextra filename.cif

An example of each of the .log and .sim files created by mextra is listed in Appendix B.

5.6 Event driven switch level simulation (ESIM).

Esim is a logic level simulator for NMOS transistor circuits, it reads all the information about the circuit from the sim file and expect

a set of input commands from the user. If you want to enter the commands interactively type

esim filename.sim

you will be prompted with

sim>

and the simulator will be waiting for the commands from the user. A simulation session for the " shreg " cell is listed in Appendix B, each of the simulator commands is followed by a comment line explaining its function. The same set of commands used in the example can be typed into a file which can be read by the esim using the argument

esim shreg.sim shregcmd.sim

where shregcmd.sim is the inputs command file. Another useful command is to write the output in another file

esim shreg.sim -shregout.sim shregcmd.sim

with shregout.sim as the output file. After each of the above commands esim will prompt you with

sim >

which enables you to continue entering more commands. In order to exit the simulator the **q** command should be used.

Example: Suppose you are working on a circuit where repeated set of commands are used during simulation such as trigerring an input clock, a command file that triggers the clock can be written and used repeatedly while simulating. To read a command file while in simulation process type

sim> **@ trigger**

where the command file name is trigger. When the new version of esim will be installed more advanced and helpful commands can be used. An example of a command file is listed in Appendix B.

5.7 Timing analysis using CRYSTAL.

Crystal is a program that reads the circuit description in sim format and extracts the information it needs to do the timing analyses. A single analysis using Crystal is sufficient to obtain information about the worst delay at any node of the circuit and to find the slowest path. Moreover, Crystal can perform static logic simulation.

In order to invoke Crystal use the command

crystal filename.sim

then define the input nodes. For example,

inputs node1 node2 . . . noden

then the delay command should be used which tells the crystal about the time when the input nodes will change values. For example

delay node1 0 4

indicates that the latest time when node1 rises is 0ns and the latest time it will fall is 4ns. Now by typing

worstdelay

the node with the worst fall or rise delay time will be displayed and the path of all the nodes that caused the worst delay will be printed. If the worst delay information is needed for a specific node, for example node2, the command becomes

worst node2

and to exit the simulation type **^D**

For more information about the available Crystal commands refer to the on line Unix manual for the VLSI tools. For illustration, a simulation session with comments statements is listed in Appendix B.

5.8 Estimate of power consumption using POWEST.

An estimate of power consumption of any MOS circuit can be obtained by applying the powest program to the MOS circuit description in the sim format. the command syntax is

powest < filename.sim

Powest will respond by producing a count of the number of pull-up transistors with an estimate of the average and maximum power consumed by the circuit. The maximum power estimate is based on the assumption that the devices are on all the time. A list of the default supply voltage and the process parameters values can be obtained using the -p option.

powest -p < filename.sim

The default parameters can be overriden using the param = value option. For example,

powest -p Gamma = 0.0 vdd = 4.5 u0=0.06 < filename.sim

The powest output of the shreg cell is listed in Appendix B.

5.9 SPICE2G6.

SPICE2G6 is a general purpose simulation program for nonlinear dc, nonlinear transient and linear ac analyses. The transient analysis portion will be used to compute the transient output of the shreg cell.

The spice2g6 software is available on the VAX-11/750 and on the SCALDstar system. The steps needed to use the software on both systems will be discussed.

VAX/spice: The input file needed for the spice simulation is divided into two parts:

1- The circuit description which can be generated from the .sim file.

2- A header file which contains the SPICE models, input voltages, control cards, etc. This file should be created by the user.

An example of the header file for the shreg cell is listed in Appendix B. Note that all node names should be enclosed between brackets. If the .sim file and the user input commands file are ready, we can produce the spice input deck using the command

pspice -rm -e shreg.hdr shreg.sim

where shreg.hdr is the input command file created by the user. The name of the output file is **shreg.spcin**.

The transient output can be obtained by running the spice2g6 program on **shreg.spcin** using the command

spice < shreg.spcin > shreg.plot

where shreg.plot is the spice2g6 output. An example of each of the above files is listed in Appendix B.

VALID/spice: There are three steps in making a SPICE deck on the VALID CAD system:

1- Run the DRC program on all cells in the subtree to bc cxpanded. Use the command file " drcex_1.0.cmd". The syntax of the command is

drc shreg drcex_1.0.cmd in UNIX

or

drc drcex_1.0.cmd in LED

where shreg is the cell name.

2- Run the SCALD compiler on the root cell to expand the netlist. The compiler reads the connectivety file (**lyout_cn.1.1**) which resides in the cell directory and produces an expansion file called " **cmpexp.dat**". This file could be quite large and take a long time to generate.

To run the compiler type:

compile shreg lyout

The error messages of the compilation will be listed in a file called " cmplst.dat ".

3- Create a header file similar to the one created on the VAX then run the spice interface program " GSPICE " to generate the output file in SPICE format. An example of the header file is listed below.

---

* Filename: spice.hdr

```
.WIDTH OUT = 80

.MODEL DEPL NMOS (VTO = -4.0 PHI=0.7)

.MODEL DEPL NMOS (VTO = 1.0 PHI=0.7)

.TRANS 2NS 200NS

VDD %VDD 0 5

VGND %GND 0 0

V0 %SIGNAL_GND 0 0

VSUB %WAFERSUBSTRATE 0 0

V1 %IN 0 0

V2 %PHI1 0 PULSE(0 5 0NS 0NS 0NS 20NS 40NS)

V3 %PHI2 0 PULSE(0 5 20NS 0NS 0NS 20NS 40NS)

.PLOT TRANS V(%OUT) V(%PHI1)  V(%PHI2)
```

---

* Note the usage of "%" in front of the signal name instead of including it between brackets (the way it is done on the VAX).

The last step is to produce a file called "**spice.netlist**". This can be done using the command

gspice

Now the spice.netlist file is ready to be run by SPICE by typing

spice < spice.netlist > spice.plot

where spice.plot in this case contains the transient response output of the shreg cell.

Command files for the compilation (**compiler.cmd**) and SPICE interfaces (**spice.cmd**) are required. An example of the compiler.cmd and spice.cmd files are listed below.

*1. compiler.cmd

```
-----------------------------------------------------------------

root_drawing 'shreg';
compile lyout;
directory 'fadi.wrk',
          '/u0/lib/lyout/lyout.lib';
output list, expand, synonym;
end.
```

```
-----------------------------------------------------------------
```

*2. spice.cmd

```
-----------------------------------------------------------------

library_file '/u0/lib/spice/spice.prt';
header_file 'spice.hdr';
```

end.

5.10 Makeplot.

The Makeplot program generates a plotter output of a cell that has been created with the layout editor. The program receives its commands from a command file (the default name of the command file is " plot.cmd ") which determines how the design is to be plotted. A typical example of the plot.cmd file is listed below:

```
{ specifying an HP plotter and a (16.5- by 11-inches) sheet size}
plotter HP 16.5 11
{ Specify the pen position to be used for text plotting }
text_pen 1
{ rotate the plot by an angle of 90 degrees }
rotate
{ The autofit command will automatically scale and if necessary rotate
the top-level cell to fit the plot sheet}
autofit
{ Specifies the work directory that contains the drawing }
```

use fadi.wrk

{The mask command defines the mask_name, fill_number, boundary_number, and pen_number. The fill_number specifies the fill pattern, if 0 is specified, the pattern is empty. Boundary_number is two digits number which defines the line style of the boundaries, and the pen_number specifies the pen position for each mask}

mask bounding_box 0 1 1

mask polysilicon 0 19 2

mask diffusion 0 19 3

mask metal 0 19 4

mask implant 0 19 5

mask cut 0 19 1

mask errors 0 19 1

{userunits is the user-specified units included in the technology file. In this case it is 100 user units per micron.}

userunits 100

The makeplot program can be run by applying the command

makeplot -c cellname

The output plot file created by the Makeplot program is " **hpplot.dat** ". The plot of the design can be obtained by filtering the **hpplot.dat** file using an hpfilter program and directing the output to port RS232C of the CAD network. The command argument is

/u0/editor/lib/hpfilter < hpplot.dat > /dev/ttyc

or simply

hpl

(The HP plotter must be connected to the RS232C port of the CAD system).

5.11 Hardcopy.

In order to obtain a hardcopy of the logic drawings created on the graphic editor, the following directives should be included in the startup.ged file.

---

set mono_hpplot

set LOcal_plot

The harcopy command should be issued when in GED. The command syntax is

ha sc

where sc is a real number by which the drawing geometry is multiplied. The hardcopy command will produce a plot file named (**hpplot.dat**), similar to the one produced by Makeplot. Similarly the output plot can be obtained using the command

hpl

(If scaling one of the plot coordinates is desired, the hpplot.dat should be edited and the first line (A transformation matrix) of the file should be modified. An example of that first line is

@ 1.0000E-01 0.0 0.0 0.0 1.0000E-01 0.0

To scale up the y coordinates by 4, the above line must be changed to

@ 1.0000E-01 0.0 0.0 0.0 4.0000E-01 0.0

(Note that the first and the fifth numbers are the scale factors of the x and y coordinates respectively).

Chapter 6

CONCLUSION

In this thesis, a model VLSI circuit was implemented on the SCALDstar CAD system using a 2.5µm nMOS-technology. The equivalent logic circuit and the physical layout structure of the design were simulated using software packages available on the VAX-11/750 and the CAD system.

The circuit design is an IIR bandpass orthogonal digital filter which was realized with a multiplierless structure by applying the cannonical sign digit technique on the coefficients in order to minimize the amount of hardware and improve the performance of the circuit.

The layout and the logic circuits were built using the layout and graphic editor programs available on the CAD system. The electrical circuit description of the layout was obtained by applying the circuit extractor program available on the VAX. Then, power estimation, logic and timing simulations were performed on the circuit.

A comparison between the results obtained from " ESIM " (the

logic simulator on the VAX) and the ones obtained on the CAD system showed that the electrical connectivity of the design on the layout level is correct. The timing simulation results showed that the 2MHz sampling rate required by our digital filter structure can be achieved. The overall average DC power estimate obtained from simulation is 350mW and the maximum power is 529mW.

APPENDIX A

* Program # 3

```
C     THIS IS A PROGRAM TO PLOT THE FREQUENCY CHARACTERISTICS
C                    OF A DIGITAL FILTER
      DIMENSION QNUM(11),DEN(11),U(11),D(11),T(11),B(11)
      DIMENSION A(11),P(11),B(11),X(202),Y(202),BUF(5000)
      DATA A /1.0,0.0,-0.280762,0.0,1.25907,0.0,0.133999,0.0,0.382950,
     $   0.0,0.15625/
      DATA P /0.124020,0.0,-0.413150,0.0,0.742131,0.0,-0.730574,
     $ 0.0,0.428410,0.0,-0.117188/
      PI=3.141592654
      F=0.0
      G=0.0
      WRITE(6,100)
100   FORMAT(5X,'NUMBER',4X,'FREQUENCY',4X,'FREQUENCY RESPONSE')
      DO 1 I=1,200
      DO 2 J=1,11
      U(1)=0
      D(1)=0
      T(1)=0
      B(1)=0
      U(J+1)=U(J)+A(J)*COS(-1*J*G)
      D(J+1)=D(J)+P(J)*COS(-1*J*G)
      T(J+1)=T(J)+A(J)*SIN(-1*J*G)
      B(J+1)=B(J)+P(J)*SIN(-1*J*G)
      DEN(J)=SQRT((U(J+1)**2)+(T(J+1)**2))
      QNUM(J)=SQRT((D(J+1)**2)+(B(J+1)**2))
    2 CONTINUE
      X(I)=F
      Y(I)=20*ALOG10(QNUM(11)/DEN(11))
      F=F+0.005
      G=PI*F
      WRITE(6,101) I, X(I),Y(I)
101   FORMAT(6X,I4,4X,F5.2,4X,F10.2)
    1 CONTINUE
      NPTS=200
C       CALL PLOTS(BUF,20000,11,-1)
C       CALL PLOT(1.0,1.0,-3)
C       CALL SCALE(X,10.0,200,1)
C       CALL SCALE(Y,8.0,200,1)
C       CALL AXIS(0.0,0.0,'KHZ',-3,10.0,0.0,0.0,X(NPTS+1),X(NPTS+2))
C       CALL AXIS(0.0,0.0,'FREQ RESP _DB',14,8.0,90.0,Y(NPTS+1),
C      * Y(NPTS+2))
C       CALL PLOT(0.0,0.0,3)
C       CALL PLINE(X,Y,200,1,0,0)
C       CALL PLOT(0.0,0.0,999)
      STOP
      END
```

* Program# 1

Page 1. File is "filter2.text". Printed on May 30, 1984.

```
program filter2 (input,output);
(*****************************************
*                                        *
*  This program will convert a           *
*  set of coeficients of a direct        *
*  form digital filter a0...aM,b0...bM   *
*  to their equivalent coeficients of    *
*  the orthogonal structure R0...RM-1,   *
*  Wh0...WhM                             *
*                                        *
*****************************************)
const
 order_of_filter = 16;
type
 table = array [0..order_of_filter,0..order_of_filter] of real;
 filt = array [0..order_of_filter] of real;
var
 ord,l,m,i,r:integer;
 pi,e,vm,k,v:filt;
 p,a,b:table;
 d,g,n,large:real;
 data,prt:text;
begin
 rewrite (prt,'printer:');
 reset(data,'data5.text');
 ord:=order_of_filter;
 for m:= 0 to ord do
   readln(data,a[ord,m],p[ord,m]);
 for m:= ord downto 1 do
   begin
       k[m-1]:=a[m,m];
       v[m]:=p[m,m];
         for i:=0 to m do
          begin
            b[m,i]:=a[m,m-i];
            a[m-1,i]:=(a[m,i]-k[m-1]*b[m,i])/(1-sqr(k[m-1]));
            p[m-1,i]:=p[m,i]-b[m,i]*v[m];
          end;
   end;
 v[0]:=p[0,0];
 l:=0;
 d:=1.0;
 large:=abs(k[1]);
    for m:= 1 to (ord-1)do
      begin
```

```
        if abs(k[m]) > large then
          begin
            large:=abs(k[m]);
            l:=m;
          end;
      end;
    m:=1;
     while m <> 0 do
      begin
        m:=m-1;
        q:=(1+abs(k[m]))/(1-abs(k[m]));
         if q*d < 1 then
          begin
           d:=q*d;
            if k[m] < 0 then
              e[m]:= -1.0
            else
              e[m]:=1.0
            end
         else
          begin
            d:=d/q;
             if k[m] < 0 then
              e[m]:=1.0
             else
              e[m]:= -1.0;
          end;
      end;
    d:=1;
    m:=1;
    while m <> ord do
     begin
      q:=(1+abs(k[m]))/(1-abs(k[m]));
             if q*d < 1 then
               begin
                 d:=d*q;
                  if k[m] < 0 then
                     e[m]:= 1.0
                  else
                     e[m]:= -1.0;
               end
              else
               begin
                  d:=d/q;
                  if k[m] < 0 then
```

```
                    e[m]:=-1.0
                  else
                    e[m]:=1.0;
              end;
                m:=m+1;
      end;
      h:=1;
      pi[ord]:=1.0;
      m:=ord-1;
      while m <> -1 do
        begin
          pi[m]:= h*(1+e[m]*k[m]);
          vh[m]:=v[m]/pi[m];
          h:=pi[m];
          m:=m-1;
        end;
      k[ord]:=0;
      vh[ord]:=v[ord];
        for m:= 0 to ord do
           begin
             write(prt,'Wh(',m,')=',vh[m]);
             writeln(prt,'R(',m,')=',e[m]);
           end;
  end.
```

Page 1. File is "filter1.text". Printed on May 30, 1984.

```
program filter (input,output);
(*****************************************
*                                        *
* This program will transform a set of   *
* coeficients of an orthogonal digital   *
* filter R0 . . . RM-1,Wh0 . . . WhM to  *
* their equivalent direct form coeficients *
* a0 . . . aM,b0 . . . bM                *
*                                        *
*****************************************)
const
  order_of_filter = 10;
type
  filt = array[0..order_of_filter,0..order_of_filter] of real;
  table = array[0..order_of_filter] of real;
var
  data,prt:text;
  a,b:filt;
  pi,e,vh,k,v,p:table;
  ord,i,m,n,l:integer;
  d,q,h,large,sum:real;
begin
    rewrite(prt,'printer:');
    reset(data,'data6.text');
    ord:=order_of_filter;
     for m:=0 to ord do
       readln(data,vh[m],k[m]);
     close(data);
     l:=0;
     d:=1.0;
     large:=abs(k[1]);
     for m:= 1 to (ord-1)do
       begin
        if abs(k[m]) > large then
         begin
          large:=abs(k[m]);
          l:=m;
         end;
       end;
      m:=1;
       while m <> 0 do
         begin
           m:=m-1;
           q:=(1+abs(k[m]))/(1-abs(k[m]));
```

```
    if q*d < 1 then
     begin
       d:=q*d;
         if k[m] < 0 then
            e[m]:= -1.0
         else
            e[m]:=1.0
         end
    else
      begin
        d:=d/q;
          if k[m] < 0 then
            e[m]:=1.0
          else
            e[m]:= -1.0;
          end;
      end;
  d:=1;
  m:=1;
   while m <> ord do
    begin
      q:=(1+abs(k[m]))/(1-abs(k[m]));
        if q*d < 1 then
         begin
          d:=d*q;
           if k[m] < 0 then
             e[m]:= 1.0
           else
             e[m]:= -1.0;
         end
        else
         begin
          d:=d/q;
            if k[m] < 0 then
              e[m]:=-1.0
            else
              e[m]:=1.0;
         end;
       m:=m+1;
    end;
  reset(data,'data6.text');
   for m:=0 to ord do
     begin
       readln(data,vh[m],k[m]);
```

```
            writeln(prt,vh[m],k[m]);
          end;
        close(data);
        h:=1;
        pi[ord]:=1.0;
        m:=ord-1;
         while m <> -1 do
           begin
             pi[m]:= h*(1+e[m]*k[m]);
             v[m]:=vh[m]*pi[m];
             h:=pi[m];
             m:=m-1;
           end;
         v[ord]:=vh[ord];
         a[0,0]:=1.0;  b[0,0]:=1.0;
          for n:=1 to 10 do
           begin
            a[n,0]:=1.0;  b[n,n]:=1.0;
            b[n,0]:=k[n-1]; a[n,n]:=k[n-1];
            l:=n;
            if (n <>1) and (l <> 0) then
             begin
              for i:=(l-1) downto 1  do
               begin
                  b[n,i]:=a[n-1,i]*k[n-1]+b[n-1,i-1];
                  a[n,i]:=a[n-1,i]+k[n-1]*b[n-1,i-1];
               end;
          l:=0;
             end
            else
          end;
      for m:=0 to 10 do
        begin
          sum:=0.0;
           for n:=10 downto m do
              begin
               p[m]:=v[n]*b[n,m]+sum;
               sum:=p[m];
              end;
         writeln(prt,p[m],'       ',a[10,m]);
        end;
      end.
```

APPENDIX B

```
VALID layout processor 8.07 F Mon May  6 09:51:02 PDT 1985

Reading the process file drc.cmd                                    124
SCALD directory is /u0/fadi/fadi.wrk

Reading data for shiftregister, version 1
 input has        55 trapezoids
 getting subcell info (layout) from /u0/fadi/dmc
 input has         3 trapezoids
 getting subcell info (layout) from /u0/fadi/dep1
 input has         6 trapezoids
using bounding box

Doing Logical Operations for Hierarchy

Doing Logical Operations for Extract

Starting Extract
There are     57 continuity objects
              13 nodes
writing schematic
 Transistors of type nmos model=enh
 Transistors of type nmos model=dep1
 12 transistors, 55 trapezoids and 13 instances in this cell.
   12 transistors and 55 trapezoids total in this cell plus all subcells.

Starting DRC operations and tests
  computing donut
  updating /u0/fadi/shiftregister/abstract.1.1
  doing drc tests

done.  Spent     69.605 seconds   on shiftregister, version 1

   ******************     **********************
   *       0 Errors *     *       0 Edge Errors *
   ******************     **********************
Total time:      76.053 seconds      on shiftregister  version 1
%
```

* DRC OUTPUT

* CIF file listing

```
DS 1 250 400;
( DS is the definition start symbol where the first integer ' 1 '
in this case is the symbol number. The symbol number serves as
a name for the layout included between the DS command
and the DF ' definition finish ' symbol for reasons that
will be clear when we reach the call statement 'C'.
The third and forth intesers are used to scale the dimensions
listed after each box statement by multiplying each coordinate
by the ratio of 250/400 );
9 DMC;
(The 9 command is used to define the name of the subcell
used in main structure. DMC is ' 4 by 4 ' lambda diffusion
-metal via subcell);
L ND;
( L specifies the layer, N specifies the technology NMOS and
D defines the first letter of the layer name which is diffusion
in this case);
        B 1600 1600 4000 -800;
L NM;
        B 1600 1600 4000 -800;
L NC;
        B 800 800 4000 -800;
( B is the box statement, which defines the geometry of the
layer. Where the first number is layer length, the second number
is the layer width, the third and the forth numbers are the x and y
coordinates of the center of the layer.
Each of the coordinates is in centimicrons units, and
each of them will be multiplied by the ratio 250/400. For example
the actual values of the cut are:
          B 5.0 5.0 25 -5.0     in micrometer );
DF;
( DF is the definition finish ' i.e where the DMC cell layout ends');
DS 2 250 400;
9 DEP1;
L NP;
        B 2400 3392 5200 2704;
L ND;
        B 1600 1600 5200 4800;
        B 800 4000 5200 2000;
L NM;
        B 1600 2400 5200 4400;
L NI;
        B 2000 4592 5200 2704;
L NC;
        B 800 1600 5200 4400;
DF;
DS 3 250 400;
9 SHIFTREGISTER;
```

```
L NF;
        B 800 16800 3200 -800;
        B 800 16800 11200 -800;
        B 4400 800 -200 3200;
        B 1600 400 1200 2600;
        B 5200 800 15400 3200;
        B 1600 400 17200 2600;
        B 800 16800 19200 -800;
        B 5200 800 23400 3200;
        B 1600 400 25200 2600;
        B 800 16800 27200 -800;
        B 5200 800 7400 3200;
        B 1600 400 9200 2600;
L ND;
        B 1600 5600 25200 0;
        B 3600 800 26200 -3200;
        B 1600 6000 -1600 1800;
        B 1600 6000 22400 1800;
        B 1600 5600 17200 0;
        B 1600 800 20800 -1600;
        B 800 800 20400 -2400;
        B 4400 800 18600 -3200;
        B 1600 6000 14400 1800;
        B 1600 5600 9200 0;
        B 1600 800 12800 -1600;
        B 800 800 12400 -2400;
        B 4400 800 10600 -3200;
        B 800 800 14400 -7200;
        B 800 800 22400 -7200;
        B 800 800 6400 -7200;
        B 800 800 -1600 -7200;
        B 1600 6000 6400 1800;
        B 1600 5600 1200 0;
        B 1600 800 4800 -1600;
        B 800 800 4400 -2400;
        B 4400 800 2600 -3200;
L NA;
        B 1200 1200 30200 -3000;
        B 2400 1600 -2400 2800;
        B 1200 3200 -1800 400;
        B 7600 1600 27000 5600;
        B 7600 1600 27000 -8400;
        B 1200 1600 -3000 -8400;
        B 1200 1600 -3000 5600;
        B 1600 2400 17200 2400;
        B 1600 2400 25200 2400;
        B 6400 1600 18400 -8400;
        B 6400 1600 10400 5600;
        B 6400 1600 10400 -8400;
        B 6400 1600 18400 5600;
        B 6400 1600 2400 5600;
        B 6400 1600 2400 -8400;
        B 1600 2400 9200 2400;
        B 1600 2400 1200 2400;
L NC;
        B 800 1600 17200 2400;
        B 800 1600 25200 2400;
        B 800 1600 9200 2400;
        B 800 1600 1200 2400;
```

```
94 IN 30000 -3200;
(94 is a label construct that allows you to define the signal
names with their x and y coordinates.  The reason the label
PHI1 ends with an '!' is to avoid problems when creating
the sim format which will modify the label names found on more than
one node by extending the label name with a '#' and a number);
94 PHI1! 27200 7200;
94 PHI2! 19200 7200;
94 PHI1! 11200 7200;
94 PHI2! 3200 7200;
94 OUT -3200 2800;
94 OUT1 20000 -3200;
94 OUT2 12000 -3200;
94 OUT3 4000 -3200;
94 Vdd -3200 -8400;
94 GND -3200 5600;
C 2 T 1200 -6800;
(The C command is used to call a symbol which is
defined by the number that follows the C character.
T stands for translate, which means that the two integers
that follows will be added to the x and y center coordinates
of the called symbol);
C 2 T -6800 -6800;
C 2 T 17200 -6800;
C 2 T 9200 -6800;
C 1 T 2400 6400;
C 1 T -5600 6400;
C 1 T -5600 -7600;
C 1 T 2400 -7600;
C 1 T 18400 6400;
C 1 T 10400 6400;
C 1 T 10400 -7600;
C 1 T 18400 -7600;
C 1 T 24800 -2000;
DF;
C 3;
End
( Some of the statements were changed, so the CIF file will
be compatible with mextra. The statements that should be changed are:
The rotation commands and the prefix preceeding each signalname. A
shell script to make the necessary changes was written [14] and should
be applied to the CIF file while in VALID );
$
```

```
Window: -2250 19250 -5750 4750
the global label 'PHI1' has 2 occurrences
the global label 'PHI2' has 2 occurrences
        4 depletion
        8 enhancement
15 nodes
```

*log file listing

```
| units: 1      tech: nmos
e 25 GND OUT1 500 1000 13500 1750
e 23 GND OUT2 500 1000 8500 1750
e 21 GND OUT3 500 1000 3500 1750
e 19 GND OUT 500 1000 -1500 1750
e PHI1 IN 25 500 500 16750 -2250
e PHI2 OUT1 23 500 500 11750 -2250
e PHI1 OUT2 21 500 500 6750 -2250
e PHI2 OUT3 19 500 500 1750 -2250
d OUT1 OUT1 Vdd 1995 500 13750 -3620
d OUT2 OUT2 Vdd 1995 500 8750 -3620
d OUT3 OUT3 Vdd 1995 500 3750 -3620
d OUT OUT Vdd 1995 500 -1250 -3620
C Vdd GND 152
C OUT GND 67
C OUT3 GND 73
C OUT2 GND 73
C OUT1 GND 73
C 19 GND 61
C 21 GND 62
C 23 GND 62
C 25 GND 62
```

* .sim file listing

```
8 transistors, 13 nodes (4 pulled up)
sim>| " THIS BAR IS USED FOR COMMENTS"
sim>I
sim>| THE COMMAND " I " IS USED TO INITIALIZE
sim>| THE NODES VALUES
initialization took 16 steps
sim>w IN
sim>| w IS USED TO SPECIFY THE NODES TO BE
sim>| DISPLAYED AFTER EACH SIMULATION STEP
sim>W OUTA OUT1 OUT2 OUT3 OUT
sim>W PHI PHI1 PHI2
sim>| W IS USED TO WATCH A BIT VECTOR
sim>| (i.e. THE VALUES OF PHI1 AND PHI2
sim>| WILL BE DISPLAYED AS A BIT VECTOR PHI)
sim>1 IN PHI2
sim>| THE " 1 " COMMAND FORCES THE LISTED
sim>| NODES TO A LOGIC 0
sim>h PHI1
sim>| " h " FORCES PHI1 TO A LOGIC 1
sim>s
sim>| THE " s " COMMAND PROPOGATES THE INPUT
sim>| VALUES THROUGHOUT THE CIRCUIT, AND THE
sim>| THE WATCHED NODES AND BIT VECTORS
sim>| WILL BE DISPLAYED
step took 8 events
PHI=10  2
OUTA=1101       13
IN=0
sim>h PHI2
sim>l PHI1
sim>s
step took 7 events
PHI=01  1
OUTA=1001       9
IN=0
sim>h PHI1
sim>l PHI2
sim>s
step took 8 events
PHI=10  2
OUTA=1011       11
IN=0
sim>h PHI2
sim>l PHI1
sim>s
step took 8 events
PHI=01  1
OUTA=1010       10
IN=0
sim>| THE LEAST SIGNIFICANT bit OF OUTA IS
sim>| 0 WHICH IS THE VALUE OF THE PROPAGATED INPUT
sim>| SIGNAL AFTER TRIGGERING THE NONOVERLAPPING
sim>| CLOCK SIGNALS FOUR TIMES
$
```

* Simulation session " ESIM"

```
[0:00u 0:00s 11k] File read-in completed.
:inputs IN
[0:00u 0:00s 11k] inputs IN
:delay IN 0 5
[0:00u 0:00s 11k] delay IN 0 5
(9 stages examined.)
:worst OUT
[0:00u 0:00s 11k] worst OUT
Delay path to OUT (see gate at -6,-18):
    OUT (see gate at -6,-18) falls at 52.4ns through the following stage:
        ... through driver FET at (-8 9) to GND (see source at 68,9).
    19 (see gate at -8,9) rises at 51.8ns through pullup at (19 -18)
    when the following stage turns off:
        ... through FET at (9 -11) to OUT3 (see gate at 19,-18).
        ... through driver FET at (18 9) to GND (see source at 68,9).
    21 (see gate at 18,9) falls at 31.2ns through the following stage:
        ... through FET at (34 -11) to OUT2 (see gate at 44,-18).
        ... through driver FET at (43 9) to GND (see source at 68,9).
    23 (see gate at 43,9) rises at 27.2ns through pullup at (69 -18)
    when the following stage turns off:
        ... through FET at (59 -11) to OUT1 (see gate at 69,-18).
        ... through driver FET at (68 9) to GND (see source at 68,9).
    25 (see gate at 68,9) falls at 6.5ns through the following stage:
        ... through FET at (84 -11) to IN (see source at 84,-11).
:worst OUT1
[0:00u 0:00s 11k] worst OUT1
Delay path to OUT1 (see gate at 69,-18):
    OUT1 (see gate at 69,-18) rises at 13.3ns through pullup at (69 -18)
    when the following stage turns off:
        ... through driver FET at (68 9) to GND (see source at 68,9).
    25 (see gate at 68,9) falls at 6.5ns through the following stage:
        ... through FET at (84 -11) to IN (see source at 84,-11).
:worstdelay
[0:00u 0:00s 11k] worstdelay
Delay path to OUT (see gate at -6,-18):
    OUT (see gate at -6,-18) falls at 52.4ns through the following stage:
        ... through driver FET at (-8 9) to GND (see source at 68,9).
    19 (see gate at -8,9) rises at 51.8ns through pullup at (19 -18)
    when the following stage turns off:
        ... through FET at (9 -11) to OUT3 (see gate at 19,-18).
        ... through driver FET at (18 9) to GND (see source at 68,9).
    21 (see gate at 18,9) falls at 31.2ns through the following stage:
        ... through FET at (34 -11) to OUT2 (see gate at 44,-18).
        ... through driver FET at (43 9) to GND (see source at 68,9).
    23 (see gate at 43,9) rises at 27.2ns through pullup at (69 -18)
    when the following stage turns off:
        ... through FET at (59 -11) to OUT1 (see gate at 69,-18).
        ... through driver FET at (68 9) to GND (see source at 68,9).
    25 (see gate at 68,9) falls at 6.5ns through the following stage:
        ... through FET at (84 -11) to IN (see source at 84,-11).
hit ctrl D in order to exit
[0:00u 0:00s 11k] Crystal done.
%
```

*Simulation session " Crystal"

```
!Filename: shregcmd.sim
! This is a command file that will produce the
! same logic simulation results obtained when
! entering the commands interactively.
! The command syntax to enter the command file is:
! esim shreg.sim shregcmd.sim
I
w IN
W OUTA OUT1 OUT2 OUT3 OUT
W PHI PHI1 PHI2
l IN PHI2
h PHI1
s
h PHI2
l PHI1
s
h PHI1
l PHI2
s
h PHI2
l PHI1
s

$
```

* Input cmmands file for " Esim "

(LOGIC SIMULATOR)

```
/* Filename: shregin.crys
This is an input command file to the timing
simulator crystal, it will produce the same results
obtained when the commands are entered interactively.
The command syntax to enter the command file is:
crystal shreg.sim < shregin.crys > shregout.crys
where shregout.crys contains the results of the
simulation */
inputs IN
delay IN 0 5
worst OUT
worst OUT1
worstdelay
$
```

*Input commnds file for "Crystal"

Timing simulator

```
/* Filename: shreg.powdiss
THIS FILE IS GENERATED BY USING THE COMMAND
powest -p < shreg.sim > shreg.powdiss
THE PROGRAM PRODUCE THE RESULTS BASED UPON
THE DEFAULT PROCESS PARAMETERS LISTED BELOW
Gamma=0.4V**.5, tox=9e-08m, u0=0.08m**2/V-s
vdd=5V, vtd=-3.5V, vte=0.8V, vsb=2V
#devs   Pdc_avg (W)      Pdc_max (W)      type

0       0.000000         0.000000         enhancement pullups
4       0.000648         0.000942         depletion pullups
0       0.000000         0.000000         special depletion pullups

4       0.000648         0.000942         TOTAL
$
```

* Powest output using default parameters.

```
/* Filename : shreg.powdiss   */
/* This file contains estimates of DC power
required by the NMOS circuit ' shreg1'.
This will be obtained by running powest on
' shreg.sim ' .
' #devs ' is the number of pullup type transistors.
' Pdc_avg & Pdc_max ' are the average and maximum
power dissipated by the circuit respectively.
The arguments given with powest are the circuit
parameters:
'Gamma=0.0 V**.5 ' Bulk Threshold Parameter .
'tox = 1.0e-7 m ' Oxide Thickness .
'u0=0.06 m**2/V-s' Electron Surface Mobility.
'vsb=0.0 V ' Source To Substrate voltage .
'vtd=-4.0 V ' Depletion Threshold Voltage .
'vte=1.0 V ' Enhancement Threshold Voltage.
'vdd=5.0 V ' Power Supply Voltage .
The command format is as follow :
powest -p Gamma=0.0 tox=1.0e-7 u0=0.06 vtd=-4.0 vte=1.0
vsb=0.0 < shreg.sim > shreg.powdiss
**********************************************************
**********************************************************
Gamma=0V**.5, tox=1e-07m, u0=0.06m**2/V-s
vdd=5V, vtd=-4V, vte=1V, vsb=0V
#devs   Pdc_avg (W)      Pdc_max (W)      type

0       0.000000         0.000000         enhancement pullups
4       0.000571         0.000830         depletion pullups
0       0.000000         0.000000         special depletion pullups

4       0.000571         0.000830         TOTAL
```

*Powest output based on user parameters.

```
* Filename: shreg.hdr
* ALL NODE NAMES LISTED BELOW WILL
* BE TRANSLATED TO NODE NUMBERS WHEN RUNNING
* PSPICE WHICH USES A PROGRAM CALLED SPCPP ;
* FOR THE TRANSLATION AND PREPARE AN INPUT DECK
* FILE TO SPICE2G6.
VDD <Vdd> 0 DC 5
* SET NODE Vdd HIGH
VPHI1 <PHI1> 0 PULSE(0 5 0NS 0NS 0NS 20NS 40NS)
VPHI2 <PHI2> 0 PULSE(0 5 20NS 0NS 0NS 20NS 40NS)
* THE ABOVE ARE TWO NONOVERLAPPING CLOCK
* SIGNALS
VIN <IN> 0 DC 0
* SET INPUT VOLTAGE IS LOW
.MODEL ENMOS NMOS (VTO=1.0)
.MODEL DNMOS NMOS (VTO=-4.0)
* THE ABOVE ARE MODEL NAMES AND THRESHOLD VOLTAGES
* FOR EACH TYPE OF TRANSISTORS
* DNMOS IS A DEPLETION TYPE TRANSISTOR MODEL
* ENMOS IS A ENHANCEMENT TYPE TRANSISTOR MODEL
.TRANS 1NS 220NS
* SIMULATING FROM 1NS TO 220NS
.PLOT TRANS V(<OUT>) V(<PHI1>) V(<PHI2>)
* NODES TO BE PLOTTED
.OPTIONS LIMPTS=1000 ITL5=8000
* THE ABOVE CARD SHOULD BE INCLUDED IF
* THE DEFAULT SIMULATION AND ITERATION
* POINTS ARE EXCEEDED
.END
‡
```

Header file for SPICE2G6.

contains spice models, input voltages, control cards,etc.

*(Created by the user)

```
*** SPICE DECK created from shreg.sim: tech=nmos
M1 5 4 0 3 ENMOS L=5.0U W=10.0U
M2 7 6 0 3 ENMOS L=5.0U W=10.0U
M3 9 8 0 3 ENMOS L=5.0U W=10.0U
M4 11 10 0 3 ENMOS L=5.0U W=10.0U
M5 4 13 12 3 ENMOS L=5.0U W=5.0U
M6 6 14 5 3 ENMOS L=5.0U W=5.0U
M7 8 13 7 3 ENMOS L=5.0U W=5.0U
M8 10 14 9 3 ENMOS L=5.0U W=5.0U
M9 1 5 5 15 DNMOS L=20.0U W=5.0U
M10 1 7 7 15 DNMOS L=20.0U W=5.0U
M11 1 9 9 15 DNMOS L=20.0U W=5.0U
M12 1 11 11 15 DNMOS L=20.0U W=5.0U
C13 1 0 152.0F
C14 11 0 67.0F
C15 9 0 73.0F
C16 7 0 73.0F
C17 5 0 73.0F
C18 10 0 61.0F
C19 8 0 62.0F
C20 6 0 62.0F
C21 4 0 62.0F
*  GND  0
*  Vdd  1
*  BULK 2
*  ENMOS        3
*  25   4
*  OUT1 5
*  23   6
*  OUT2 7
*  21   8
*  OUT3 9
*  19   10
*  OUT  11
*  IN   12
*  PHI1 13
*  PHI2 14
*  DNMOS        15
*VDD <Vdd> 0 DC 5
 VDD 1     0 DC 5
*VPHI1 <PHI1> 0 PULSE(0 5 0NS 0NS 0NS 20NS 40NS)
 VPHI1 13     0 PULSE(0 5 0NS 0NS 0NS 20NS 40NS)
*VPHI2 <PHI2> 0 PULSE(0 5 20NS 0NS 0NS 20NS 40NS)
 VPHI2 14     0 PULSE(0 5 20NS 0NS 0NS 20NS 40NS)
*VIN <IN> 0 DC 0
 VIN 12   0 DC 0
 .MODEL ENMOS NMOS (VTO=1.0)
 .MODEL DNMOS NMOS (VTO=-4.0)
 .TRANS 1NS 220NS
*.PLOT TRANS V(<OUT>) V(<PHI1>) V(<PHI2>)
 .PLOT TRANS V(11   ) V(13    ) V(14    )
 .OPTIONS LIMPTS=1000 ITL5=9000
 .END
$
```

*SPICE DECK input file for spice2g6

This file is prepared by "pspice"

based on the header file.

* SPICE2G6 OUTPUT LISTING

```
cat shreg.plot
1*******12/31/85 ********  SPICE 2G.6    3/15/83 ********14:37:03*****

0*** SPICE DECK CREATED FROM SHREG.SIM, TECH=NMOS

0****     INPUT LISTING                    TEMPERATURE =   27.000 DEG C

0*************************************************************************


 M1 5 4 0 3 ENMOS L=5.0U W=10.0U
 M2 7 6 0 3 ENMOS L=5.0U W=10.0U
 M3 9 8 0 3 ENMOS L=5.0U W=10.0U
 M4 11 10 0 3 ENMOS L=5.0U W=10.0U
 M5 4 13 12 3 ENMOS L=5.0U W=5.0U
 M6 6 14 5 3 ENMOS L=5.0U W=5.0U
 M7 8 13 7 3 ENMOS L=5.0U W=5.0U
 M8 10 14 9 3 ENMOS L=5.0U W=5.0U
 M9 1 5 5 15 DNMOS L=20.0U W=5.0U
 M10 1 7 7 15 DNMOS L=20.0U W=5.0U
 M11 1 9 9 15 DNMOS L=20.0U W=5.0U
 M12 1 11 11 15 DNMOS L=20.0U W=5.0U
 C13 1 0 152.0F
 C14 11 0 67.0F
 C15 9 0 73.0F
 C16 7 0 73.0F
 C17 5 0 73.0F
 C18 10 0 61.0F
 C19 8 0 62.0F
 C20 6 0 62.0F
 C21 4 0 62.0F
 *  GND 0
 *  VDD 1
 *  BULK       2
 *  ENMOS      3
 *  25  4
 *  OUT1       5
 *  23  6
 *  OUT2       7
 *  21  8
 *  OUT3       9
 *  19  10
 *  OUT 11
 *  IN  12
 *  PHI1       13
 *  PHI2       14
 *  DNMOS      15
 *VDD <VDD> 0 DC 5
  VDD 1     0 DC 5
 *VPHI1 <PHI1> 0 PULSE(0 5 0NS 0NS 0NS 20NS 40NS)
  VPHI1 13     0 PULSE(0 5 0NS 0NS 0NS 20NS 40NS)
 *VPHI2 <PHI2> 0 PULSE(0 5 20NS 0NS 0NS 20NS 40NS)
  VPHI2 14     0 PULSE(0 5 20NS 0NS 0NS 20NS 40NS)
 *VIN <IN> 0 DC 0
  VIN 12   0 DC 0
  .MODEL ENMOS NMOS (VTO=1.0)
  .MODEL DNMOS NMOS (VTO=-4.0)
  .TRANS 1NS 220NS
 *.PLOT TRANS V(<OUT>) V(<PHI1>) V(<PHI2>)
  .PLOT TRANS V(11    ) V(13    ) V(14    )
  .OPTIONS LIMPTS=1000 ITL5=8000
  .END
1*******12/31/85 ********  SPICE 2G.6    3/15/83 ********14:37:03*****

0*** SPICE DECK CREATED FROM SHREG.SIM, TECH=NMOS
0****              MOSFET MODEL PARAMETERS                TEMPERATURE =   27.000 DEG

0*************************************************************************
```

```
                ENHOS       DNHOS
.OTYPE          NHOS        NHOS
.OLEVEL          1.000       1.000
.OVTO            1.000      -4.000
.OKP          2.00d-05    2.00d-05
1*********************12/31/85 ********************************  SPICE 2B.6    3/15/83 ***************************14:37:03*****************

0### SPICE DECK CREATED FROM SHREG.SIM, TECH=NMOS
0####                  INITIAL TRANSIENT SOLUTION                        TEMPERATURE =   27.000 DEG C

0*************************************************************************************************************************************

  NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE

(  1)    5.0000    (  3)    0.1765    (  4)    0.1765    (  5)    5.0000    (  6)    0.1765    (  7)    5.0000    (  8)    0.1765

(  9)    5.0000    ( 10)    0.1765    ( 11)    5.0000    ( 12)    0.        ( 13)    0.        ( 14)    0.        ( 15)    5.0000


    VOLTAGE SOURCE CURRENTS

    NAME        CURRENT

    VDD       -4.682d-11

    VFHI1      0.    d+00

    VFHI2      0.    d+00

    VIN        9.364d-12

    TOTAL POWER DISSIPATION   2.34d-10  WATTS
1*********************12/31/85 ********************************  SPICE 2B.6    3/15/83 ***************************14:37:03*****************

0### SPICE DECK CREATED FROM SHREG.SIM, TECH=NMOS
0####                  OPERATING POINT INFORMATION                       TEMPERATURE =   27.000 DEG C

0*************************************************************************************************************************************

0
0#### MOSFETS

0              M1          M2          M3          M4          M5          M6          M7          M8          M9          M10         M11         M12
.MODEL      ENHOS       ENHOS       ENHOS       ENHOS       ENHOS       ENHOS       ENHOS       ENHOS       DNHOS       DNHOS       DNHOS       DNHOS
 ID       1.86e-12    1.86e-12    1.86e-12    1.86e-12 -1.67e-30  0.  e+00  0.  e+00  0.  e+00  1.34e-11    1.34e-11    1.34e-11    4.69e-12
 VGS         0.176       0.176       0.176       0.176       0.          -5.000      -5.000      -5.000      0.          0.          0.          0.
 VDS         5.000       5.000       5.000       5.000       0.176       -4.824      -4.824      -4.824      0.000       0.000       0.000       0.000
 VBS         0.176       0.176       0.176       0.176       0.176       -4.824      -4.824      -4.824      0.001       0.001       0.001       0.001
1*********************12/31/85 ********************************  SPICE 2B.6    3/15/83 ***************************14:37:03*****************

0### SPICE DECK CREATED FROM SHREG.SIM, TECH=NMOS
0####                  TRANSIENT ANALYSIS                                TEMPERATURE =   27.000 DEG C

0*************************************************************************************************************************************

0LEGEND:
```

12/31/85 SPICE 2G.6 3/15/83 14:37:03.

**** SPICE DECK CREATED FROM SHREG.SIn, TECH=nnOS
**** TRANSIENT ANALYSIS TEMPERATURE = 27.000 DEG C

LEGEND:

*: V(11)
+: V(13)
=: V(14)
X

TIME	V(11)	(*+=)---------- 0. d+00 ---------- 2.000d+00 ---------- 4.000d+00 ---------- 6.000d+00
0. d+00	5.000d+00	X
1.000d-09	5.000d+00	=
2.000d-09	5.000d+00	=
3.000d-09	5.000d+00	=
4.000d-09	5.000d+00	=
5.000d-09	5.000d+00	=
6.000d-09	5.000d+00	=
7.000d-09	5.000d+00	=
8.000d-09	5.000d+00	=
9.000d-09	5.000d+00	=
1.000d-08	5.000d+00	=
1.100d-08	5.000d+00	=
1.200d-08	5.000d+00	=
1.300d-08	5.000d+00	=
1.400d-08	5.000d+00	=
1.500d-08	5.000d+00	=
1.600d-08	5.000d+00	=
1.700d-08	5.000d+00	=
1.800d-08	5.000d+00	=
1.900d-08	5.000d+00	=
2.000d-08	5.000d+00	=
2.100d-08	5.000d+00	.
2.200d-08	5.000d+00	+
2.300d-08	5.000d+00	+
2.400d-08	5.000d+00	+
2.500d-08	5.000d+00	+
2.600d-08	5.000d+00	+
2.700d-08	5.000d+00	+
2.800d-08	5.000d+00	+
2.900d-08	5.000d+00	+
3.000d-08	5.000d+00	+
3.100d-08	5.000d+00	+
3.200d-08	5.000d+00	+
3.300d-08	5.000d+00	+
3.400d-08	5.000d+00	+
3.500d-08	5.000d+00	+
3.600d-08	5.000d+00	+
3.700d-08	5.000d+00	+
3.800d-08	5.000d+00	+
3.900d-08	5.000d+00	+
4.000d-08	5.000d+00	+
4.100d-08	5.000d+00	.
4.200d-08	5.000d+00	=
4.300d-08	5.000d+00	=
4.400d-08	5.000d+00	=
4.500d-08	5.000d+00	=
4.600d-08	5.000d+00	=
4.700d-08	5.000d+00	=
4.800d-08	5.000d+00	=
4.900d-08	5.000d+00	=
5.000d-08	5.000d+00	=
5.100d-08	5.000d+00	=
5.200d-08	5.000d+00	=
5.300d-08	5.000d+00	=
5.400d-08	5.000d+00	=
5.500d-08	5.000d+00	=
5.600d-08	5.000d+00	=
5.700d-08	5.000d+00	=
5.800d-08	5.000d+00	=
5.900d-08	5.000d+00	=
6.000d-08	5.000d+00	=
6.100d-08	5.000d+00	.
6.200d-08	4.910d+00	+
6.300d-08	4.527d+00	+
6.400d-08	3.946d+00	+
6.500d-08	3.203d+00	+
6.600d-08	2.452d+00	+
6.700d-08	1.621d+00	+
6.800d-08	1.003d+00	+
6.900d-08	6.538d-01	+
7.000d-08	5.209d-01	+
7.100d-08	4.648d-01	+
7.200d-08	4.433d-01	+
7.300d-08	4.316d-01	+
7.400d-08	4.245d-01	+
7.500d-08	4.188d-01	+
7.600d-08	4.142d-01	+
7.700d-08	4.095d-01	+
7.800d-08	4.062d-01	+
7.900d-08	4.031d-01	+
8.000d-08	4.000d-01	+

8.000d-08	4.000d-01	+	x
8.100d-08	3.975d-01	.	x
8.200d-08	3.963d-01	=	x
8.300d-08	3.960d-01	=	x
8.400d-08	3.959d-01	=	x
8.500d-08	3.959d-01	=	x
8.600d-08	3.959d-01	=	x
8.700d-08	3.959d-01	=	x
8.800d-08	3.959d-01	=	x
8.900d-08	3.959d-01	=	x
9.000d-08	3.959d-01	=	x
9.100d-08	3.959d-01	=	x
9.200d-08	3.959d-01	=	x
9.300d-08	3.959d-01	=	x
9.400d-08	3.959d-01	=	x
9.500d-08	3.959d-01	=	x
9.600d-08	3.959d-01	=	x
9.700d-08	3.959d-01	=	x
9.800d-08	3.959d-01	=	x
9.900d-08	3.959d-01	=	x
1.000d-07	3.959d-01	=	x
1.010d-07	3.958d-01	.	x
1.020d-07	3.947d-01	+	x
1.030d-07	3.930d-01	+	x
1.040d-07	3.913d-01	+	x
1.050d-07	3.897d-01	+	x
1.060d-07	3.882d-01	+	x
1.070d-07	3.870d-01	+	x
1.080d-07	3.858d-01	+	x
1.090d-07	3.845d-01	+	x
1.100d-07	3.833d-01	+	x
1.110d-07	3.824d-01	+	x
1.120d-07	3.815d-01	+	x
1.130d-07	3.806d-01	+	x
1.140d-07	3.797d-01	+	x
1.150d-07	3.789d-01	+	x
1.160d-07	3.782d-01	+	x
1.170d-07	3.775d-01	+	x
1.180d-07	3.769d-01	+	x
1.190d-07	3.762d-01	+	x
1.200d-07	3.756d-01	+	x
1.210d-07	3.751d-01	.	x
1.220d-07	3.748d-01	=	x
1.230d-07	3.747d-01	=	x
1.240d-07	3.747d-01	=	x
1.250d-07	3.747d-01	=	x
1.260d-07	3.747d-01	=	x
1.270d-07	3.747d-01	=	x
1.280d-07	3.747d-01	=	x
1.290d-07	3.747d-01	=	x
1.300d-07	3.747d-01	=	x
1.310d-07	3.747d-01	=	x
1.320d-07	3.747d-01	=	x
1.330d-07	3.747d-01	=	x
1.340d-07	3.747d-01	=	x
1.350d-07	3.747d-01	=	x
1.360d-07	3.747d-01	=	x
1.370d-07	3.747d-01	=	x
1.380d-07	3.747d-01	=	x
1.390d-07	3.747d-01	=	x
1.400d-07	3.747d-01	=	x
1.410d-07	3.747d-01	.	x
1.420d-07	3.744d-01	+	x
1.430d-07	3.740d-01	+	x
1.440d-07	3.735d-01	+	x
1.450d-07	3.731d-01	+	x
1.460d-07	3.727d-01	+	x
1.470d-07	3.723d-01	+	x
1.480d-07	3.719d-01	+	x
1.490d-07	3.715d-01	+	x
1.500d-07	3.711d-01	+	x
1.510d-07	3.708d-01	+	x
1.520d-07	3.705d-01	+	x
1.530d-07	3.702d-01	+	x
1.540d-07	3.698d-01	+	x
1.550d-07	3.695d-01	+	x
1.560d-07	3.693d-01	+	x
1.570d-07	3.690d-01	+	x
1.580d-07	3.687d-01	+	x
1.590d-07	3.685d-01	+	x
1.600d-07	3.682d-01	+	x
1.610d-07	3.680d-01	.	x
1.620d-07	3.679d-01	=	x
1.630d-07	3.678d-01	=	x
1.640d-07	3.678d-01	=	x
1.650d-07	3.678d-01	=	x
1.660d-07	3.678d-01	=	x
1.670d-07	3.678d-01	=	x
1.680d-07	3.678d-01	=	x
1.690d-07	3.678d-01	=	x

1.660d-07	3.678d-01	=	*
1.670d-07	3.678d-01	=	*
1.680d-07	3.678d-01	=	*
1.690d-07	3.678d-01	=	*
1.700d-07	3.678d-01	=	*
1.710d-07	3.678d-01	=	*
1.720d-07	3.678d-01	=	*
1.730d-07	3.678d-01	=	*
1.740d-07	3.678d-01	=	*
1.750d-07	3.678d-01	=	*
1.760d-07	3.678d-01	=	*
1.770d-07	3.678d-01	=	*
1.780d-07	3.678d-01	=	*
1.790d-07	3.678d-01	=	*
1.800d-07	3.678d-01	=	*
1.810d-07	3.678d-01	.	*
1.820d-07	3.677d-01	+	*
1.830d-07	3.675d-01	+	*
1.840d-07	3.673d-01	+	*
1.850d-07	3.671d-01	+	*
1.860d-07	3.669d-01	+	*
1.870d-07	3.667d-01	+	*
1.880d-07	3.665d-01	+	*
1.890d-07	3.663d-01	+	*
1.900d-07	3.662d-01	+	*
1.910d-07	3.660d-01	+	*
1.920d-07	3.658d-01	+	*
1.930d-07	3.657d-01	+	*
1.940d-07	3.655d-01	+	*
1.950d-07	3.653d-01	+	*
1.960d-07	3.652d-01	+	*
1.970d-07	3.651d-01	+	*
1.980d-07	3.649d-01	+	*
1.990d-07	3.648d-01	+	*
2.000d-07	3.646d-01	+	*
2.010d-07	3.645d-01	.	*
2.020d-07	3.644d-01	=	*
2.030d-07	3.644d-01	=	*
2.040d-07	3.644d-01	=	*
2.050d-07	3.644d-01	=	*
2.060d-07	3.644d-01	=	*
2.070d-07	3.644d-01	=	*
2.080d-07	3.644d-01	=	*
2.090d-07	3.644d-01	=	*
2.100d-07	3.644d-01	=	*
2.110d-07	3.644d-01	=	*
2.120d-07	3.644d-01	=	*
2.130d-07	3.644d-01	=	*
2.140d-07	3.644d-01	=	*
2.150d-07	3.644d-01	=	*
2.160d-07	3.644d-01	=	*
2.170d-07	3.644d-01	=	*
2.180d-07	3.644d-01	=	*
2.190d-07	3.644d-01	=	*
2.200d-07	3.644d-01	=	*

JOB CONCLUDED
TOTAL JOB TIME 139.22

APPENDIX C

* POWEST OUTPUT


```
/* Filename : g9.powdiss   */
/* This file contains estimates of DC power
required by the NMOS circuit " g9 "   .
This will be obtained by running powest on
" g9.sim " .
" #devs " is the number of pullup type transistors.
" Pdc_avg & Pdc_max " are the average and maximum
power dissipated by the circuit respectively.
The arguments given with powest are the circuit
parameters:
"gamma=0.0 V**.5 " Bulk Threshold Parameter .
"tox = 1.0e-7 m " Oxide Thickness .
"u0=0.06 m**2/V-s" Electron Surface Mobility.
"vsb=0.0 V " Source To Substrate voltage .
"vtd=-4.0 V " Depletion Threshold Voltage .
"vte=1.0 V " Enhancement Threshold Voltage.
"vdd=5.0 V " Power Supply Voltage .
The command format is as follow :
powest -p Gamma=0.0 tox=1.0e-7 u0=0.06 vtd=-4.0 vte=1.0 vsb=0.0 < g9.sim
powest resides in /usr/cad/bin directory .
**************************************
**************************************
gamma=0V**.5, tox=1e-07m, u0=0.06m**2/V-s
vdd=5V, vtd=-4V, vte=1V, vsb=0V
#devs    Pdc_avg (W)      Pdc_max (W)      type

0        0.000000         0.000000         enhancement pullups
390      0.052985         0.080967         depletion pullups
0        0.000000         0.000000         special depletion pullups

430      0.059168         0.089272         TOTAL
$
```

```
/* Filename : s7.powdiss   */
/* This file contains estimates of DC power
required by the NMOS circuit ' s7 '   .
This will be obtained by running powest on
' s7.sim ' .
' $devs ' is the number of pullup type transistors.
' Pdc_avg & Pdc_max ' are the average and maximum
power dissipated by the circuit respectively.
The arguments given with powest are the circuit
parameters:
'gamma=0.37 V**.5 ' Bulk Threshold Parameter .
'tox = 1.0e-7 m ' Oxide Thickness .
'u0=0.06 m**2/V-s' Electron Surface Mobility.
'vsb=0.0 V ' Source To Substrate voltage .
'vtd=-4.0 V ' Depletion Threshold Voltage .
'vte=1.0 V ' Enhancement Threshold Voltage.
'vdd=5.0 V ' Power Supply Voltage .
The command format is as follow :
powest -p Gamma=0.0 tox=1.0e-7 u0=0.06 vtd=-4.0 vte=1.0 vsb=0.0 < s7.sim
powest resides in /usr/cad/bin directory .
*****************************************
*****************************************
gamma=0V**.5, tox=1e-07m, u0=0.06m**2/V-s
vdd=5V, vtd=-4V, vte=1V, vsb=0V
$devs   Pdc_avg (W)     Pdc_max (W)     type

0       0.000000        0.000000        enhancement pullups
510     0.069289        0.105881        depletion pullups
0       0.000000        0.000000        special depletion pullups

510     0.069289        0.105881        TOTAL
$
```

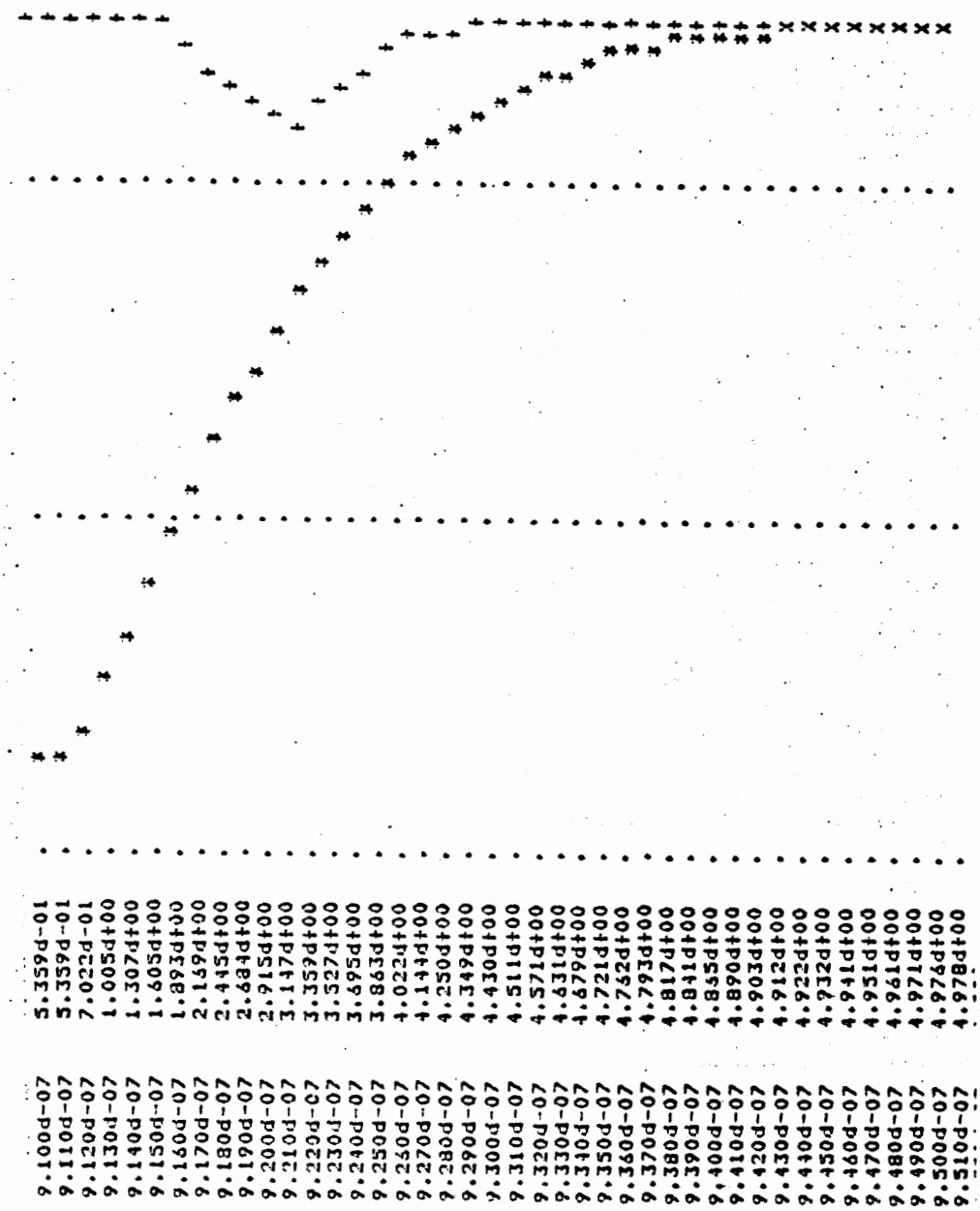
```
/* Filename : s5.powdiss   */
/* This file contains estimates of DC power
required by the NMOS circuit ' s5 ' .
This will be obtained by running powest on
' s5.sim ' .
' #devs ' is the number of pullup type transistors.
' Pdc_avg & Pdc_max ' are the average and maximum
power dissipated by the circuit respectively.
The arguments given with powest are the circuit
parameters:
'Gamma=0.0 V**.5 ' Bulk Threshold Parameter .
'tox = 1.0e-7 m ' Oxide Thickness .
'u0=0.06 m**2/V-s' Electron Surface Mobility.
'vsb=0.0 V ' Source To Substrate voltage .
'vtd=-4.0 V ' Depletion Threshold Voltage .
'vte=1.0 V ' Enhancement Threshold Voltage.
'vdd=5.0 V ' Power Supply Voltage .
The command format is as follow :
powest -p Gamma=0.0 tox=1.0e-7 u0=0.06 vtd=-4.0 vte=1.0 vsb=0.0 < s5.sim
powest resides in /usr/cad/bin directory .
*****************************************
*****************************************
Gamma=0V**.5, tox=1e-07m, u0=0.06m**2/V-s
vdd=5V, vtd=-4V, vte=1V, vsb=0V
#devs   Pdc_avg (W)     Pdc_max (W)     type

0       0.000000        0.000000        enhancement pullups
510     0.069289        0.105881        depletion pullups
0       0.000000        0.000000        special depletion pullups

510     0.069289        0.105881        TOTAL
```

```
/* Filename :s3.powdiss   */
/* This file contains estimates of DC power
required by the NMOS circuit ' s3 '   .
This will be obtained by running powest on
' s3.sim ' .
' #devs ' is the number of pullup type transistors.
' Pdc_avg & Pdc_max ' are the average and maximum
power dissipated by the circuit respectively.
The arguments given with powest are the circuit
parameters:
'gamma=0.0 V**.5 ' Bulk Threshold Parameter .
'tox = 1.0e-7 m ' Oxide Thickness .
'u0=0.06 m**2/V-s' Electron Surface Mobility.
'vsb=0.0 V ' Source To Substrate voltage .
'vtd=-4.0 V ' Depletion Threshold Voltage .
'vte=1.0 V ' Enhancement Threshold Voltage.
'vdd=5.0 V ' Power Supply Voltage .
The command format is as follow :
powest -p Gamma=0.0 tox=1.0e-7 u0=0.06 vtd=-4.0 vte=1.0 vsb=0.0 < s3.sim
powest resides in /usr/cad/bin directory .
*****************************************
*****************************************
gamma=0V**.5, tox=1e-07m, u0=0.06m**2/V-s
vdd=5V, vtd=-4V, vte=1V, vsb=0V
#devs   Pdc_avg (W)      Pdc_max (W)      type

0       0.000000         0.000000         enhancement pullups
470     0.065137         0.097577         depletion pullups
0       0.000000         0.000000         special depletion pullups

470     0.065137         0.097577         TOTAL
$
```

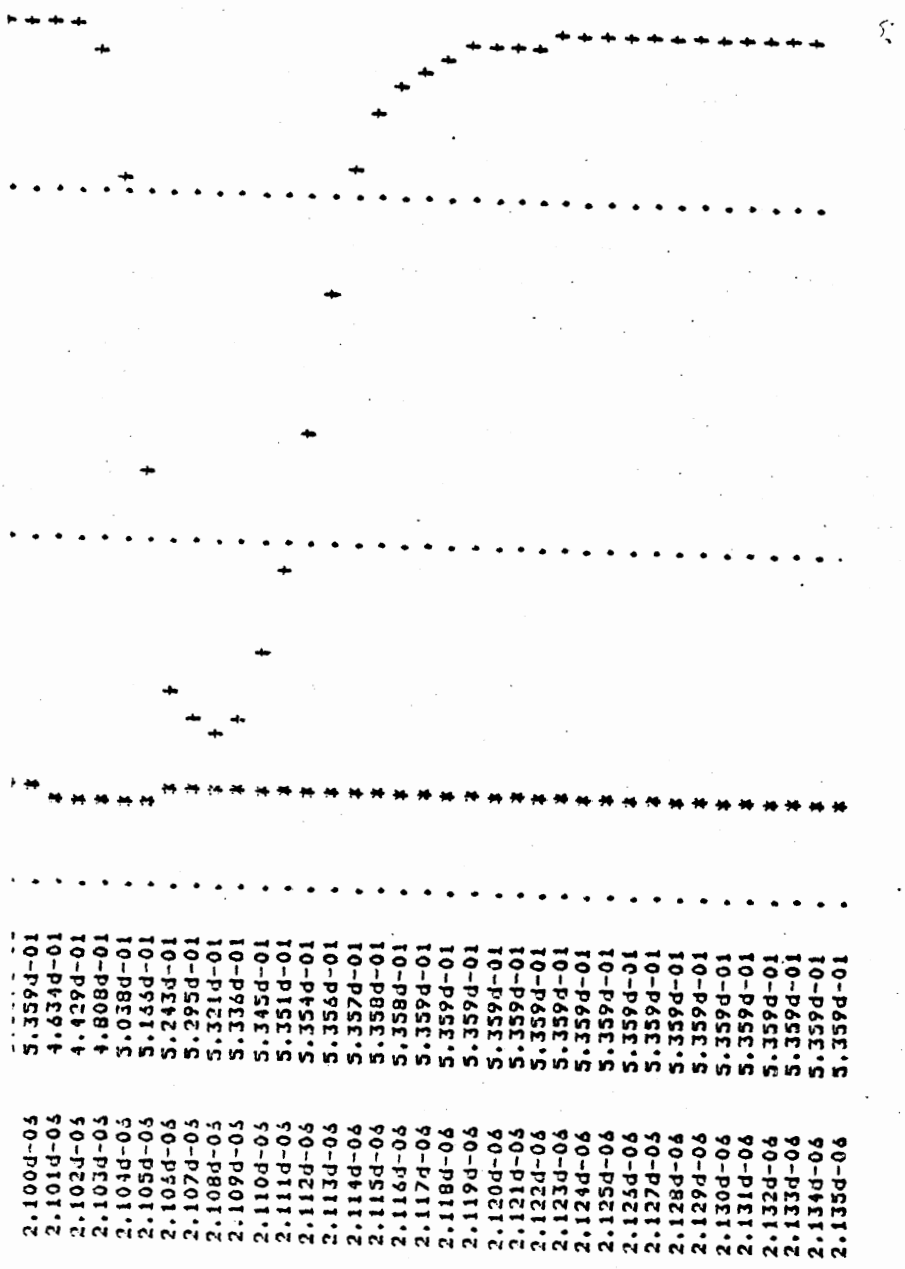
```
/* Filename : s1.powdiss   */
/* This file contains estimates of DC power
required by the NMOS circuit ' s1 '    .
This will be obtained by running powest on
' s1.sim ' .
' #devs ' is the number of pullup type transistors.
' Pdc_avg & Pdc_max ' are the average and maximum
power dissipated by the circuit respectively.
The arguments given with powest are the circuit
parameters:
'gamma=0.0 V**.5 ' Bulk Threshold Parameter .
'tox = 1.0e-7 m ' Oxide Thickness .
'u0=0.06 m**2/V-s' Electron Surface Mobility.
'vsb=0.0 V ' Source To Substrate voltage .
'vtd=-4.0 V ' Depletion Threshold Voltage .
'vte=1.0 V ' Enhancement Threshold Voltage.
'vdd=5.0 V ' Power Supply Voltage .
The command format is as follow :
powest -p Gamma=0.0 tox=1.0e-7 u0=0.06 vtd=-4.0 vte=1.0 vsb=0.0 < s1.sim
powest resides in /usr/cad/bin directory .
**********************************************
**********************************************
gamma=0V**.5, tox=1e-07m, u0=0.06m**2/V-s
vdd=5V, vtd=-4V, vte=1V, vsb=0V
#devs    Pdc_avg (W)      Pdc_max (W)      type

0        0.000000         0.000000         enhancement pullups
630      0.087196         0.130794         depletion pullups
0        0.000000         0.000000         special depletion pullups

630      0.087196         0.130794         TOTAL
$
```

TIMING ANALYSIS USING SPICE2G6

*(output at node b at t = 910ns)



*(Output at node COUT at t =2100)



2.100d-05	5.359d-01
2.101d-05	4.634d-01
2.102d-05	4.429d-01
2.103d-05	4.808d-01
2.104d-05	5.038d-01
2.105d-05	5.183d-01
2.106d-05	5.243d-01
2.107d-05	5.295d-01
2.108d-05	5.321d-01
2.109d-05	5.336d-01
2.110d-05	5.345d-01
2.111d-05	5.351d-01
2.112d-05	5.354d-01
2.113d-05	5.356d-01
2.114d-05	5.357d-01
2.115d-05	5.358d-01
2.116d-05	5.358d-01
2.117d-05	5.359d-01
2.118d-05	5.359d-01
2.119d-05	5.359d-01
2.120d-05	5.359d-01
2.121d-05	5.359d-01
2.122d-05	5.359d-01
2.123d-05	5.359d-01
2.124d-05	5.359d-01
2.125d-05	5.359d-01
2.126d-05	5.359d-01
2.127d-05	5.359d-01
2.128d-05	5.359d-01
2.129d-05	5.359d-01
2.130d-05	5.359d-01
2.131d-05	5.359d-01
2.132d-05	5.359d-01
2.133d-05	5.359d-01
2.134d-05	5.359d-01
2.135d-05	5.359d-01

```
!Filename: g9cmd.sim
!This file contains the commands to
!be executed by " esim " .
!The cell to be simulated is " G9 "
!G9 is the ninth block of the digital
!filter with R parameter ( R9 = 0.15625 )
!and tap parameter W'10 = - 0.1171875
!AA,BB,CC and DD are defined vectors for
!nodes (a10 ... a1),(b10 ... b1),(c10 ... c1),
!(d10 ... d1) and (e10 ... e1) respectively.
!AA = The sampled input X10(n) to block9
!BB = The sampled output (vh10)*(X10(n+1))
!CC = The sampled input X9(n+1) to block9
!DD = The sampled input X9(n) to block 7
!X10(n) is the sampled input of the digital
!filter.
I
W AA a10 a9 a8 a7 a6 a5 a4 a3 a2 a1
W BB b10 b9 b8 b7 b6 b5 b4 b3 b2 b1
W CC c10 c9 c8 c7 c6 c5 c4 c3 c2 c1
W DD d10 d9 d8 d7 d6 d5 d4 d3 d2 d1
! Defining vectors AA,BB,CC and DD
h a5 a6 a9 a10
l a1 a2 a3 a4 a7 a8
h c1 c5 c6 c9 c10
l c2 c3 c4 c7 c8
s
! Setting values for AA and CC
h c1 c5 c8 c7 c9
l a1 a6 a7 c2 c3 c4 c10 c6
h a2 a3 a4 a5 a8 a9 a10
s
! Same as above but for different values of
! AA and CC
$
```

*(Next pages contain input commands files and simulution output of "ESIM" for cell g9,g7,and g3)

```
/* Filename : s9out.sim */
/* This file contains the results of
the simulation performed by " esim "
on cell ' s9 ' with the input commands read
from the command file s9cmd.sim */
initialization took 2359 steps
step took 709 events
DD=0011111000   248
CC=1100110001   817
BB=0011001111   207
AA=1100110000   816
step took 506 events
DD=0101010001   337
CC=0111010001   465
BB=1100101111   815
AA=1110011110   926
985 transistors, 714 nodes (390 pulled up)
$
```

```
!Filename: s7cmd.sim
!This file contains the commands to
!be executed by ' esim ' .
!The cell to be simulated is ' G7 '
!G7 is the fifth block of the digital
!filter with R parameter ( R7 = 0.4375 )
!and tap parameter W'3 - 0.484375
!AA,BB,CC,DD and EE are defined vectors
!for nodes (a10 ... a1),(b10 ... b1),(c10 ... c1),
!(d10 ... d1) and (e10 ... e1) respectively.
!AA = The sampled input X8(n) to block7
!BB = The sampled output (vh8)*(X8(n+1))
!CC = The sampled input X7(n+1) to block7
!DD = The sampled input X7(n) to block 6
!EE = The sampled input X8(n+1) to block 7
!PHI is a defined vector of nodes p2 and p1.
!p2 & p1 are the two nonoverlapping clocks inputs
!to the digital filter .
!
W AA a10 a9 a8 a7 a6 a5 a4 a3 a2 a1
W BB b10 b9 b8 b7 b6 b5 b4 b3 b2 b1
W CC c10 c9 c8 c7 c6 c5 c4 c3 c2 c1
W DD d10 d9 d8 d7 d6 d5 d4 d3 d2 d1
W EE e10 e9 e8 e7 e6 e5 e4 e3 e2 e1
W PHI p2 p1
! Defining vectors AA,BB,CC,DD and EE .
h a5 a6 a9 a10
l a1 a2 a3 a4 a7 a8
h c1 c5 c6 c9 c10
l c2 c3 c4 c7 c8
h p2
l p1
s
h p1
l p2
s
h p2
l p1
s
h p1
l p2
s
! Setting values for AA and CC
! Triggering the nonoverlapping to
! propogate the output EE
h c1 c5 c8 c7 c9
l a1 a6 a7 c2 c3 c4 c10 c6
h a2 a3 a4 a5 a8 a9 a10
h p2
l p1
s
h p1
l p2
s
h p2
l p1
s
h p1
l p2
s
! Same as above but for different values of
! AA and CC
!
!
```

```
/* Filename : s7out.sim */
/* This file contains the results
of the simulation performed by ' esim '
on cell ' s7 ' with the commands file
read from the commands file ' s7cmd.sim '  */
initialization took 3063 steps
step took 888 events
PHI=10  2
EE=1111111111   1023
DD=0101111100   380
CC=1100110001   817
BB=0000001011   11
AA=1100110000   816
step took 62 events
PHI=01  1
EE=1111111111   1023
DD=0101111100   380
CC=1100110001   817
BB=0000001011   11
AA=1100110000   816
step took 62 events
PHI=10  2
EE=1111111111   1023
DD=0101111100   380
CC=1100110001   817
BB=0000001011   11
AA=1100110000   816
step took 67 events
PHI=01  1
EE=0110011001   409
DD=0101111100   380
CC=1100110001   817
BB=0000001011   11
AA=1100110000   816
step took 1008 events
PHI=10  2
EE=0110011001   409
DD=0010100101   165
CC=0111010001   465
BB=1110100111   935
AA=1110011110   926
step took 62 events
PHI=01  1
EE=0110011001   409
DD=0010100101   165
CC=0111010001   465
BB=1110100111   935
AA=1110011110   926
step took 62 events
PHI=10  2
EE=0110011001   409
DD=0010100101   165
CC=0111010001   465
BB=1110100111   935
AA=1110011110   926
step took 67 events
PHI=01  1
EE=1000010011   531
DD=0010100101   165
CC=0111010001   465
BB=1110100111   935
AA=1110011110   926
1230 transistors, 914 nodes (510 pulled up)
$
```

```
!Filename: g3cmd.sim
!This file contains the commands to
!be executed by ' gsim ' .
!The cell to be simulated is ' G3'
!G3 is the third block of the digital
!filter with R parameter ( R3 = 0.9375 )
!and tap parameter W'4 = 0.1875     .
!AA,BB,CC,DD and EE are defined vectors
!for nodes (a10 ... a1),(b10 ... b1),(c10 ... c1),
!(d10 ... d1) and (e10 ... e1) respectively.
!AA = The sampled input X4(n) to block 3
!BB = The sampled output (vh4)*(X4(n+1))
!CC = The sampled input X3(n+1) to block 3
!DD = The sampled input X3(n) to block 1
!EE = The sampled input X4(n+1) to block 5
!PHI is a defined vector of nodes p2 and p1.
!p2 & p1 are the two nonoverlapping clocks inputs
!to the digital filter .
!
W AA a10 a9 a8 a7 a6 a5 a4 a3 a2 a1
W BB b10 b9 b8 b7 b6 b5 b4 b3 b2 b1
W CC c10 c9 c8 c7 c6 c5 c4 c3 c2 c1
W DD d10 d9 d8 d7 d6 d5 d4 d3 d2 d1
W EE e10 e9 e8 e7 e6 e5 e4 e3 e2 e1
W PHI p2 p1
! Defining vectors AA,BB,CC,DD and EE .
h a5 a6 a9 a10
l a1 a2 a3 a4 a7 a8
h c1 c5 c6 c9 c10
l c2 c3 c4 c7 c8
h p2
l p1
s
h p1
l p2
s
h p2
l p1
s
h p1
l p2
s
! Setting values for AA and CC
! Triggering the nonoverlapping to
! propogate the output EE
h c1 c5 c8 c7 c9
l a1 a6 a7 c2 c3 c4 c10 c6
h a2 a3 a4 a5 a8 a9 a10
h p2
l p1
s
h p1
l p2
s
h p2
l p1
s
h p1
l p2
s
! Same as above but for different values of
! AA and CC
$
```

```
/* Filename : g3out.sim */
/* This file contains the results of
the simulation performed by ' esim '
on cell ' g3 ' with the input commands
read from the commands file g3cmd.sim */
initialization took 2639 steps
step took 725 events
PHI=10  2
EE=1111111111   1023
DD=1010010110   662
CC=1100110001   817
BB=0110011111   415
AA=1100110000   816
step took 62 events
PHI=01  1
EE=1111111111   1023
DD=1010010110   662
CC=1100110001   817
BB=0110011111   415
AA=1100110000   816
step took 62 events
PHI=10  2
EE=1111111111   1023
DD=1010010110   662
CC=1100110001   817
BB=0110011111   415
AA=1100110000   816
step took 63 events
PHI=01  1
EE=1111111011   1019
DD=1010010110   662
CC=1100110001   817
BB=0110011111   415
AA=1100110000   816
step took 817 events
PHI=10  2
EE=1111111011   1019
DD=0011001001   201
CC=0111010001   465
BB=0111010000   464
AA=1110011110   926
step took 62 events
PHI=01  1
EE=1111111011   1019
DD=0011001001   201
CC=0111010001   465
BB=0111010000   464
AA=1110011110   926
step took 62 events
PHI=10  2
EE=1111111011   1019
DD=0011001001   201
CC=0111010001   465
BB=0111010000   464
AA=1110011110   926
step took 66 events
PHI=01  1
EE=1101111100   892
DD=0011001001   201
CC=0111010001   465
BB=0111010000   464
AA=1110011110   926
1190 transistors, 876 nodes (470 pulled up)
:
```

REFERENCES

[1] Abraham Peled, "On the Hardware Implementation of Digital Signal Processors", IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-24, no. 1, Feb. 1976.

[2] J.D. Markel and A.H. Gray, Jr., "On Autocorrelation Equations as Applied to Speech Analysis", IEEE Trans. on Electroacoustics, vol. AU-21, no. 2, April 1975.

[3] P. Dewilde, E. Deprettere, and R. Nouta, "Parallel and Pipelined VLSI Implementation of Signal Processing Algorithms", Delft University of Technology, Delft The Netherland.

[4] A.H. Gray, Jr., "A Normalized Digital Filter Structure", IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-23, no. 3 June 1975.

[5] P. Dewilde, Ed.F. Deprettre and C.V.K. Prabhakara Rao, "Orthogonal Digital Filters", Delft University of Technology, Department of Electrical Engineering, Mekelweg 4, 2628 CD Delft, The Netherlands.

[6] J.D. Markel, "Fixed-Point Implementation Algorithm for a Class of Orthogonal Polynomial Filter Structures", IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-23, no. 5, October 1975.

[7] Harvey L. Garner, "Number Systems and Arithmetic", in Advances in Computers, vol. 6, F. L. Alt, Ed. New York: Academic, 1965, pp. 163-168.

[8] W. Ulbrich, T. Noll, and B. Zehner, "Mos-VLSI Pipelined Digital Filters for Video Applications", Proc, IEEE International Conference on Acoustics, Speech & Signal Processing, 1984.

[9] A.H. Gray, Jr. and J.D. Markel, "Digital Lattice and Ladder Filter Synthesis", IEEE Trans. on Audio and Electroacoustics, vol. AU-21, no. 6, December 1973.

[10] Karlheinz Hofer, "Minimization of the Number of Additions in Digital Filters without Multipliers", Institut fur Netzwerk- and Systemtheorie, Universitat Stuttgart. European conf. on circuit theory and design, 1983.

[11] T. Noll and W. Ulbrich, "Digital Filter Structures with Parallel Arithmatic for Custom Designs", Proc. European Conf. on Circuit Theory and Design, ECCTD, Stuttgart, 1983.

[12] C. Mead and L. Conway, "Introduction to VLSI Systems", Addison-Wisely Publishing Company, Reading Massachsetts, 1980.

[13] A.V. Oppenheim and R.W. Schafer, "Digital Signal Processing", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[14] Robert Nobles, "Development and Integration of a Distributed Computer System for Integrated Circuit Design using VAX-11/750 and SCALDsystem Computers", Masters Thesis ECE Dept Ohio University, January 1986.

[15] Jeffrey D. Ullman, "Computational Aspects of VLSI", Computer Science Press,Inc,1984.