DIGITAL DUAL TONE MULTIFREQUENCY RECEIVER

A THESIS
SUBMITTED TO THE DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Ahmet Suat EKİNCİ

September 1994

DIGITAL DUAL TONE MULTIFREQUENCY RECEIVER

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Ahmet Suat Ekinci

September 1994

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Abdullah Atalar (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Assoc. Prof. Dr. M. Ali Tan (Co-supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Murat Aşkar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Dr. Orhan Arıkan

Approved for the Institute of Engineering and Sciences:

[illegible]

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

DIGITAL DUAL TONE MULTIFREQUENCY RECEIVER

Ahmet Suat Ekinci
M.S. in Electrical and Electronics Engineering
Supervisors: Dr. Abdullah Atalar
and
Dr. M. Ali Tan
September 1994

In this thesis, a suitable algorithm for detection of Dual Tone Multifrequency tones is proposed and implemented as a VLSI chip. The algorithm is based on an approximation of correlation. The input signal is correlated with the hardlimited versions of three sinusoids having $\pi/3$ phase difference. Also, a level detector is added to the algorithm. The algorithm only requires addition and subtraction, but no multiplication; and this reduces the complexity of the circuit. The implementation of the algorithm proposed has been realized as a fully integrated digital DTMF receiver chip using semi-custom layout techniques. The final chip is fabricated in 1-μm CMOS technology, and it has a total area of 24.78 mm^2.

Keywords : Dual Tone Multifrequency, correlation receivers, frequency detection.

ÖZET

SAYISAL ÇİFT TONLU ÇOK FREKANSLI SİNYAL ALICISI

Ahmet Suat Ekinci
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticileri: Dr. Abdullah Atalar
ve
Dr. M. Ali Tan
Eylül 1994

Bu tezde, çift tonlu çok frekanslı sinyallerin saptanması için bir algoritma önerilmiş ve çok büyük çapta tümleşik yonga olarak gerçeklenmiştir. Algoritma bir korelasyon işlemi yaklaşığına dayanmaktadır. Giriş sinyali aralarında $\pi/3$ faz farkı olan üç sinuzoidal dalganın iki değere nicemlenmesiyle elde edilen işaretlerle ilintilendirilmiştir. Algoritmaya bir de düzey saptayıcısı eklenmiştir. Algoritma toplama ve çıkarma işlemlerinden oluşup, çarpma işlemi içermemektedir; bu da devre karmaşıklığını azaltmaktadır. Önerilen algoritmayla gerçekleştirilen sayısal çift tonlu çok frekanslı alıcı yongası ölçümlü gözeler kullanılarak yapılmıştır. Yonga, 1-μm CMOS teknolojisi kullanılarak üretilmiştir, toplam alanı 24.78 mm^2 dir.

Anahtar Kelimeler : Çift tonlu çok frekanslı, korelasyon tipi alıcılar, frekans saptama.

To Koray Karahan (1970-1991)

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my supervisors Dr. Mehmet Ali Tan and Dr. Abdullah Atalar for their guidance, suggestions, and invaluable encouragement throughout the development of this thesis. I would like to thank to Dr. Murat Aşkar and Dr. Orhan Arıkan for reading and commenting on the thesis.

And special thanks to all Graduate Students in this department for their continuous support.

I would like to thank to TUBITAK AEAGE for providing all the facilities which have made the completion of this study possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Telephony, first invented by Alexander Graham Bell and patented in the USA in 1876, is perhaps the most important means of long-distance communication. Telephony is the transmission of sound to a distant place. The literal translation of 'telephone' from the Greek means 'long distance-voice' [1].

On February 14, 1876 Bell applied for a patent, and on March 7 it was issued. However, first demonstration was not until 10th of March. The first service was a *private-line* service. Then a number of telephones were connected to the same line, but everyone could hear everyone's talk, there was no privacy (*party-line* service). (Shown in Fig. 1.1). Then, the thought of centralized switching arose. Many lines come to a "central office" or "exchange", and the switching occurs there. Exchange telephone service was invented in 1878 allowing 21 customers to reach each other by means of a central switchboard. Most of the exchanges serve at most 10,000 customers (i.e. 4 digits should be used to represent the numbers). The switchboard was operated manually, the first automated switchboard was invented by Almon B. Strowger in 1892. As the number of telephones grows up, more central offices were created each serving a nearby area and the lines connecting these offices are called trunks. The centralized switching of trunks are performed at a switching office called tandem office. New switching needed to be developed to serve long-distance or toll circuits between cities. Toll offices were thus devised to perform switching of toll trunks only.[2, 3]

In late 1879, telephone numbers were used for the first time. In 1896 A.E. Keith, J. Erickson, and C.J. Erickson, all associates of Strowger, invented a

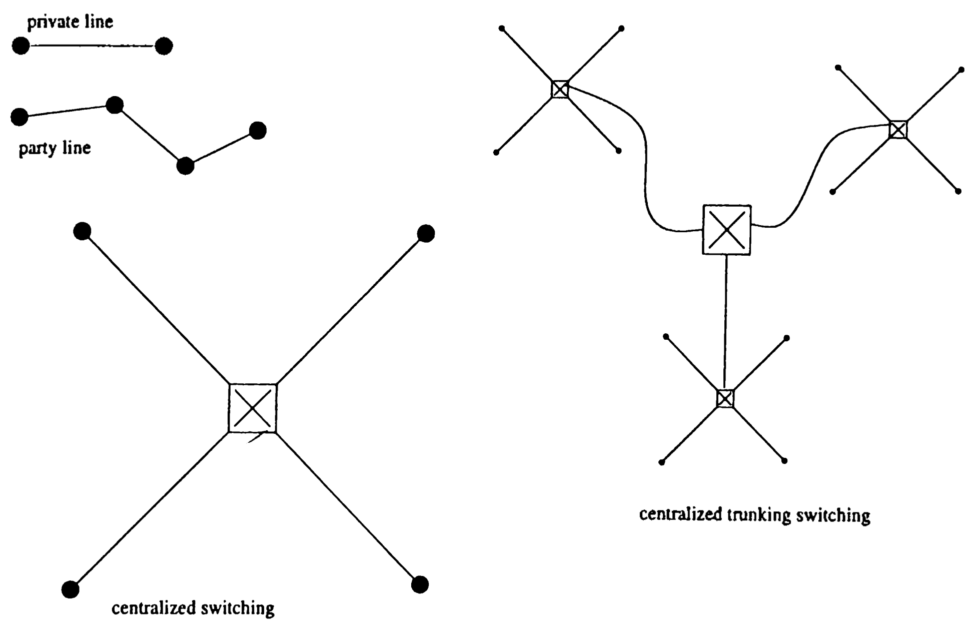


Figure 1.1: Switching configurations.

rotating finger-wheel for dialing the desired telephone number.

In todays telephony systems, "calling customer" dials the number on the telephone handset and the number is relayed to the exchange by one of the following two signalling methods:

- "loop" and "disconnect" pulsing
- "multifrequency" tone signalling

"Loop" and "disconnect" pulsing or in other words LD signalling is the older of the two methods. The line is disconnected and re-looped in a very rapid sequence. A pulse of disconnection is obtained. The number of pulses are used to indicate digits of the destination number. One pulse for "1" , two pulses for "2" , . . . , ten pulses for "0". The pulses are repeated at a frequency of 10 per second. So 0 will take a whole second. There is also a gap between two sequences of pulses representing two numbers which is called inter-digit pause (IDP).

Multifrequency signalling, in which the digits are relayed much faster, is a more recent invention, and it has been spreading all around the world. The exchange is alerted by looping the line, but the digits are dialled as a number of short bursts of tones. Rather than detecting the disconnect pulses, the exchange should detect the frequency of the tones. Multifrequency signalling is also called DTMF signalling where DTMF stands for dual-tone multifrequency. [1, 2, 3].

The problem of tone detection arises here. Many scientists have dealt with detection of DTMF tones problem.

First, analog MF receivers have been used. These have employed a bank of six filters to determine the signalling frequencies. The filters have been followed by detectors producing outputs proportional to power of the frequency components. And finally these outputs and the power of the input signal have been used to determine the correct tones. The likelihood of digit simulation which is the false detection because of the similar tones at the presence of speech has been reduced by the timing specifications of digit and pause durations.

It has been then proposed to use digital receivers to reduce the costs [4]. Koval has used DFT in his receiver, and "its performance was as good as conventional analog receiver".

M. J. Callahan has made a single integrated-circuit receiver[5]. He has proposed two filters to separate the high group and low group components. Then the periods of the signals have been counted to come to a decision. The filters have been located off chip. Then, another DTMF receiver; combining filters, zero crossing detectors and amplitude detectors, has been fabricated [6].

There are also other methods proposed containing digital filters. Instead of analog reception digital reception is used in order to reduce the filter orders, because PCM data is converted from the input signals that are bandlimited to 4kHz [7]. The output of this input filter is then separated by two band-elimination filters. These filters are followed by limiters which drive the band-pass filters which have been tuned to the correct frequencies. All of these are done by a single DSP chip.

Nonrecursive digital filters can also be used in DTMF receivers [8]. Here two linear phase filters with $\pi/2$ phase difference should be used to eliminate the input phase dependence. The outputs of these filters are squared and added. It is seen that linear phase filters are as good as filters without phase constraints.

In an approach by Denenberg [9], spectral moment estimates of the input have been used for determining the presence of a tone. These three moments measure the power, power mean frequency and RMS power bandwidth of the signal. This implementation has required quadrature detection, two lowpass filters and digital differentiation logic to obtain three moments. Detection logic has followed these.

A well-known method of detection of tones with unknown phase in white Gaussian noise is to correlate the input signal with the sine and cosine of the frequency to be detected. To reduce the complexity of the circuit several methods have been proposed. [10, 11, 12]

The algorithm that has been described in [10] is based on the correlation principle where the signals to be correlated are transformed into binary signals that have approximately the same correlation function. Such transformations have been described in [13]. In this paper, two linear combinations of Rademacher functions have been used as auxiliary signals in correlator. A.D. Proudfoot has explained [12] an algorithm based on a correlation with hardlimited versions of sine and cosine. It is extended in this thesis and used to implement a receiver. In [11] a digital correlator for detecting the multiple signal has been described. The correlation is simplified to a summing accumulator using four multiplications at a time. This introduces more hardware than the previous ones as it includes multiplicators (or ROMs) in it.

A DTMF receiver using synchronous additions and subtractions has been proposed in [14], which is a simplification of the correlation into a series of additions and subtractions. It has been done by sampling the input signal at four times each frequency involved. The results have been supported by minimum signal to noise ratio, the ratio of maximum accumulated and sampled values, number of zero crossings and the number of extremes in order to reduce digit simulation and detector has been implemented by using an 8-bit microcomputer without any special hardware.

In the next chapter, the basis for the correlation algorithms are explained with some different approaches. The implementation of the design is described in Chapter 3, and computer simulation results are shown in Chapter 4.

Chapter 2

DTMF RECEIVERS

Technical features of the push-button telephone sets are explained in the recommendations of the CCITT[1]. In recommended multifrequency code the dialing signal is composed of two frequencies emitted simultaneously when a button is pressed. It is planned to have 10 decimal digits and 6 reserve signals. The two frequencies composing each signal are taken from two mutually exclusive frequency groups of four frequencies each, a code known as the "2 (1/4) code"[2]. The low and high group frequencies are shown in Table 2.1. In Fig. 2.1, the

low group	high group
697	1209
770	1336
852	1477
941	1633

Table 2.1: The signaling frequencies in Hz.

push button frequencies are shown. There are 10 decimal digits and 6 reserve signals making 16 signals altogether.

Characteristics of multifrequency push-button (MFPB) telephone sets using voice frequency signals are included in Recommendation Q.23, and Recommendation Q.24 is for application in local exchanges for the reception of MFPB signals. There are recommendations for various characteristics of such

[1]Recommendation Q.23
[2]Recommendation Q.23

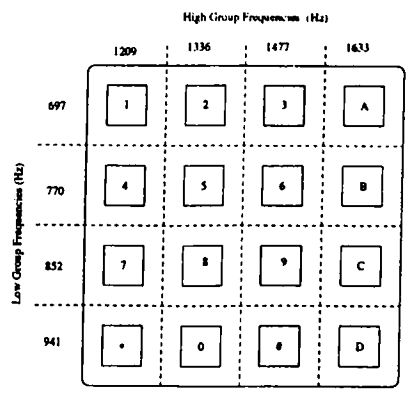


Figure 2.1: Push buttons and touch tones.

receivers. Each signal consists of two frequencies taken from mutually exclusive frequency groups of four frequencies each, as shown in Fig. 2.1 and Table 2.1. The exchange should check for simultaneous presence of one and only one frequency from each group. It should respond to signals having frequencies within the tolerances for MFPB sending (± 1.8 % of the frequency). Wider tolerances may be appropriate to tolerate the disturbances during the transmission. The reception characteristics may take the advantage of a limitation, if specified, on maximum difference in power levels of the two frequencies composing the tone. This is also called the twist. The exchange should recognize signals whose duration exceed the minimum expected value from subscribers to guard against false signal indications. Similarly pause intervals greater than a specified value should be recognized by the exchange. The receiver should operate properly at the presence of speech and a dial tone. These recommendations do not supersede the existing national specifications. In Table 2.2, the values of multifrequency push-button receiving parameters adopted by various administrations are shown.

The algorithm proposed in this thesis satisfies the specifications (Table 2.2) of frequency tolerance and power level difference between frequencies.

2.1 Methods for Receiving

For detecting a tone the first thing to be done is finding the frequency components of the input signal. As there are finite number of tones to be detected the problem reduces to detection of only those frequencies that can form a valid

Parameters		NTT	AT&T
Frequency tolerance	Operation	$\leq 1.8\%$	$\leq 1.5\%$
	Non-operation	$\geq 3.0\%$	$\geq 3.5\%$
Power levels per frequency	Operation	-3 to -24 dBm	0 to -25 dBm
	Non-operation	Max. -29 dBm	Max. -55 dBm
Power level difference between frequencies		Max. 5dB	+4dB to -8dB
Signal duration	Operation	Min. 40 ms	Min. 40 ms
	Non-operation	Max. 24 ms	Max. 23 ms
Pause Duration		Min. 30 ms	Min. 40 ms

Table 2.2: Specifications for multifrequency push-button receiving

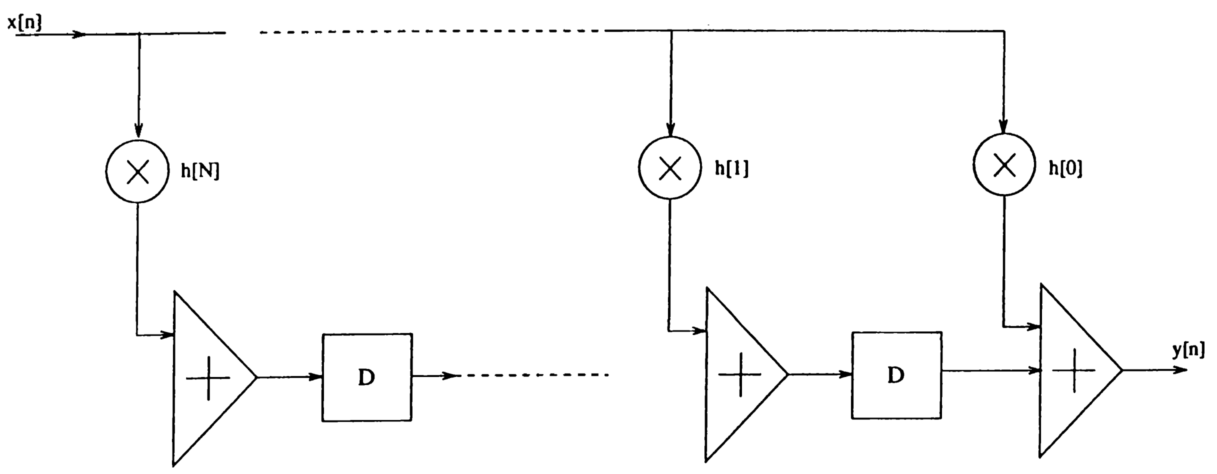


Figure 2.2: Realization of the filter.

tone. For this purpose several detection algorithms are proposed.

In [8] the use of nonrecursive digital bandpass filters are explained. These are filters having N+1 nonzero entries at most.

$$\begin{aligned} y[n] &= h[n] * x[n] \\ y[n] &= \sum_{k=0}^{N} h[k] \cdot x[n-k] \end{aligned}$$

Here $x[n]$ is the input sequence, $y[n]$ is the output sequence and $h[n]$ is the impulse response of the filter. $h[n]$ is zero for $n > N$ and $n < 0$.

Such filters can be built in the way shown in figure Fig. 2.2. The same filter can be realized as a set of $N+1$ correlators (Fig. 2.3). In each of these correlators the input sequence is cross-correlated with the impulse response of the nonrecursive filter. The k'th accumulator is connected to the output at

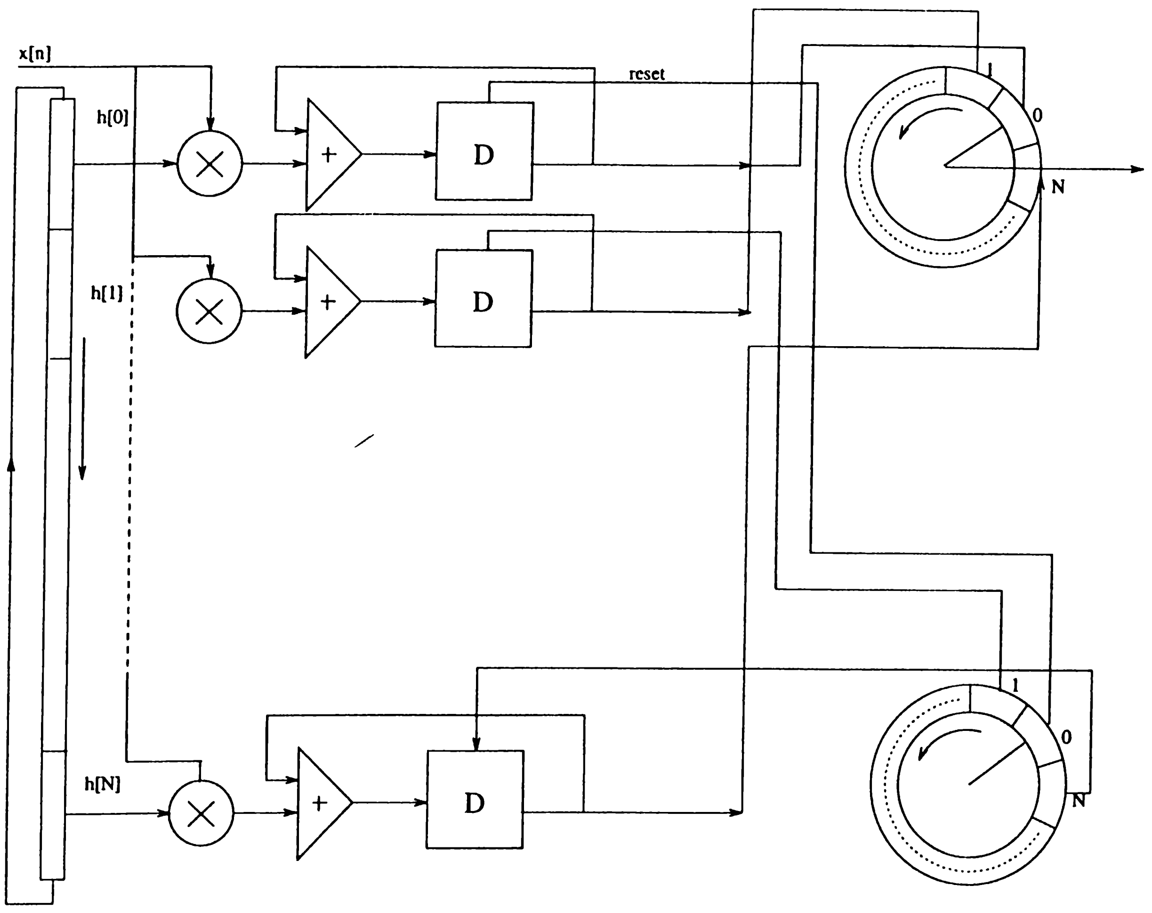


Figure 2.3: Another realization of the same filter.

$t = k \cdot T$ and then reset to zero where T is the time interval between two samples.

If the output rate is reduced by some constant L, then the complexity of the circuit decreases. When $L = N + 1$, only one correlator is used, and this means one multiplication and one addition have to be carried out and only one value has to be stored for every output value.

For eliminating the phase dependence, two filters with 90° phase shift have been used. The results have been squared and added. The filters have been reduced to two correlators. This is the base for the correlation receivers.

The purpose of a correlator is to determine the integral [10]

$$\overline{\rho}_{\overline{x}\,\overline{y}}(\tau) = \frac{1}{T_0} \int_0^{T_0} \overline{x}(t)\overline{y}(t-\tau)dt \qquad (2.1)$$

where $\overline{x}(t)$ and $\overline{y}(t)$ are the two signals to be correlated. The digital version of such a correlator determines instead of the integral the sum given by:

$$\rho_{xy}(m) = \frac{1}{N_0} \sum_{n=0}^{N_0-1} x(n)y(n-m) \tag{2.2}$$

A hardware realization of this requires a multiplier and an accumulator. There exists a simple procedure [10] for transforming each signal into a binary signal in such a way that the correlation function of these binary signals is approximately equal to the correlation function of the original functions. Using auxiliary functions a_1 and a_2,

$$\begin{aligned} u(n) &= \text{sign}(x(n) + a_1(n)) \\ v(n) &= \text{sign}(y(n) + a_2(n)) \end{aligned} \tag{2.3}$$

is obtained and their correlation function is

$$\rho_{uv}(m) = \frac{1}{N_0} \sum_{n=0}^{N_0-1} u(n)v(n-m) \tag{2.4}$$

It has been shown that sets $\{a_1, a_2\}$ exist such that for large N_0, $\rho_{uv}(m)$ converges to $\rho_{xy}(m)$ apart from a constant[10].

$$\rho_{uv}(m) \rightarrow C \cdot \rho_{xy}(m), \qquad N_0 \rightarrow \infty \tag{2.5}$$

It has been shown that an approximation of (2.5) can be obtained by taking for a_1 and a_2 suitable linear combinations of Rademacher Functions (Fig. 2.4), and that the approximation can be made more accurate by taking more Rademacher functions in each of the signals. Let T_k denote the period of the k'th Rademacher function,

$$T_k = T_{k-1}/2, \qquad k = 2, 3, \cdots \tag{2.6}$$

$$\bar{a}_1(t) = \frac{1}{8}(\bar{r}_1(t) + 2\bar{r}_2(t) + 4\bar{r}_3(t)) \tag{2.7}$$

$$\bar{a}_2(t) = \frac{1}{8}(\bar{r}_4(t) + 2\bar{r}_5(t) + 4\bar{r}_6(t)) \tag{2.8}$$

where $\bar{r}_k$ is continuous-time k'th order Rademacher function, $\bar{a}_1$ and $\bar{a}_2$ are the continuous-time auxiliary functions.(Fig. 2.5)

Another approach is by A.D. Proudfoot [12]. This approach has been motivated by Fourier-analysis techniques. It has been shown that the multifrequency tones may be recognized in 12 ms. The receiver is shown in Fig. 2.6.

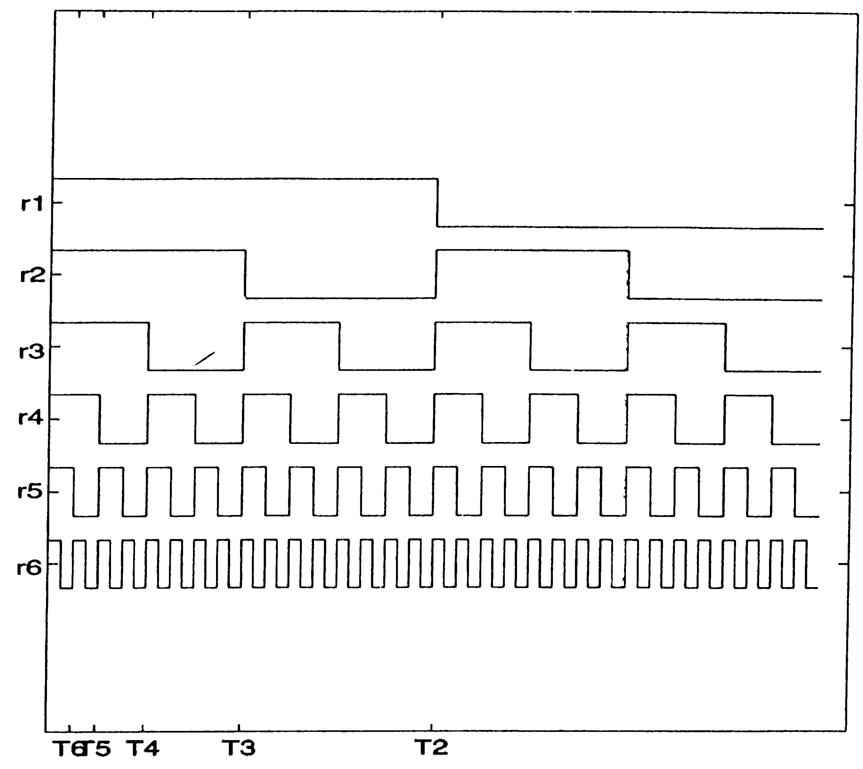


Figure 2.4: Rademacher functions

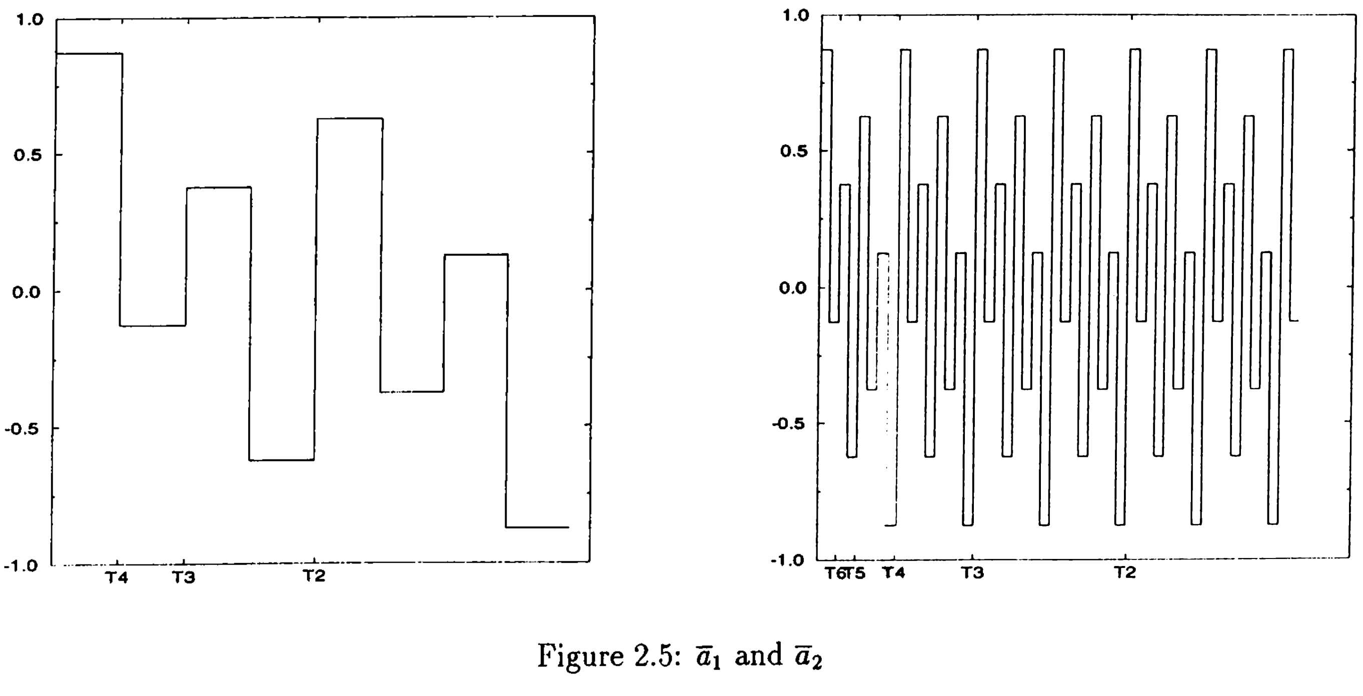


Figure 2.5: $\overline{a}_1$ and $\overline{a}_2$

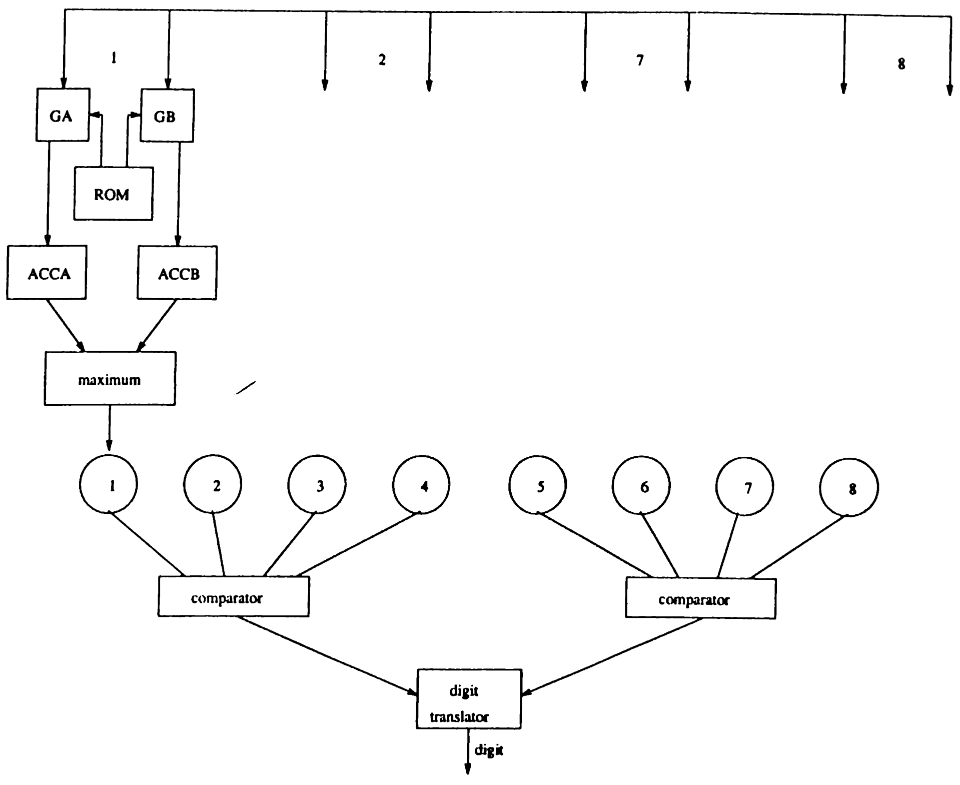


Figure 2.6: Functional diagram of the digital multifrequency tone detector in Proudfoot.

ROM represents a stored sequence of approximately 100 sign bits. In fact there are 100 sign bits for each sinusoid, but the cosine function is obtained by taking a later bit. The contents of first ROM may be represented by the sequence A (665666566656665666), denoting six '+', six '−', five '+', ...This means the first 6 samples of the sinusoid having frequency 697 Hz are positive, the next 6 samples are negative, ...GA represents a sign gate and pulse counter. It counts the number of pulses and changes the sign according to the input coming from the ROM. GB is the same but it introduces the 90^o phase delay. Absolute maximum of the values of accumulators A and B is found. Then the first comparator finds the maximum of the outputs 1 to 4 and the second comparator finds the maximum of the outputs 5 to 8. Finally a translation is made from the coming indexes into a number. It is also added that if the number of the sign bits is increased to 200 then the frequency tolerance will decrease to $\pm 0.4\%$.

In Proudfoot's approach there are two approximations to correlation algorithms. First one is using square waves instead of sine waves (sign bits are used); the second approximation is taking the maximum of the accumulated values instead of taking the squares and then finding the square-root of the sum.

The results of these approximations are discussed in section 2.2.

2.2 The Method Used

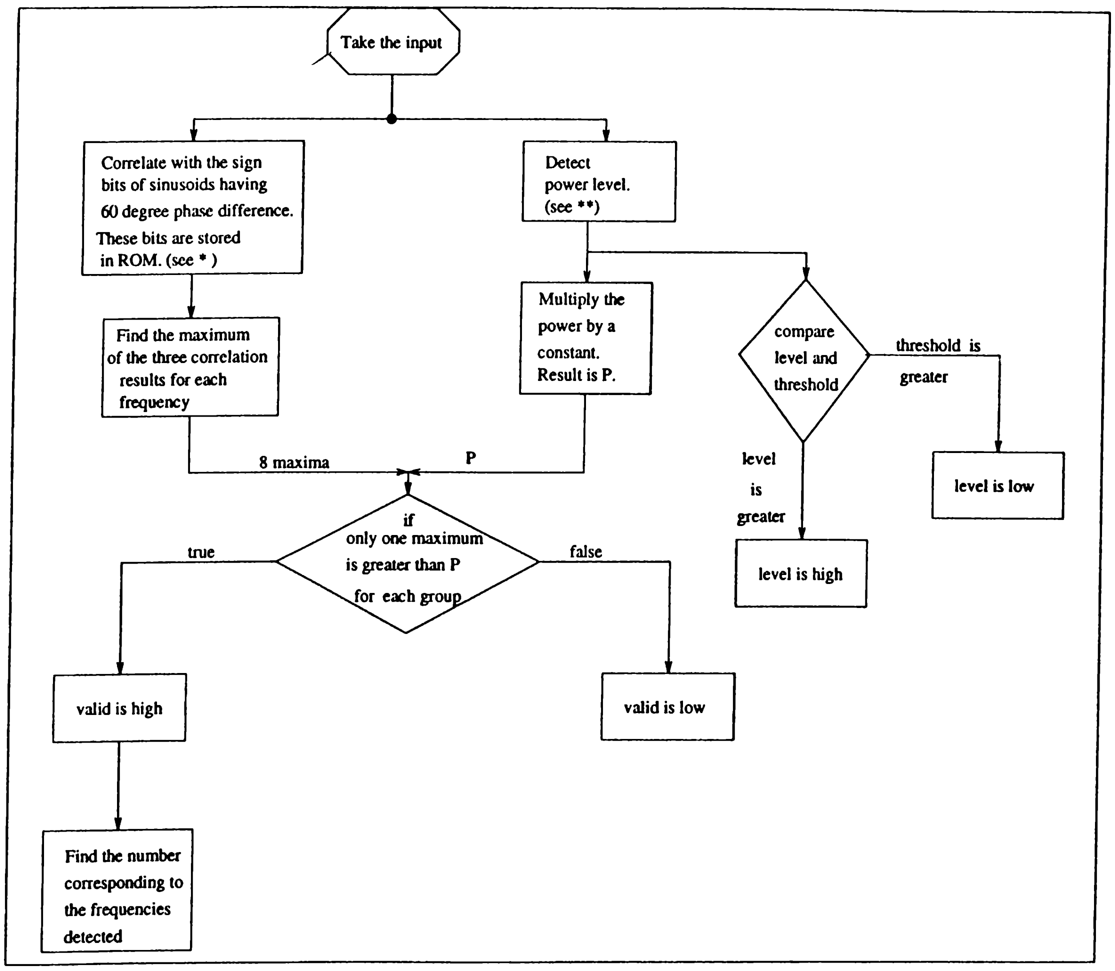
In this thesis, a correlation algorithm is examined and implemented. Some algorithms which can be used are described in Section 2.1.

First method was classical correlation algorithm. The input is first correlated with the sinusoids which are stored in the ROM. There are 8 frequencies and 2 sinusoids for each frequency. Besides, each entry of the ROM is a word. So it can be concluded that this method requires a lot of silicon area. This method also introduces some multiplications which also consume a lot of area.

The second method was using binary signals instead of correlating the original signals. The input signal and the sinusoids to be correlated are added to linear combinations of Rademacher functions, the sums are hardlimited and then correlated. This method requires the area for ROM containing 1-bit sequences for each sinusoid, as the data is one bit, the multiplication reduces to an AND operation. It also includes power detection, using absolute sum of the input stream. Then the result is multiplied with the auxiliary signal for the input in order to compensate the power level difference between the incoming signal and the sinusoid to be correlated with. This multiplication requires a lot of silicon area. Finally, the correlation results are squared and added. which needs a ROM or some approximations requiring more area.

The third method, proposed by A.D. Proudfoot; requires less area, because it has no multiplication or squaring and adding in it. However, it is more dependent on the phase difference between the input and the sinusoid to be correlated with. To reduce this, three sinusoids are used instead of two. The results are depicted in Fig. 2.12 and Fig. 2.13.

In the approach proposed in this thesis; as Proudfoot has proposed, the input signal is first multiplied by the sign-bits stored in the ROM , then a number of products are added. This number is chosen according to the specifications of the detector. A power detector has also been added to the circuitry. This



Take the input
Correlate with the sign bits of sinusoids having 60 degree phase difference. These bits are stored in ROM. (see *)
Detect power level. (see **)
Find the maximum of the three correlation results for each frequency
Multiply the power by a constant. Result is P.
compare level and threshold
threshold is greater
level is low
level is greater
level is high
8 maxima
P
if only one maximum is greater than P for each group
true
false
valid is high
valid is low
Find the number corresponding to the frequencies detected

Figure 2.7: Flow chart of the detection algorithm. (* is shown in Fig. 2.21, and ** is shown in equation 2.9)

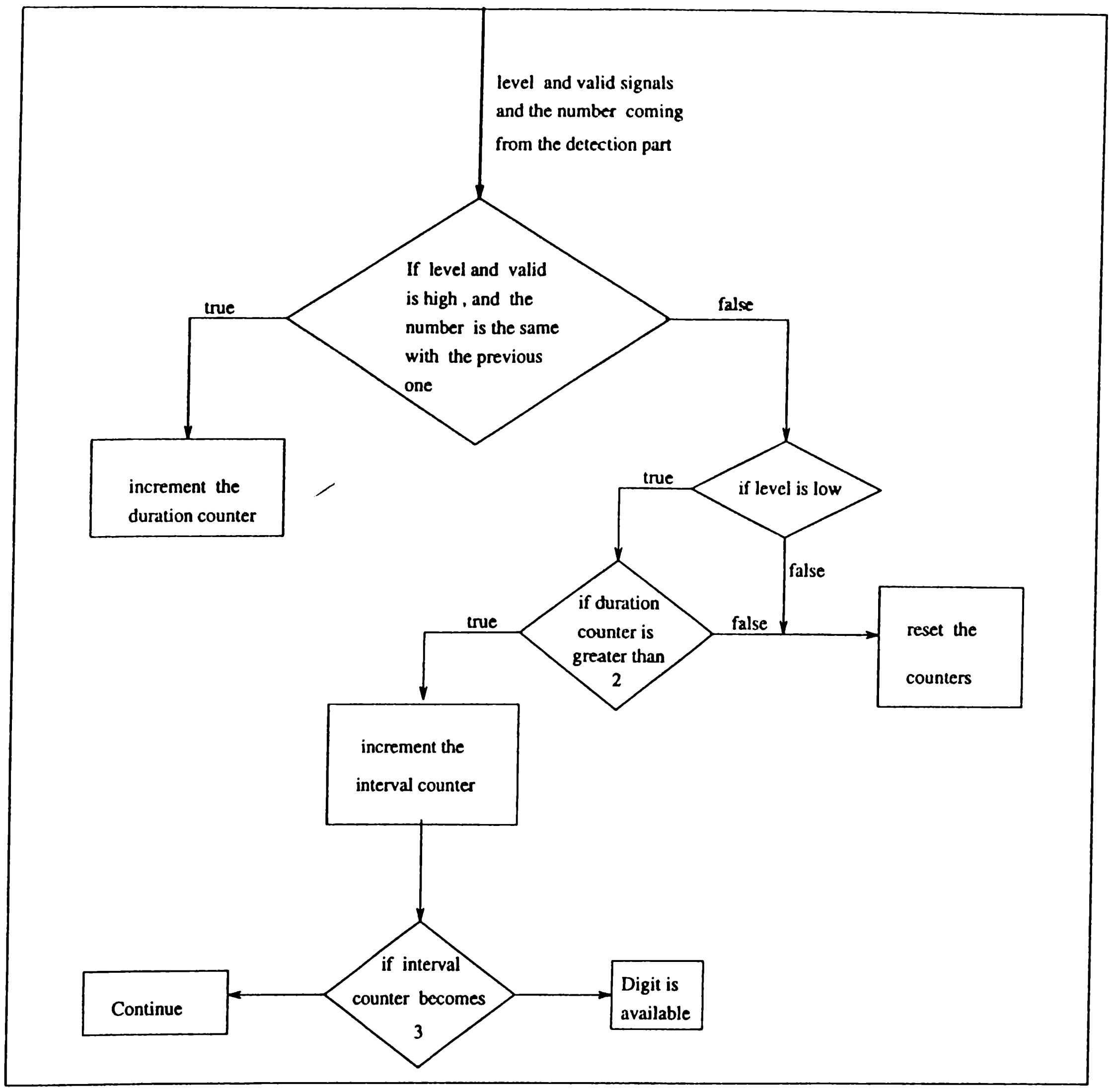

level and valid signals and the number coming from the detection part
If level and valid is high , and the number is the same with the previous one
true
false
increment the duration counter
if level is low
true
false
if duration counter is greater than 2
true
false
reset the counters
increment the interval counter
if interval counter becomes 3
Continue
Digit is available

Figure 2.8: Flow chart of the decision algorithm.

is because the receiver would receive signals having amplitudes different from 1. The whole algorithm is shown by the flow charts Fig. 2.7 and Fig. 2.8. The general view of the detector is shown in Fig. 2.9.

The results of the different approaches to the correlation receivers are shown in Fig. 2.10. The input is a pure sinusoid with frequency running from 550Hz to 850Hz. Correlation stands for the usual method of correlation. The input is correlated with the sinusoids

$$\sin(2\pi f_o t)$$

and

$$\cos(2\pi f_o t).$$

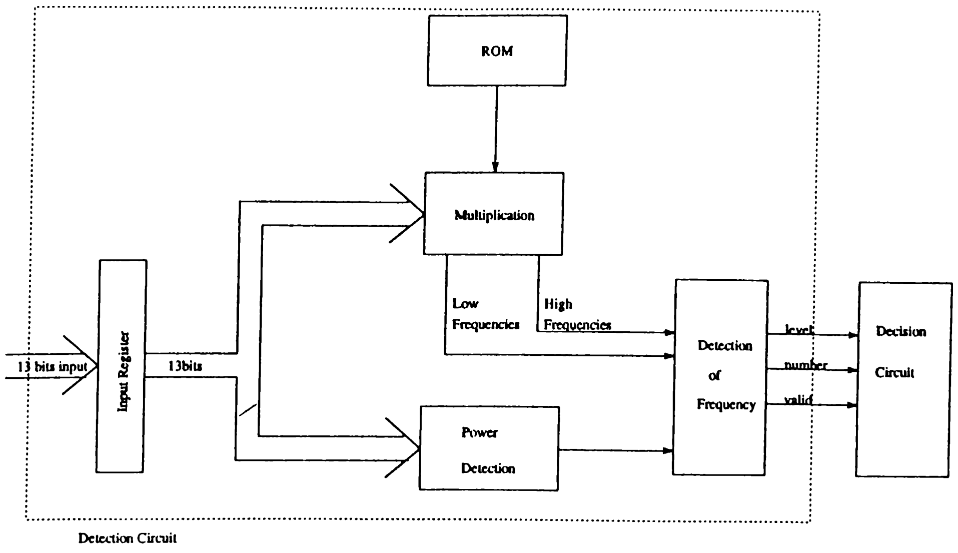


Figure 2.9: The receiver block diagram.

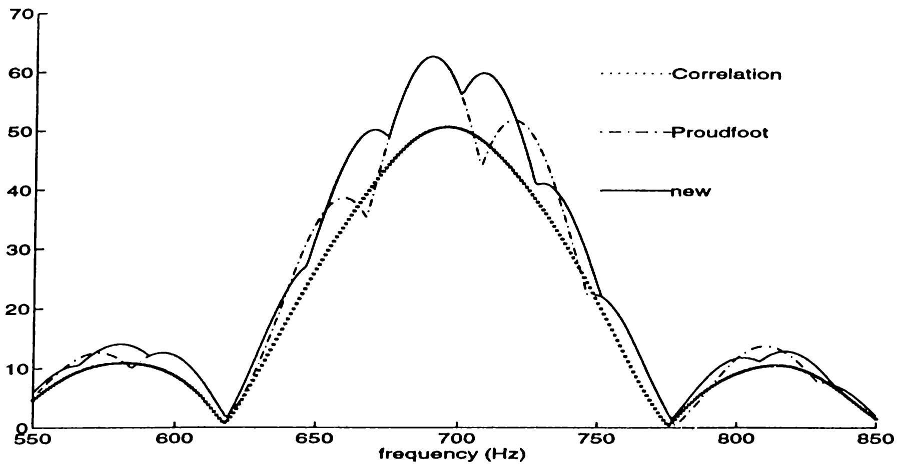


Figure 2.10: Frequency response for the different algorithms.

where t runs from $-49.5/8000$ to $49.5/8000$; there are total of 100 samples for each sinusoid.

In figures Fig. 2.11, Fig. 2.12, and Fig. 2.13 the input is

$$\sin(2\pi f t + \phi)$$

where the frequency runs from 550Hz to 850Hz while the phase term runs from 0 to π. In method proposed by A.D. Proudfoot, the input is correlated with the hardlimited versions of sinusoids.

$$\text{sign}(\sin(2\pi f_o t))$$

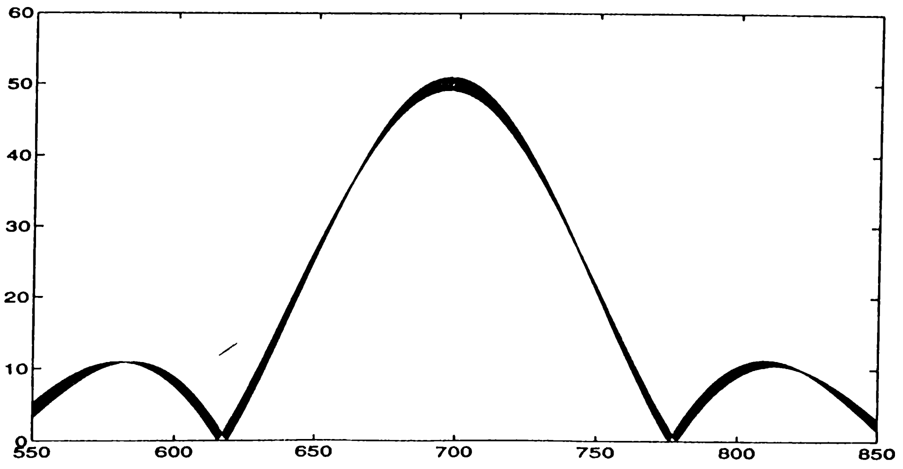


Figure 2.11: Frequency response for the conventional correlation receiver.

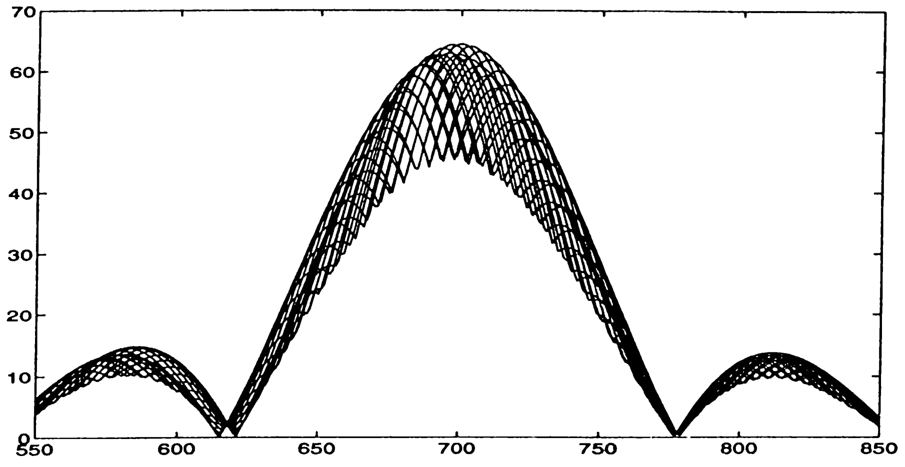


Figure 2.12: Frequency response for the algorithm proposed by Proudfoot.

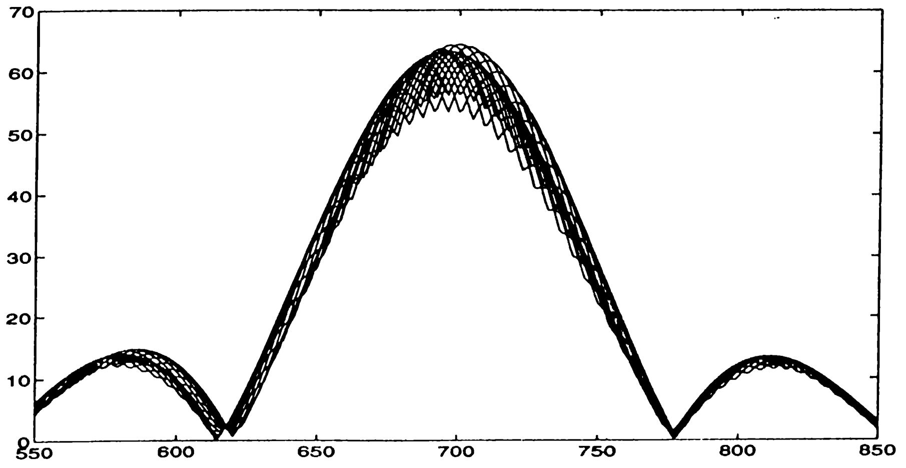


Figure 2.13: Frequency response for our algorithm.

and

$$\text{sign}(\cos(2\pi f_o t)).$$

In the method proposed in this thesis the input is correlated with the signals:

$$\text{sign}(\sin(2\pi f_o t))$$

$$\text{sign}(\sin(2\pi f_o t + \pi/3))$$

$$\text{sign}(\sin(2\pi f_o t + 2\pi/3)).$$

The second approach [10], using the binary signals for the correlation, is comparable to the approach in this thesis in area requirements. Their performances are compared in Fig. 2.14, Fig. 2.15, Fig. 2.16 and Fig. 2.17 where the outputs of the correlators of the two approaches are shown.

In Fig. 2.14 and Fig. 2.15 the responses to the input $\sin(2\pi ft + \phi)$; where f runs from 500 Hz to 1100 Hz, while the phase term ϕ varies from 0 to π, are shown. In each figure there are two graphs showing the upper and lower limits for the frequency response due to the changes in phase. The lengths of the correlation windows are comparable. The correlation window length is chosen to be 100 throughout the whole thesis. The length used for the simulations of the algorithm proposed in [10] is 128.

In Fig. 2.16 and Fig. 2.17 the responses of the same correlators to the input $\sin(2\pi ft + \phi)$, where this time f runs from 1100Hz to 1800Hz, are shown. The phase is again varying from 0 to π. It can be seen from the graphs that the sensitivity of the first correlator, which is proposed in this thesis, to high frequencies is less than the corresponding correlator.

2.3 How The System is Built

There are two main parts of the circuit (See figures Fig. 2.9, Fig. 2.18, and Fig. 2.19).

- Detection circuitry
- Decision circuitry

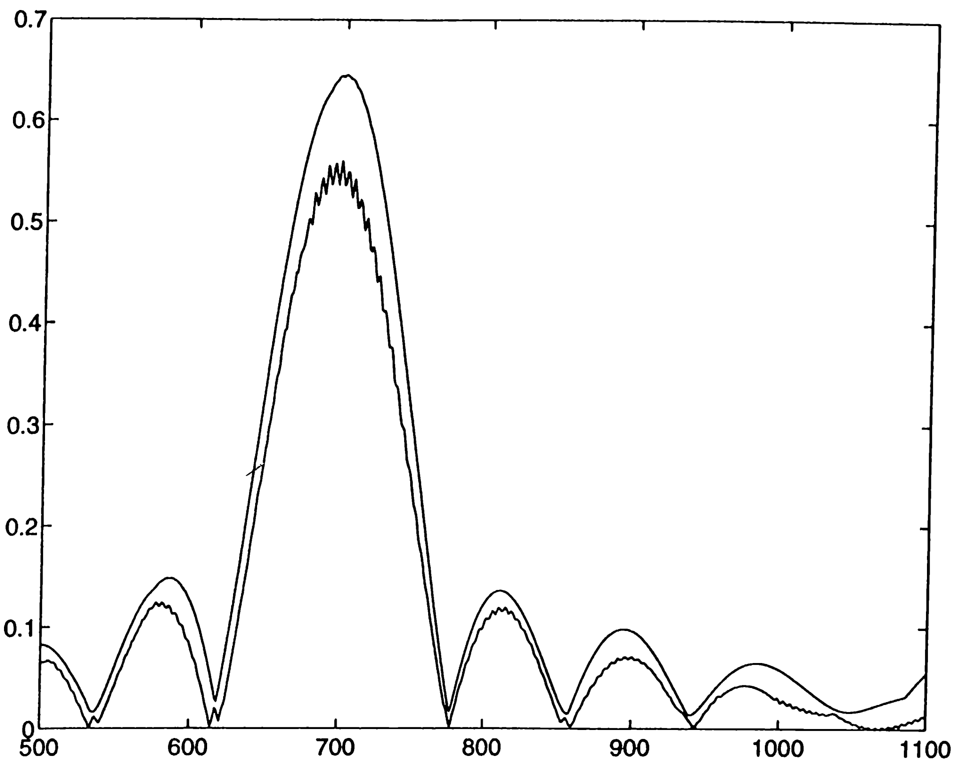


Figure 2.14: Output of the first correlator. (Low frequencies)

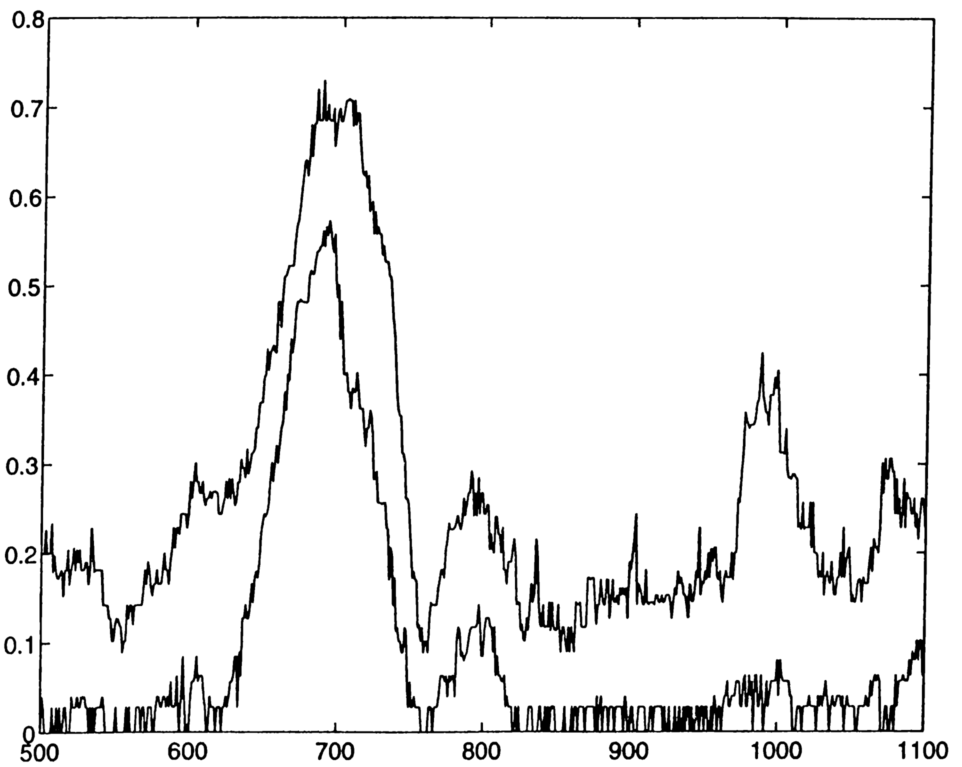


Figure 2.15: Output of the corresponding correlator in [10] for the same inputs.

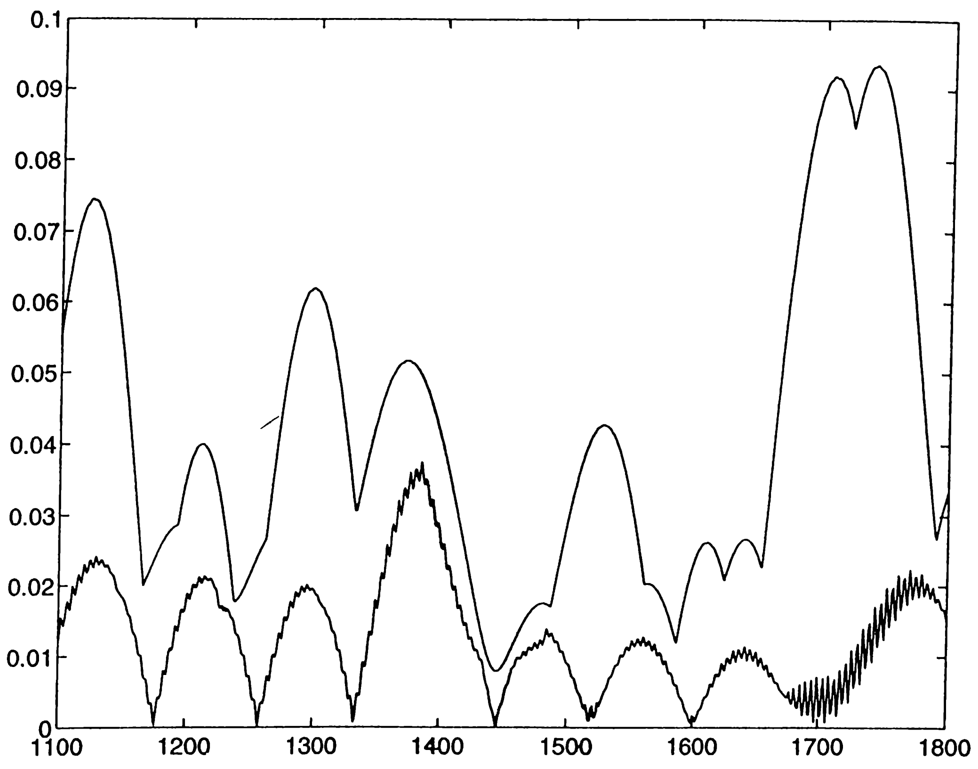


Figure 2.16: Output of the first correlator. (High frequencies)

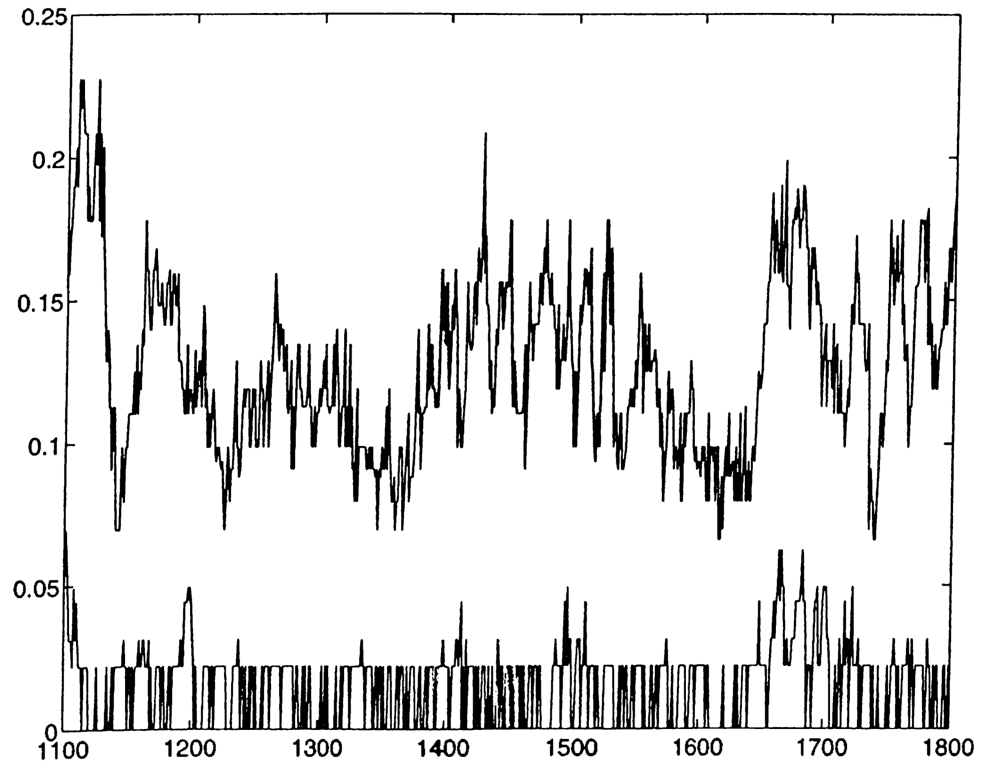


Figure 2.17: Output of the corresponding correlator in [10] for the same inputs.

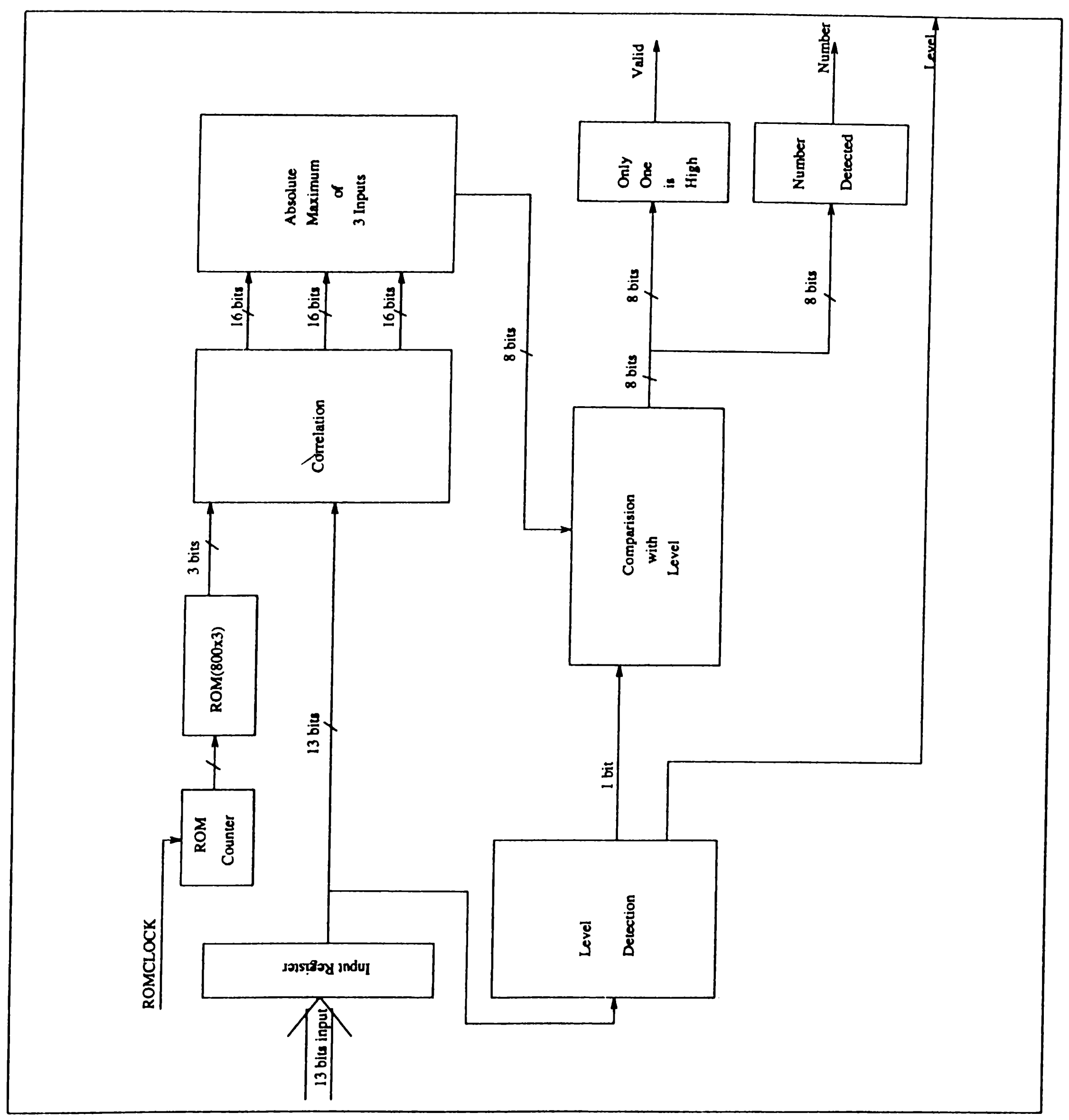


Figure 2.18: Frequency detection.

2.3.1 Detection Circuit

In this part of the circuit the frequency components of the input signal is multiplied with the data stored in ROM. ROM contains the sign-bits of sinusoids having the desired frequencies (697,770,852,941 and 1209,1336,1477,1633 Hz.). Three different phases for each frequency are used. The frequency values giving the maximum product is found for low group and for high group. This product and the power are proportional to the amplitude of the sinusoids. Detection block diagram is shown in Fig. 2.18.

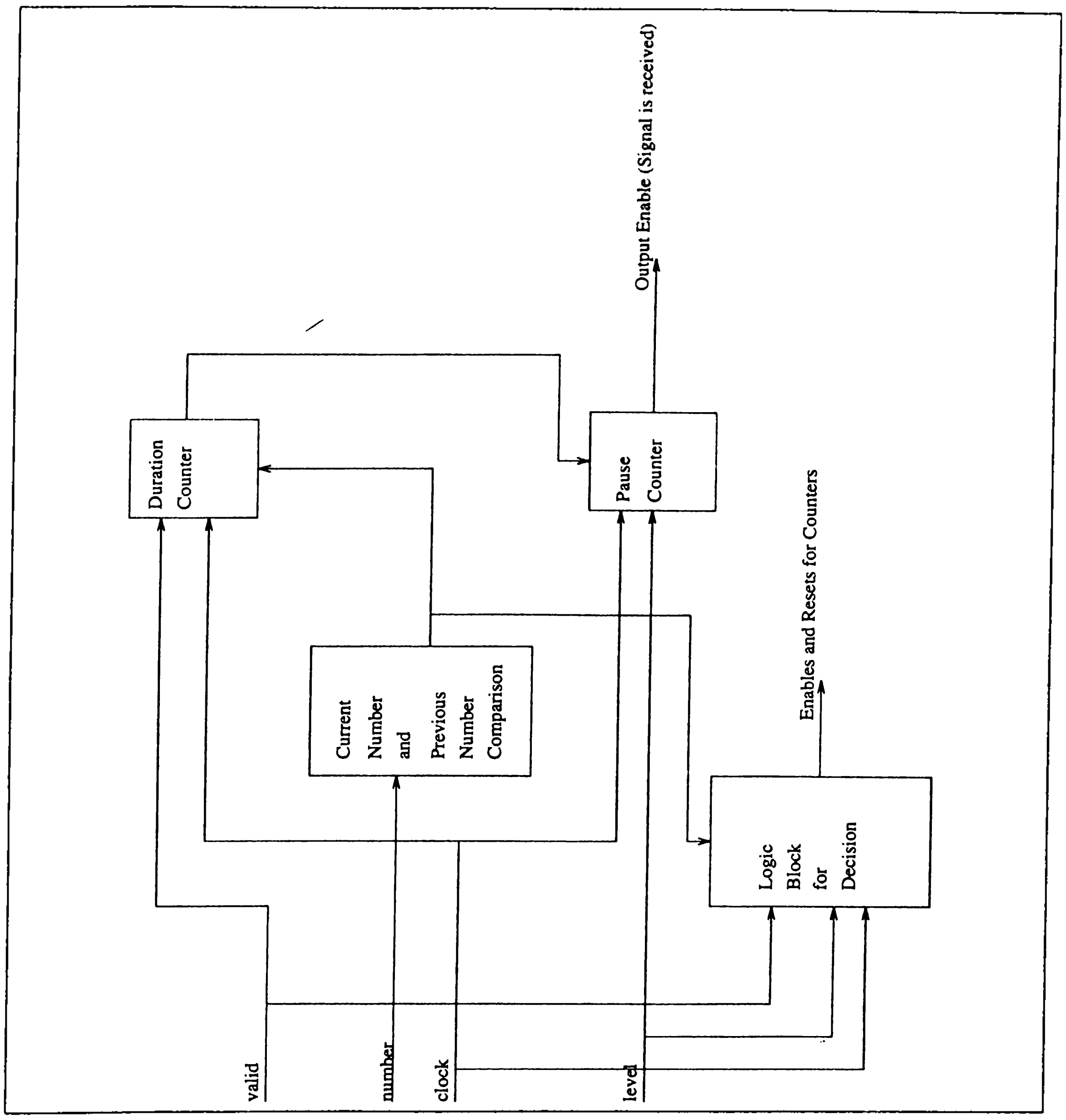

Output Enable (Signal is received)
Duration Counter
Pause Counter
Enables and Resets for Counters
Current Number and Previous Number Comparison
Logic Block for Decision
valid
number
clock
level

Figure 2.19: Decision Block Diagram.

2.3.2 Decision Circuit

In detection circuit the frequencies having the greatest components are found.In decision part the signal is decided to be correct tone or not. This is done using the properties of the valid tones. (See Table 2.2). The duration of tone should be at least 40 milliseconds, and a pause longer than 30 milliseconds should follow. Fig. 2.19 shows the decision block diagram.

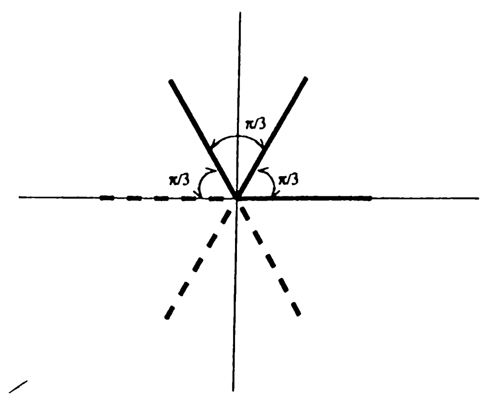


Figure 2.20: Phases

2.4 Algorithm

In this receiver the input stream is multiplied by 24 different bits stored in ROM (at each time step). As the multiplier is one-bit the multiplication is simple. Each product is added to the value at the output of the accumulator.

ROM contains 2400 bits (100 bits for each sinusoid). There are 3 sinusoids of each frequency:

$$\sin(2\pi f t)$$

$$\sin(2\pi f t + \pi/3)$$

$$\sin(2\pi f t + 2\pi/3)$$

The phase relations are shown in Fig. 2.20. Dashed lines show phases that are π greater and sines have negative sign. The next step is taking the absolute value of the accumulated values and finding maximum of the three outputs. Fig. 2.21

The result here is the maximum of

$$\sum_{m=0}^{n-1} \mathrm{x}(m) \cdot \mathrm{z}_1(m)$$

$$\sum_{m=0}^{n-1} \mathrm{x}(m) \cdot \mathrm{z}_1(m)$$

$$\sum_{m=0}^{n-1} \mathrm{x}(m) \cdot \mathrm{z}_1(m)$$

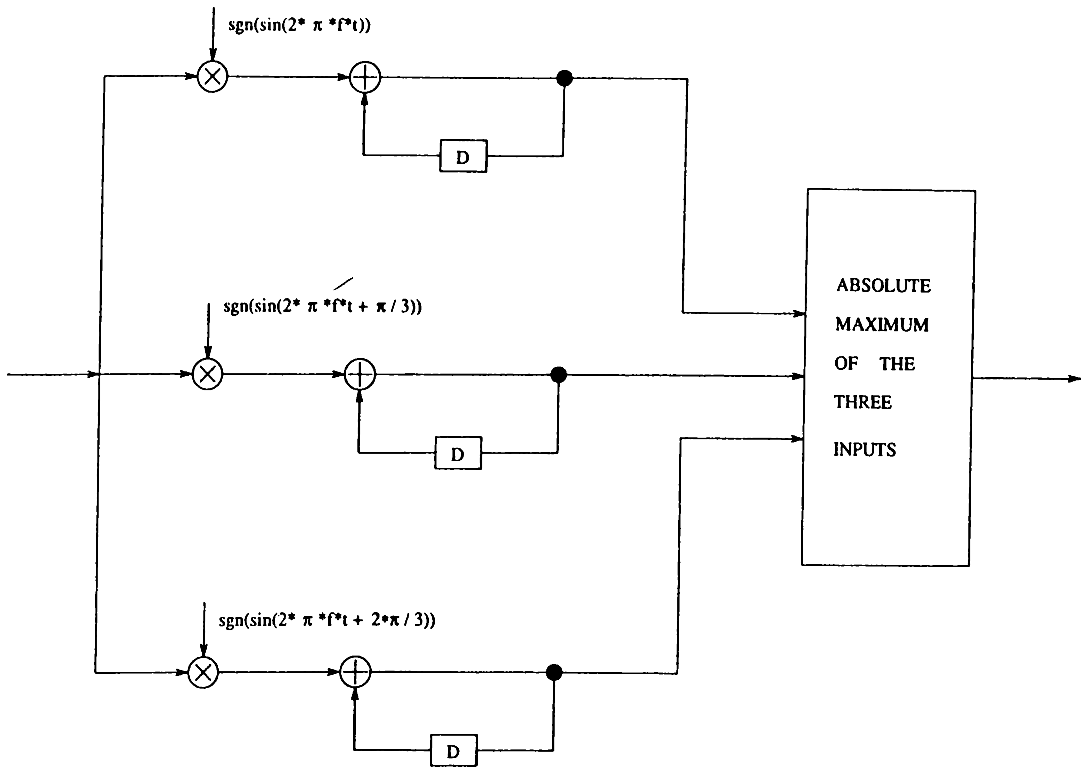

sgn(sin(2* π *f*t))
sgn(sin(2* π *f*t + π / 3))
sgn(sin(2* π *f*t + 2*π / 3))
D
D
D
ABSOLUTE MAXIMUM OF THE THREE INPUTS

Figure 2.21: Multiplication and accumulation

where $z_1(m), z_2(m), z_3(m)$ are the obtained by sampling sign(sin($2\pi ft$)) , sign(sin($2\pi ft + \frac{\pi}{3}$)) and sign(sin($2\pi ft + \frac{2\pi}{3}$)) respectively. n is the length of the stream to be summed up. x is the input.

The dependency of the result on the frequency is shown in section 4.1 and the dependency of the result on the phase of the input signal is presented in section 4.2.

Maximum value for absolute values of these three results is obtained. Then there are 8 sums to be processed; 4 for the high group of frequencies and 4 for the low group of frequencies. These sums are compared with the data coming from the power section. For a valid DTMF tone, only one of the absolute sums is greater than the power data for each group. If zero or more than one of the sums are greater than the power data, the signal is rejected. The data coming from the power section is a constant times the absolute sum of the input for the same 100 samples.

$$P = c \cdot \sum_{i=0}^{N-1} |x(i)| \qquad (2.9)$$

This approach is used in [10] for compensating the level difference between input and the signal to be correlated with.

For every 100 samples this process goes on. From table 2.2 it can be seen that the duration of a valid DTMF tone is more than 40 milliseconds which corresponds to 320 samples. Tones with duration smaller than 24 milliseconds (or 23 milliseconds) should be rejected and this corresponds to 192 samples. In this detection algorithm 320 means 3 valid signals for a valid tone. The minimum duration that can pass this 3 consecutive valid pulse specification is 250 samples. After 3 consecutive valid pulses, there should be 3 consecutive silence pulses (similar argument). When 3 valid pulses are followed by 3 silence pulses the DTMF tone is accepted.

Chapter 3

IMPLEMENTATION

The DTMF receiver proposed is implemented in ES2 CMOS 1μ technology using Cadence Environment, which makes possible to develop the design hierarchically. First the schematics of the implementation is drawn using standard cells, then the layout is obtained using standard place and route algorithms of the software. In this chapter the realization of the design is explained.

The first hierarchy level of the digital DTMF receiver includes the following subblocks:

- Detection of the Frequencies
- Decision

The cells used to implement the whole circuit are shown in tables 3.1 and 3.2.

The clock is chosen to be 1024 kHz.

3.1 Detection Circuit

The input is taken, it is correlated with the ROM data and a guess for the input frequency is done in the detection part.

cell name	includes	description
topp	topl and pads	top schematic with pads
topl	detectcell	top circuit
	declog2	
	multiplexors	
detectcell	testpow	detection part of the circuit
	eightzeros	
	corrcelnew	
	powlev	
	level2	
	fmaxdown16	
	outreg2	
	down8003	
	outnum	
	outreg	
	findmax	
	onlyone	
	equalreg	
	cmpwithlev	
	compclk	
	rom	
	comparat	
	down16	
	ckdiv8	
	inreg	
comparat	comp3	
comp3	abs2 maxof3	

Table 3.1: Cells in the hierarchy of the DTMF receiver circuit

cell name	included cells	description
rom	down800	rom
	rom3out	
corrcelnew	muxedregmat	Correlation
	corr	
	andarr	
muxedregmat	muxedreg	
powlev	testpow	Signal Level Detection
	comparator	
	levdown128	
	abssummer	
	levelreg	
comparator	compcell	
compcell	compcella	
	compcellb	
abssummer	count100	
	andarr	
	corr	
levelreg	div	
declog2	eff	Decision
	durcount2	
	intcount2	
	sysdata	
	cmp	

Table 3.2: Cells in the hierarchy of the DTMF receiver circuit

The input is latched into the register block *inreg*. 1MHz clock is divided by 128 and input is latched with the rate of 8 kHz. The clocking blocks for this division are *down16* and *ckdiv8*. First one divides by 16, and second divides by 8. 1 MHz clock is for shifting data serially.

Then the output of the register is fed to level detector and correlators.

3.1.1 Correlators

Actually there are 24 adders needed for correlators. Instead of using 24 adders, 3 adders are used and the rate of the input taken is 8 times larger. Here, three adders are preferred to a single adder with an input rate 24 times larger because the available clocks have the rate of a power of 2.

Input is multiplied by the bit coming from the ROM, the result is added to the output of a register matrix and result is shifted into the same matrix again. (Block diagram is in Fig. 3.1). Schematic of the correlator is in Appendix B.1

Each correlation block has one 20-bit adder and a register matrix for storing the sums (20 bits by 8). 3 correlation blocks performs the correlation with 3 different phases respectively. (And can be expanded to 4 or more with small changes in the next stage).

Corrcelnew is the hierarchy name for the correlation block containing the correlator,register matrix -*muxedregmat*-, and *andarr* for masking the output. This masking is needed to clear the sum for every 100 samples. This masking signal is obtained from a synchronization pulse generated at the ROM block.

3.1.2 ROM

Rom is the section for generating the sign bits of sinusoids $\sin(2\pi ft)$, $\sin(2\pi ft + \pi/3)$ and $\sin(2\pi ft + 2\pi/3)$. This block is generated with the synthesis tool of Cadence. At every time step three sign bits are generated and the frequency is changed from the highest one to the lowest one. The time step here is 1/8 of the input period.i.e. rate is 8 times the input frequency. The 1 MHz clock is divided by 16 and the clock for the ROM is generated. The same clock is used to shift the contents of the register matrix.

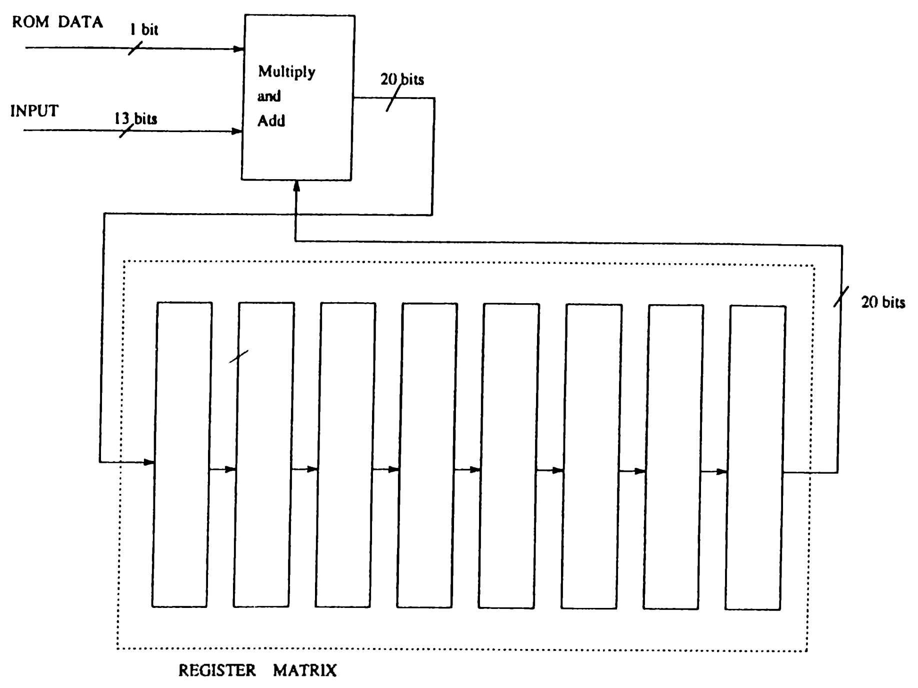


Figure 3.1: The correlator block diagram.

3.1.3 Level Detection

The schematics related to level detection are in Appendix B.1. It sums the absolute value of the input data for every period of 100 samples. The absolute sum is calculated with a similar way to the correlation. The same input is taken but here the sign bit of the signal is used as multiplier instead of the bit coming from the ROM. i.e., if the input data is positive it is added to the sum; otherwise it is subtracted from the sum.

In the comparator block the level is compared to the smallest allowable level. It is a 20 bit comparator and it is an entirely combinational circuit. In figures 3.2, 3.3 and 3.4 there is a 2-bit comparator. 20-bit comparator is obtained by cascading this cell. If A is high output *Cout* is logic-1; if B is large output *Dout* is logic-1 else both outputs are logic-0.

Levdown128 is the frequency divider. As its name indicates it divides by 128. (used clock is 8 kHz).

Levelreg block multiplies the power value by a constant which can be selected by choosing S1 and S0. (Table 3.3) This multiplication is done by choosing 1/4 or 1/2 or 1/8 of the input value and adding to another one. This

S0	S1	multiplier
0	0	3/8
0	1	1/2
1	0	5/8
1	1	3/4

Table 3.3: The multiplier

result is converted into serial data. Only the first 16 bits are used.

3.1.4 Comparison of three phases

This comparison is performed serially. As it can be seen in the Fig. 3.5 first the absolute value of the 3 numbers are calculated. If the sign bit is positive the serial data appear at the output, otherwise the inverse of every bit exists at the output serially. The calculated value is 1 lower than the absolute value of the number for negative inputs. But this was preferred for the sake of smaller area. The positive numbers are then latched into the *maxof3* subblock. The first two serial inputs are compared serially in the first clock period then the larger one is compared with the other one in the next clock period.

The results are compared to the serial output coming from the level detection block at the *cmpwithlev* block. This performs the same algorithm with the previous comparator. (Fig. 3.7)

3.2 Decision Circuit

In this section of the circuit it is decided that if the input stream defines a DTMF signal or not. The implemented schematic of Fig. 2.19 is in Fig. 3.8

There are 4 important blocks in the decision part.

- Duration counter: *durcount*
- Interval counter: *intcount*

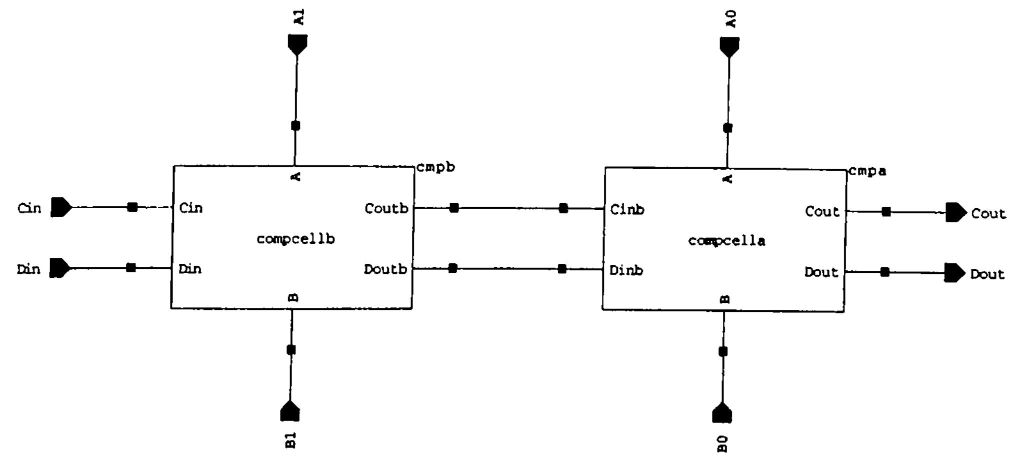


Figure 3.2: two-bit comparator

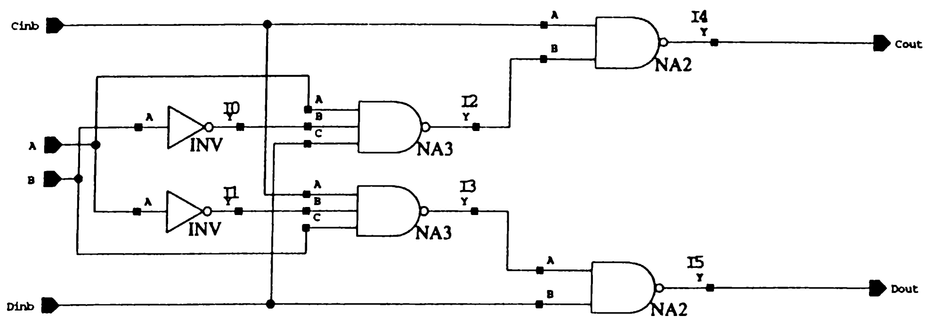


Figure 3.3: subblock compcella.

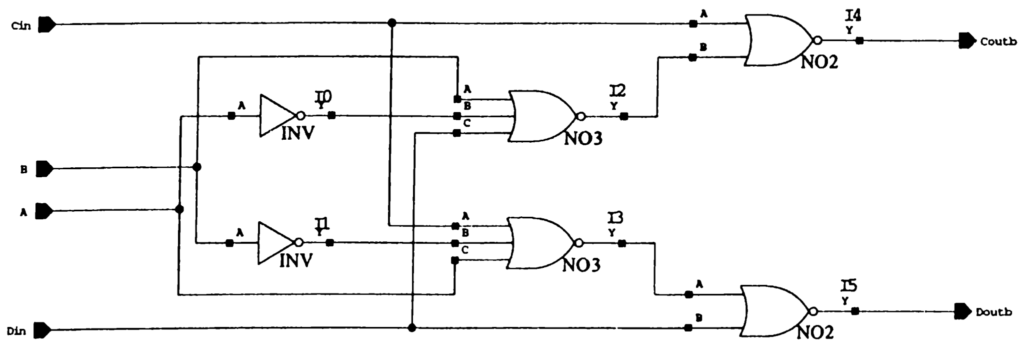


Figure 3.4: subblock compcellb.

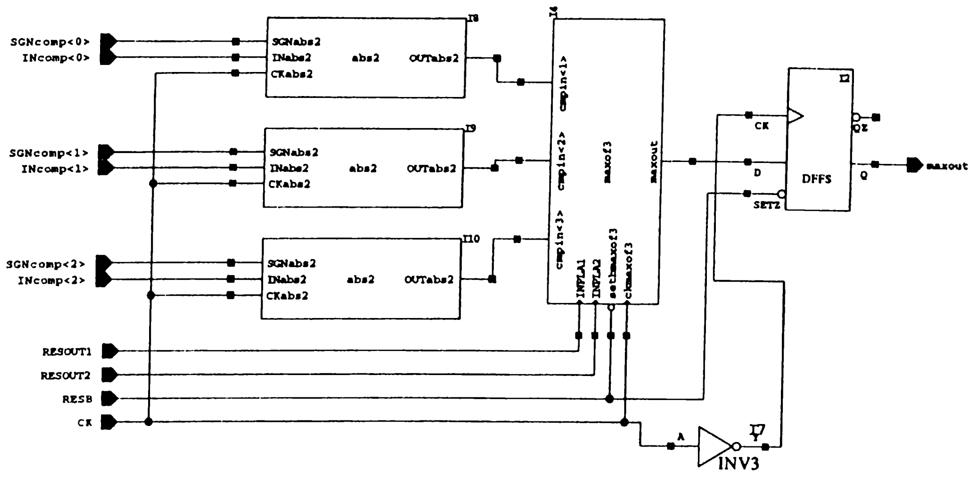


Figure 3.5: The absolute comparison of three inputs.

- Comparison of the previous and current numbers: *cmp*
- Output converter into system data :*sysdata*

3.2.1 Duration and Interval Counter

Duration counter starts counting when the previous and the current numbers are equal. This means that the signal with the same frequency components is continuing. Otherwise the counter is always zero. It stops counting when counter reaches 2. (That is three samples are the same).

Interval counter starts counting when the duration counter is 2 and the signal level is low.(i.e. pause). It counts from 1 up to 4 and stops at 4. when signal level becomes high it returns back to its initial value 1.

3.2.2 Comparison

This block compares the previous and the current numbers. This comparison is performed by a simple subtraction. If all the bits become zero, the output changes from 0 to 1.

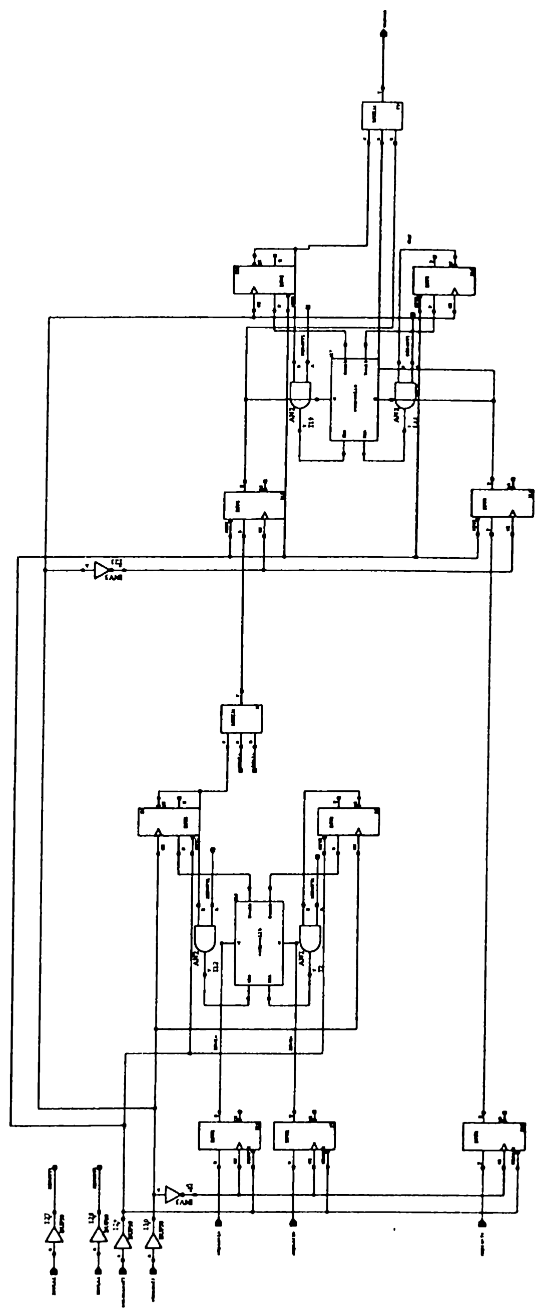


Figure 3.6: Circuit giving the maximum of three inputs.

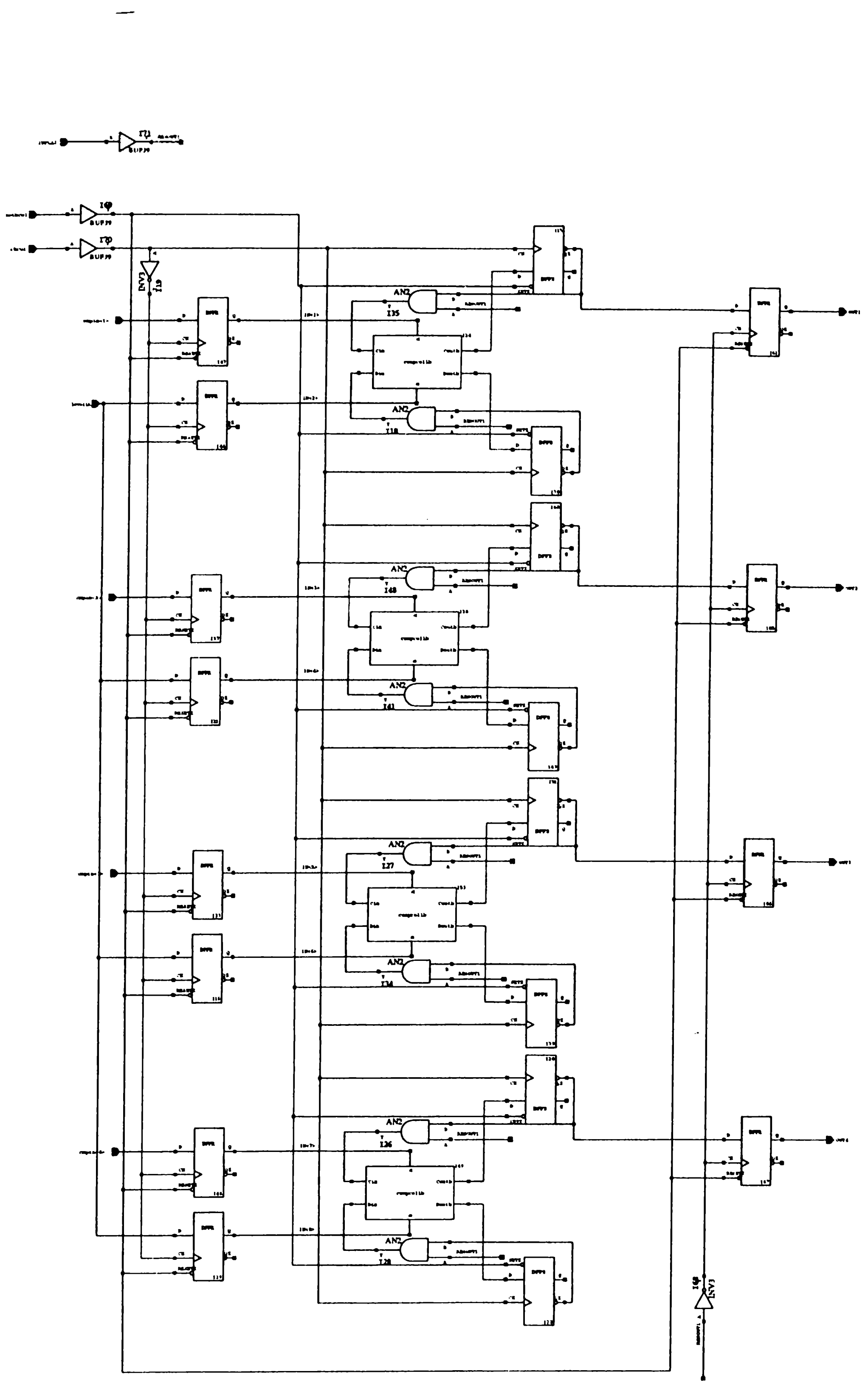


Figure 3.7: Comparison with level.

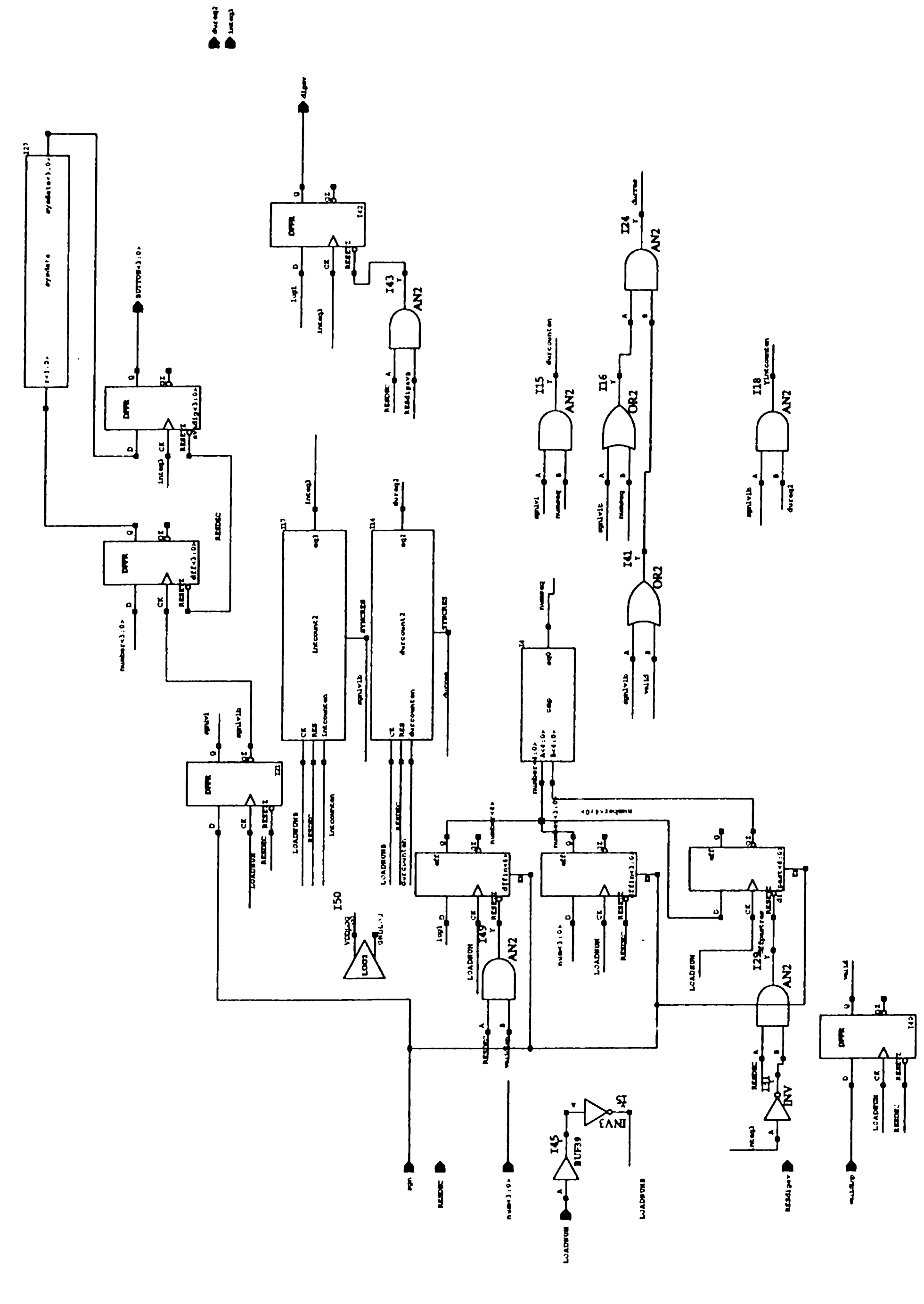


Figure 3.8: Decision circuit schematic

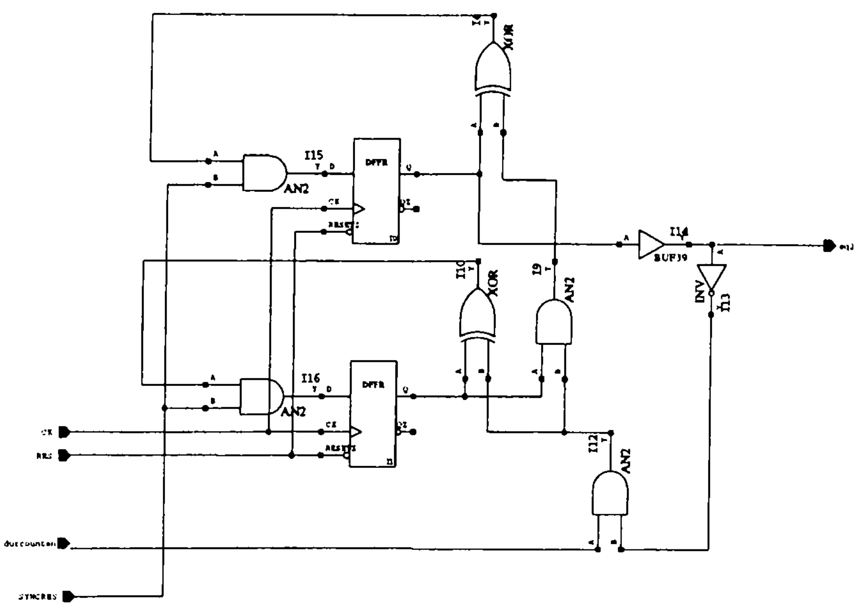


Figure 3.9: The duration counter.

3.2.3 System Data

This block converts the decided number into a form that the microprocessor will understand. This subblock is generated by the help of the Synthesis tool of Cadence. Input output relation can be seen in Table 3.4. At the input side the leftmost two bits represent the low frequencies and the rightmost two bits represent the high frequencies.

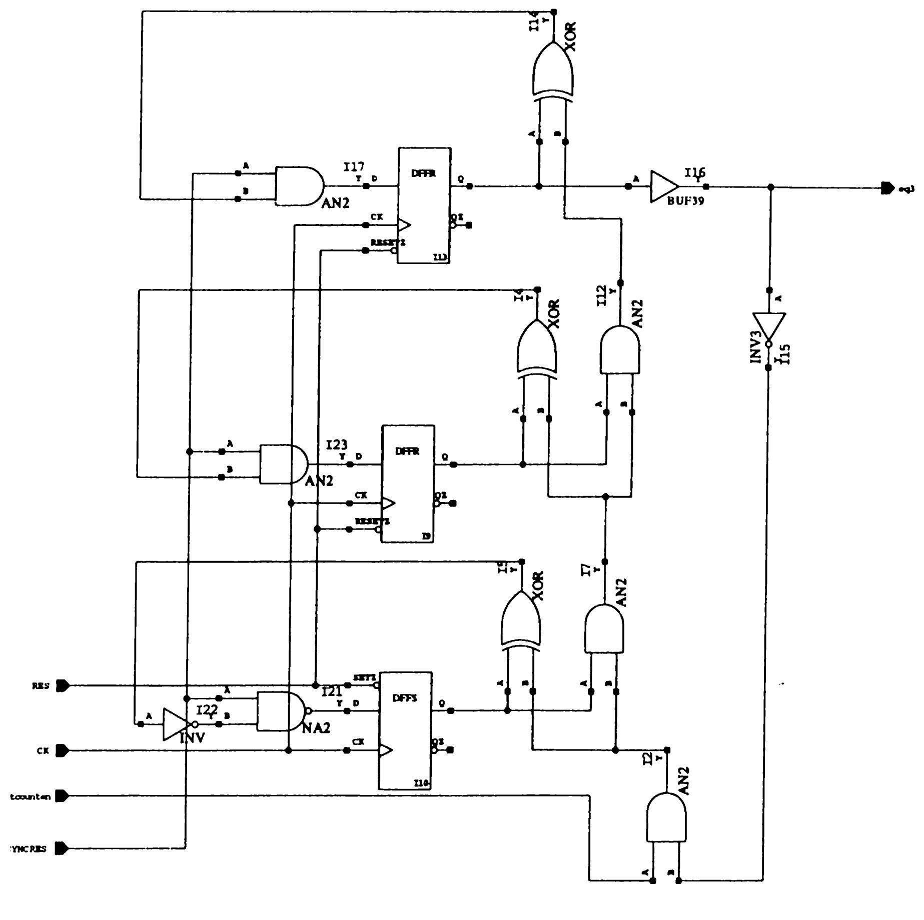


Figure 3.10: The interval counter.

button	in	out
1	0000	0001
2	0001	0010
3	0010	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	1000	0111
8	1001	1000
9	1010	1001
0	1101	1010
$*$	1100	1011
#	1110	1100
A	0011	1101
B	0111	1110
C	1011	1111
D	1111	0000

Table 3.4: Conversion Map

Chapter 4

SIMULATION RESULTS AND TESTING THE IMPLEMENTATION

In the design step, the frequency and phase dependence are examined by using Matlab simulations. Then the decision logic is developed by using C-language. In this chapter, the results of these simulations are explained and the additions for test purposes are described.

4.1 Frequency Dependence

Actually there are 24 filters used here, each of which is tuned to a different frequency. Also every correlator has different responses to the inputs having different phases. In Fig. 4.1, the outputs of the correlations with

$$\mathrm{sign}(\sin(2\pi 697t)$$

$$\mathrm{sign}(\sin(2\pi 697t + \pi/3)$$

$$\mathrm{sign}(\sin(2\pi 697t + 2\pi/3)$$

are shown. Here the input is $\cos(2\pi ft)$ where the input frequency f runs from 500Hz to 1000Hz. The next figure (Fig. 4.2) shows absolute values of the same outputs for the same input. The second graph in the same figure shows the

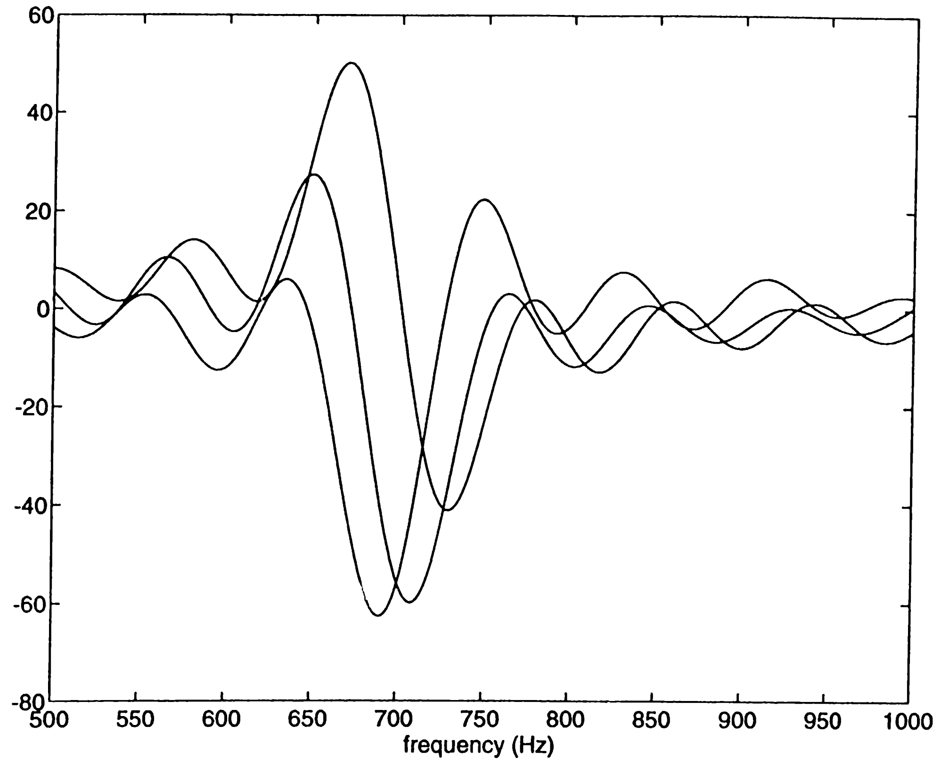


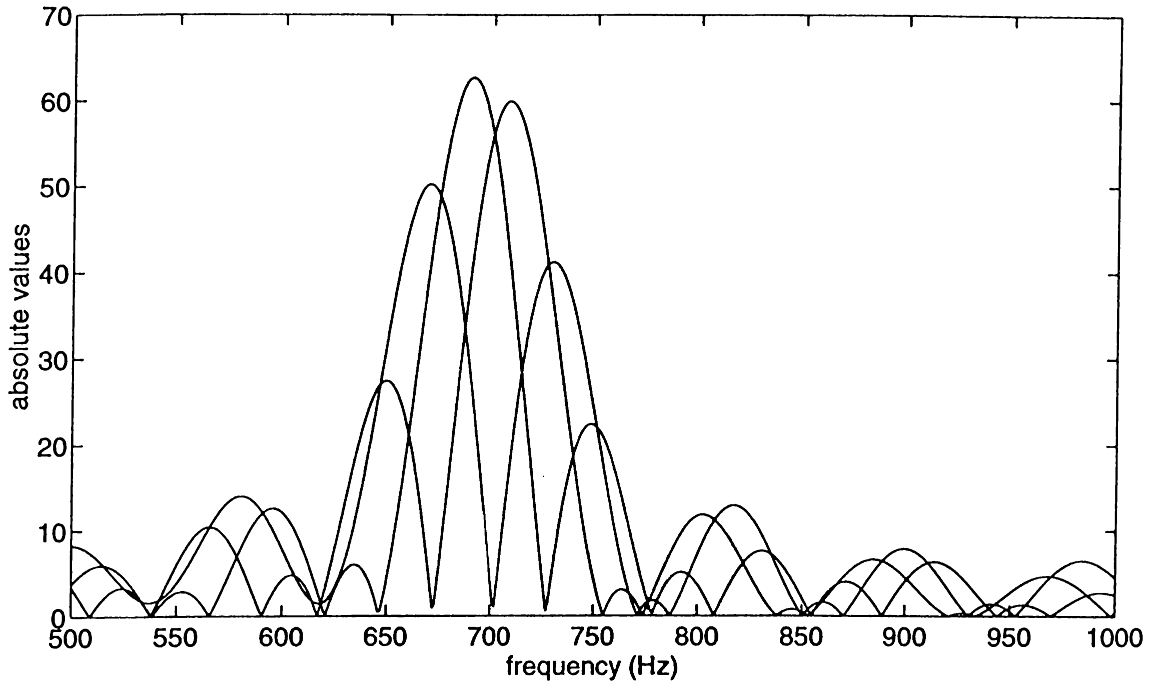
Figure 4.1: Frequency dependence of the first three correlators

absolute values of the same correlator outputs for input signals $\cos(2\pi ft + \phi)$ where f runs from 500Hz to 800Hz and ϕ runs from 0 to π.

In Fig. 4.3 the outputs of all the correlators are shown. The peaks correspond to the correlators tuned to the frequencies belonging to the low group. (Centers are located at 697,770 852,952 Hz). In this figure, the input frequency runs from 550 to 1100 Hz. The larger values (peaks) are the responses of the correlator nearest to the input frequency. In Fig. 4.4 the outputs of all the correlators are shown. The peaks correspond to the correlators tuned to the frequencies belonging to the high group. (Centers are located at 1209, 1336, 1477, 1633 Hz). The input frequency runs from 1100 to 1800 Hz. The shaded regions are because of the different phases (i.e. 0 to π) used at the input.

In Fig. 4.5 output of the absolute maximum for the input $\cos(2\pi ft + \phi)$. ($f = 1150..1700$ Hz and $\phi = 0..\pi$) is shown.

The next figure (Fig. 4.6 shows the frequencies that should be rejected or accepted. The regions between rejected and accepted are the tolerance regions which include the frequencies that can be either accepted or rejected.(It is



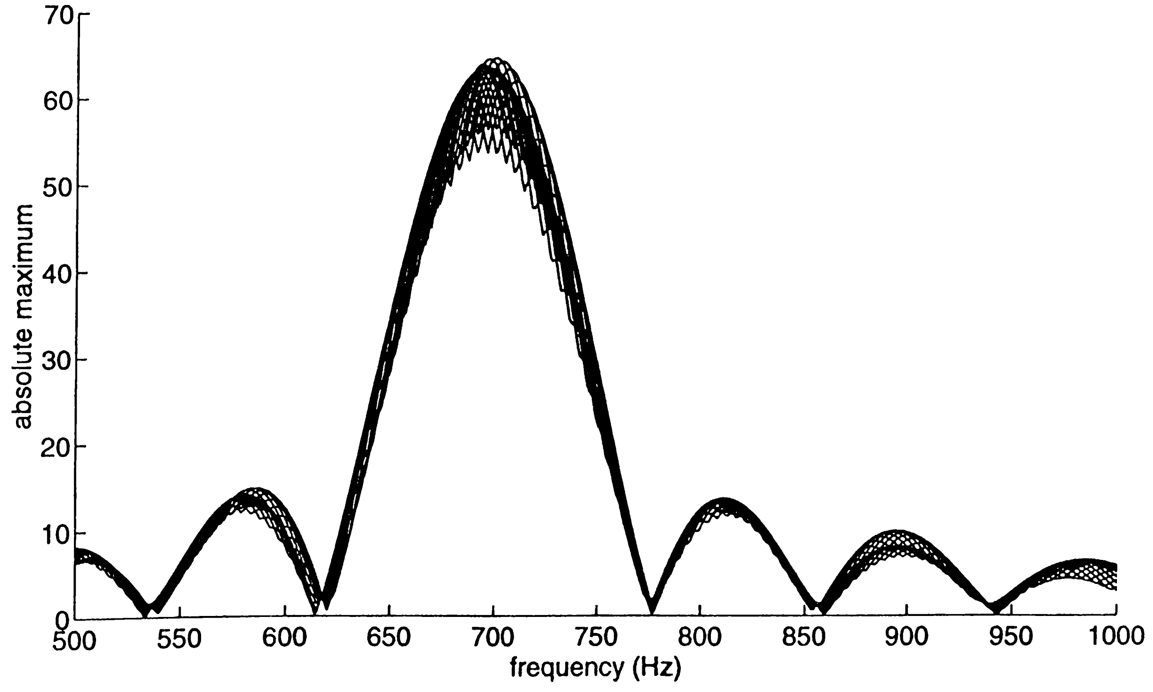


Figure 4.2: Frequency dependence of the first three correlators

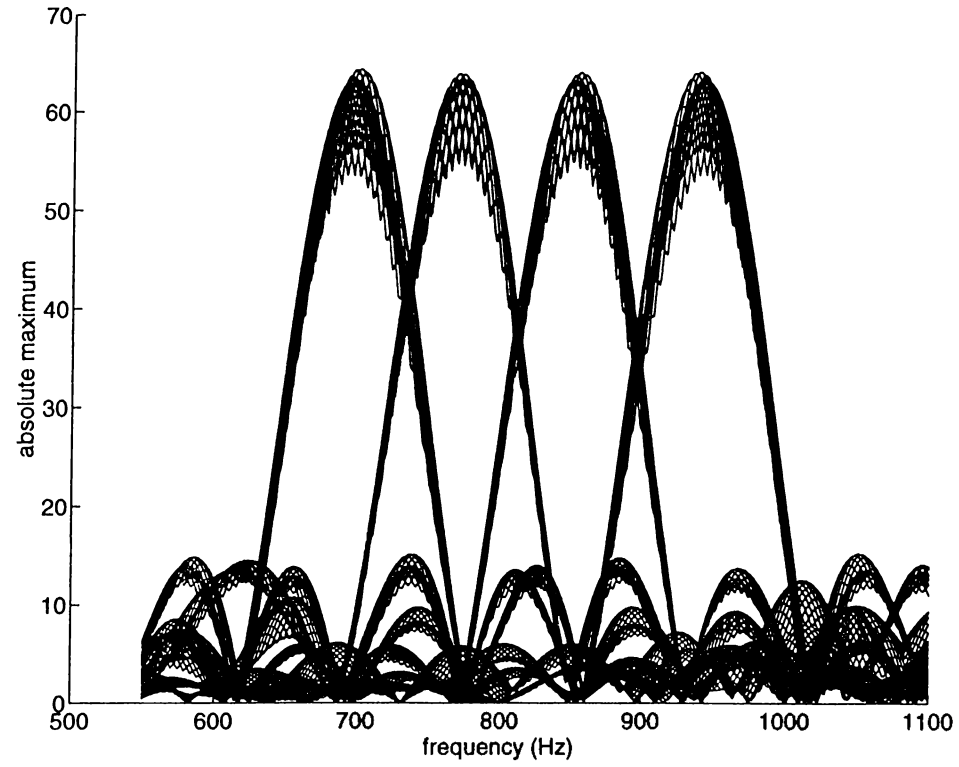


Figure 4.3: Outputs after the absolute maximum (low group)

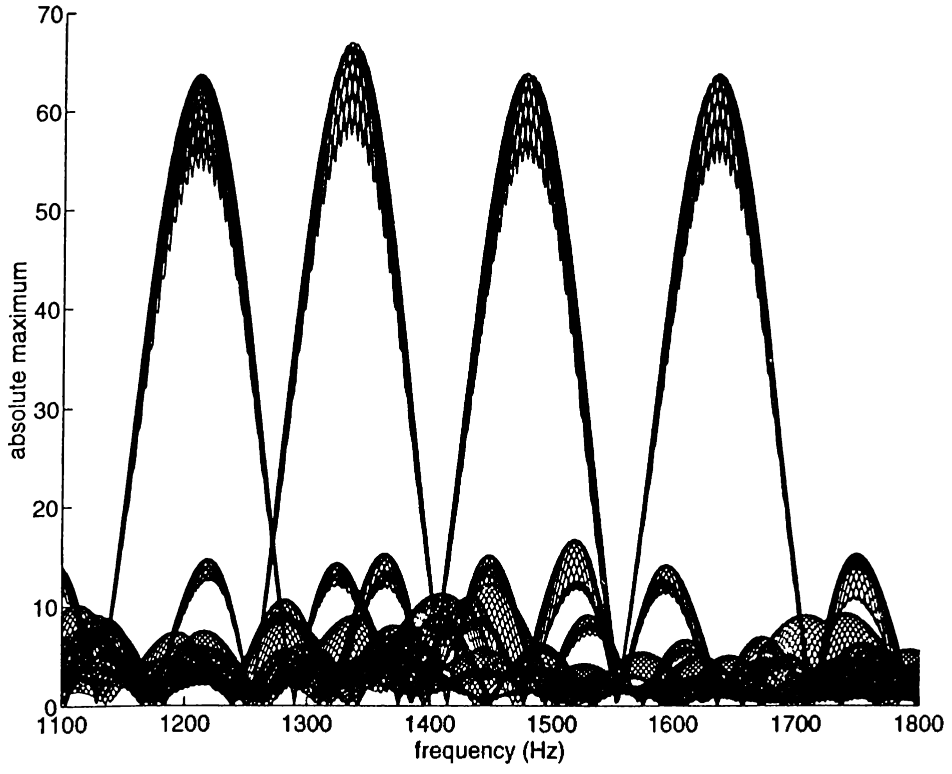


Figure 4.4: Outputs after the absolute maximum (high group)

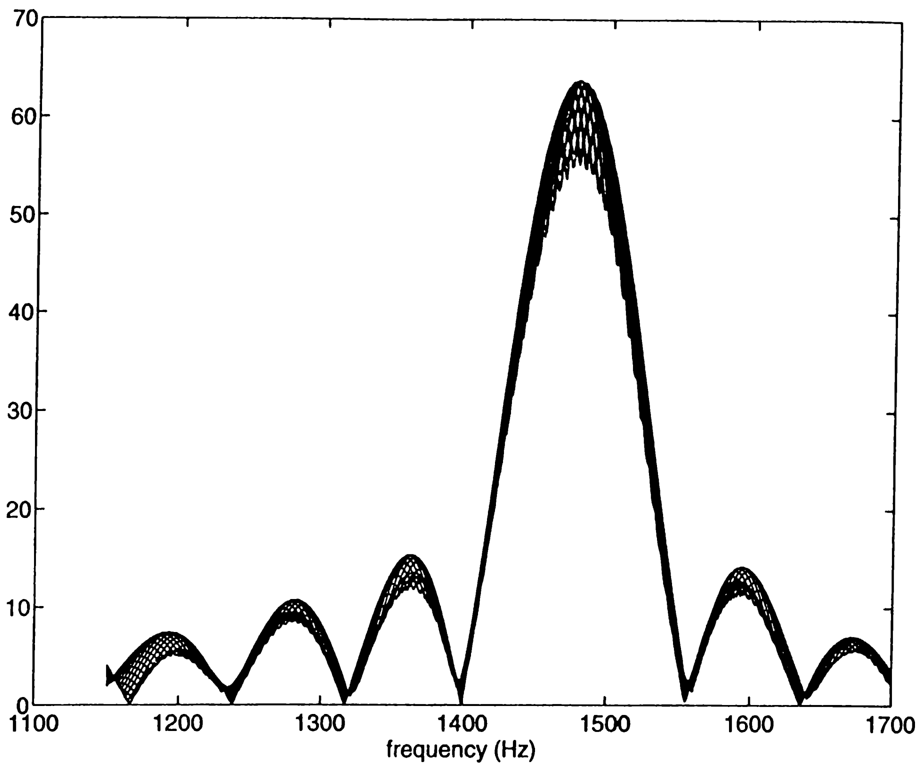


Figure 4.5: Frequency dependence of the 7'th three correlators

similar for the other correlator outputs)

4.2 Phase Dependence

Because 3 correlators with 60 degrees phase difference are used, the phase difference of the input with one of the correlating signals are 30 degrees at most. Because of this the width of the shaded regions in the previous figures is narrower than with correlations with sine and cosine.($\pi/2$ phase difference). Figure Fig. 4.7 shows the phase dependence of the output at the 6th correlation output.

4.3 Simulation Results

After simulations it is seen that the receiver satisfies the specifications of CCITT. Some simulations for the frequency tolerances are made by the help

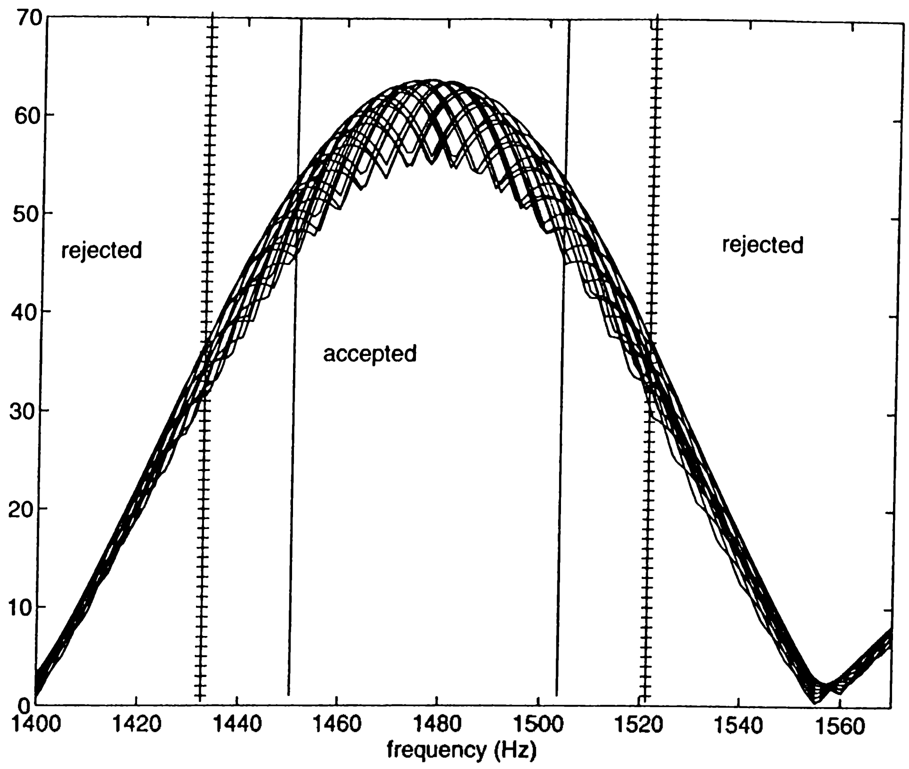


Figure 4.6: Frequency dependence of the seventh three correlators

of C programming language, and MATLAB. The results of these simulations are given in Table 4.1.

In Table 4.1, it can be seen that the frequency tolerance for low group frequencies is group frequencies it is responses are more steep for the high group.

4.4 Testing the Design

Testing is the problem of determining, in a cost effective way, whether a chip, module, board or system has been manufactured correctly. Increasing circuit complexities increase the cost of testing. To reduce this cost a collection of techniques known as "Design For Testability (DFT)" is carried on (Making internal nodes in a circuit more accessible; transforming sequential circuits into combinational circuits, or decomposing complex circuits into less complex sub-functions for the purposes of testing reducing the amount of test data which is needed to test the circuit). The main thing is to insert a reset into all

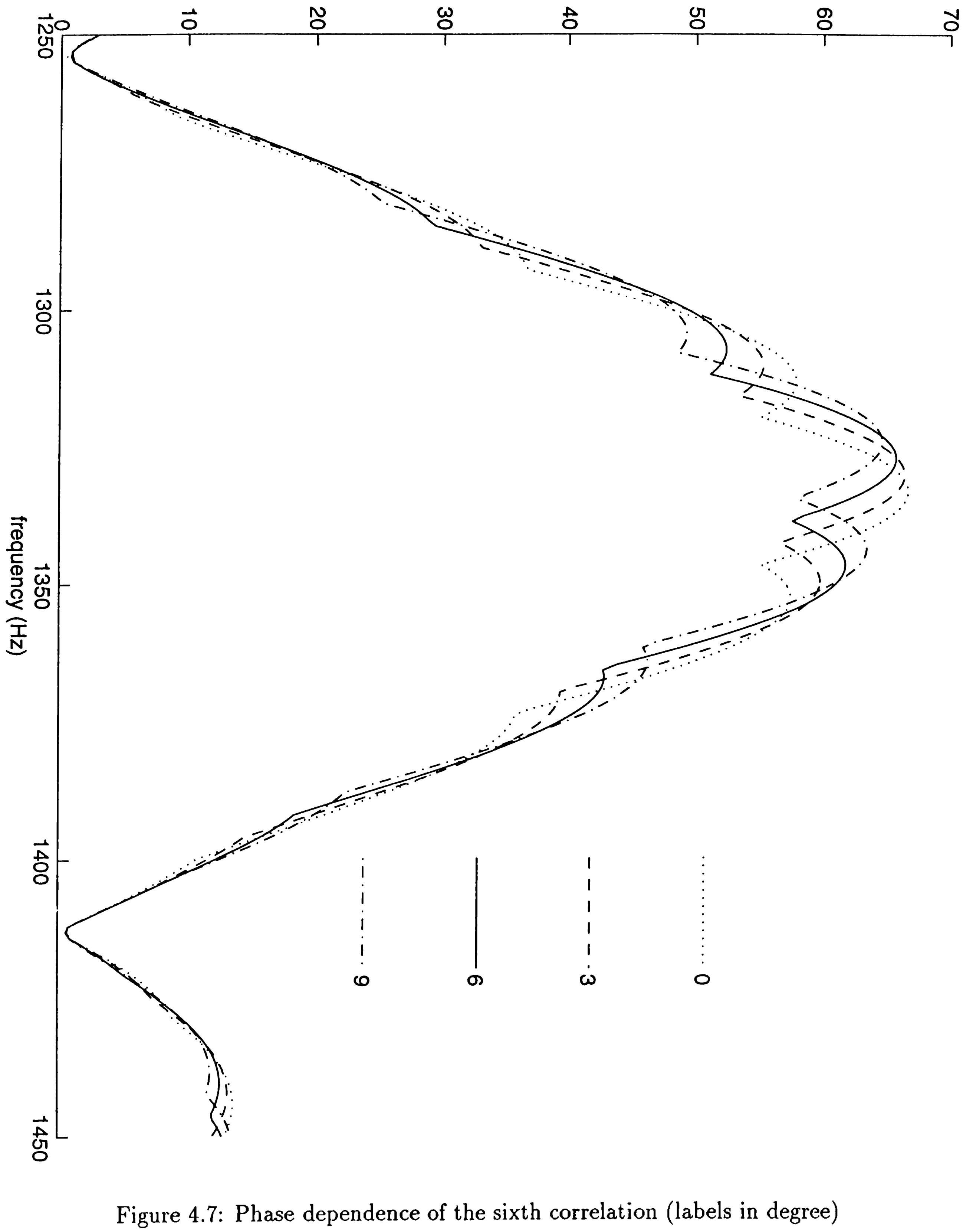


Figure 4.7: Phase dependence of the sixth correlation (labels in degree)

high frequency tolerance \ low frequency tolerance	-%5	-%3	-%2	exact	%2	%3	%5
-%5	ND	ND	ND	ND	ND	ND	ND
-%3	ND	ND	ND	ND	ND	ND	ND
-%2	ND	D	D	D	D	D	ND
exact	ND	D	D	D	D	D	ND
%2	ND	D	D	D	D	D	ND
%3	ND	ND	ND	ND	ND	ND	ND
%5	ND	ND	ND	ND	ND	ND	ND

Table 4.1: Response of the system for different input freq uencies. (ND : not detected; D : detected).

digital systems, this has the advantage of starting all states from known values. In this chip, in order to function properly the system should begin from zero, and all internal nodes are set to a value.

This chip will be in fact used as a part of a telecommunication chip. The full testing circuitry has not been designed yet. But this chip contains some separate circuits for testing. For level detection *testpow* is designed. It takes the value, converts it into serial data and shifts it to the output.

Other outputs of the subblocks are also multiplexed and connected to the output pads (Fig. 4.8). This is for making more nodes observable at the output.

For selecting the bits at the multiplexer inputs 3 select signals are introduced. *S0Test*, *S1Test* and *S2Test*. These select signals choose 4 of 24 inputs including level, level data, comparator outputs, correlations, ROM outputs, valid and level signals, counter outputs and clocks generated.

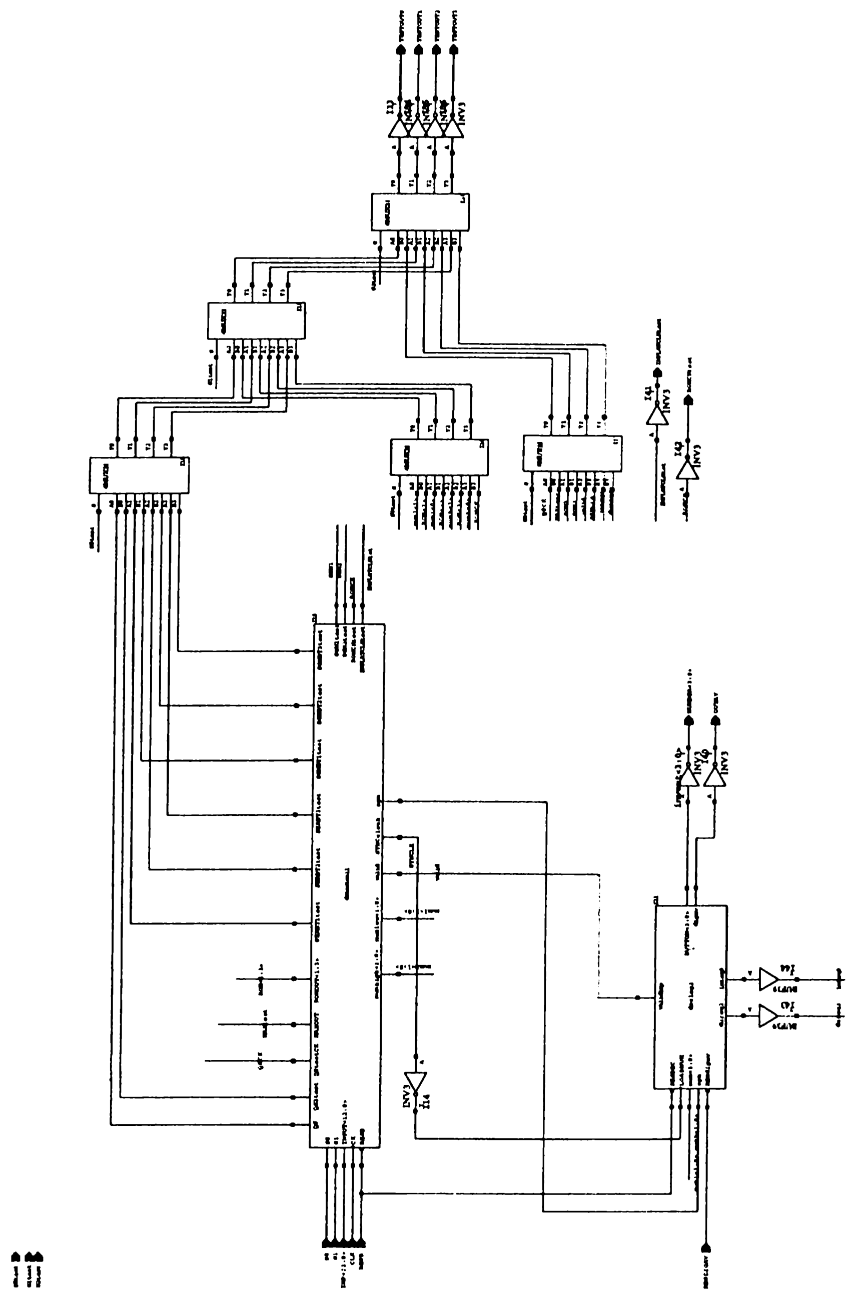


Figure 4.8: top schematic including the multiplexors for test

Chapter 5

CONCLUSIONS

In this thesis, a new implementation of a single chip digital DTMF receiver is explained.

Different from usual correlation receivers using 2 correlations with the functions having $\pi/2$ phase difference, 3 correlations with $\pi/3$ phase difference is used. This has been done in order to reduce the errors coming from taking maximum instead of squaring and adding. These errors make it difficult to choose a threshold. The maxima of the results of these correlations are found instead of squaring and adding to decrease the area requirements and circuit complexity. Also, an approximate power detector is added to decrease false alarm rate.

The correlation part of the detector can be used in other detectors as it does not include any multiplication which increases the complexity. The implementation is simpler as well.

The simulations of design are done by using Matlab and C-programming language. Design is implemented in Cadence -Design Framework II- environment and simulated by Verilog simulator.

The designed chip is in ES2 1-μm CMOS technology. It is 4.802 mm by 5.161 mm having a total area about 24.78mm^2. The registers and correlators require the largest area, 6.2mm^2, where core area is 16.3mm^2. It has 40 pins including 4 ground and 4 supply pins. It contains 3787 cells and 58259 equivalent transistors.

REFERENCES

[1] Martin P. Clark. *Networks and Telecommmunications.* John Wiley & Sons, Inc., 1991.

[2] A. Michael Noll. *Introduction to Telephones and Telephone Systems.* Artech House, 1986.

[3] William Sinnema and Tom McGovern. *Digital, Analog, and Data Communication.* Prentice Hall Inc., second edition, 1986. Chapter 2 and 3.

[4] George Gara Ivan Koval "Digital MF receiver using discrete fourier transform," *IEEE Transactions on Communications*, vol. COM-21, December 1973.

[5] JR. Michael J. Callahan "Integrated dtmf receiver," *IEEE Journal of Solid-State Circuits*, vol. SC-14, pp. 85–90, February 1979.

[6] George F. Landsburg Bertram J. White, Gordon M. Jacobs "Monolithic dual tone multifrequency receiver," *IEEE Journal of Solid-State Circuits*, vol. SC-14, pp. 991–997, December 1979.

[7] J. Tow J. R. Boddie, N. Sachs "Receiver for TOUCH-TONE service," *The Bell System Technical Journal*, vol. 60, pp. 1573–1583, September 1981.

[8] Fritz G. Braun "Nonrecursive digital filters for detecting multifrequency code signals," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-23, pp. 250–256, June 1975.

[9] J.N. Denenberg "Spectral moment estimator: A new approach to tone detection," *The Bell System Technical Journal*, vol. 55, pp. 143–155, February 1976.

[10] Theo A. C. M. Claasen and J. B. H. Peek "A digital receiver for tone detection applications," *IEEE Transactions on Communications*, vol. COM-24, December 1976.

[11] David Agnew "Simplified tone detector for pcm channel," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-31, pp. 8–16, February 1983.

[12] A.D. Proudfoot "Simple multifrequency-tone detector," *Electronic Letters*, vol. 8, pp. 524–525, October 1972.

[13] Pieter Eijkhoff. *System identification : parameter and state estimation.* Wiley-Interscience, 1974. section 8.3, pages 299–302.

[14] Yoshihito Haneda Yoshiaki Tadokoro "Dual tone multifrequency receiver using synchronous additions and subtractions," *IEEE Transactions on Communications*, vol. COM 35, pp. 414–418, April 1987.

APPENDIX A

SOURCE FILES

A.1 Algorithm Implemented

```
#include <stdio.h>
#include <math.h>
#define max(a,b) ((a)>(b) ? (a) : (b))
#define max3(a,b,c) max(a,max(b,c))
#define NUM 100
#include </manisah/ee/ekinci/dtmfrec/cfiles/coralg/save.h>
char digit[4][4];
int    sign(x)
double    x;
{
    if (x >= 0)
        return(1);
    else
        return(-1);
}

int findnum(number)
int number[];
{
    int i,flag1,flag2,ret;
    ret = flag1=flag2=0;
    for(i=0;i<4;i++){
        if (number[i]>0) {
            flag1 = flag1 + 1;
            ret = 10*(i+1);
        }
    }
    for(i=4;i<8;i++){
        if (number[i]>0) {
            flag2 = flag2 + 1;
            ret = ret + i - 3;
        }
    }
    if ((flag1 ==1) && (flag2 == 1))
        return(ret);
    else return(-1);
}
detect(x,y,N)
double **x,**y;
int N;
{
    int level,i,j,durcount,intcount,numnow,numpast,index1,index2;
    int number[8];
    numnow = numpast = -1;
    for (i = 0;i<N;i++){
        if (x[i][0]>2000) level = 1;
        else level = 0;
        if (level > 0){
            for (j = 0;j<8;j++){
                number[j] = ((y[i][j]>(x[i][0]*4/8)) ? 1:0);
            }
            intcount = 0;
            numnow = findnum(number);
            if (i > 0)
            if (numpast != -1){
              index1 = (int)(numpast/10)-1;
              index2 = (numpast % 10) + 3;
              if ((y[i][index1]<(y[i-1][index1]*3/4)) ||
                              (y[i][index2]<(y[i-1][index2]*3/4)))  {
```

```
                  level = 0;
                 numnow = numpast;
               }
             }
             if ((level > 0)&&(numnow != -1)){
                 if (numnow == numpast)
                     durcount = durcount +1;
                 else
                     durcount = 0;
             }
             if (numnow == -1)
                durcount = 0;
         }
         if (level == 0){
             if (durcount >=2)
                 intcount = intcount + 1;
               index1 = (int)(numnow/10)-1;
               index2 = (numnow % 10) -1;
             if (intcount==3)
                 printf("at point %d number %c is detected \n",
                                              i,digit[index1][index2]);
         }
         printf("%d\n",level);
         numpast = numnow;
     }


}


main(argc, argv)
int    argc;
char    *argv[];
{
    char    *fname1, *fname2, *fname3, *fname4;
    int    i, j, test[24][NUM], N1, N2, N3, noofinp,inc,savinp;
    float    input;
    double    inp, dummy;
```

```
FILE  *fopen(), *f1, *f2;
double    freqs[8];
double    **z, **zt,**zmax;
double **res;
double t[NUM];
if (argc != 2)
    exit(1);
f1 = fopen(argv[1], "r");
fname1 = (char *)calloc(10, sizeof(char));
fname2 = (char *)calloc(10, sizeof(char));
fname3 = (char *)calloc(10, sizeof(char));
fname4 = (char *)calloc(10, sizeof(char));
fname1 = "z";
fname2 = "ztest";
fname3 = "level";
fname4 = "zmax";
digit[0][0] = '1';
digit[0][1] = '2';
digit[0][2] = '3';
digit[0][3] = 'A';
digit[1][0] = '4';
digit[1][1] = '5';
digit[1][2] = '6';
digit[1][3] = 'B';
digit[2][0] = '7';
digit[2][1] = '8';
digit[2][2] = '9';
digit[2][3] = 'C';
digit[3][0] = '*';
digit[3][1] = '0';
digit[3][2] = '#';
digit[3][3] = 'D';
scanf("%d", &noofinp);
scanf("%d",&savinp);
N1 = noofinp;
N2 = 24;
N3 = (int)(N1/100)+1;
res = (double **)calloc(N3,sizeof(double *));
```

```
    for (i=0; i < N3;i++)
        res[i] = (double *)calloc(1,sizeof(double));
    z = (double **)calloc(N1, sizeof(double * ));
    for (i = 0; i < N1; i++)
        z[i] = (double *)calloc(N2, sizeof(double));
    zmax = (double **)calloc(N3,sizeof(double *));
    for (i = 0; i < N3; i++)
        zmax[i] = (double *)calloc(8,sizeof(double));
    zt = (double **)calloc(NUM, sizeof(double * ));
    for (i = 0; i < NUM; i++)
        zt[i] = (double *)calloc(N2, sizeof(double));

    freqs[0] = 697;
    freqs[1] = 770;
    freqs[2] = 852;
    freqs[3] = 941;
    freqs[4] = 1209;
    freqs[5] = 1336;
    freqs[6] = 1477;
    freqs[7] = 1633;
    for (i = 0; i < NUM; i++)
        t[i] = i - (NUM - 1) / 2.0;


    for (i = 0; i < 8; i++) {
        for (j = 0; j < NUM; j++) {
            dummy = sin(2 * M_PI * t[j]* freqs[i] / 8000.0);
            test[i][j] = sign(dummy);
            dummy = sin(2 * M_PI * t[j] * freqs[i] / 8000.0 +
M_PI / 3.0);
            test[i+8][j] = sign(dummy);
            dummy = sin(2 * M_PI * t[j] * freqs[i] / 8000.0 +
    2 * M_PI / 3.0);
            test[i+16][j] = sign(dummy);
        }
    }

    inc = -1;
    for (i = 0; i < N1; i++) {
        fscanf(f1, "%f", &input);
        if ((i % 100) == 0){
            inc = inc + 1;
            res[inc][0] = fabs(input);

            for (j = 0; j < 8; j++) {
                z[i][j] = input * (test[j][0]);
                z[i][j+8] = input * (test[j+8][0]);
                z[i][j+16] = input * (test[j+16][0]);
            }
        }
        else{
            res[inc][0] = res[inc][0] + fabs(input);
            for (j = 0; j < 8; j++) {
                z[i][j] = z[i-1][j] + input * (test[j][i%NUM]);
                z[i][j+8] = z[i-1][j+8] + input *
      (test[j+8][i%NUM]);
                z[i][j+16] = z[i-1][j+16] + input *
      (test[j+16][i%NUM]);
            }
        }
    }
    for (i = 0; i < (N3 -1); i++){
        for(j = 0; j <8;j++){
            zmax[i][j] = max3(fabs(z[i*100+99][j]),
fabs(z[i*100+99][j+8]), fabs(z[i*100+99][j+16]));
        }

    }
    detect(res,zmax,N3);
    if (savinp != 0){
    save_array(z, N1, N2, fname1);
    for (i = 0; i < NUM; i++)
        for (j = 0; j < N2; j++) {
            zt[i][j] = (double)test[j][i];
        }
    save_array(zt, NUM, N2, fname2);
    save_array(res,N3,1,fname3);
    save_array(zmax, N3,8, fname4);
    }
}
```

A.2 Frequency and Phase Dependence

A.2.1 C-file

```
#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include "/manisah/ee/ekinci/dtmfrec/cfiles/coralg/save.h"
#define NUM 100

int    sign(x)
double    x;
{
    if (x >= 0)
        return(1);
    else
        return(-1);
}


inputsin(amp, freq, phase, inp)
double    amp, freq, phase;
double    inp[NUM];
{
    int    i;
    double    phi;

    phi = phase * M_PI / 180.0;
    for (i = 0; i < NUM; i++) {
        inp[i] = amp * cos(2 * M_PI * freq * i / 8000 + phi);
    }
}



main(argc, argv)
int     argc;
char    *argv[];
{
    double    dummy;
    double    **z;
    double    freqs[8], inpfre, firstfre, lastfre, stepfre;
    double    inpphi, firstphi, lastphi, stepphi;
    int     i, j, test[24][NUM], indexfre, lenfre, indexphi,
      lenphi, index;
    double    t[NUM];
    double    input[NUM];
    char    *fname1;

    fname1 = (char *)calloc(10, sizeof(char));

    scanf("%lf", &firstfre);
    scanf("%lf", &lastfre);
    scanf("%lf", &stepfre);
    scanf("%lf", &firstphi);
    scanf("%lf", &lastphi);
    scanf("%lf", &stepphi);
    lenphi = (int)((lastphi - firstphi) / stepphi + 1);
    lenfre = (int)((lastfre - firstfre) / stepfre + 1);
    z = (double **)calloc(24, sizeof(double * ));
    for (i = 0; i < 24; i++)
        z[i] = (double *)calloc(lenfre * lenphi, sizeof(double));

    fname1 = argv[1];
    freqs[0] = 697;
    freqs[1] = 770;
    freqs[2] = 852;
    freqs[3] = 941;
    freqs[4] = 1209;
    freqs[5] = 1336;
    freqs[6] = 1477;
    freqs[7] = 1633;

    for (i = 0; i < NUM; i++)   /* Time -49.5/8000 to 49.5/8000 */
        t[i] = i - (NUM - 1) / 2.0;

    for (i = 0; i < 8; i++) {
        for (j = 0; j < NUM; j++) {
            dummy = sin(2 * M_PI * t[j] * freqs[i] / 8000.0);
            test[i][j] = sign(dummy);
            dummy = sin(2 * M_PI * t[j] * freqs[i] / 8000.0 +
M_PI / 3.0);
            test[i+8][j] = sign(dummy);
            dummy = sin(2 * M_PI * t[j] * freqs[i] / 8000.0 +
   2 * M_PI / 3.0);
            test[i+16][j] = sign(dummy);
        }
    }

    for (inpphi = firstphi; inpphi <= lastphi;
inpphi = inpphi + stepphi) {
        indexphi = (int)((inpphi - firstphi) / stepphi);
        for (inpfre = firstfre; inpfre <= lastfre;
inpfre = inpfre + stepfre) {
            inputsin(1.0, inpfre, inpphi, input);
            indexfre = (int)((inpfre - firstfre) / stepfre);
            index = indexphi * lenfre + indexfre;
            for (j = 0; j < 24; j++)
                z[j][index] = 0;
            for (i = 0; i < NUM; i ++) {
                for (j = 0; j < 8; j++) {
                 z[j][index] = z[j][index] + input[i] *
(test[j][i]);
                 z[j+8][index] = z[j+8][index] + input[i] *
(test[j+8][i]);
                    z[j+16][index] = z[j+16][index] + input[i] *
(test[j+16][i]);
                  }
              }
          }
      }
      save_array(z, 24, lenfre * lenphi, fname1);
}
```

A.2.2 Matlab

```
w = 2*pi* input('w = ');
a = 2*pi* input('a = ');
h = input('amplitude = ');
fig = input('figure = ');
phi = input('phi = ');
lphi = length(phi);
T = 1/8000;
n = 100;
ns = (-n/2:1:(n/2-1))+1/2;
%figure(fig);clg;hold on;
clear S1 S2 S3
for k = 1:length(w),
y1 = sgn(sin(ns*T*w(k)));
y1 = 2*y1-1;
y2 = sgn(sin(ns*T*w(k)+pi/3));
y2 = 2*y2 -1;
y3 = sgn(sin(ns*T*w(k)+2*pi/3));
y3 = 2*y3 -1;
for j = 1:length(phi),
for i = 1:length(a),
x = h*cos((0:(n-1))*T*a(i)+phi(j));
%SS(i+(j-1)*length(a),k) = sum(y1.*x);
```

```
%CC(i,(k-1)*lphi+j) = sum(y2.*x);
S1(i+(j-1)*length(a),k) = sum(y1.*x);
S2(i+(j-1)*length(a),k) = sum(y2.*x);
S3(i+(j-1)*length(a),k) = sum(y3.*x);
end % i
end % j
end % k
```

A.3 Circuit Simulation Files

The simulation of the circuit has been performed by Verilog.

A.3.1 Top Circuit

```
`timescale 100ps / 100ps

module test;


wire  BUTTON_3_, BUTTON_2_, BUTTON_1_, BUTTON_0_, DIGITAV,
      INPLATCLKTEST, ROMCKTEST, tstOUT0, tstOUT1, tstOUT2, tstOUT3;


reg  CLOCK, RESDIGAV, RESET, S0, S0tst, S1, S1tst, S2tst;

wire  DATIN_12_, DATIN_11_, DATIN_10_, DATIN_9_, DATIN_8_, DATIN_7_
     , DATIN_6_, DATIN_5_, DATIN_4_, DATIN_3_, DATIN_2_, DATIN_1_,
     DATIN_0_;

reg [12:0]  innum[1:2000];
integer index;
integer dstb1,dstb2,dstb3,dstb4,dstb5;
integer i,j;

assign {DATIN_12_, DATIN_11_, DATIN_10_, DATIN_9_,
          DATIN_8_, DATIN_7_, DATIN_6_, DATIN_5_,
          DATIN_4_, DATIN_3_,
DATIN_2_, DATIN_1_, DATIN_0_} = $getpattern(innum[index]);

  .
initial $get_design("test.top", ,);
topp top(BUTTON_3_, BUTTON_2_, BUTTON_1_, BUTTON_0_, DIGITAV,
         INPLATCLKTEST, ROMCKTEST, tstOUT0, tstOUT1, tstOUT2,
         tstOUT3, CLOCK, DATIN_12_, DATIN_11_, DATIN_10_,
         DATIN_9_, DATIN_8_, DATIN_7_, DATIN_6_, DATIN_5_,
         DATIN_4_, DATIN_3_, DATIN_2_, DATIN_1_, DATIN_0_,
         RESDIGAV, RESET, S0, S0tst, S1, S1tst, S2tst);
initial $check_design();

// Default verilog stimulus

initial
begin

   CLOCK = 1'b0;
//   DATIN_12_ = 1'b0;
//   DATIN_11_ = 1'b0;
//   DATIN_10_ = 1'b0;
//   DATIN_9_ = 1'b0;
//   DATIN_8_ = 1'b0;
//   DATIN_7_ = 1'b0;
//   DATIN_6_ = 1'b0;
//   DATIN_5_ = 1'b0;
//   DATIN_4_ = 1'b0;
//   DATIN_3_ = 1'b0;
//   DATIN_2_ = 1'b0;
//   DATIN_1_ = 1'b0;
//   DATIN_0_ = 1'b0;
   RESDIGAV = 1'b1;
   RESET = 1'b0;
   S0 = 1'b0;
   S0tst = 1'b0;
   S1 = 1'b1;
   S1tst = 1'b1;
   S2tst = 1'b0;
end


 // please enter any additional stimulus here
initial
$readmemb("/home/vlsi/ekinci/dtmf/simul/patt.num",innum);
initial
       begin
   dstb1 = $fopen("/home/vlsi/ekinci/dtmf/simul/mul1.out");
   dstb2 = $fopen("/home/vlsi/ekinci/dtmf/simul/mul2.out");
   dstb3 = $fopen("/home/vlsi/ekinci/dtmf/simul/mul3.out");
   dstb4 = $fopen("/home/vlsi/ekinci/dtmf/simul/mul4.out");
   dstb5 = $fopen("/home/vlsi/ekinci/dtmf/simul/mul5.out");
// dstb6 = $fopen("/home/vlsi/ekinci/dtmf/simul/mul6.out");
       end

initial
    begin
index = 1;
#16800 RESET = 1;
    end
always #4800 CLOCK = !CLOCK;
always
      begin
#1228800 index = index + 1;
if (index > 5000 ) $finish;
      end

initial
begin
#100800 i = 1;
for(i=0;i<500000;i=i+1)
begin
     #9600
     $fdisplay(dstb1,test.top.I50.MAXtest);
 $fdisplay(dstb2,test.top.I50.I15.QS2test);
```

```
end

end
initial
begin
#283200 j=1;
for(j=0;j<100000;j=j+1)
begin
     #153600
     $fdisplay(dstb4,BUTTON_3_,BUTTON_2_,
      BUTTON_1_,BUTTON_0_,DIGITAV);
     $fdisplay(dstb3,tstOUT0,, tstOUT1,,
                                        tstOUT2,, tstOUT3);
     $fdisplay(dstb5,test.top.I50.I15.sgn,,
                                        test.top.I50.I15.valid);
end
end
endmodule
```

APPENDIX B

SCHEMATICS

B.1 Correlator and Level Detector

The schematics drawn for the *correlator* and for the *level detector* are shown in this chapter.

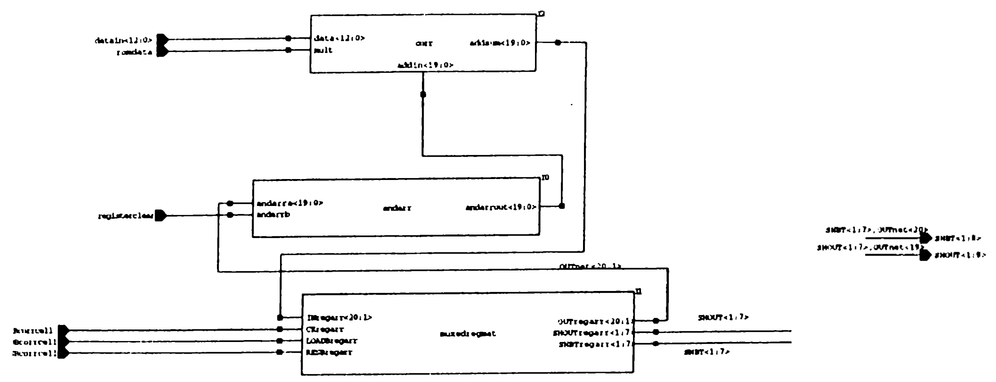


Figure B.1: The correlator schematic.

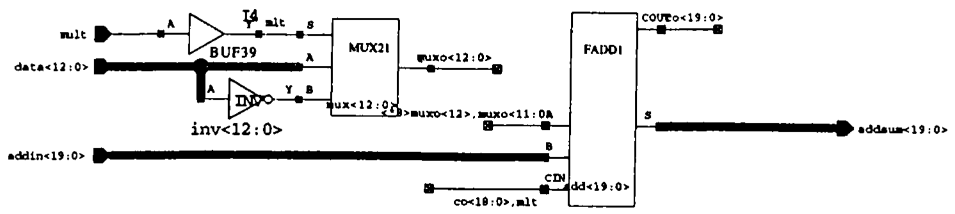


Figure B.2: Correlation.

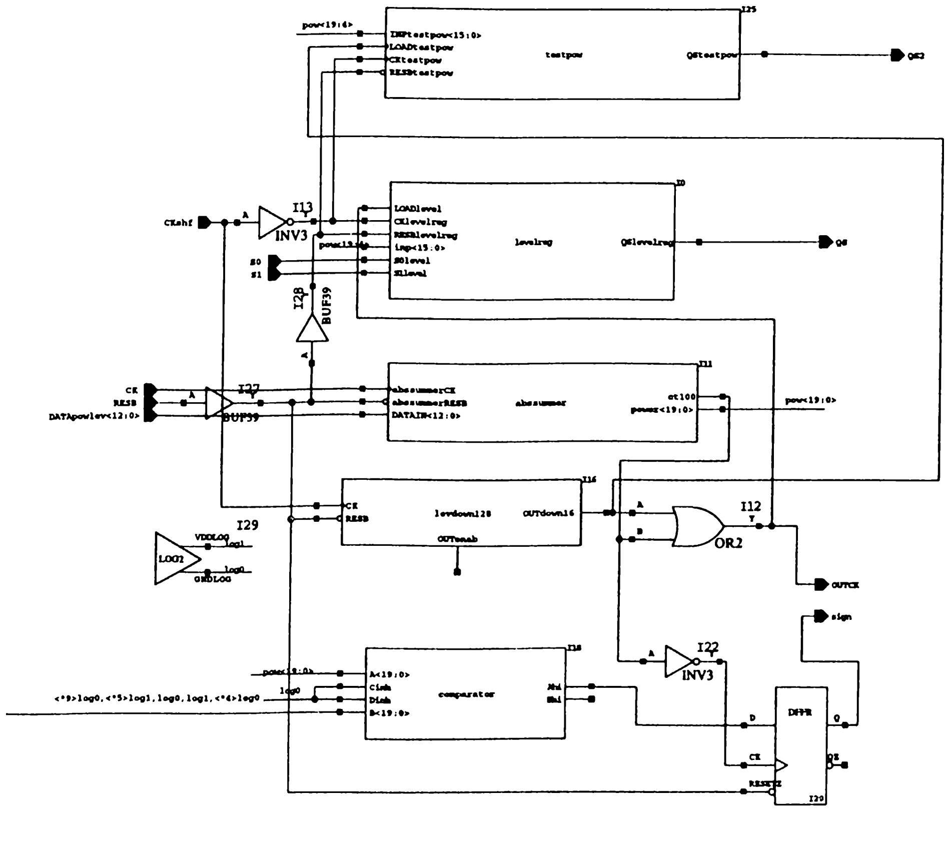


Figure B.3: The level detector top schematic