موضوع پروژه:

پیاده سازی بلادرنگ کدک صحبت استاندارد G.728

برروی پردازنده TMS320C5402

K.N.T. University of Technology

Real Time Implementation of G.728 Speech Codec using TMS320C5402

By: Asghar Pourhosein

Advisor: Dr. M.E. Kalantari

A Thesis submitted to faculty of Electrical Engineering in partial fulfillment Of the requirements of the degree of M.Sc.

Summer 2002

Abstract

G.728 speech codec is a low delay ITU standard codec which could provide toll quality speech in 16kb/s. It is specially designed for delay sensitive applications like satellite telephony, Internet, and mobile networks.

In this thesis real time implementation of full duplex G.728 encoder and decoder on TMS320C5402 is presented. Using a combinatorial technique for TMS programming, the programming time and complexity have been decreased to 30% comparing with traditional assembly programming. First a fixed point simulation of the codec algorithm has been programmed in C and it is compiled to assembly using CCS (Code Composer Studio) and manually optimized. Then some of the critical functions regarding MIPS, have been programmed in assembly for achieving real time implementation. Finally, implementation results have been presented.

Keywords: Speech Coding & Compression, Real Time Implementation, DSP, TMS320C5402, DSK Board

چکیده

کدک صحبت استاندارد G.728 ، یک کدک کم تاخیر است که صحبت با کیفیت عالی را در نرخ بیت 16 kbps ارائه می دهد و برای شـــبکه های تلفن ماهواره ای و اینترنت و موبایل که به تاخیر زیاد حســـاس هستند ، مناسب است. در این رساله به پیاده سازی بلادرنگ اینکدر و دیکدر G.728 بصورت دوطرفه کامل (Full Duplex) بر روی پردازنده TMS320C5402 می پردازیم .

روشی ترکیبی برای برنامه نویسی TMS ارائه می شود که در آن زمان وپیچیدگی برنامه نویسی نسبت به برنامه نویسی دستی به ۳۰٪ کاهش می یابد . در این روش پس از برنامه نویسی و شبیه سازی ممیزثابت الگوریتم کدک به زبان C ، با استفاده از نرم افزار (Code Composer Studio) CCS ، برنامه به زبان اسمبلی ترجمه شده و بهینه سازی دستی در کل کد اسمبلی صورت می گیرد . سپس بعضی از توابع مهم برنامه از نظر MIPS ، بصورت دستی به زبان اسمبلی بازنویسی می شوند تا برنامه بصورت بلادرنگ قابل اجرا گردد . در پایان نتایج این پیاده سازی ارائه می شود .

کلمات کلیدی

کدینگ و فشرده سازی صحبت ، پیاده سازی بلادرنگ ، DSP ، TMS320C5402 ، برد DSK

فهرست

- مقدمه

امروزه در عصر ارتباطات و گسترش روزافزون استفاده از شبکه های تلفن ،موبایل و اینترنت در جهان ومحدودیت پهنای باند در شبکه های مخابراتی ، کدینگ و فشرده سازی صحبت امری اجتناب ناپذیر است . در چند دهه اخیر روشهای کدینگ مختلفی پدیدآمده اند ولی بهترین و پرکاربردترین آنها کدک های آنالیزباسنتز هستند که توسط Atal & Remedeدر سال ۱۹۸۲ معرفی شدند [2] . اخیرا مناسبترین الگوریتم برای کدینگ صحبت با کیفیت خوب در نرخ بیت های پائین و زیر 16 kbps ، روش پیشگویی خطی باتحریک کد (CELP) می با شد که در سال ۱۹۸۵ تو سط Schroeder & Atal معرفی شد [8] و تا کنون چندین استاندارد مهم کدینگ صحبت بر اساس CELP تعریف شده اند .

در سال ۱۹۸۸ CCITT برنامه ای برای استانداردسازی یک کدک 16 kbps با تاخیراندک و کیفیت بالا در برابر خطاهای کانال آغاز نمود و برای آن کاربردهای زیادی همچون شبکه PSTN، ISDN،تلفن تصویری و غیره در نظر گرفت . این کدک در سال ۱۹۹۲ توسط Chen et al. تحت عنوان LD-CELP معرفی شد[6] و بصورت ا ستاندارد G.728 در آمد[9] و در سال ۱۹۹۴ م شخصات ممیز ثابت این کدک تو سط ITU ارائه شد[10] . با توجه به کیفیت بالای این کدک که در آن صحبت سنتزشده از صحبت اولیه تقریبا غیرقابل تشخیص است و کاربردهای آن در شبکه های تلفن و اینترنت و ماهواره ای در این گزارش به پیاده سازی این کدک می پردازیم .

در فصل اول به معرفی وآنالیز سیگنال صحبت پرداخته می شود و در فصل دوم روش ها و استانداردهای کدینگ بیان می شوند . در فصل سوم کدک LD-CELP را بیشتر بررسی می کنیم و در فصل چهارم شبیه سازی ممیز ثابت الگوریتم به زبان C را بیان می نمائیم. ودر پایان در فصل ۵ به نحوه پیاده سازی بلادرنگ کدکG.728 بر روی پردازنده TMS320C5402 می پردازیم.

فصل ۱

بررسی و مدل سازی سیگنال صحبت

۱،۱،۱،۱،۱،۱،۱،۱ ۱-۱-معرفی سیگنال صحبت

صحبت در اثر دمیدن هوا از ریه ها به سمت حنجره و فضای دهان تولید می شود. در طول این مسیر در انتهای حنجره، تارهای صوتی[۱] قرار دارند. فضای دهان را از بعد از تارهای صوتی ، لوله صوتی[۲] می نا مند که در یک مرد متوسط حدود ۱۷cm طول دارد . در تولید برخی اصوات تارهای صوتی کاملاً باز هستند و مانعی بر سر راه عبور هوا ایجاد نمی کنند که این اصوات را اصطلاحاً اصوات بی واک[۳] می نامند. در د سته دیگر ا صوات ، تارهای صوتی مانع خروج طبیعی هوا از حنجره می گردند که این باعث به ارتعاش درآمدن تارها شده و هوا به طور غیر یکنواخت و تقریباً پالس شکل وارد فضای دهان می‌شود. این دسته از اصوات را اصطلاحاً باواک[۴] می گویند.

فرکانس ارتعاش تارهای صوتی در اصوات باواک را فرکانس Pitch و دوره تناوب ارتعاش تارهای صوتی را پریود Pitch می‌نامند. هنگام انتشار امواج هوا در لوله صوتی، طیف فرکانس این امواج

---

[۱] Vocal Cords
[۲] Vocal Tracts
[۳] Unvoiced
[۴] Voiced

توسط لوله صوتی شکل می‌گیرد و بسته به شکل لوله ، پدیده تشدید در فرکانس های خاصی رخ می‌دهد که به این فرکانس های تشدید فرمنت[1] می‌گویند.

از آنجا که شکل لوله صوتی برای تولید اصوات مختلف، متفاوت است پس فرمنت ها برای اصوات گوناگون با هم فرق می کنند. با توجه به اینکه صحبت یک فرآیند متغییر با زمان است پس پارامترهای تعریف شده فوق اعم از فرمنت ها و پریود Pitch در طول زمان تغییر می کنند به علاوه مد صحبت به طور نامنظمی از باواک به بی واک و بالعکس تغییر می‌کند. لوله صوتی ، همبستگی های زمان-کوتاه ، در حدود 1 ms ، درون سیگنال صحبت را در بر می‌گیرد. و بخش مهمی از کار کدکننده های صوتی مدل کردن لوله صوتی به صورت یک فیلتر زمان-کوتاه می‌باشد. همان طور که شکل لوله صوتی نسبتاً آهسته تغییر می کند، تابع انتقال این فیلتر مدل کننده هم نیاز به تجدید[2] ، معمولاً در هر 20ms یکبارخواهد داشت.

در شکل (۱-۱ الف) یک قطعه صحبت باواک که با فرکانس 8KHz نمونه برداری شده است دیده می‌شود. اصوات باواک دارای تناوب زمان بلند به خاطر پریود Pitch هستند که نوعاً بین 2ms تا 20ms می‌باشد. در اینجا پریود Pitch در حدود 8ms یا ۶۴ نمونه است. چگالی طیف توان این قطعه از صحبت در شکل (۱-۱ ب) دیده می‌شود[3].

اصوات بی واک نتیجه تحریک نویز مانند لوله صوتی هستند و تناوب زمان- بلند اندکی را در بر دارند ، همانگونه که در شکل های (۱-۱ ج) و (۱-۱ د) دیده می شود ولی همبستگی زمان کوتاه به خاطر لوله صوتی در آنها هنوز وجود دارد.

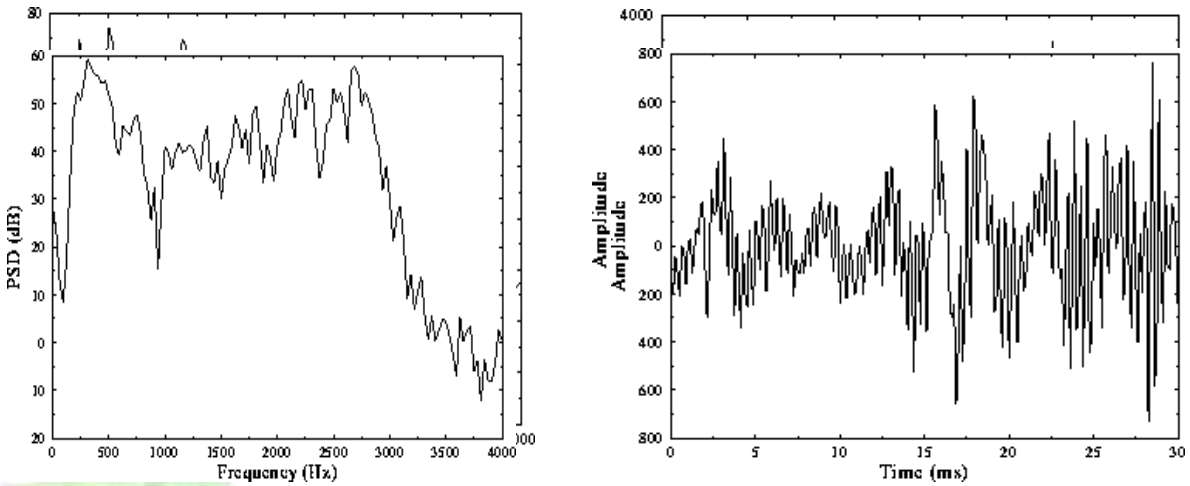
[1] Formant

[2] Update

بطورکلی سیگنال صحبت دارای افزونگی[1] زیادی است که ناشی از عوامل ذیل هستند:

ـ وابستگی های زمان-کوتاه : این وابستگی ها عمدتاً به کندی تغییرات صحبت با زمان و ساختار

(ب) (الف)



(د) (ج)

شکل (۱-۱) : مقایسه اصوات باواک و بی واک. (الف)و(ب) : باواک ، (ج)و(د) : بی واک

نسبتاً منظم فرمنت ها مربوط می‌شوند.

ـــــ وابستگی های زمان- بلند : که عمدتاً از طبیعت نیمه متناوب ا صوات با واک و تغییرات آرام پریود Pitch ناشی می‌شوند.

ـــــتابع چگالی احتمال صحبت : علیرغم پیچیدگی آماری صحبت می‌توان آن را با توابع چگالی احتمال شناخته شده تقریب زد. شکل لوله صوتی و مد تحریک آن به صورت نسبتاً آرام تغییر می‌کند و بنابراین صحبت را می‌توان به صورت شبه ایستان در دوره های کوتاه زمانی (حدود 20ms) در نظر

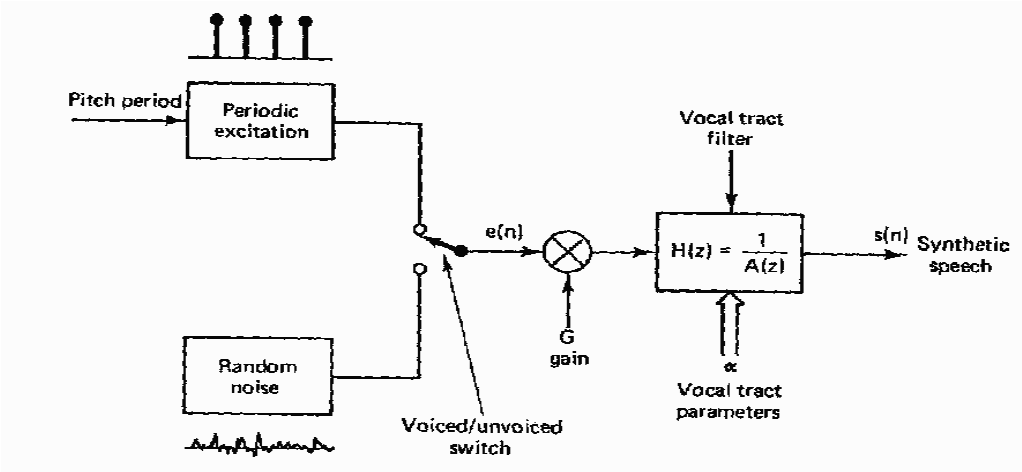
[1] Redundancy

گرفت و با یک فرآیند تصادفی ارگادیک در یک قطعه زمانی کوچک مدل نمود و طیف مشخصی برای آن در این قطعه زمانی بدست آورد.

علاوه بر افزونگی های فوق عامل مهم دیگری که کاهش نرخ داده سیگنال صحبت را ممکن می سازد، طبیعت غیر حساس گوش انسان نسبت به بسیاری از ویژگیهای این سیگنال می‌باشد.

۱-۲- مدل سازی پیشگویی خطی

روش کدینگ پیشگویی خطی (LPC[1]) مبتنی بر مدل تولید صحبت در کد کننده های صوتی می‌باشد که در اینجا در شکل (۱-۲) نشان داده شده است. برای استفاده از مدل لازم است که معلوم شود سیگنال با واک است یا بی‌واک و اگر با واک است پریود Pitch محاسبه گردد. تفاوت اصلی بین LPC و سایر کدکننده های صوتی در مدل کردن لوله صوتی است. در تحلیل LPC ، لوله صوتی به صورت یک فیلتر دیجیتال تمام قطب در نظر گرفته می‌شود.[4,1].



[1] Linear Predictive Coding

شکل (۱-۲): مدل تولید صحبت در LPC

با شرکت دادن بهره G در این فیلتر داریم:

$$H(Z)=\frac{G}{1+a_1Z^{-1}+\ldots+a_pZ^{-p}}=\frac{S(Z)}{E(Z)}$$

که در آن p مرتبه فیلتر است. اگر S(n) خروجی فیلتر مدل صحبت و e(n) تحریک ورودی باشد، معادله فوق را در حوزه زمان به صورت زیر می‌توان نوشت:

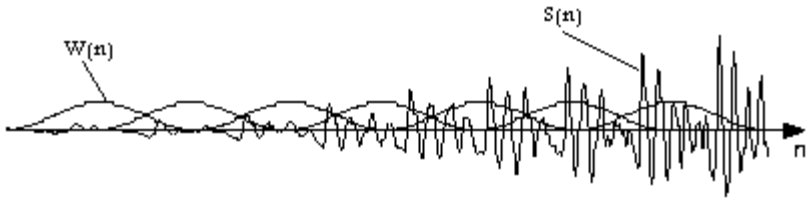
$$S(n)=Ge(n)-a_1S(n-1)-\cdots-a_pS(n-p)$$

به عبارت دیگر هر نمونه صحبت به صورت ترکیب خطی از نمونه های قبلی قابل بیان است و این دلیل نام گذاری کدینگ پیشگویی خطی (LPC) می‌باشد.

۱-۲-۱- پنجره کردن سیگنال صحبت

روش LPC هنگامی دقیق ا ست که به سیگنالهای ای ستان[۱] اعمال شود، یعنی به سیگنالهایی که رفتار آنها در زمان تغییر نمی‌کند. هر چند که این مو ضوع در مورد صحبت صادق نیست، اما برای اینکه بتوانیم روش LPC را بکار ببریم، سیگنال صحبت را به قسمت های کوچکی بنام "فریم" تقسیم می‌کنیم که این فریم ها شبه ایستان هستند. شکل (۱-۳) مثالی از قسمت بندی سیگنال صحبت را نشان می‌دهد. این قسمت بندی با ضرب کردن سیگنال صحبت S(n) ، در سیگنال پنجره W(n) انجام می‌شود.

[۱] Stationary



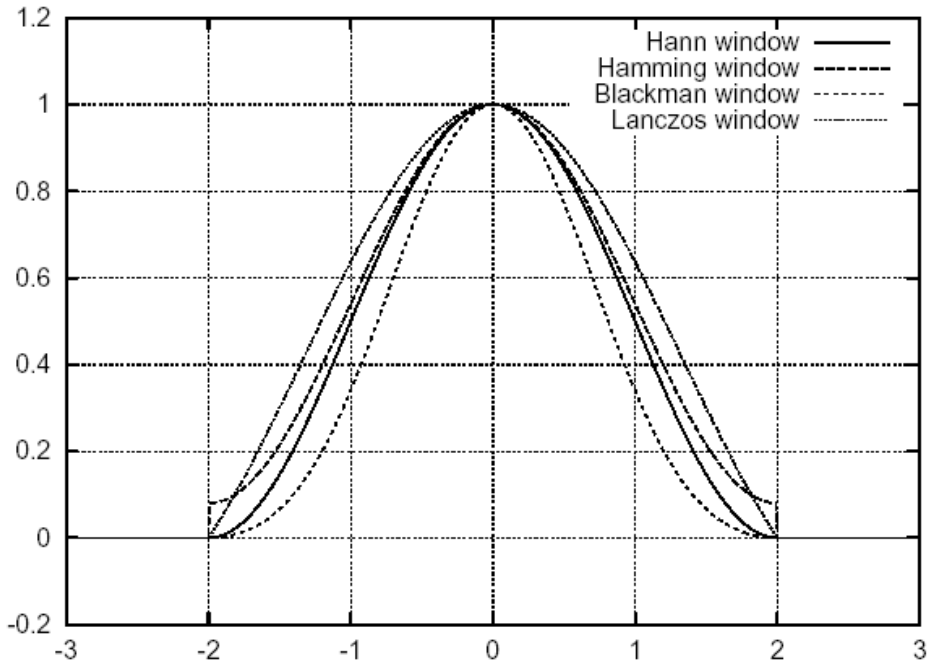
شکل (۱-۳) : قسمت بندی سیگنال صحبت

معروف ترین انتخاب برای پنجره ، پنجره همینگ (Hamming) به صورت زیر است:

$$W(n) = 0.54 - 0.46\cos\frac{2\pi n}{N} \quad , \quad 0 \le n \le N-1$$
$$W(n) = 0 \quad , \quad otherwise$$

در اینجا N ، طول پنجره دلخواه به نمونه و عموماً در محدوده 160-320 انتخاب می گردد که 240 یک مقدار نوعی می‌باشد . در شکل (۱-٤) چند پنجره معروف نشان داده شده است.

معمولاً پنجره های متوالی برروی هم همپوشانی دارند و فاصله بین آنها را پریود فریم می‌گویند. مقادیر نوعی برای پریود فریم 10-30ms می باشد. این انتخاب به نرخ بیت و کیفیت صحبت دلخواه ما بستگی خواهد داشت. هر چه پریود فریم کوچکتر باشد، کیفیت بهتری خواهیم داشت.



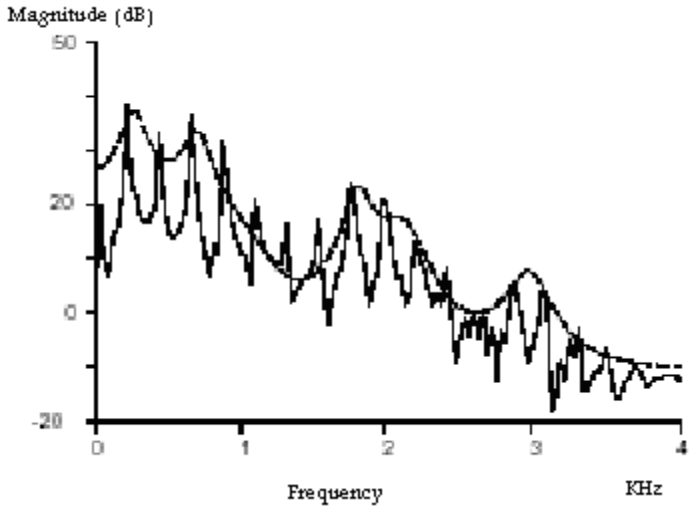
شکل (۱-۴): نمایش چند پنجره معروف

۱-۲-۲- پیش تاکید سیگنال صحبت

شکل (۱-۵) یک توزیع طیفی نمونه سیگنال صحبت را برای ا صوات باواک ن شان می دهد. با توجه به افت طیف در فرکانس های بالا وضعیف بودن فرکانس های بالا در طیف صحبت ، تحلیل LPC در فرکانس های بالا عملکرد ضعیفی خواهد داشت. برای تقویت مؤلفه های فرکانس بالا صحبت ، آن را از یک فیلتر بالا گذر با تابع انتقال $1-az^{-1}$ که فیلتر پیش تاکید نامیده می شـــود، عبور می دهیم. مقدار نوعی ضریب a معمولاً $\frac{15}{16}=0.9375$ در نظر گرفته می‌شود.

اگر S(n) سیگنال ورودی باشد، سیگنال پیش تأکید شده $S'(n)$ خواهد شد:

$$S'(n)=S(n)-0.9375S(n-1)$$



شکل (۱-۵) : پوشش طیفی نمونه اصوات باواک

۱-۲-۳- تخمین پارامترهای LPC

در اینجا لازم است که پارامترهای مدل LPC یعنی ضرایب ai فیلتر و بهره G تعیین گردند. اگر

$$\hat{S}(n) = -a_1 S(n-1) - \cdots - a_p s(n-p)$$

تخمین S(n) از روی نمونه های قبلی باشد، ضرایب ai را چنان تعیین می‌کنیم که خطای

$$\sum_n \left[S(n) - \hat{S}(n)\right]^2$$

روی همه نمونه های موجود مینیمم گردد. این مینیمم سازی ما را به معادلات خطی زیر می‌رساند:

$$\begin{aligned}
a_1 r(0) + a_2 r(1) + \cdots + a_p r(p-1) &= -r(1) \\
a_1 r(1) + a_2 r(1) + \cdots + a_p r(p-2) &= -r(2) \\
&\vdots \\
a_1 r(p-1) + a_2 r(p-2) + \cdots + a_p r(0) &= -r(p)
\end{aligned}$$

و یا در فرم ماتریسی

$R.a = -r$

در معادلات فوق تعریف زیر را داریم:

$$r(i) = r(-i) = \sum_{n=0}^{N-i-1} S(n)S(n+i)$$

که r(i) ، i امین اتوکورلیشن سیگنال می‌باشد و فرض شده که S(n) به طول N پنجره شده است. این فرمولاسیون به روش اتوکورلیشن معروف است و ماتریس R در آن یک ماتریس Toeplitz می‌باشد. چنین ماتریسی غیرمنفرد و همیشه معکوس پذیر است و در نتیجه همواره می‌توانیم جوابی به صورت $a = -R^{-1}r$ داشته باشیم.

روش دیگری نیز بنام روش کواریانس وجود دارد. در این روش سیگنال صحبت S(n) پنجره نمی‌شود و به جای اتوکورلیش های r(i) ، کواریانس های r(i,j) برای عنصر (i,j) ماتریس R محاسبه می‌گردد:

$$r(i, j) = \sum_{n} S(n+i)S(n+j)$$

در اینجا تضمین نمی شود که ماتریس R معکوس پذیر باشد و ممکن است که سیستم معادلات فوق جواب نداشته باشد. در این حالت فیلتر LPC ناپایدار می‌شود. از این رو در اینجا بیش از این به روش کواریانس نمی‌پردازیم.

راه سوم روش Burg است که امتیاز عدم استفاده از پنجره را در روش کواریانس با امتیاز روش اتوکورلیشن یعنی تضمین پایداری فیلتر ، ترکیب می‌کند. این روش از ساختار مشبک[1] فیلتر تمام قطب استفاده می‌کند[1] .

[1] Lattice

جواب د ستگاه معادلات فوق را می توان با یکی از روش های کلا سیک آنالیز عددی مثل حذف گوسی بدست آورد. اما چون R یک ماتریس Toeplitz است می توان از روشی مؤثر بنام روش تکرار Durbin سود جست که بصورت زیر ضرائب فیلتر را تولید می کند :

$$E(0)=r(0)$$
$$for \quad i=1,...,p$$
$$K_i=-\frac{r(i)+a_1^{(i-1)}r(i-1)+\cdots+a_i^{(i-1)}r(1)}{E(i-1)},$$
$$a_i^{(i)}=K_i$$
$$for \quad j=1,...,i-1 \qquad a_j^{(i)}=a_j^{(i-1)}+K_i a_{i-j}^{(i-1)}$$
$$E(i)=(1-K_i^2)E(i-1)$$

که در آن $a_j^{(i)}, j=1,\cdots,i$ ، ضریب j ام فیلتردر تکرار i ام و E(i) خطای پیشگویی مرتبه i است و بدین ترتیب ضرایب فیلتر بصورت زیر بدست خواهند آمد:

$$a_j=a_j^{(p)}, j=1,\cdots,p$$

روش تکرار Durbin پارامترهای $K_i, i=1,\cdots,p$ را که ضرایب انعکاس نامیده می شوند و E(p) را بدست می‌دهد که مربع بهره پیشگویی G و مورد نیاز فیلتر سنتز می‌باشد:

$$G^2=E(p)$$

و چون داریم :

$$E(p)=(1-K_1^2)(1-K_2^2)\cdots(1-K_p^2)r(0)$$

می توانیم به جای E(p)، r(0) را کد کرده و ارسال داریم و از آنجا به بهره G برسیم و این ترجیح داده می‌شود زیرا حساسیت r(0) به نویز کوانتیزاسیون کمتر از G است.

ضرایب انعکاس K_i یا PARCOR (برای PARtial CORrelation) نقش مهمی در تحلیل LPC دارند و دارای خواص زیر هستند:

۱- ضـرایب انعکاس Ki معادل با ضـرایب فیلتر ai هسـتند . به عبارت دیگر می توان K را به a و برعکس تبدیل کرد :

K به a :

$$a_i^{(i)} = K_i$$
$$a_j^{(i)} = a_j^{(i-1)} + K_i a_{i-1}^{(i-1)} \qquad i = 1,\cdots,p \qquad j = 1,\cdots,i-1$$

a به K :

$$K_i = a_i^{(i)}$$
$$a_j^{(i-1)} = \frac{a_j^{(i)} - a_i^{(i)} a_{i-j}^{(i)}}{1 - K_i^2} \qquad i = p,\cdots,1 \qquad j = 1,\cdots,i-1$$

۲ـ برای یک فیلتر پایدار یعنی یک فیلتر LPC که همه قطب های آن داخل دایره واحد باشد داریم:

$$-1 < K_i < 1 \quad , \quad i = 1,\cdots,p$$

که این شرط بسیار مهمی ا ست چرا که با اطمینان از اینکه Ki بین 1– و 1+ ا ست حتی بعد از کوانیتزاسیون ، پایداری فیلتر تضمین خواهد شد. به علاوه محدوده (1+ , 1-) کار کوانیتزاسیون را ساده‌تر می‌کند. ولی ai ها دارای چنین ویژگی نیسـتند که پایداری فیلتر را تضـمین نمایند و کوانیتزاسـیون ai ها می‌تواند موجب ناپایداری شود.

فـصل ۲

روش ها و استانداردهای کدینگ صحبت

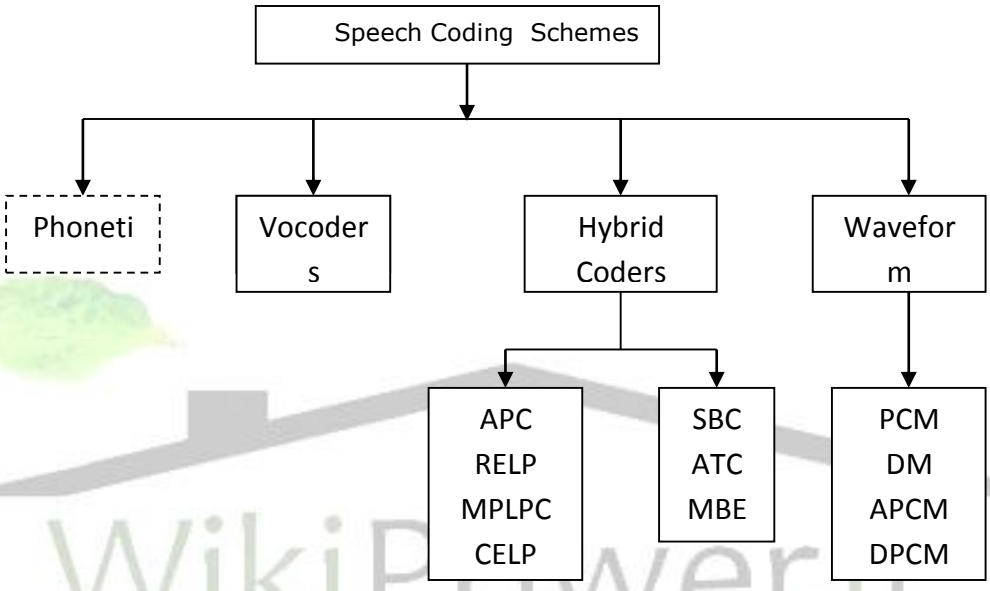
۲-۱- مقدمه

کدینگ دیجیتال صحبت موضوع تحقیقات بیش از سه دهه اخیر بوده و روش های زیادی برای کدینگ صحبت پدید آمده است .کیفیت صحبت و نرخ بیت دو عامل اساسی هستند که بطور مستقیم با هم درگیر می باشند و هر چه نرخ بیت پایین تر بیاید ، از کیفیت صحبت کاسته می شود. برای سیستم هایی که به شبکه تلفن متصل می شوند ، صحبت کد شده باید دارای کیفیت خوب باشد تا با استانداردهای ITU[1] مطابقت داشته باشد اما برای سیستم های محدود مانند شبکه های تجاری خصوصی و سیستم های نظامی ، ممکن است عامل نرخ بیت مهمتر از کیفیت بالای صحبت باشد . از ویژگیهای سیستم های کدینگ صحبت ، تأخیر کدینگ می باشد که مقدار آن به کیفیت مورد نیاز سیستم ارتباط نزدیک دارد .تأخیر کدینگ شامل تاخیر الگوریتمی (بافر کردن صحبت برای آنالیز)، تاخیر محاسباتی (زمان لازم برای پردازش و ذخیره کردن نمونه های صحبت) و تاخیر مربوط به ارسال و انتقال می باشد . تاخیر اندک در سیستم های مخابراتی باعث کاهش اثراکو در آن سیستم می شود. از ویژگیهای دیگر سیستم ها توانایی آنها برای ارسال داده در باند صوتی است . چرا که مشخصه های آماری و طیف فرکانسی داده باند صوتی کاملا با سیگنال صحبت تفاوت دارد .

۱،۱،۱،۱،۱،۱،۱،۱،۲-۲- روشهای کدینگ

[1] International Telecommunication Union (CCITT)

روشهای کدینگ صحبت را می توان به چند دسته اصلی که در شکل (۲-۱) نشان داده شده است تقسیم بندی نمود. از این میان سه دسته اصلی که با خط پر نشان داده شده اند موضوع تحقیقاتی بیشتری هستند . این روشها سیگنال صحبت را آنالیز کرده افزونگی های آنرا حذف نموده و بخش های غیر زائد صحبت را به روشی کد می کنند که از نظر شنیداری قابل قبول باشد.



شکل (۲-۱) : دسته بندی روشهای کدینگ

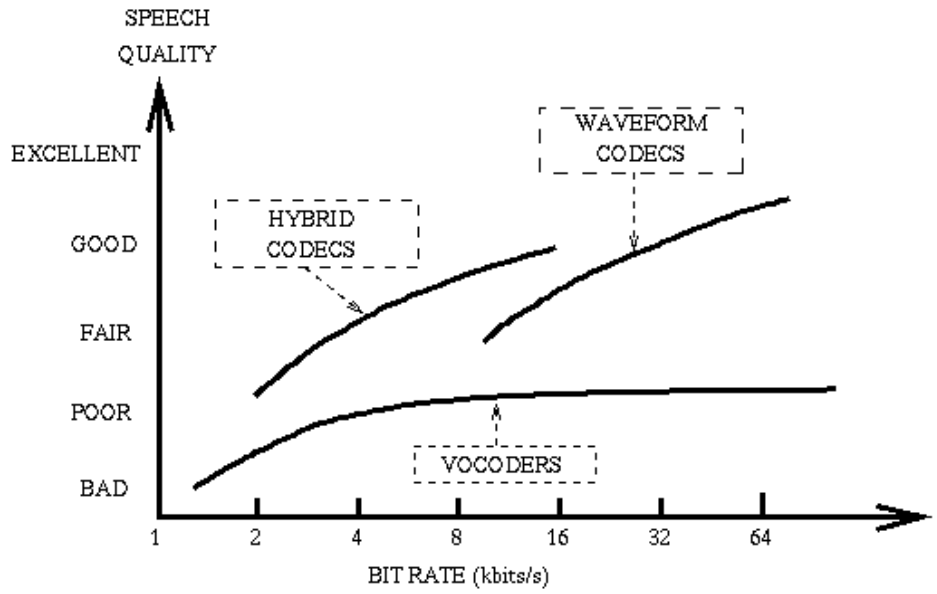
کدرهای شکل موج[۱] نوعا نرخ بیت بالایی دارند و صحبت باز سازی شده را با کیفیت خیلی خوب ارائه می دهند. کدرهای صوتی[۲] در نرخ بیت های خیلی پایین کار می کنند و صحبت را از طریق سنتز

[۱] Waveform Coders

[۲] Vocoders

بازسازی می کنند . کدرهای مختلط ترکیبی از تکنیک های صوتی و شکل موج را بکار می گیرند و صحبت با کیفیت خوب را در نرخ بیت های میانی ارائه می دهند.

در شکل (۲-۲) کیفیت صحبت بر حسب نرخ بیت برای سه دسته اصلی کدینگ یعنی کدینگ شکل موج ، کدینگ صوتی و کدینگ مختلط نشان داده شده است . همچنین خلاصه ای از کاربردهای روشهای در حال کار و آنهایی که در حال توسعه می باشند در جدول (۲-۱) گردآوری شده است [5] .



شکل (۲-۲) : مقا یسه کیفیت صحبت روشهای کدینگ صحبت

Rate (kbps)	Application	Type of Coder	Year of Operation
64	PSTN (1st Generation)	PCM	1972
32	PSTN (2nd Generation)	ADPCM	1984

16	PSTN (3rd Generation)	LD-CELP	1992
16	INMARSAT-B	APC	1985
13	GSM	RPE-LTP	1991
9.6	Skyphone	MPLPC	1990
8	North American Mobile	VSELP	1992
6.4	INMARSAT-M (land mobile)	MBE-CELP	1993
4.8	U.S. Gov.Fed.Standard	CELP	1991
4.8	NASA MSAT-X (mobile satellite)	Vector Adaptive Predictive Coding VAPC	1991

جدول (۲-۱): استاندارد های کدینگ صحبت

۲-۲-۱- کدرهای شکل موج

کدرهای شکل موج تلاش می کنند که شکل کلی سیگنال صحبت را حفظ نمایند. این کدرها می توانند هر شکل موجی در باند صوتی را قبول کنند و صرفا مختص صحبت نیستند . از آنجا که

در این کدرها کدینگ به صورت نمونه به نمونه انجام می شود ، عملکرد آنها همانند کوانتیزاسیون بوسیله نسبت سیگنال به نویز (SNR) اندازه گیری می شود. کدینگ های شکل موج برای صحبت، به نرخ بیت های بالای 16kbps محدود می شود و بخاطر سادگی و پیاده سازی آسان ، فراگیر شده اند .

اولین استاندارد جهانی کدینگ صحبت استاندارد G.711 PCM 64 kbps ، با کمپندینگ u-law برای امریکای شمالی و A-law برای اروپا، یک کدر شکل موج بوده و هنوز هم کاربرد زیادی در سیستم های مخابراتی دیجیتال دارد . از آنجا که کیفیت صحبت PCM 64 kbps عالی می باشد ، معمولا مرجعی برای مقایسه دیگر کدرهای صحبت با نرخ بیت پایین تر قرار می گیرد . کدر بعدی که توسط CCITT استاندارد شده PCM تفاضلی تطبیقی (ADPCM [1]) 32 kbps می باشد . کاهش نرخ بیت به نصف در این کدر، از طریق اعمال پیشگویی و کوانتیز اسیون تطبیقی حاصل شده است.

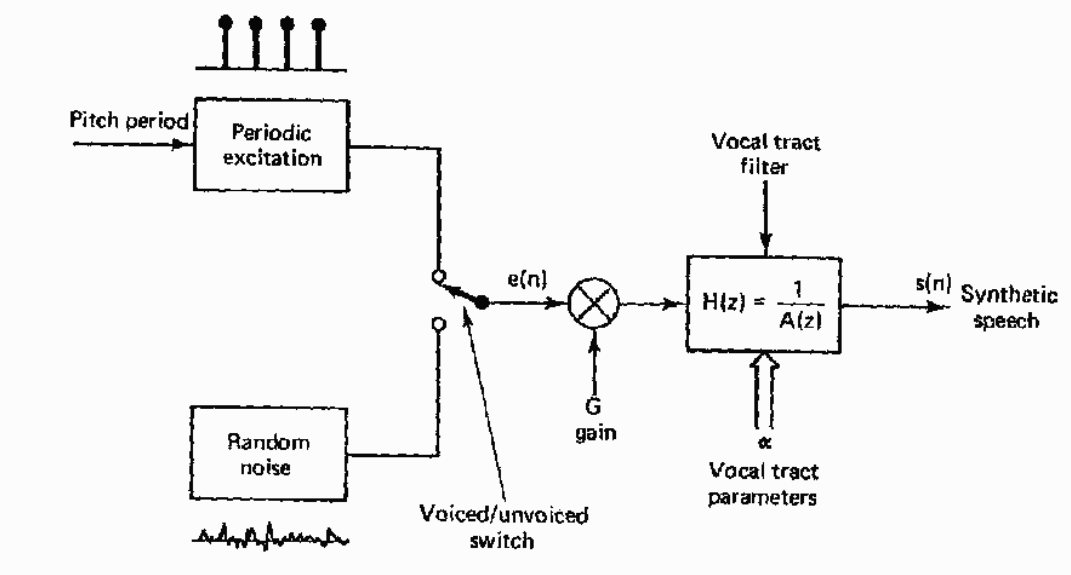
۲-۲-۲- کدرهای صوتی (Vocoders)

بر خلاف کدرهای شکل موج ، کدرهای صوتی خیلی به صحبت وابسته هستند و در اصولشان هم سعی ندارند که شکل موج اصلی صحبت را حفظ نمایند . یک کدرصوتی از یک آنالیزکننده و یک سنتزکننده تشکیل شده است . آنالیزکننده از صحبت اصلی یک دسته پارامترا را که نمایش دهنده مدل تولید صحبت هستند ، استخراج نموده و آنها را ارسال می کند و در گیرنده ، صحبت با استفاده از این پارامترها بازسازی می شود.

در شکل(۲-۳) مدل تولید صحبت در کدرهای صوتی نشان داده شده است . لوله صوتی بصورت یک فیلتر متغیر با زمان نمایش داده می شود. برای قطعات بی واک صحبت،این فیلتر بایک منبع نویز سفید تحریک می گردد و برای قطعات باواک صحبت بوسیله یک قطار پالس باپریود pitch تحریک می شود.

[1] Adaptive Prediction PCM

کدرهای صـــوتی معمولا در محدوده 2.4kbps کار می کنند و کیفیت قابل قبول آنها از صـــدای طبیعی فاصله دارد . از معروفترین آنها می توان کدرصوتی استاندارد 2.4kbps LPC-10 را نام برد[5].



شکل (۲-۳) : مدل تولید صحبت درکدرهای صوتی

سیـ ستم های حوزه فرکانس هم، مانند کدرهای صوتی فرمنت درپردازش صحبت بکار می روند . کدرهای صوتی فرمنت صحبت را با تحریک یک د سته از فیلترهای میانگذر که پهنای باند و فرکانس مرکزی آنها با فرمنت های صـــحبت منطبق هســـتند ، ســـنتر می کنند و عملکرد آنها به تعیین مکان دقیق فرمنت ها بستگی دارد.

۱،۱،۱،۱،۱،۱،۱،۲ ۲-۲-۳- کدرهای مختلط

کدرهای مختلط که از ویژگی های هر دوروش کدینگ صـــوتی و کدینگ شـــکل موج اســـتفاده می کنند به دو دسته حوزه فرکانس و حوزه زمان تقسیم بندی می شوند :

۱،۱،۱،۱،۱،۱،۱،۲،۱الف- کدرهای مختلط حوزه فرکانس

ایده اصلی در کدینگ حوزه فرکانس ،تقسیم طیف صحبت به باندهای فرکانسی یا مؤلفه هایی با استفاده از بانک فیلتر و یا یک بلوک تبدیل می باشد .در مرحله دیکدینگ ، این مؤلفه های فرکانسی برای باز سازی یک کپی از شکل موج ورودی ، از طریق مجموع بانک فیلترها یا تبدیل معکوس ، ا ستفاده می شوند.

۱،۱،۱،۱،۱،۱،۱،۲،۲– کدینگ زیر باندی و تبدیل تطبیقی

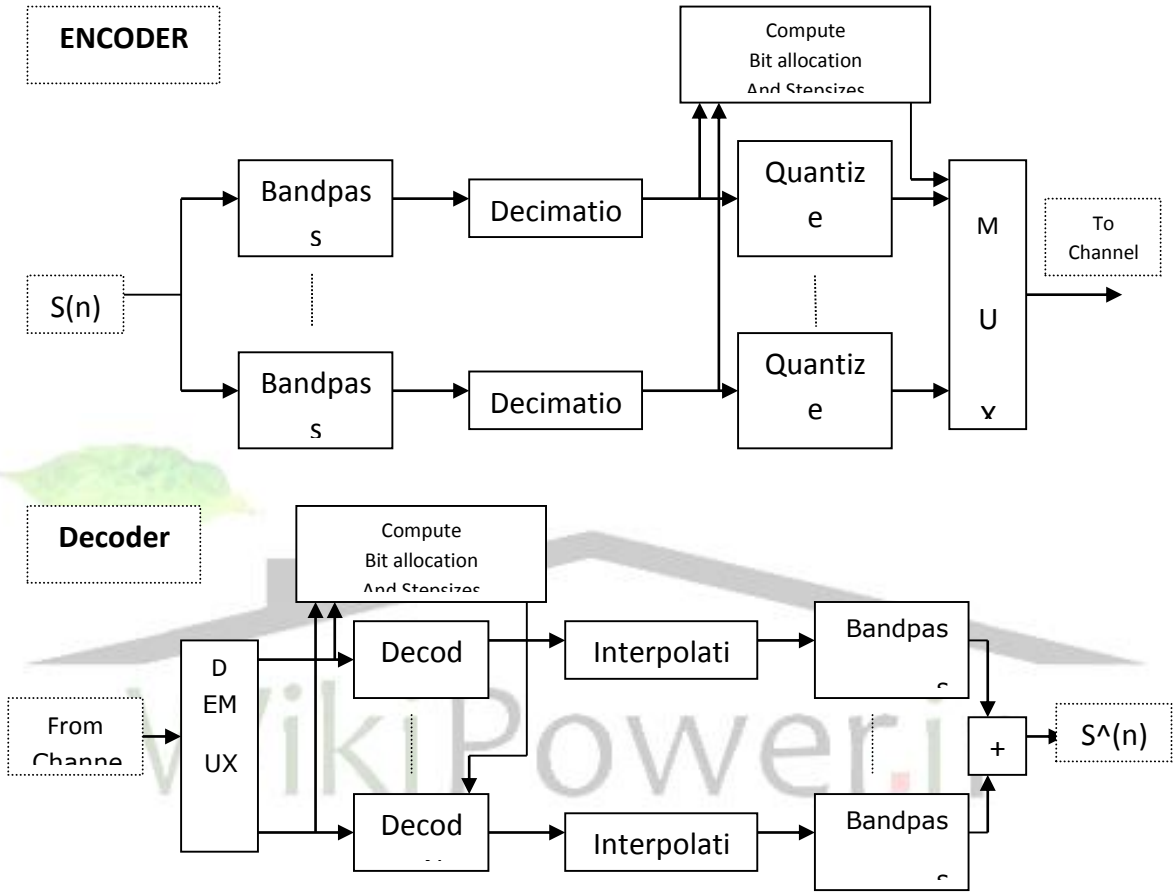
دو روش معروف کدینگ صحبت در حوزه فرکانس ، کدینگ زیر باندی (SBC[1]) و کدینگ تبدیل تطبیقی (ATC[2]) هستند . اصول پایه در هر دو روش تقسیم طیف صحبت ورودی به تعدادی باند فرکانسی است که هر یک جداگانه کد می شوند . درSBC یک بانک فیلتر به صحبت ورودی اعمال می شود تا آنرا به ٤تا١٦ باند فرکانسی تقسیم نماید. ATC یا بعبارت دیگر روش بلوک تبدیل ، تبدیلی به اندازه نوعی ۱۲۸تا ۲٥٦ نمونه را بکار می برد تا تفکیک فرکانسی بهتری را فراهم آورد .

در کدینگ زیر باندی بعد از تقسیم طیف صحبت ورودی به تعدادی زیر باند ، هرزیر باند چندتایکی[3]،کوانتیزه ، کد و مالتی پلکس و سپس ارسال می گردد. در گیرنده این زیر باندها دیمالتی پلکس ، دیکد و بوسیله درونیابی[4] به فرکانس اولیه خود برمی گردند . سیگنالهای زیر باندی حاصل

1 Sub-Band Coding
2 Adaptive Transform Coding

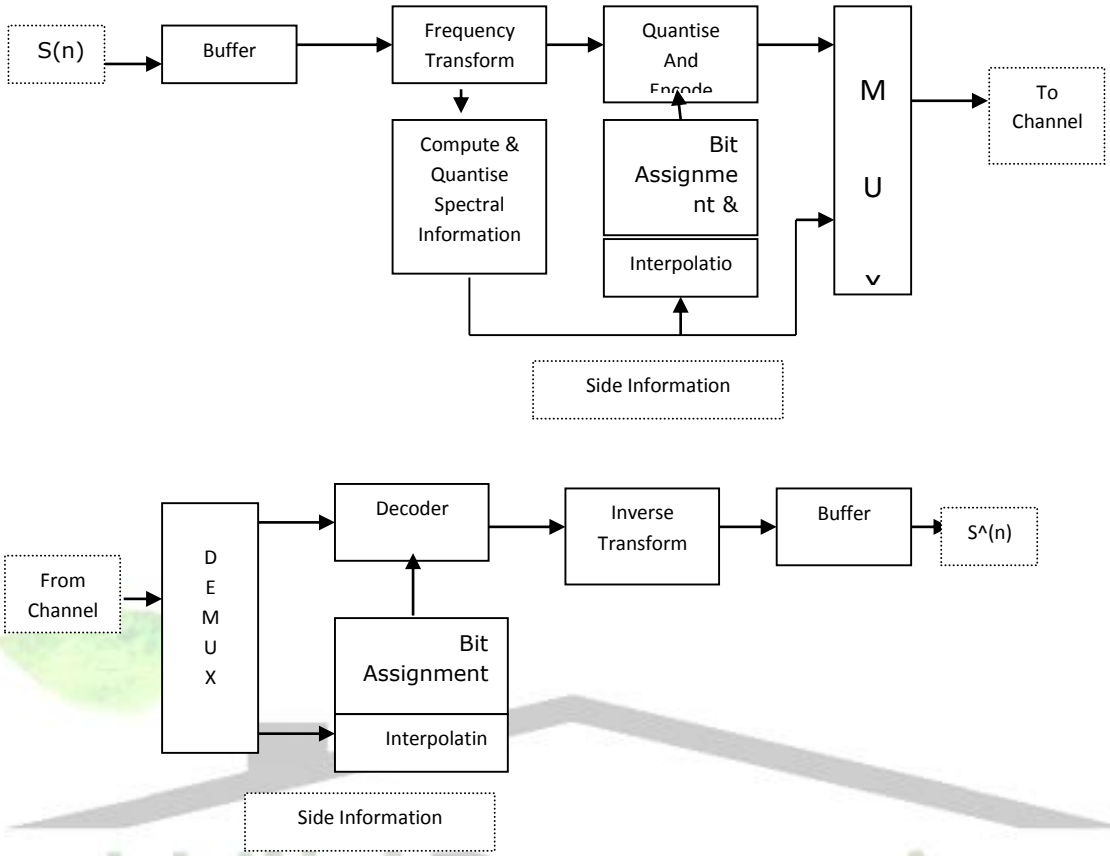
1- Decimation
2- Interpolation

سپس با هم جمع می شوند تا تقریبی از صحبت اصلی را بوجود آورند . در شکل (۲-٤) بلوک دیاگرام اینکدر و دیکدر زیر باندی نشان داده شده است .



شکل(۲-٤) بلوک دیاگرام یک کدر زیر باندی نمونه

کدرهای تبدیل تطبیقی (ATC) یک روش پیچیده تر آنالیز حوزه فرکانس هستند که شامل بلوکی از تبدیل های قطعات پنجره شده صحبت ورودی می باشند . هر قطعه با یک دسته از ضرائب تبدیل نمایش داده می شود که جداگانه کوانتیزه و ارسال می گردند .درگیرنده این ضرائب کوانتیزه و تبدیل معکوس شده تا یک کپی از قطعه اصلی را باز سازی نماید. سپس قطعات مجاور به هم متصل شده تا صحبت سنتز شده را شکل دهند . در شکل (۲-۵) بلوک دیاگرام یک کدر ATC نشان داده شده است .



شکل (۲-۵) : بلوک دیاگرام یک کدر تبدیل تطبیقی

کدرهای SBC وATC که در بالا بحث شدند، صحبت با کیفیت بالا در 16kpbs تولید می کنند اما کیفیت صحبت در نرخ بیت حدود 8kbps بدلیل عدم حضور پیشگویی Pitch افت می کند بنابراین محدوده عملیاتی آنها را 16kpbs تا 9.6kbps در نظر می گیرند .

۱,۱,۱,۱,۱,۱,۱,۱,۲,۳

ب۱،۱،۱،۱،۱،۱،۱،۲،٤) کدرهای مختلط حوزه زمان

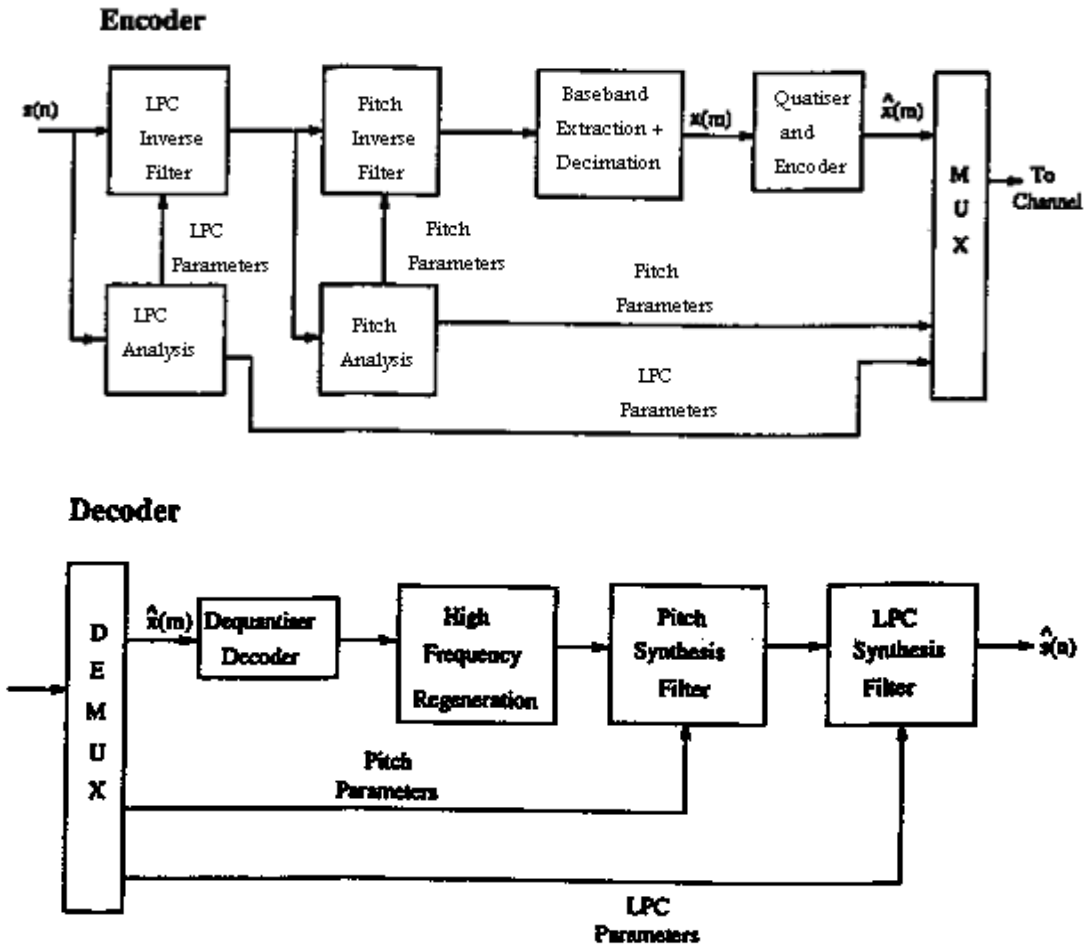
کدرهای مختلط حوزه زمان از روش پیشـــگویی خطی اســـتفاده می کنند . با بکار گیری یک مدل منبع-فیلتر که فرض می کند صـــحبت نتیجه تحریک یک فیلتر متغیر با زمان بوســـیله یک قطار پالس پریودیک برای اصوات باواک و یا یک منبع نویز تصادفی برای اصوات بی واک است ، می توان مشخصات آماری سیگنال صحبت را بسیار دقیق مدل کرد.

روش کدینگ پیشـــگویی تطبیقی(APC[1]) که در شـــکل (۲-٦) نشـــان داده شـــده اســـت ، در اصـــل توســـط schroeder , Atal ابداع شده ا ست واز پیـشگویی های زمان کوتاه و زمان بلند هردودرآن ا ستفاده شده است[1]. سیگنال تحریک حاصل بعد از فیلتر معکوس شدن به صورت نمونه به نمونه کوانتیزه می شود. روش APC برای نرخ بیت حدود 16 kbps طراحی شده و در سی ستم اینمار ست B ازAPC 16kbps به همراه کدینگ کانال Reed-Solomon استفاده شده است[۵].

عملکرد APC در نرخ بیت های پایین افت می کند زیرا بیشتر ظرفیت کدینگ آن صرف کدینگ سیگنال مانده می شـــود . به منظور کاهش ظرفیت مورد نیاز برای کد کردن ســـیگنال مانده ، کدرهایی با تحریک مانده با عنوان RELP[2] مورد بررسی قرار گرفتند .

[1] Adaptive Prediction Coding

[2] Regular Excited Linear Prediction



شکل (۲-۷) : بلوک دیاگرام کدر RELP

از اینرو در اینکدر سیگنال باند پایه بو سیله فیلتر پایین گذر و کوانتیزا سیون ا ستخراج می شود . در دیکدر سیگنال باند پایه با ا ستفاده از روش باز سازی فرکانس بالا [۵] به سیگنال تمام باند، تبدیل می شـــود .ویژگی اصـــلی روش RELP ، توانایی کار در شـــرایط خیلی بد نویز زمینه می باشـــد و عملکرد شنیداری خوب آن محدود به 9.6 kbps و بالاتر است .

روشهای کدینگ صحبت که تا کنون بررسی شده اند بر مبنای آنالیز و سنتز هستند . یعنی سیگنال صحبت آنالیز می شود تا پارامترهای سیگنال بدون افزونگی ، از آن ا ستخراج گردد و سپس سیگنال باقیمانده کوانتیزه و ارســـال می گردد و در دیکدر عمل عکس صـــورت می گیرد . این روش کدینگ،

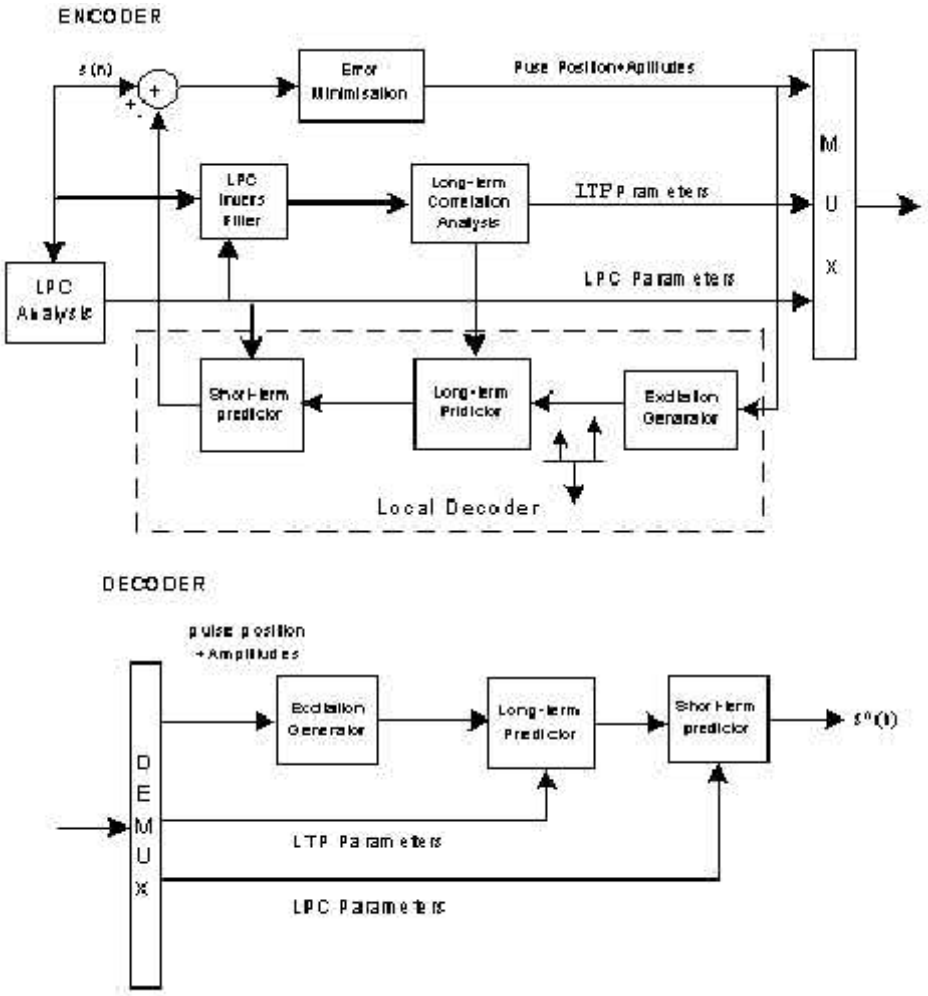
ا ستخراج پارامترها را از پرو سه کوانتیزا سیون جدا می سازد و بنابراین کنترل بر روی اعوجاج ها به کنترل زیر سیستم های جدا از هم محدود می شود .

برای داشـــتن کنترل بهتر بر روی تمام پروســـه کدینگ، یعنی برای مینیم کردن خطای کلی در ســـیگنال صحبت سنتز شده ، رو شهای آنالیز با سنتز(AbS [1]) منظور می شوند . در روش AbS در طرف فر ستنده یک دیکدر محلی وجود دارد که در آن صحبت سنتز شده برای آنالیز موجود می باشد .

روش های AbS-LPC مختلفی وجود دارد و اولین روش گزارش شـده LPC چند پالسـی (MPLPC [2]) می باشد که در شکل (۲–۸) نشان داده شده است . تعیین محل پالسها و دامنه آنها بو سیله یک پرو سه AbS انجام می شـــود . یک نمونه ازMPLPC در ســـرویس هوایی Skyphone با نرخ بیت 9.6kbps و کدینگ FEC کانولو شنال با نرخ $\frac{1}{2}$ ، بکار گرفته شده ا ست . عیب عمده MPLPC بار محا سباتی نسبتا زیاد آن است. ترکیبی از MPLPC و RELP که برای پیاده سازی ساده تر باشد توسط Kroon گزارش شده که به LPC با تحریک پالس منظم [RPELPC [3]] موسوم است .

تحریک بهینه در این کدر ، پالس هایی با فاصله برابر و دامنه های مختلف می باشد که بار محاسباتی آنرا کاهش می دهد. این الگوریتم کارآمد از نظر محاسباتی ، برای سیستم موبایل GSM از میان بقیه الگوریتم های موجود انتخاب شده است [۵] .

1 Analysis by Synthesis
2 Multi Pulse LPC
3 Regular Pulse Excited LPC



شکل (۲-۸) : بلوک دیاگرام MPLPC استفاده شده در Skyphone

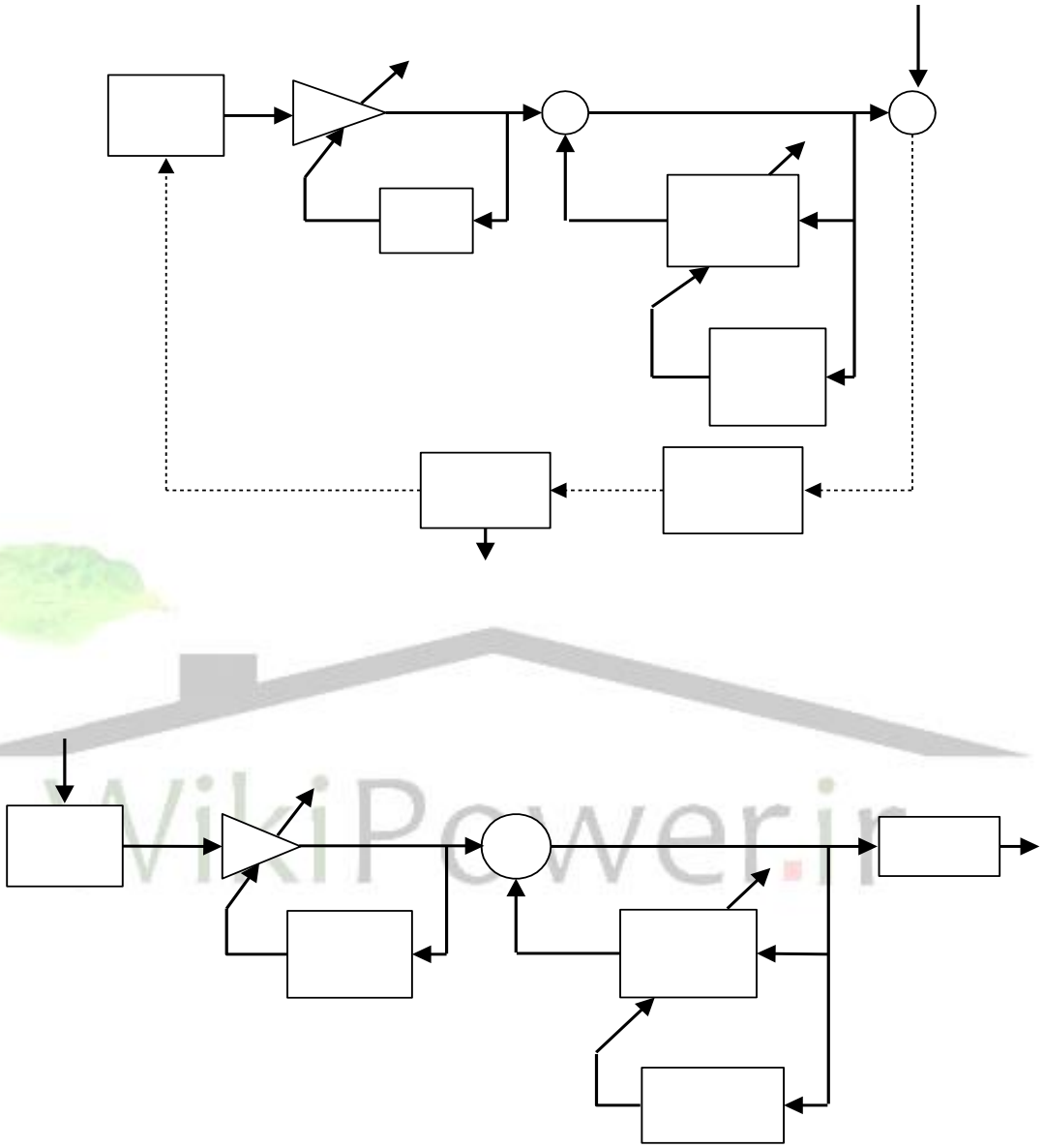
با افزایش قدرت پردازشی چیپ های DSP ، سیستم های AbS کاملی که قبلا برای پیاده سازی غیر عملی بودند مورد توجه قرار گرفتند ، همچون LPC با تحریک کد (CELP [۱] که برای نرخ بیت های پایین بسیار مناسب است . در CELP تحریک به صورت یک بردار است که از کتاب کد تحریک انتخاب می شود . هربردار تحریکی که حداقل خطای وزنی را داشته باشد ، بعنوان تحریک بهینه انتخاب می گردد . پروسه AbS درCELP

[۱] Code Excited Linear Prediction

بار محا سباتی زیادی دارد. هر چند که CELP پیچیده ا ست اما قادر به تولید صحبت با کیفیت عالی حتی در نرخ بیت های پایین می باشد . یک گونه از CELP بعد ازبررسی روشهای مختلف کدینگ ، برای سیستم مخابراتی صدا در وزارت دفاع آمریکا با عنوان 4.8 kbps DOD انتخاب شده تا جانشین کدر قدیمی 2.4kbps LPC-l0e شود [۵].

هر چند CELP عمدتا برروی نرخ بیت های پایین متمرکز است ، برای نرخ بیت های بالا تر هم مناسب بوده و استاندارد CCITT 16kbps LD-CELP نیز نوعی CELP است . بلوک دیاگرام کدر LD-CELP در شکل (۲-۹) نشان داده شده است .

همانطور که در دیاگرام دیده می شود ، فیلتر طیفی بصورت برگشتی محا سبه می شود و بنابراین لازم نیست که ضرائب فیلتر ار سال شوند . بعلاوه مقیاس کردن یا تنظیم بهره نیز تنها با پیشگویی انجام می شود و نتیجه مستقیم آن این است که نرخ روزآمد شدن تحریک می تواند خیلی بالا باشد (هر ۵ نمونه) و این کدر را قادر می سازد تا تاخیر کدینگ اندک حتی در حدود 2ms داشته باشد . برای یکبار کدینگ در کانالهای نویزی یا بدون نویز ، کیفیت صحبت LD-CELP همانند و یا بهتر از استاندارد G. 721 یعنی 32kbps ADPCM گزارش شده است[6].



شکل (۲-۹) : بلوک دیاگرام کدر LD-CELP

فـصل ۳

کـدر کـم تـــاخـیـر LD-CELP

۳-۱- مقدمه

امروزه با توجه به استفاده روزافزون از شبکه های تلفن و موبایل و نیاز به استفاده موثرتر از ظرفیت شبکه های مخابراتی ، کدرهایی که در نرخ بیت متو سط (16 kbps) ، صحبتی با کیفیت بالا و حداقل تاخیر ممکن تولید نمایند ، مورد توجه خا صی قرار گرفته اند . تاخیر کلی در یک سیستم مخابراتی را می توان ناشی از عوامل زیر دانست :

تاخیر بافر کردن در اینکدر و دیکدر که بعلت جمع آوری نمونه های مورد نیاز آنالیز LPC اســـت . تاخیر پردازش در اینکدر و دیکدر که در اثر پردازش روی نمونه های بافر شـــده می باشـــد . پردازش نمونه های بافر شده باید در فر صت بافر کردن، کامل شود چرا که باید برای پردازش نمونه های بعدی آماده بود . تاخیر ارسال که در سیستم های ماهواره ای بسیار مهم است و قابل کنترل بوسیله کدر صحبت نمی باشد . بنابراین تاخیر کلی سیستم را می توان مجموع عوامل فوق دانست و معمولا حداکثر تاخیر را ٤ برابر تاخیر بافر کردن در نظر می گیرند [۵،۶] . در نتیجه با کنترل طول بافر میتوان تآخیر کدر را

کاهش داد . بعنوان مثال در کدر LD-CELP طول بافر باندازه ۵ نمونه اســـت که با فرض نمونه برداری 8 KHz تاخیر کلی حداکثر 2.5ms می شــود ولی در ســیســتم GSM با طول بافر 20ms تاخیر کلی حداکثر 80ms خواهد بود .

در سال ۱۹۸۸ CCITT برنامه ای را برای استاندارد سازی یک کدر 16kbps آغاز نمود از ویژگی های آن کیفیت بالا و تاخیر اندک بود و برای آن کاربردهای زیادی همچون شــبکه PSTN شــبکه ISDN ، تلفن تصویری و غیره در نظر گرفته شد. در جدول (۳-۱) لیستی از شرایط مورد نیاز CCITT برای کدر مورد نظر آورده شده است[۵] .

شــرط تاخیر کمتر از 5ms مســأله ای بود که هیچ یک از کدرهای موجود قادر به برآوردن آن نبودند . از این رو کدرLD-CELP در سال ۱۹۹۱ بدین منظور معرفی شد و توانست همه شرایط CCITT را برآورده سازد و حتی با 32kbps ADPCM رقابت کند . این کدر در سال ۱۹۹۲ بصورت استاندارد CCITT در آمد و در توصیه نامه G.728 ITU-T(CCITT) ارائه شد [۸].

Parameter	CCITT requirements	CCITT objective
Coding Delay	<= 5 ms	<= 2 ms
Quality at BER = 0	Distortion< 4 qdu	
Quality at BER=10^-3	No worse than G.721*	
Quality at BER =10^- 3	No worse than G.721	
TAndeming	3 Async Tandem with distortion <= 14 qdu	Sync Tandem without distortion accumulation
Signaling Tones	DTMF	

* G.721 : CCITT standard for 32kbps ADPCM

جدول (۳-۱) : شرایط CCITT برای کدر 16 kbps

۳-۲- بررسی کدر کم تاخیر LD-CELP

در کدرهای معمولی CELP که تطبیقی مستقیم هستند ، پارامترهای پیشگویی بهمراه بردار تحریک به گیرنده ار سال می شوند ولی در LD-CELP فقط سیگنال تحریک ار سال می گردد . ضرائب پیشگویی از طریق انجام آنالیز LPC بر روی نمونه های صحبت قبلی ، روز آمد می شوند. بنابراین کدر LD-CELP در اصل یک نسخه تطبیقی معکوس از کدر CELP معمولی می باشد و اساس CELP که جستجوی کتاب کد به روش آنالیز با سنتز است در آن حفظ شده است .در شکل (۳-۱) بلوک دیاگرام ساده شده اینکدر و دیکدر LD-CELP نشان داده شده است.

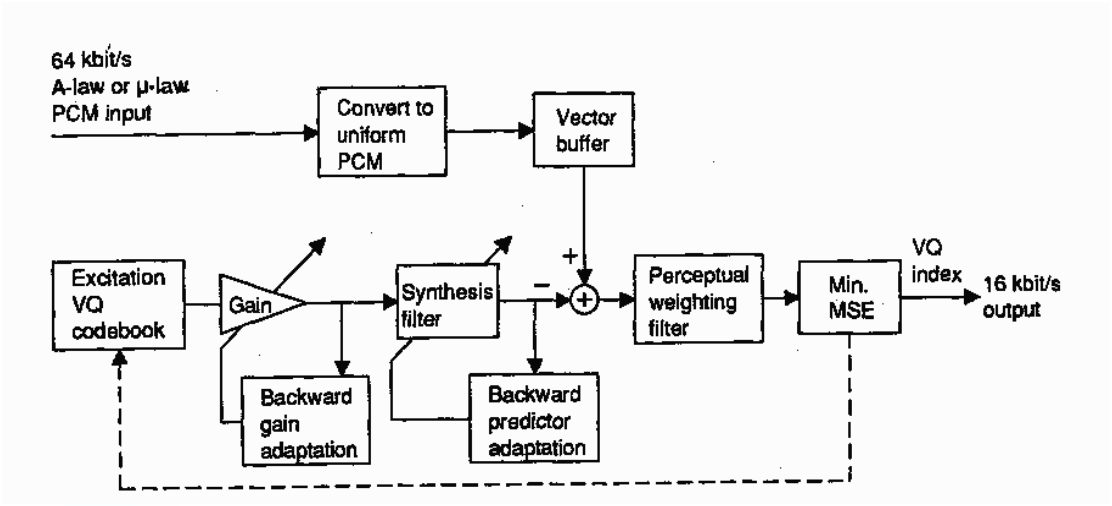
بردار تحریک در اینجا به اندازه ۵ نمونه است و پیشگویی زمان بلند رایج در CELP با پیشگویی زمان کوتاه مرتبه بالای ۵۰ ، برروی نمونه های صحبت کوانتیزه که ضرائب آن برای هر ٤ بردار تحریک روز آمد می شوند ، تعویض شده است .

بهره تحریک با استفاده از پیشگویی خطی تطبیقی مرتبه ۱۰ بصورت لگاریتمی، برای هر بردار روز آمد می شود .ضرائب این پیشگویی لگاریتمی بهره ، هر ٤ بردار یک بار با انجام آنالیز LPC بر روی بهره های لگاریتمی بردارهای تحریک قبلی روز آمد می گردد.

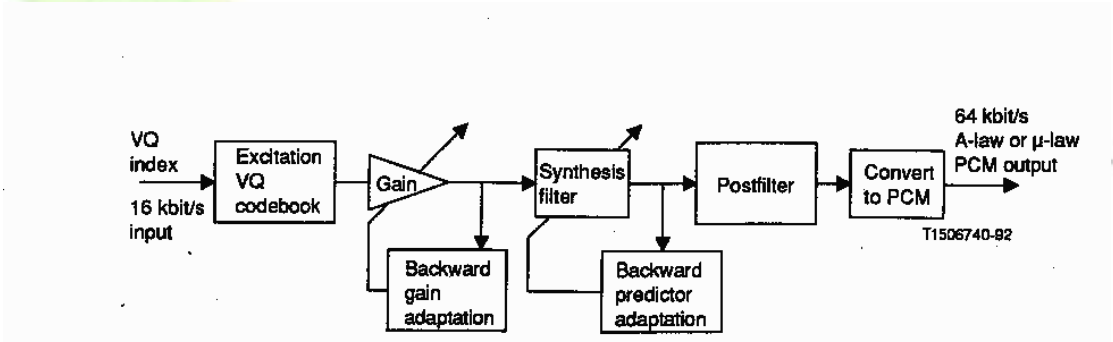
فیلتر مرتبه ۱۰ شنیداری نیز در اینکدر لحاظ شده که هر٤ بردار تحریک یک بار با انجام آنالیز LPCروی نمونه های صحبت اصلی در اینکدر روزآمد می شود.

کتاب کد کوانتیزه برداری از کتاب کدهای ۳ بیتی بهره و ۷ بیتی شکل[۱] در 16kbps تشکیل شده و تنها اندیس ۱۰ بیتی این کتاب کد به گیرنده ارسال می شود .

[۱] Shape



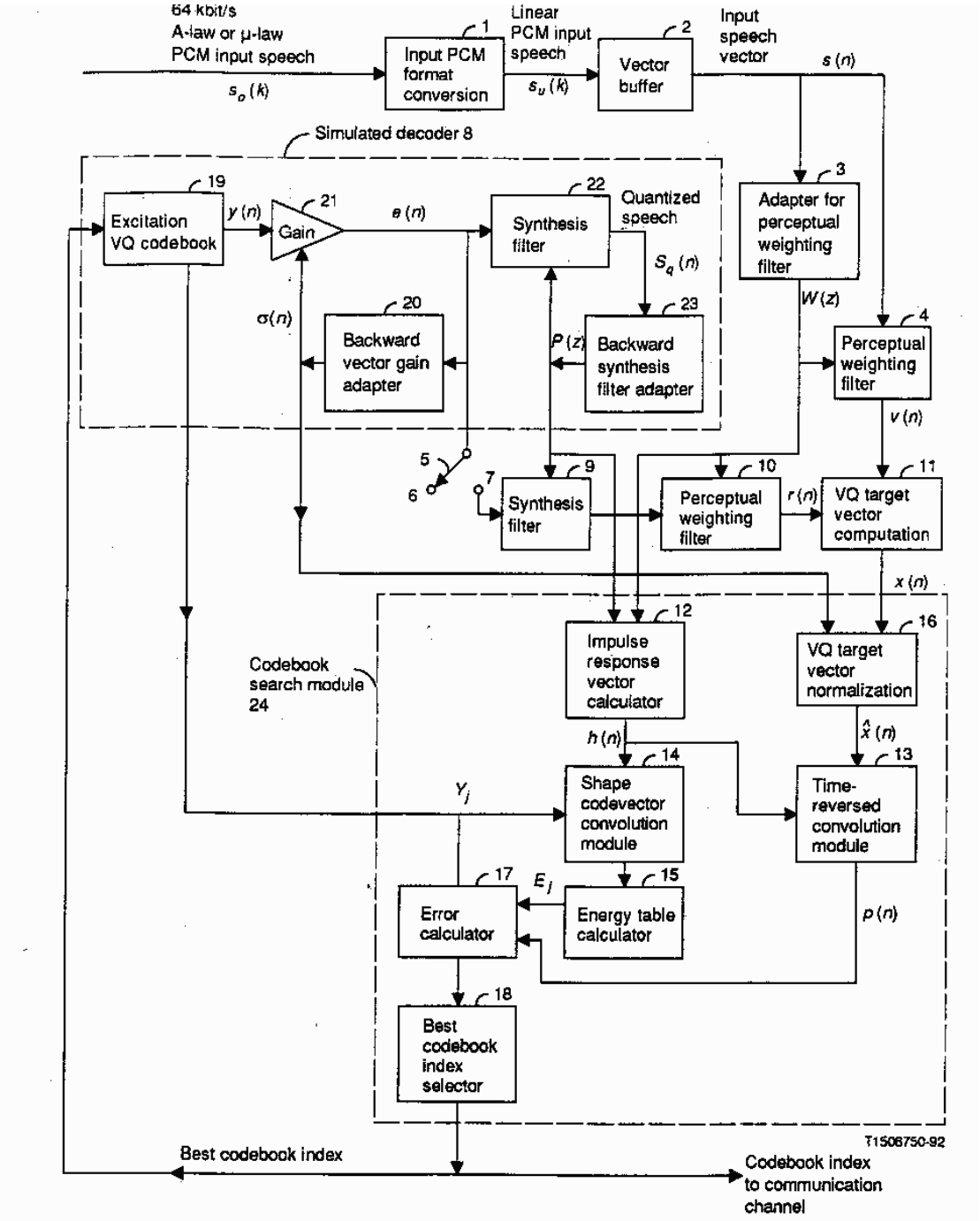
Encoder



Decoder

شکل (۳-۱) : بلوک دیاگرام ساده شده 16kbps LD-CELP

در شکل (۳-۲) بلوک دیاگرام اینکدر LD-CELP ، با جزئیات بیشتری نشان داده شده است. دراین قسمت به بررسی بیشتر آن می پردازیم .



شکل (۳-۲) : طرح اینکدر LD-CELP

۳-۲-۱- LPC معکوس مرتبه بالا

از آنجا که پیشگویی زمان بلند pitch (LTP[1]) ، سیستم را خیلی به خطاهای کانال حساس می سازد ، در الگوریتم LD-CELP , LTP حذف شده و بجای آن از LPC معکوس مرتبه بالا ا ستفاده شده ا ست .

کاربرد LPC مرتبه ۵۰ دارای امتیازهای زیر است :

- حساسیت خیلی پایین کدر به خطاهای کانال
- اطلاعات جنبی دیگر ارسال نمی شوند .
- با حذف LTP ، الگوریتم کمتر به صحبت وابسته می شود و قادر به دریافت داده باند صوتی می شود .

البته استفاده از LPC مرتبه ۵۰ ، باعث افزایش تعداد محاسبات کورلیشن و پیچیدگی زیاد الگوریتم می شود و برای کاهش پیچیدگی از روش پنجره بازگشتی Branwell استفاده شده است [5].

هر چند که این روش باعث کاهش پیچیدگی می شود ولی بخاطر عملیات ضرب باعث بروز مسائلی در پیاده سازی ممیز ثابت می شود از اینرو از یک پنجره مختلط جدید استفاده می شود .

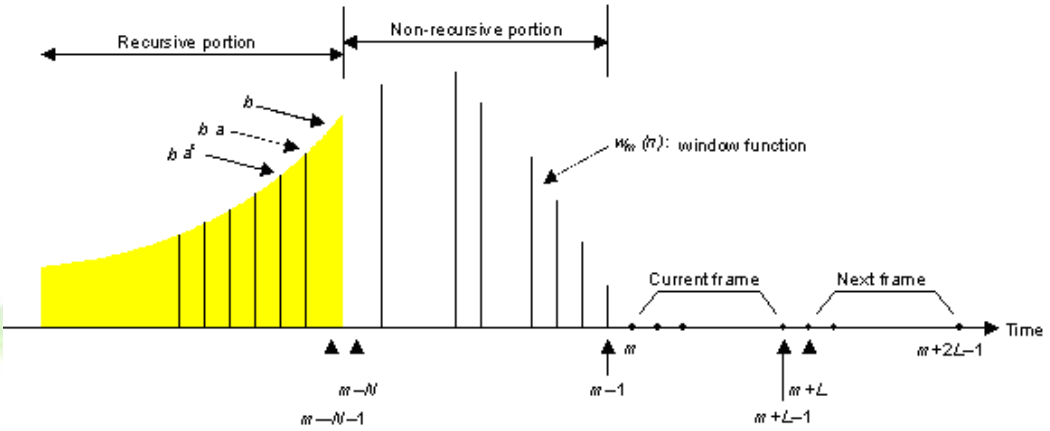
فرض کنید که آنالیز LPC برای هر L نمونه صحبت انجام شود و نمونه های مطابق با سیکل تطبیق فعلی $\hat{S}_u(m),\hat{S}_u(m+1),\ldots,\hat{S}_u(m+L-1)$ باشند و پنجره مختلط به نمونه های قبلی با اندیس کمتر از m اعمال شود ، در زمان m تابع پنجره مختلط $W_m(k)$ بصورت زیر تعریف می شود :

[1] Long Term Prediction

$$W_m(k)=\begin{cases} f_m(k)=b\alpha^{-[k-(m-N-1]};k\le m-N-1 \\ g_m(k)=-\sin[c(k-m)];m-N\le k\le m-1 \\ 0;k\ge m \end{cases}$$

که در آن $0\langle b\langle 1$ و $0\langle \alpha\langle 1$ و N طول بخش غیر بازگشتی پنجره هستند .

در شکل (۳-۳) نمایش پنجره مختلط نشان داده شده است[8] .



شکل(۳-۳): پنجره مختلط

صحبت پنجره شده بصورت زیر محاسبه می شود :

$$S_m(k)=\hat{S}_u(k)W_m(k)=\begin{cases} \hat{S}_u(k)f_m(k);k\le m-N-1 \\ \hat{S}_u(k)g_m(k);m-N\le k\le m-1 \\ 0;k\ge m \end{cases}$$

فیلترLPC مرتبه M ام از M+1 ضریب اتوکورلیشن بصورت زیر استفاده می کند :

$$R_m(i)=\sum_{k=-\infty}^{m-1}S_m(k)S_m(k-i)=r_m(i)+\sum_{k=m-N}^{m-1}S_m(k)S_m(k-i)$$

که در آن $r_m(i)$ ، بخش بازگشتی R $m(i)$ چنین است :

$$r_m(i)=\sum_{k=-\infty}^{m-N-1}S_m(k)S_m(k-i)=\sum_{K=-\infty}^{m-N-1}\hat{S}_u(k)\hat{S}_u(k-i)f_m(k)f_m(k-i)$$

فرض کنید که مقادیر $r_m(i)$ در فریم حا ضر معلوم هستند و ما می خواهیم، فریم بعدی را محا سبه کنیم که از $\hat{S}_u(m+L)$ شروع می شود.صحبت پنجره شده در این حالت:

$$S_{m+L}(k) = \hat{S}_u(k) W_{m+L}(k)$$

و بخش بازگشتی $R_{m+L}(i)$ را می توان بصورت زیر نوشت:

$$\text{یا } r_{m+L}(i) = \sum_{K=-\infty}^{m+L-N-1} S_{m+L}(k) S_{m+L}(k-i)$$

$$r_{m+L}(i) = \sum_{k=-\infty}^{m-N-1} S_{m+L}(k) S_{m+L}(k-i) + \sum_{k=m-N}^{m+L-N-1} S_{m+L}(k) S_{m+L}(k-i)$$

با قرار دادن , $\hat{S}_u(k)$ و تابع پنجره در بخش اول داریم :

$$r_{m+L}(i) = \alpha^{2L} r_m(i) + \sum_{k=m-N}^{m+L-N-1} S_{m+L}(k) S_{m+L}(k-i)$$

و سرانجام ضرایب اتوکورلیشن بصورت زیر محاسبه می شوند:

$$R_{m+L}(i) = r_{m+L}(i) + \sum_{k=m-N}^{m+L-N-1} S_{m+L}(k) S_{m+L}(k-i)$$

بعد از محاسبه ضرایب اتوکوریشن با N=35 و $\alpha = (\frac{3}{4})^{\frac{1}{40}}$ و L=20 ، از الگوریتم Durbin برای محاسبه پارامترهای LPC استفاده می کنیم [8]. همچنین ضریب تحصیح نویز $\frac{257}{256}$ را برای اصلاح R(0) بکار می بریم که معادل اضافه کردن نویز سفید 24dB زیر سطح متوسط صحبت می باشد. این کار باعث پر شدن چاله های طیفی با نویز سفید وکاهش رنج دینامیک طیفی می شود. قبل از بکارگیری

پارامترهای LPC محاسبه شده ، آنها را با ضریب گسترش پهنای باند $\lambda = \frac{253}{256}$، بصورت زیر اصلاح می کنیم:

$$\hat{a}_i = \lambda^i a_i, i = 0,1,...49$$

که در آن a_i و $\hat{a}_i$ به ترتیب مقدار اولیه و گسترش یافته پارامترهای LPC می باشند[5] .

۳-۲-۲-فیلتر وزنی شنیداری

فیلتر وزنی شنیداری استفاده شده در LD-CELP (بلوک ۱۰ در شکل (۳-۲)) ،شبیه فیلتر شکل دهنده نویز به فرم کلی زیر است:

$$W(Z) = \frac{1 + \sum_{i=1}^{10} q_i \gamma_1{}^i Z^{-i}}{1 + \sum_{i=1}^{10} q_i \gamma_2{}^i Z^{-i}}$$

که در آن γ_1 و γ_2 ضرایب تنظیم هستند که به ترتیب برابر 0.9 و 0.6 قرار می گیرند. ضرایب q_i فیلتر به همان روش ضرایب فیلتر سنتز ، با M=10 ، L=20 و N=30 و $\alpha = (\frac{1}{2})^{\frac{1}{40}}$ محاسبه می شوند .

همچنین ضریب تصحیح نویز $\frac{257}{256}$ نیز در اینجا اعمال می شود.

۳-۲-۳- ساختار کتاب کد

در شکل (۳-۲) بلوک های ۱۲تا۱۸ تشکیل دهنده بلوک جستجوی کتاب کد (۲٤) هستند . این بلوک ، ۱۰۲٤ بردار تحریک کتاب کد کوانتیزه برداری (VQ) بلوک ۱۹ را بررسی کرده و اندیس

بهترین بردار که بردار صحبت کوانتیزه متناظر آن ، به بردار صحبت ورودی از همه نزدیکتر باشد را مشخص می سازد .جهت ساده شدن جستجو ، کتاب کد ۱۰۲٤ مدخلی ۱۰ بیتی به دو کتاب کد کوچکتر تقسیم می شود: یک کتاب کد شکل ۷بیتی که شامل ۱۲۸ بردارکد مستقل ویک کتاب کد بهره ۳ بیتی شامل ۸ مقدار اسکالر متقارن نسبت به صفر (یعنی ۱ بیت برای علامت و ۲ بیت برای اندازه) . بردار کد نهایی حاصلضرب بهترین بردار شکل و بهترین سطح بهره خواهد بود .

۳-۲-۳-۱- جستجوی کتاب کد

بلوک جستجوی ۲٤ درواقع هر بردار کد را در بهره تحریک کنونی $\sigma(n)$ ضرب کرده و از فیلتر های متوالی سنتز F(z) و وزن دهنده شنیداری W(z) عبور می دهد .

فیلتر کردن بردارهای کد VQ را میتوان بدین صورت بیان کرد : فرض کنید که y j ، j امین بردارکد در کتاب کد شکل ۷ بیتی و gi ،i امین سطح در کتاب کد بهره با شد . اگر h(n) پا سخ ضربه فیلتر متوالی H(z)=F(z)W(z) با شد ، وقتیکه بردارکد م شخص شده با اندیس های jو i به H(z) اعمال شود خروجی فیلتر بصورت زیرخواهد شد :

$$\tilde{x}_{ij} = H\sigma(n)g_i y_j$$

که در آن

$$H = \begin{bmatrix} h(0) & 0 & 0 & 0 & 0 \\ h(1) & h(0) & 0 & 0 & 0 \\ h(2) & h(1) & h(0) & 0 & 0 \\ h(3) & h(2) & h(1) & h(0) & 0 \\ h(4) & h(3) & h(2) & h(1) & h(0) \end{bmatrix}$$

$$D = \left\| x(n) - \tilde{x}_{ij} \right\|^2 = \sigma^2(n) \left\| \hat{x}(n) - g_i H y_j \right\|^2$$

بلوک ۲٤ بدنبال ترکیبی از jو i می گردد که خطای میانگین مربعی (MSE) زیر را مینیمم سازد :

که در آن $\hat{x}(n) = x(n)/\sigma(n)$ بردار هدف نرمالیزه بهره می باشد :

$$D = \sigma^2(n)\left[\| \hat{x}(n) \|^2 - 2 g_i \hat{x}^t(n) H y_j + g_i^2 \| H y_j \|^2 \right]$$

چون عبارت های در طول جستجوی کتاب کد ثابت هستند باید :

$$\hat{D} = -2 g_i p^t(n) y_j + g_i^2 E_j$$

مینیمم شود که در آن :

$$p(n) = H^t \hat{x}(n), E_j = \| H y_j \|^2$$

توجه کنید که Ej انرژی j امین بردار شکل است و به بردار هدف $\hat{x}(n)$ بستگی ندارد . همچنین بردار شکل yj وماتریس H تنها به فیلترسنتز و فیلتر وزنی بستگی دارد که در دوره ٤ بردار صحبت ثابت است ودر نتیجه Ej هم در این دوره ثابت می ماند .

برای کاهش بیشتر محاسبات می توان آرایه های bi =2 gi و g_i^2 (i=0,1,...,7) Ci= را ازقبل محاسبه و ذخیره ساخت .

در نتیجه :

$$\hat{D} = -b_i P_j + c_i E_j$$

که در آن :

$$P_j = p^t(n) y_j$$

بنابر این برای جستجوی کتاب کد ابتدا باید برای هر بردار شکل y j ، بهترین اندیس بهره i که $\hat{D}_i$ را مینیمم سازد مشخص کرده و سپس این کار را برای همه y j ها انجام داده تا در نهایت $\hat{D}_{\min}$ مربوطه بدست آید . $i_{\min}$, $j_{\min}$

۳-۲-٤- شبه دیکدر

تا کنون کدکننده بهترین اندیس کتاب کد تحریک را ارسال داشته ولی هنوز بعضی کارهای آماده سازی برای کدینگ بردارهای بعدی باقی مانده است .ابتدا این اندیس به کتاب کد VQ تحریک اعمال شده تا بهترین بردار تحریک $y_{i\min}$ یا $y_{j\min}$ بدست آید . سپس این بردار در بهره تحریک کنونی $\sigma(n)$ ضرب می شود (بهره از (۲۱) بدست آمده است) . بردار تحریک بدست آمده برابر است با :

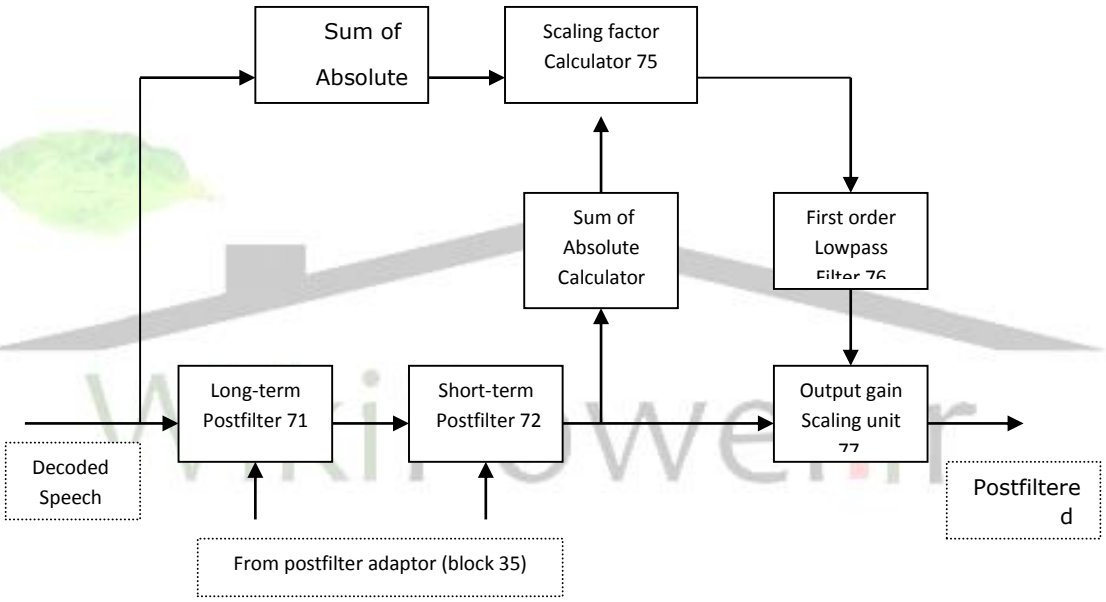
$$e(n) = \sigma(n)y(n)$$

بردار تحریک از فیلتر سنتز (۲۲) عبور می کند تا بردار صحبت کوانتیزه فعلی $S_q(n)$ تولید کند . توجه کنید که بلوک های ۱۹ تا ۲۳ شبه دیکدر(۸) را شکل می دهند . ازاین رو بردار صحبت کوانتیزه $S_q(n)$ برابر صحبت کوانتیزه شده شبیه سازی در کانال بدون خطا است .

در شکل (۳-۲) تطبیق دهنده فیلتر سنتز (۲۳) ، بردار $S_q(n)$ را برای تجدید ضرائب فیلتر سنتز نیاز دارد. همچنین تطبیق دهنده بهره (۲۰)، بردار تحریک e (n) را جهت تجدید ضرائب log-gain لازم دارد .

۱-۲-۳-۲-۵-پست فیلتر[۱]

این بلوک جهت افزایش کیفیت شنیداری ، صحبت دیکد شده را فیلتر می کند . جزئیات بیشتر این بلوک در شکل (۳-٤) نشان داده شده است . پست فیلتر در اصل شامل ۳ بخش است : پست فیلتر زمان-بلند ۷۱ ، پست فیلترزمان-کوتاه ۷۲ و واحد مقیاس کننده بهره خروجی ۷۷ . بقیه بلوکهای شکل (۳-٤) برای محاسبات بلوک ۷۷ هستند.



شکل (۳-٤) : طرح بلوکی پست فیلتر

پست فیلتر زمان- بلند (۷۱) که گاهی پست فیلتر pitch نامیده می شود ، یک فیلتر شانه ای[۲] است که پیکهای طیفی آن در مضارب فرکانس pitch صحبتی که باید پست فیلتر شود ، واقع شده

[۱] Postfilter
[۲] Comb

است . پریود pitch را میتوان از صحبت دیکد شده با استفاده از یک آشکارساز pitch استخراج کرد. اگر p پریود pitch (به نمونه) بدست آمده از آشکارساز pitch باشد ، آنگاه تابع انتقال پست فیلتر زمان-بلند بصورت زیر میتواند بیان شود :

$$H(z) = g_l(1 + b\,z^{-p})$$

که درآن ضـرائب gl ،b, وپریود pitch در تطبیق دهنده پسـت فیلتر سـاخته ودر هر ٤بردار (یک فریم) تجدید می شوند .

پست فیلتر زمان-کوتاه (۷۲) شامل یک فیلتر مرتبه۱۰ صفر-قطب بدنبال یک فیلتر تمام-صفر مرتبه ۱ است . فیلتر مرتبه ۱۰ صفر-قطب ، مؤلفه های فرکانسی بین پیک های فرمنت را تضعیف می کند . در حالیکه فیلتر مرتبه اول تمام- صفر می کوشد تا پاسخ فرکانسی فیلتر صفر-قطب مرتبه ۱۰ را هموار سازد .

فرض کنید که ai ,i=1,2,...,10 ضرائب LPC مرتبه ۱۰ و K1 اولین ضریب انعکاس حا صل از آنالیزمعکوس LPC صحبت دیکد شده با شند آنگاه ai و K1 را میتوان باهم از آنالیز PC معکوس مرتبه ۵۰ (بلوک ۳۳ در شـکل(۳-۲)) بدسـت آورد . فقط باید تکرار Durbin مرتبه ۵۰ را در مرتبه ۱۰ متوقف کرده وضرائب فوق را ذخیره کرد و

سپس تکرار Durbin را از مرتبه ۱۱ تا ۵۰ ادامه داد .

تابع انتقال پست فیلتر زمان-کوتاه بصورت زیر است :

$$H_s(z) = \frac{1 - \sum_{i=1}^{10} b_i z^{-i}}{1 - \sum_{i=1}^{10} a_i z^{-i}} \left[1 + \mu z^{-1}\right]$$

$$b_i = \tilde{a}_i (0.65)^i, i = 1,2,...,10$$

$$a_i = \tilde{a}_i (0.75)^i, i = 1,2,...,10$$

$$\mu = 0.15 K_1$$

که در آن ضرائب ai ,bi در هر فریم یک بار و در اولین بردار آن تجدید می شوند . در حالت کلی بعد از اینکه صحبت دیکد شده از پست فیلتر زمان-بلند و زمان-کوتاه می گذرد ، صحبت فیلتر شده همان انرژی اولیه را نداردو بدین منظور بلوکهای ۷۳ تا ۷۷ کار کنترل اتوماتیک بهره وحفظ انرژی صحبت را انجام می دهند .

فصل ٤

شبیه سازي ممیزثابت الگوریتم به زبان C

٤-١- مقدمه

شبیه سازی یک الگوریتم بر روی کامپیوتر بو سیله یک زبان سطح بالا مانند C،گام نخ ست جهت پیاده سازی آن الگوریتم برروی DSP ب شمار می رود . همانطور که در ف صل ٥ بیان خواهد شد، با توجه به پیاده سازی کدک G.728 برروی DSP های ممیزثابت ، لازم ا ست که ابتدا آلگوریتم کدک را بصورت ممیزثابت شبیه سازی نماییم. در این فصل ابتدا ویژگی های برنامه نویسی ممیز ثابت را شرح میدهیم . سپس ساده سازی محاسبات برای برنامه نویسی ممیزثابت را بیان نموده و پس از آن بلوک ها ومتغییرهای برنامه را به اختصار شرح می دهیم. در قسمت آخر هم روندنمای برنامه اینکدر و دیکدر را توصیف می کنیم .

٤-٢- ویژگی های برنامه نویسی ممیز ثابت

٤-٢-١- نمایش اعداد

واحد پایه در پیاده سازی ١٦ بیتی ، کلمه ١٦ بیتی ا ست که برای اعداد صحیح علامت دار بین ٣٢٧٦٧ تا ٣٢٧٦٨- تغییر می کند و بصورت مکمل ٢ ذخیره می گردد. برای نمایش اعداد اعشاری یک نقطه اعشاری بایدبین دوبیت کلمه در نظر گرفته شود .مثلاً برای نمایش اعداد بین 1+,1 - بایستی نقطه اعشاری بین بیت های ١٤و١٥ قرار گیرد. این فرمت خاص Q15 نامیده می شود زیرا ١٥ بیت در سمت راست نقطه اعشاری وجود دارد . Qn فرمت بدین صورت تعریف می شود که n بیت در سمت راست نقطه اعشاری وجود داشته باشد . اعداد صحیح بصورت Q0 نمایش داده می شوند.

برخی داده ها به دقت بیشتری نیاز دارند از اینکه با یک کلمه ١٦ بیتی نمایش داده شوند . مانند رجیسترهای ضرب و اکومولاتور در چیپ های DSP که می توانند اعداد با دقت از ١ تا 2^{31} را نمایش دهند که به دقت مضاعف معروف است .

برخی داده ها رنج وسیعتری دارند از اینکه بتوان با یک فرمت ١٦ بیتی ثابت نمایش داده شوند . شاید ١٦ بیت دقت کافی باشد ولی باید از مقیاس کردن دینامیک استفاده شود. این نوع داده ها را می توان بصورت ممیز شناور و با دقت معمولی نشان داد . یعنی اینکه داده با ٢ کلمه نشان داده می شود، اولین کلمه حاوی عددی است که اندازه آن بین ١٦٣٨٤و٣٢٧٦٧ می باشد . این مانتیس مقدار است و می گوئیم که مقدارش در فرمت نرمالیزه نمایش داده می شود اگر مقدار مثبت باشد ، بیت ١٤ مانتیس ١ است . کلمه دوم حاوی تعداد شیفت به چپ (NLS [1]) هایی است که بکار رفته تا عدد را در مقدار نرمالیزه اش قرار دهد .

[1] Number of Left Shift

بنابراین کلمه دوم مشخص کننده Q فرمت مانتیس می باشد. اگر این فرمت برای یک مقدار منفرد بکار رود ممیز شناور اسکالر نامیده می شود . همچنین امکان دارد که یک آرایه از n مقدار را با n+1 کلمه، با استفاده از ممیز شناور بلوکی نمایش داد. با استفاده از این فرمت ، بزرگترین مقدار در آرایه باید به همان صورتی که برای ممیز شناور اسکالر گفته شد ، نمایش داده شود . بقیه مقادیر در آرایه باید از همان NLS استفاده کنند و مانتیس آنها لزوما نباید در فرمت نرمالیزه باشد. گسترش این نمایش بصورت ممیز شناور بلوکی قسمت شده می باشد . در این حالت یک آرایه mn مقداری بوسیله m(n+1) کلمه نمایش داده می شود . این آرایه تقسیم می شود به m زیر آرایه با اندازه n ،که هر زیر آرایه بصورت ممیز شناور بلوکی با n کلمه برای اندازه ها و ۱ کلمه برای NLS نمایش داده می شود .

نوع دیگری از نمایش بکار رفته بصورت ممیز شناور با دقت مضاعف است. در این حالت اعداد صحیح با دقت مضاعف بعنوان مانتیس بکار می روند و یک کلمه با دقت معمولی برای NLS استفاده می شود. در مجموع انواع مختلف نمایش ها عبارتند از ممیز ثابت با دقت معمولی، ممیز ثابت با دقت مضاعف برای اکومولاتورها و رجیسترهای ضرب، ممیز شناور اسکالر با دقت معمولی و ممیز شناور بلوکی با دقت معمولی و مضاعف.

۴-۲-۲- عملیات حسابی

از آنجا که ضرب دو کلمه ۱۶ بیتی یک عدد ۳۲ بیتی تولید می کند، رجیسترهای ضرب و اکومولاتور ها باید حداقل ۳۲بیتی باشند . در محاسبات مجموع حاصلضرب ها مانند کانولوشن و فیلتر کردن IIR,FIR ممکن است که آکومولاتور سرریز (overflow) شود که بطور جداگانه مشخص می گردد . در فیلتر

کردن IIR مجموع حاصلضربها یا نتیجه عملیات ضرب و انباشت (MAC) بخشی از حافظه فیلتر برای مرحله بعدی است و ۱۶ بیت بالای خروجی در ورودی ضرب کننده استفاده می شود . یک سرریز که موجب می شود تا یک عدد بزرگ مثبت به یک عدد بزرگ منفی تبدیل شود، موجب اختلاف زیادی در خروجی فیلتر می گردد . از اینرو از مد اشباع در فیلترهای IIR استفاده می کنیم . مد اشباع یعنی اینکه اگر کلمه بالایی بزرگتر از ۳۲۷۶۷ یاکوچکتراز۳۲۷۶۸- باشد آنگاه مقدار آن برش می خورد (به این دو حد محدود می گردد).

الف- شیفت و رند کردن

حاصلضرب یک مقدار Qn فرمت در یک مقدار Qm فرمت در رجیستر حاصلضرب بصورت (n+m) Q فرمت و با دقت مضاعف ذخیره می شود . اگر لازم باشد که این نتیجه ذخیره گردد یا با دقت دیگری جمع شود ، آنگاه نتیجه باید شیفت داده شود و یا رند گردد تا به دقت منا سب بر سد . نکته ای که در عملیات شیفت به راست مکمل ۲ وجود دارد اینست که فرض کنید عدد۳ را یک بیت به راست می خواهیم شیفت دهیم، نمایش ۱۶ بیتی ۳ بصورت 0000000000000011 است که بعد از شیفت، به 0000000000000001 تبدیل می گردد .

حال اگر عدد۳- را بخواهیم یک شیفت به راست دهیم داریم :

-3=1111111111111101 >>1 1111111111111110=-2

در این حالت اندازه مقادیربا هم برابر نیستند و در پیاده سازی باید از این نکته آگاه بود .

همچنین ممکن ا ست که در یک الگوریتم تعداد شیفت به را ست از اندازه کلمه بی شتر گردد . مثلا یک کلمه ١٦ بیتی باندازه ١٨ بیت شیفت داده شود که با انجام ١٨ بار شیفت یک بیتی حاصل ٠ یا ١- می شود البته ب سته به علامت اولیه آن عدد. اما در بع ضی از کمپایلر ها ممکن ا ست که این کار خطا گرفته شود و این موضوع را باید در نظر داشت .

رند کردن فرایند تبدیل از دقت مضاعف به دقت معمولی در اکومولاتور است. که قبل از ذخیره به صورت ١٦ بیتی در حافظه ،صورت می گیرد. معمولا با ارزشترین بیت (MSB) در کلمه پایین اکومولاتور چک می شود، اگر این بیت ١ با شد آنگاه یکی به مقدار کلمه بالایی ا ضافه می شود و این کلمه بعنوان حاصل رند کردن در نظر گرفته می شود . البته در این حالت باید از سرریز نشدن اکومولاتور مطمئن شد .

ب- تقسیم

تقسیم به اندازه جمع و ضرب زیاد استفاده نمی شود . تنها تقسیمی که بکار می رود بصورت ممیز شناور اسکالر می باشد .صورت و مخرج و خارج قسمت به فرمت نرمالیزه نمایش داده می شوند . NLS خارج قسمت با تفریق NLS مخرج از NLS صورت و جمع با ١٤ محاسبه می شود . چرا که اگر صورت کمی از مخرج بزرگتر باشد و هر دو دارای NLS=0 باشند خارج قسمت باید دارای NLS=14 باشد تا بصورت صحیح نرمالیزه باشد. اگر مانتیس صورت کوچکتر از مانتیس مخرج باشد آنگاه صورت یکی به چپ شیفت داده می شود و به NLS آن یکی اضافه می شود سپس NLS مخرج محاسبه می گردد و این تضمین می کند که مانتیس خارج قسمت بصورت نرمالیزه باشد.

٤-٣- ساده سازی محاسبات الگوریتم

در این قسمت لازم است که برخی از محاسبات الگوریتم را برای پیاده سازی ممیز ثابت مناسب ساخته و بار محاسباتی آنها را کاهش دهیم .

٤-٣-١- تطبیق دهنده بهره

در اینجا لازم است که ابتدا عملیات ممیز شناور تطبیق دهنده بهره را به اختصار شرح دهیم . آرایه حالت بهره GSTATE که با علامت δ نشان داده می شود حاوی ۱۰ بهره لگاریتمی بدون افست می با شد . علامت $\delta(n)$ بهره را برای بردار n نشان می دهد.خروجی پیشگویی بهره برای بردار n بصورت زیر است :

$$\hat{\delta}(n) = -\sum_{i=1}^{10} \alpha_i \delta(n-i) \qquad (۱)$$

مطابق با شکل (٤-۱) قبل از تبدیل $\hat{\delta}(n)$ به حالت خطی یک افست بهره 32db باید به آن ا ضافه شده و آزمایش شود که :

$$0 \le \hat{\delta}(n) + 32 \le 60 \qquad (۲)$$

تخمین بهره در حالت خطی بصورت زیر خواهد بود:

$$\sigma(n) = 10^{(\hat{\delta}(n)+32)/20} \qquad (۳)$$

مقدار σ (n) در ابتدا برای نرمالیزه کردن بردار هدف VQ تحریک بکار می رود . بعد از اینکه جستجوی کتاب کد کامل شد ، σ (n) برای مقیاس کردن بهترین بردار کد انتخابی بکار می رود . اگر فرض کنیم که اندیس بهره i و اندیس شکل j برای بردار n انتخاب شده باشد بردار تحریک e(n) بصورت زیر خواهد بود :

$$e(n) = \sigma(n) g_i y_j \qquad (۴)$$

که در آن yj ،j امین بردار شکل و gi ،i امین سطح بهره در کتاب کد بهره می باشند .آنگاه بردار تحریک e(n) برای محاسبه δ (n) بکار می رود. توان e(n) بصورت زیر است :

$$P[e(n)] = \frac{1}{5}\sum_{k=1}^{5} e_k^2(n) \qquad (۵)$$

قبل از تبدیل P[e(n)] به مقیاس لگاریتمی P[e(n)] را به مقادیر بزرگتر یا مساوی ۱ برش می زنیم یعنی :

$$P[e(n)] \geq 1 \qquad (۶)$$

این از سرریز شدن در تبدیل لگاریتمی جلوگیری می کند و در نهایت $\delta(n)$ بصورت زیر برای پیشگویی بهره تحریک بعدی حاصل می گردد:

$$\delta(n) = 10\log_{10} P[e(n)] - 32 \qquad (۷)$$

حالا روشی که از نظر ریاضی معادل روش فوق و مناسب برای پیاده سازی ممیز ثابت می باشد را شرح می دهیم . فرض کنید که, k y_{jk} امین عنصر j امین بردار شکل باشد. از ترکیب معادلات (۴و۵) داریم :

$$\begin{aligned} P[e(n)] &= \frac{1}{5}\sum_{k=1}^{5} (\sigma(n) g_i y_{jk})^2 \\ &= \sigma^2(n) g_i^2 [\frac{1}{5}\sum_{k=1}^{5} y_{jk}^2] \\ &= \sigma^2(n) g_i^2 P[yi] \end{aligned} \qquad (۸)$$

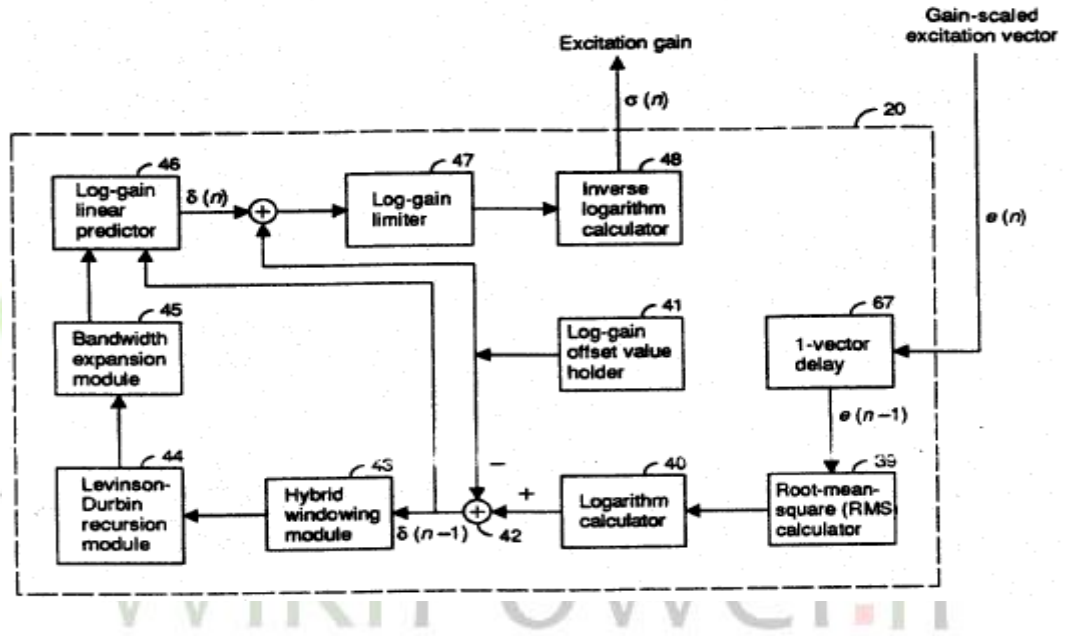
با جایگذاری معادله (۸) در معادله (۷) داریم :

$$\delta(n) = 20\log\sigma(n) - 32 + 20\log|g_i| + 10\log P[y_j]$$

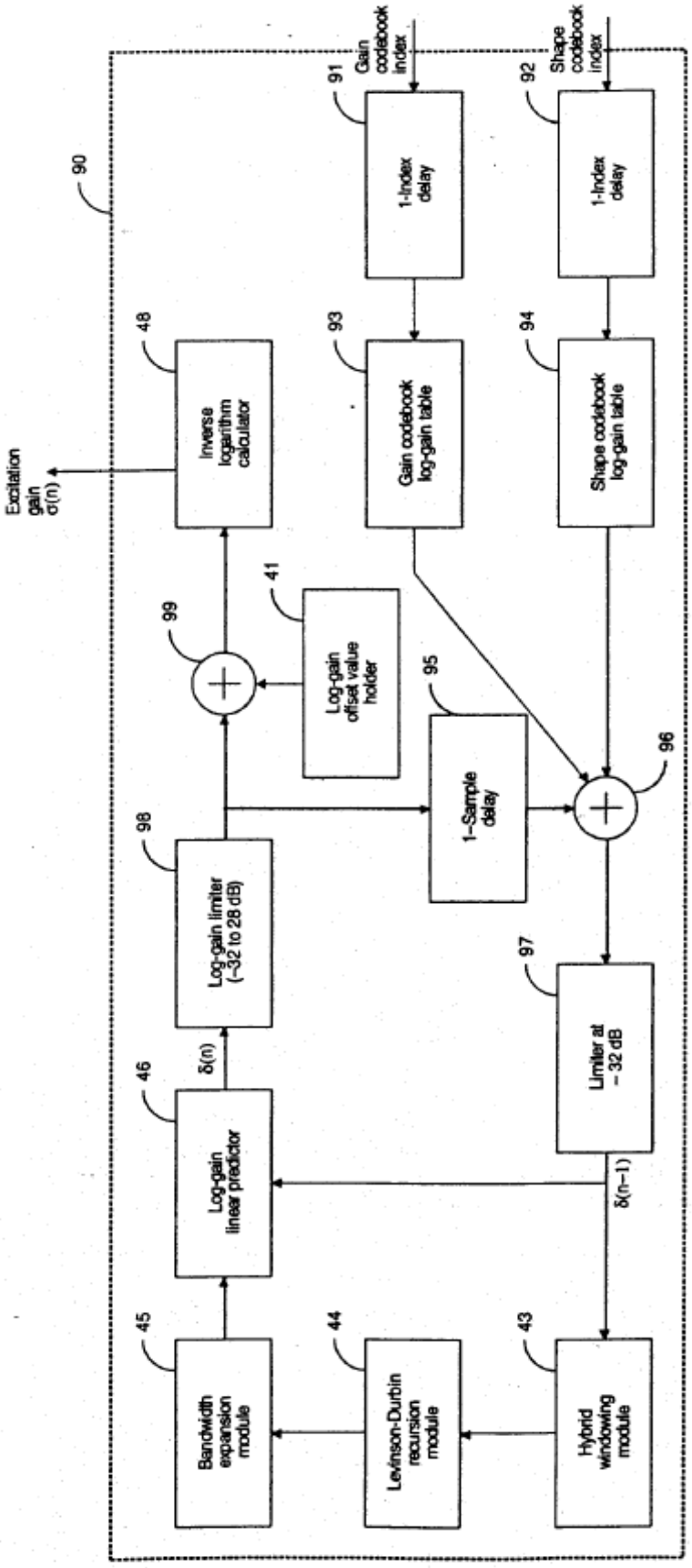
اکنون با استفاده از معادله (۳) داریم:

$$\delta(n) = \hat{\delta}(n) + 20\log|g_i| + \log P[y_j]$$

که درآن 20 log |gi| مقدار بهترین بهره انتخابی و 10 log P[yj] مقدار توان بهترین بردار شکل انتخابی بر حسب dB هستند. شکل (۴-۲) طرح بلوکی این روش را نشان می دهد .از آنجا که فقط ۴ مقدار ممکن |gi| و ۱۲۸ مقدار ممکن P[yj] داریم ، می توان مقادیرآنهارابرحسب dB محاسبه و در جدول های بهره بلوک های ۹۳و۹۴ این شکل ذخیره ساخت.واحدهای تاخیر۹۱و۹۲ بهترین اندیس بهره و شکل حاصل ازجستجوی کتاب کدتحریک برای بردارقبلی راآماده میسازند.واحد یک نمونه تاخیر۹۵ پیشگویی قبلی $\sigma\hat{}(n)$ را نگهداری میکند.خروجی محدود کننده ۹۷ از نظر ریاضی معادل است با خروجی جمع کننده۴۲ در شکل(۴-۱) و بقیه بلوکها تقریبا مشابه یکدیگرهستند .



شکل(٤-١) : تطبیق دهنده بهره



شکل(٤-٢)

روش معادل نشان داده شده در شکل (٤-٢)، ٢ امتیاز مهم بر روش اولیه دارد :

١- نیاز به محاسبه تابع لگاریتم (بلوک ٤٠ شکل(٤-١)) که در DSP ها با محاسبه سری های توانی پیاده سازی می شود و سیکل دستور زیادی را برای محاسبه می گیرد، ندارد .

٢- به دلیل حلقه فیدبک طولانی در روش اول (بخاطر تطبیق معکوس) محاسبات بیشتری در آن انجام می شود و در نتیجه از دقت محاسبات کاسته می شود و بعلاوه در یک پردازنده ممیز ثابت محاسبه تابع لگاریتم همواره با دقت مناسب صورت نمی گیرد .

البته در روش جدید به خاطر محاسبه و ذخیره ٤ بردار بهره و ١٢٨ بردار شکل به 128+4=132 کلمه بیشتر از حافظه ROM احتیاج است که البته مقدار ناچیزی می باشد.

٤-٣-٢- محاسبه لگاریتم معکوس (بلوک ٤٨)

شبه کد ممیزشناور بلوک (٤٨) شکل (٤-١) بصورت زیر است :

$$GAIN = 10^{(Z/20)}, Z = LOGAIN + GOFF$$

که در آن GOFF ، افست بهره می باشد. با تبدیل به مبنای ٢ داریم :

$$10^{0.05Z} = 2^{0.05\log_2(10)Z} = 2^{0.1660964Z}$$

فرض کنید X=0.1660964Z باشد که در محدوده 0 تا 9.97 تغییر می کندو X=[X]+x که [X] جز صحیح X و x بخش کسری آن است .مقدار $2^{[X]}$ براحتی محاسبه می شود و آنچه که می ماند محاسبه بخش کسری X است.

در محاسبه فرض کنید که 0.1660964 در فرمت Q21 نمایش داده شود.این مطابق است با عددی که به صورت ١٠ در ١٦ بیت بالایی و 20649 در ١٥ بیت پایینی نمایش داده شود. ما هردو جزء Z را بطور

جداگانه ضـرب می کنیم تا به دقت خوبی برای X برسـیم و آنگاه [X] و x را جدا می کنیم .در محاسـبه نمای جز کســـری می دانیم که 0<x<1 و بنابراین 1<2 ^{x} 2> . در نتیجه می توان نمایش ثابت زیر را بکار برد :x بصورت Q15 و 2^{x} ،Q14 خواهد بود. برای محاسبه 2^{x} از بسط سری تیلور زیر استفاده می کنیم :

$$2^{x} = [((C4x + C3)x + C2)x + C1]x + C0$$
$$= C4x^{4} + C3x^{3} + C2x^{2} + C1x + C0$$

که در آن ضرایب C بصورت زیر هستند [10] :

C4=323 (Q15)

C3=1874 (Q15)

C2=7866 (Q15)

C1=22702 (Q15)

C0=16384 (Q14)

۴-۴- روند نمای برنامه

با توجه به بلوک دیاگرام طراحی شده برای کدکG.728 (شکل(۳-۱)) و همچنین ملاحظات و تغییرات لازم برای برنامه نویسی ممیز ثابت، برنامه اینکدر به زبان C استاندارد نوشته شده و همراه با برنامه دیکدر در شبیه سازی مورد استفاده قرار گرفته و آزمایش شده است.این برنامه ها در ضمیمه (الف) بطور کامل موجود می باشـــد.در این قســـمت ابتدا برخی از متغیرها و بلوک های مهم برنامه را در جدول های (۴-۱)و(۴-۲) معرفی نموده و سپس روندنمای اینکدر و دیکدر را شرح می دهیم .

۱-۲-۱-۱- نام متغیر	۱-۲-۱-۲- ابعاد (آرایه)	
A	۱ تا ۵۱	ضرایب فیلتر سنتز
ATMP	۱ تا ۵۱	آرایه موقت ضرایب فیلتر سنتز
AWP	۱ تا ۱۱	ضرائب مخرج فیلتر وزنی
AWZ	۱ تا ۱۱	ضرائب صورت فیلتر وزنی
AWZTMP	۱ تا ۱۱	آرایه موقت فیلتر وزنی
ET	۱ تا ۵	بردار تحریک
G2	۱ تا ۸	۲ برابر سطوح بهره در کتاب کد
GAIN	۱	بهره تحریک خطی
GB	۱ تا ۷	نقطه وسط سطوح بهره
GSQ	۱ تا ۸	مربع سطوح بهره
GP	۱ تا ۱۱	
GPTMP	۱ تا ۱۱	آرایه موقت ضرایب پیشگویی بهره
GTMP	۱ تا ٤	آرایه موقت بهره
H	۱ تا ۵	بردار پاسخ ضربه F(z)W(z)
ICHAN	۱	بهترین اندیس کتاب کد برای ارسال
IG	۱	بهترین اندیس بهره کتاب کد
IS	۱	بهترین اندیس شکل کتاب کد
PN	۱ تا ۵	بردار کورلیشن جستجوی کتاب کد
R	۱ تا ۱۱	ضرائب اتوکورلیشن

RTMP	۱ تا ۱۱	آرایه موقت ضرائب اتوکورلیشن
S	۱ تا ۵	بردار صحبت ورودی
ST	۱ تا ۵	بردار صحبت کوانتیزه
STATELPC	۱ تا ۵۰	حافظه فیلتر سنتز
STMP	۱ تا ۲۰	بافر پنجره هیبرید فیلتر وزنی
STTMP	۱ تا ۲۰	بافر پنجره هیبرید فیلتر سنتز
SW	۱ تا ۵	بردار صحبت وزن داده شده شنیداری
TARGET	۱ تا ۵	بردار هدف VQ
Y	۱ تا ٦٤٠	آرایه کتاب کد شکل
Y2	۱ تا ۱۲۸	انرژی بردارهای کانولوشن شده شکل
ZIR	۱ تا ۵	پاسخ ورودی صفر

جدول (۴-۱) : متغیرهای برنامه

بلوک	ورودی(ها)	خروجی(ها)
٤- فیلتر وزنی شنیداری	S,AWZ,AWP	SW
۹و۱۰-BLOCKZIR،پاسخ ورودی صفر فیلتر سنتز وفیلتر وزنی شنیداری	A,STATELPC,AWZ,AWP	ZIR
۱۱-محاسبه بردار هدف	SW,ZIR	TARGET
۱۲-محاسبه پاسخ ضربه	A,AWZ,AWP	H
١٤و١٥-کانولوشن بردار شکل ومحاسبه انرژی	H,Y	Y2
١٦- نرمالیزه کننده بردار هدف	TARGET,GAIN	TARGET
١٣- کانولوشن معکوس زمانی	H,TARGET	PN
١٧و١٨- محاسبه خطاوانتخاب بهترین اندیس کتاب کد	PN,Y,Y2,GB,G2,GSQ	IG,IS,ICHAN
١٩و٢١- کتاب کد تحریک و مقیاس بهره	IG,IS,GAIN	ET
۹و۱۰- تجدید حافظه فیلتر	ET,A,AWZ,AWP,STATELPC	ST

جدول (۴-۲): بلوک های برنامه

بلوک	ورودی(ها)	خروجی(ها)
۳۶- پنجره هیبرید فیلتر وزنی شنیداری	STMP	R
۳۷- تکرار Durbin فیلتر وزنی شنیداری	R	AWZTMP
۳۸- محاسبه ضرایب وزن داده فیلتر	AWZTMP	AWZ,AWP
۴۹- پنجره هیبرید فیلتر سنتز	STTMP	RTMP
۵۰- تکرار Durbin فیلتر سنتز	RTMP	ATMP
۵۱- گسترش دهنده پهنای باند	ATMP	A
۴۳- پنجره هیبرید بهره	GTMP	R
۴۴- تکرار Durbin بهره	R	GPTMP
۴۵- گسترش پهنای باند بهره	GPTMP	GP

جدول (۴-۲): بلوک های برنامه (ادامه)

۴-۴-۱- اینکدر

در شکل (٤-۳) روند نمای کلی برنامه اینکدر ترسیم شده است، در اینجا مراحل اجرای این روند نما و توصیف کلی بلوکهای استفاده شده در آنرا بیان می کنیم.

در ابتدای برنامه و قبل از وارد شدن به حلقه اصلی برنامه (main-loop)، باید متغیرهای سراسری برنامه مقدار دهی اولیه شوند ، برخی از این متغیر ها مانند بردار پاسخ ضربه H باید در ابتدا دارای مقدار معینی بوده و برخی دیگر مانند ضرایب A , LPC به صفر مقدار دهی اولیه می شود.

همچنین در ابتدای برنامه فلگ های وضعیت خرابی ILLCOND برای فیلتر سنتز ،ILLCONDG برای پیشگویی بهره ، ILLCONDP برای پست فیلتر و ILLCONDW برای فیلتر وزنی به FALSE مقدار دهی اولیه می شوند.این فلگ ها در صورتیکه در محا سبه ضرایب هر یک از فیلتر های مربوطه ا شکال پدید آید، دارای مقدار TRUE می شوند و در نتیجه ضرایب این فیلتر ها در سیکل معین شده، تجدید نمی شوند.و تا سیکل بعدی از همان ضرایب قبلی استفاده می شود.

سپس متغیر شمارنده برنامه یعنی ICOUNT به صفر مقدار دهی می شود.این شمارنده دارای مقادیر ۱، ۲، ۳، ٤ می باشد و مشخص می کند که در کدام بردار فریم کنونی قرار داریم(هر فریم شامل ٤ بردار است).

پس از انجام مقدار دهی های اولیه به حلقه اصلی برنامه اینکدر یعنی main-loop می رسیم.

در ابتدا شمارنده ICOUNT چک می شود که اگر در دور قبل دارای مقدار ٤ بوده است به مقدار ۱ برگردد. سپس یک بردار ۵ نمونه ای از صحبت ورودی از بافر ورودی اینکدر خوانده می شود و

به بردار S[1:5] تخصـــیص داده می شـــود. در اینجا همه بردار های صـــحبت ورودی اعم از PCM خطی، u-LAW و یاA-LAW به محدوده [16383,+16384-] تبدیل می شوند. یعنی نمایش Q2 فرمت [- 4096,4095.75] .

در مرحله بعد اگر ICOUNT=3 باشدو ILLCOND=FALSE باشد بلوک گسترش دهنده پهنای باند ۵۱ اجرا می شود و الا اگر ILLCONDW=FALSE باشد بلوک محاسبه ضرایب فیلتر وزنی ۳۸ اجرا می گردد.سپس بلوک های محاسبه پاسخ ضربه ۱۲وانرژی کتاب کد۱٤و ۱۵ اجرا می شوند.

در مرحله بعدی اگر ICOUNT=2 و همچنین فلگ ILLCONDG=FALSE با شد بلوک گ سترش پهنای باند ٤۵ اجرا می شود.در ادامه به شروع پردازش های مربوط به هر بردار می ر سیم. ابتدا بلوکهای پیشگویی بهره ٤٦ ، محدود کننده بهره ۹۸- ۹۹ و محاســبه لگاریتم معکوس ٤۸ بطور همزمان و در یک بلوک اجرا می شود .سپس blockzir یعنی بلوکهای فیلتر سنتز ۹و فیلتروزنی شنیداری ۱۰ برای پاسخ ورودی صفر، اجرا می گردد.

در ادامه به ترتیب بلوکهای فیلتر وزنی شنیداری ٤ و محاسبه بردار هدف ۱۱ و نرمالیزه کننده بردار هدف ۱٦ و کانولوشن معکوس زمانی ۱۳ و جستجوی کتاب کد نرمالیزه ۱۷و ۱۸ اجرا می گردند.در این مرحله خروجی اینکدر یعنی اندیس کانال ICHAN بد ست می آید و در بافر خروجی اینکدر قرار می گیرد.

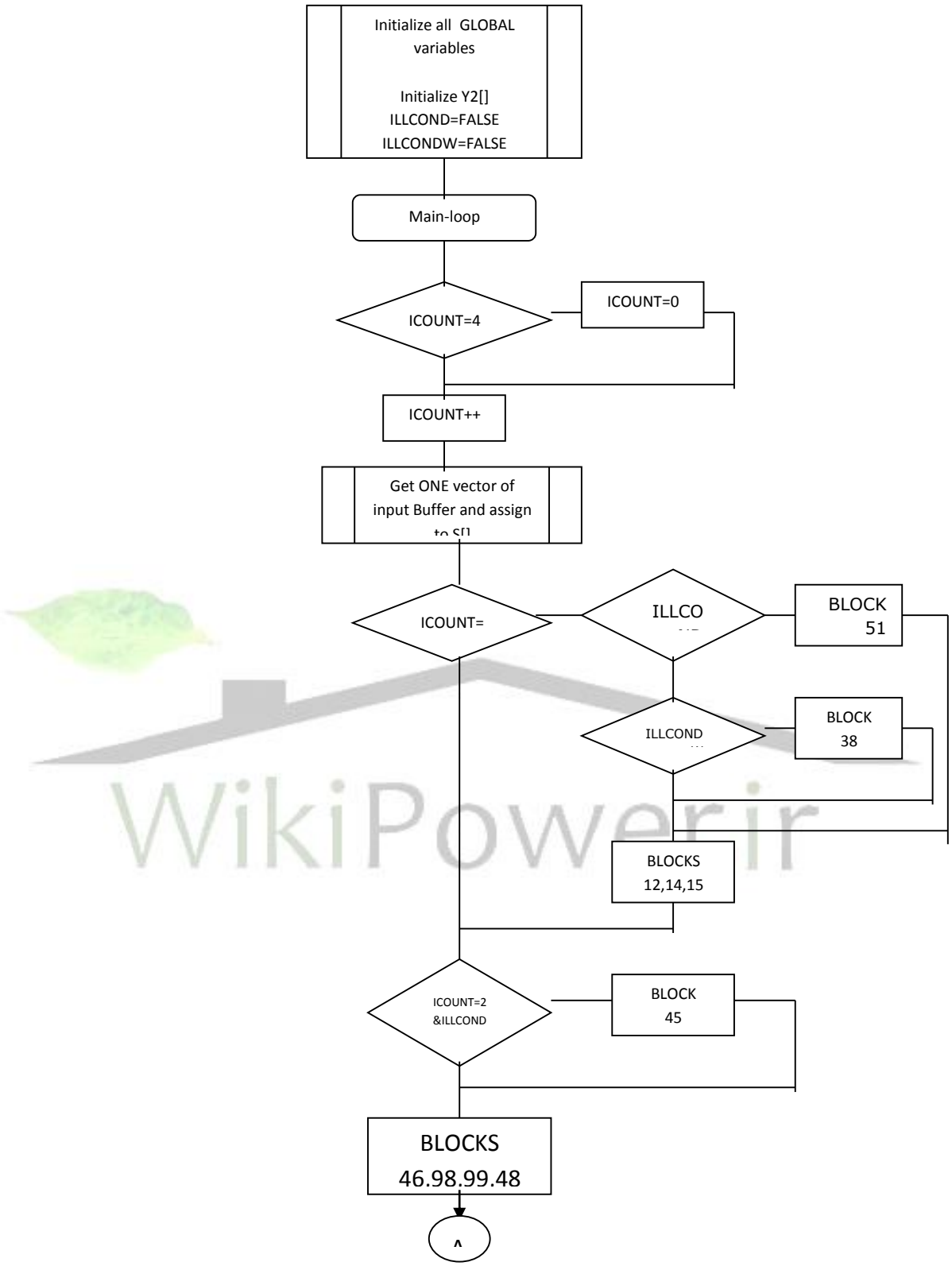
پس از تحویل اندیس کانال ۱۰ بیتی ICHAN به کانال مخابراتی نوبت به تطبیق معکوس اینکدر می رسد. در ابتدا بلوکهای مقیاس بردار تحریک ۱۹و ۲۱ و سپس تجدید حافظه فیلتر های سنتز و شنیداری ۹و ۱۰ اجرا می شود.سپس بلوکهای ۹۳،۹٤،۹٦و ۹۷ برای تجدید بهره اجرا می گردند.

در ادامه برنامه، بردار صـــحبت کوانتیزه []ST در بافر []STTMP ذخیره می گردد.STTMP برای محاســـبه ضرایب اتوکورلیشن در فیلتر سنتز بکار می رود.سپس بردار سیگنال []S در بافر []STMP ذخیره می گردد تا در محا سبه ضرایب اتوکورلی شن بکار رود .در اینجا پردازش های موبوط به هر بردار پایان می یابد و نوبت به پردازش های مربوط به فریم می رسد.

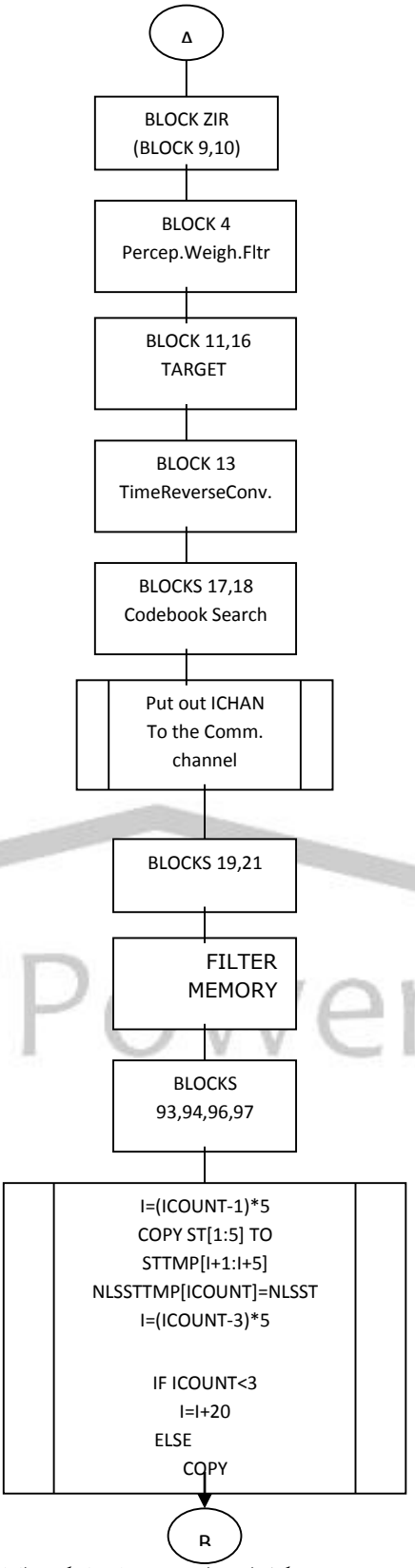
ابتدا اگر ICOUNT=4 باشد بلوک پنجره هیبرید ٤٩ و تکرار Durbin ٥٠ اجرا می شود.

اگر ICOUNT=2 باشد، بلوک پنجره هیبرید ٣٦ و تکرار Durbin ٣٧ اجرا می گردد.

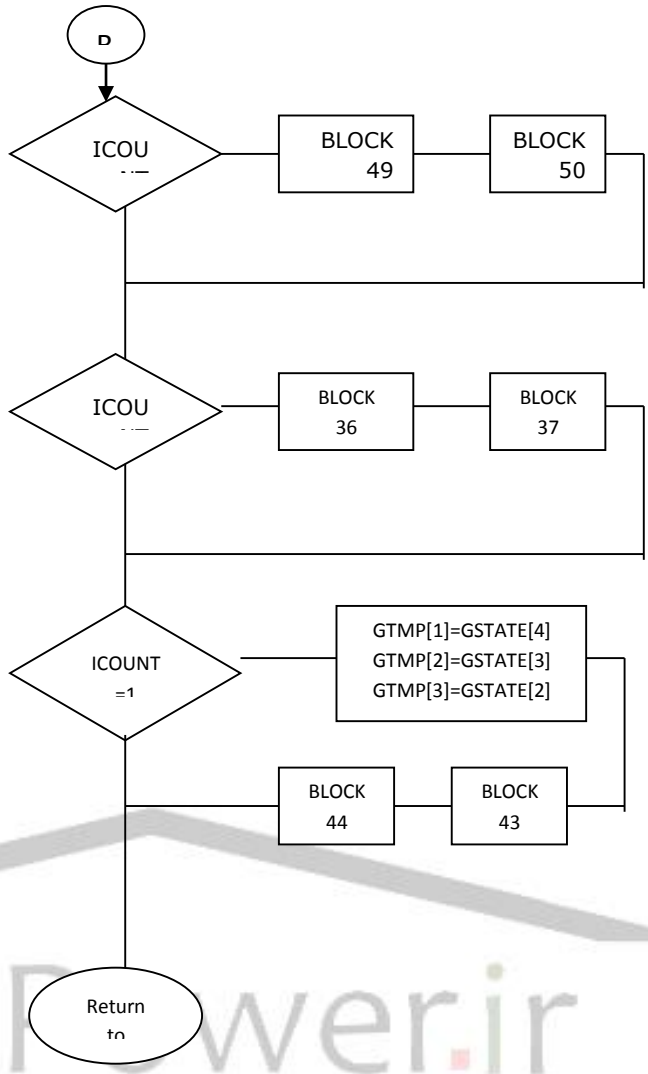
ســـرانجام اگر ICOUNT=1 باشـــد ابتدا آرایه موقت پیشـــگویی بهره ، GTMP با حافظه پیشـــگویی بهره GSTATE ، تجدید شـــده و ســـپس بلوکهای پنجره هیبرید ٤٣ و تکرار Durbin ٤٤ اجرا می شـــوند.در اینجا پردازش های مربوط به هر فریم هم پایان می یابدو حلقه اصـــلی برنامه اینکدر کامل می گرددو برنامه به نقطه شروع این حلقه باز می گردد.



شکل(٤-٣): روندنمای اینکدر



شکل(٤-٣): روندنمای اینکدر(ادامه)



شکل(٤-٣): روندنمای اینکدر(ادامه)

۴-۴-۲- دیکدر

شکل (٤-٤) روندنمای کلی حاکم بر برنامه دیکدر را نشان می دهد.در این قسمت مراحل اجرای این روندنما را بیان می کنیم.

در ابتدای برنامه، متغیر های سراسری مقدار دهی اولیه می شوند.همچنین فلگ های وضعیت خرابی فیلتر سنتز ILLCOND ، بهره ILLCONDG و پست فیلتر ILLCONDP به FALSE مقدار دهی اولیه می گردند.و سر انجام شمارشگر برنامه ICOUNT به صفر مقدار دهی می شود.

پس از مقدار دهی اولیه برنامه وارد حلقه اصلی دیکدر (main-loop) می شویم، ابتدا شمارشگر برنامه که نشان می دهد هم اکنون در کدام بردار از فریم کنونی هستیم چک می شود که اگر در دور قبل دارای مقدار ٤ بوده به مقدار ۱ برگردد. سپس اندیس کانال ICHAN از بافر ورودی دیکدر خوانده می شود.آنگاه اندیس شکل IS و اندیس بهره IG از ICHAN بدست می آید. حال نوبت به تجدید ضرایب فیلتر ها می رسد. اگرICOUNT=3 باشد و همچنین فلگ خرابی ILLCOND برابر FALSE باشد بلوک گسترش پهنای باند ۵۱ اجرا می گردد.در مرحله بعد اگر ICOUNT=2 با شد و فلگ ILLCONDG هم برابر FALSE باشد ، بلوک گسترش باند ٤۵ اجرا می شود.

پس از آن پردازش های مربوط به یک بردار شروع می شود.ابتدا بلوک های ٤٦،۹۸،۹۹و ٤٨ مربوط به پیشگویی بهره انجام می شود . سپس بلوکهای ۱۹و ۲۱ مربوط به بردار تحریک انتخاب شده و بعد از آن بلوک فیلتر سنتز ۳۲ اجرا می شوند .

اگر ICOUNT=1 باشد ضرائب پست فیلتر زمان کوتاه در بلوک ۸۵ تجدید می شود. سپس فیلتر معکوس LPC مرتبه ۱۰ در بلوک ۸۱ اجرا می شود آنگاه اگر ICOUNT=3 باشد بلوک های استخراج پریودpitch ۸۲ ، محا سبه ضریب پریود pitch ۸۳ و تجدید ضرائب پست فیلتر زمان بلند ۸٤ اجرا می

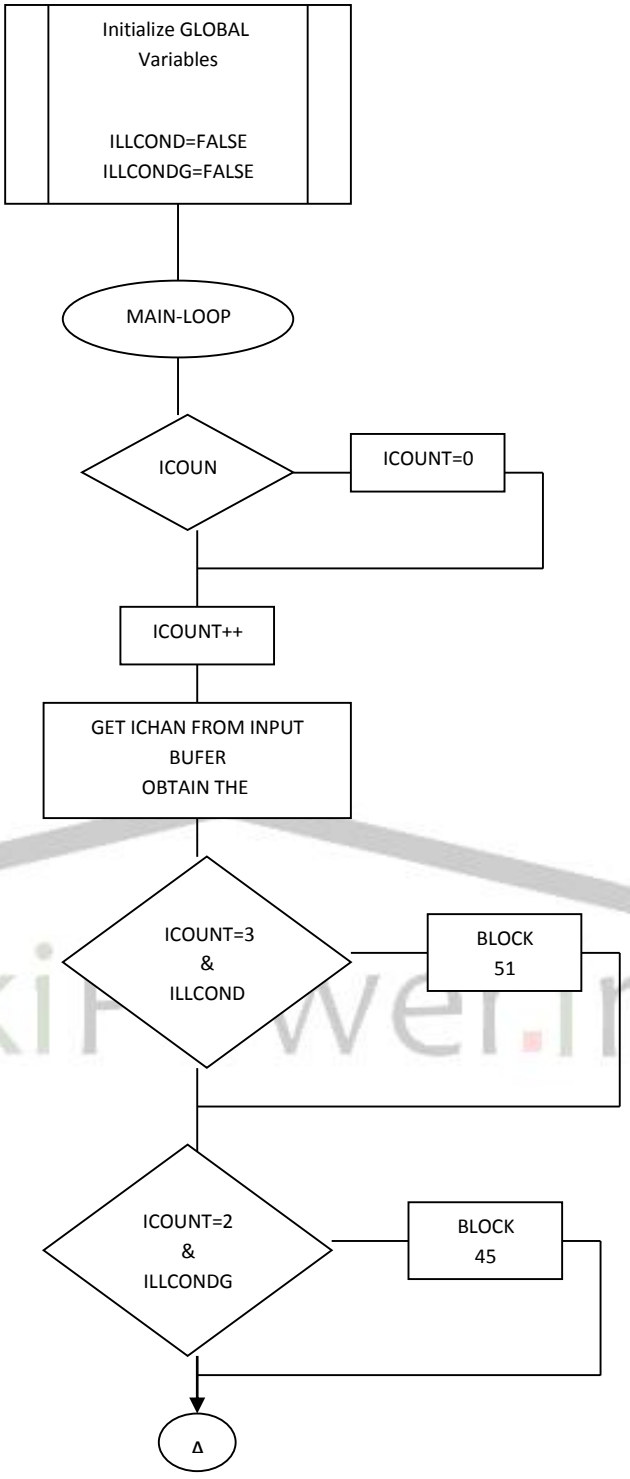
شوند . سپس بلوکهای پست فیلتر زمان بلند ۷۱ و پست فیلتر زمان کوتاه۷۲ و محاسبه مجموع قدر مطلق های ۷۳و۷٤ و همچنین بلوکهای محاسبه ضریب مقیاس ۷۵ و فیلتر پایین گذر ۷٦ و کنترل بهره خروجی پست فیلتر ۷۷ اجرا می گردند.

پس از آن پیشـــگویی بهره و حافظه آن در بلوک های ۹۳و۹٤و۹٦و۹۷ تجدید می شـــود . در مرحله بعد بافر صحبت کوانتیزه و سنتز شده ST یعنی STTMP تجدید می گردد . و در اینجا پردازش های مربوط به یک بردار پایان می یابد و پردازش های مربوط به هر فریم آغاز می شود.

ابتدا اگر ICOUNT=4 باشـــد بلوک پنجره هیبرید فیلتر ســـنتز ٤۹ و پس از آن بلوک تکرار Durbin ۵۰ از مرتبه ۱ تا مرتبه ۱۰ اجرا می شود . سپس ضرائب پیشگویی مرتبه ۱۰ در APF ذخیره می شـوند تا پســت فیلتر بعداً از آنها اسـتفاده نماید . پس از آن عملیات بلوک ۵۰ از مرتبه ۱۱ تا مرتبه ۵۰ ادامه می یابد .

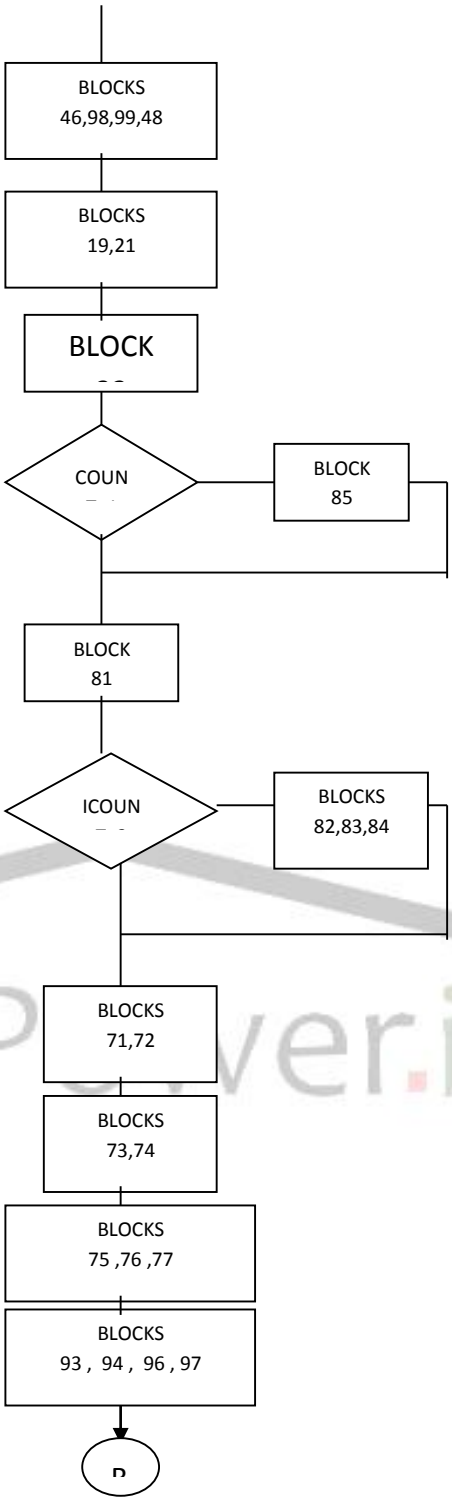
در پایان اگر ICOUNT=1 با شد بافر بهره GTMP بو سیله حافظه GSTATE در یک مرحله تجدید می شود و بلوک های پنجره هیبرید بهره ٤۳ و تکرار Durbin بهره ٤٤ اجرا می گردند .

در اینجا پردازش های مربوط به هر فریم در دیکدر پایان می یابدو برنامه به حلقه اصـــلی (main-loop) باز می گردد .

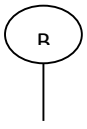


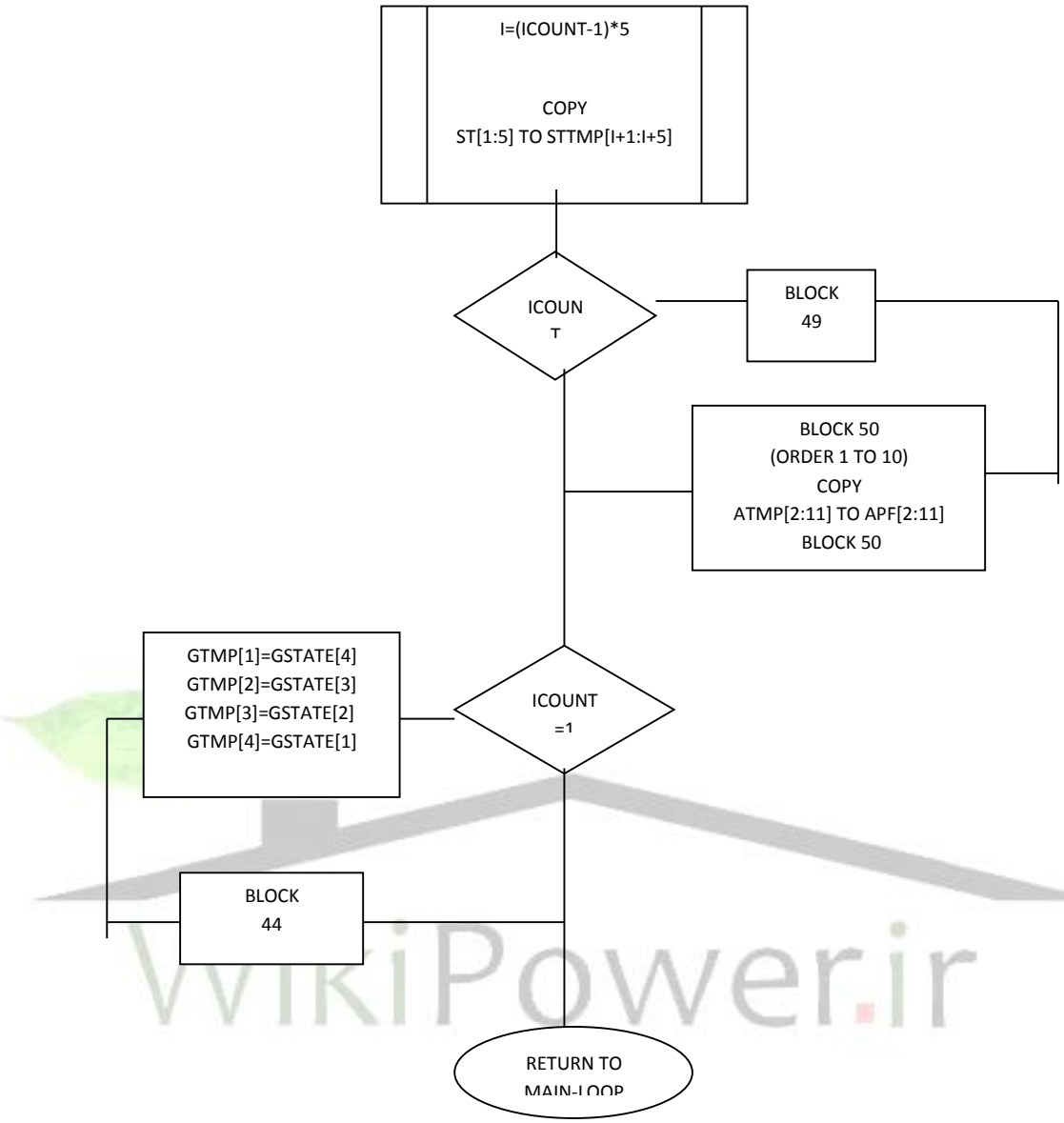
شکل(٤-٤) : روندنمای دیکدر

Δ



شکل(٤-٤): روندنمای دیکدر(ادامه)





شکل(٤-٤): روندنمای دیکدر(ادامه)

فصل ۵

۱-۲-۱-۳- پیاده سازی الگوریتم بر روی DSP

۵-۱- مقدمه

پس از شبیه سازی الگوریتم کدک G.728 به زبان C ، نوبت به انتخاب DSP و پیاده سازی برنامه برروی آن می رسد . در این فصل ابتدا مروری بر نحوه پیاده سازی بلادرنگ داریم . سپس به معرفی DSP های سری TMS می پردازیم. در قسمت بعدی نحوه برنامه نویسی TMS و تبدیل کد برنامه از زبان C به زبان اسمبلی TMS و بهینه سازی آن را به منظور پیاده سازی بی درنگ شرح می دهیم .

۵-۲- مروری بر پیاده سازی بلادرنگ

یک پروسه بلادرنگ ، فرآیندی است که باید در زمان مشخص انجام پذیرد . این حد زمانی ممکن است از مرتبه 200us تا حدود20 ms برسد . در کدینگ صحبت با فرض نرخ نمونه برداری 8 khz، برای کدری که بصورت نمونه به نمونه کارمی کند ، پردازش بلادرنگ باید در زمان 0.125 ms انجام شود . اما در کدرهای جدید مانند کدک CELP ، فرآیند کدینگ ، بصورت بلوکی و با طول بلوک اصلی

در حدود 20-30 ms (فریم اصلی) و طول بلوک فرعی 4-8 ms (فریم فرعی) صورت می گیرد که باعث افزایش محدودیت زمانی مجاز می شود .

به موازات پیشرفت در الگوریتم های کدینگ صحبت ، تکنولوژی پردازنده های DSP هم در چند سال اخیر پیشرفت زیادی داشته و با استفاده از DSP های قدرتمند ممیز ثابت[۱] و ممیز شناور[۲] ، امکان پیاده سازی بلادرنگ الگوریتم های صحبت بسیار پیچیده همچون CELP فراهم شده است DSP های ممیز ثابت ارزانتر هستند ولی از نظر برنامه نویسی مشکل تر می باشند . هرچند که از نظر قیمت ،استفاده از یک DSP ممیز ثابت بهتر به نظر می رسد اما طراحی یک نسخه ممیز ثابت از یک الگوریتم کدینگ صحبت که ممیز شناور است کاری دشوار و گاهی غیر ممکن می باشد زیرا باید اثرات مقیاس کردن ، نرمالیزه کردن و سرریز شدن را در نظر گرفت.

دو عامل مهم در انتخاب DSP جهت پیاده سازی الگوریتم کدینگ صحبت عبارتند از سیکل زمانی دستور العمل DSP (سرعت محاسبه)و مناسب بودن دستورالعملهای آن برای پردازش بلوک های اصلی آن الگوریتم . بعنوان مثال در الگوریتم کدینگ CELP بیشتر زمان پردازش DSP با دستوراتی به شکل MAC (ضرب و انباشت) گرفته می شود. برای چنین الگوریتمی DSP هایی که این کار را در یک سیکل دستوالعمل انجام دهند مناسب خواهند بود.

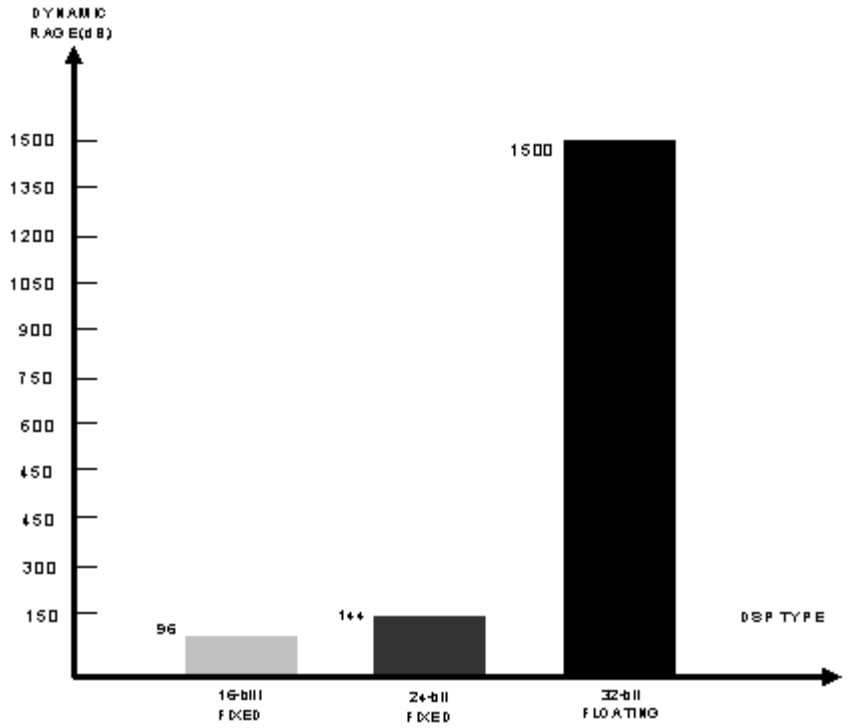
۵-۳-چیپ های DSP

[۱] Fixed Point
[۲] Floating Point

DSP های برنامه پذیر با توجه به دقت و عملیات حسابی به دو گروه ممیز ثابت و ممیزشناور دسته بندی می شوند . DSP های ممیز ثابت سریعتر و ارزانتر هستند ولی از نظر برنامه نویسی مشکل تربوده و دقت پایین تری را فراهم می سازند . امتیاز های پیاده سازی ممیز شناور بر پیاده سازی ممیز ثابت زیاد است . دقت در اعداد ممیز شناور بخاطر نرمالیزه شدن اتوماتیک مانتیس بو سیله پردازنده در طول برنامه ثابت می ماند . در حالیکه دقت داده های ممیز ثابت، بسته به اندازه ذخیره سازی آن داده پس از عملیات روی آن، تغییر می کند . بخاطر نرمالیزه شدن اتوماتیک مانتیس در DSP های ممیز شناور ، رند کردن و بریدن اعداد ، خطای کلی کمتری نسبت به ممیز ثابت بوجود می آورد . همچنین دقت ثابت بهمراه توانایی نمایش اعداد خیلی بزرگ یا خیلی کوچک ، باعث می شود مشکلاتی که در پیاده سازی فیلترها از نظر فاصله مکان صفرها و قطب ها وجود دارد ، برطرف گردد.

همانطور که در شکل(۵-۱) دیده می شود ، رنج دینامیکی وسیع در DSP های ممیز شناور اجازه می دهد که اعداد خیلی کوچک و خیلی بزرگ با دقت بالا نمایش داده شوند [5] . این بخصوص در محاسبات میانی FFT ها و فیلتر های بازگشتی مرتبه بالا لازم می شود . همچنین رنج دینامیکی وسیع موجب حذف نیاز مقیاس کردن میانی در DSP های ممیز ثابت ، جهت جلوگیری

از سرریز شدن داده، می گردد. از این رو استفاده از DSP های ممیزشناور ، باعث کاهش اندازه و پیچیدگی برنامه می شود بعلاوه اغلب الگوریتم های DSP ابتدا بر روی کامپیوتر های شخصی و با استفاده از امکانات ممیز شناور شبیه سازی می شوند و در صورت استفاده از DSP های ممیز شناور نیاز به شبیه سازی مجدد ممیز ثابت این الگوریتم ها بر طرف می گردد .



شکل (۵-۱) . رنج دینامیک در DSP های ممیز ثابت و ممیز شناور

۵-۳-۱- DSP های ممیزثابت

در DSP های ممیز ثابت باید توجه خاصی به سرریز شدن و مقیاس کردن داده ها نمود . از طریق چک کردن اعداد و مقیاس کردن مناسب آنها می توان به دقت یکنواختی در تمام پیاده سازی دست یافت ، کاهش دقت اغلب در هنگام ضرب دو عدد رخ می دهد چرا که تعداد بیت لازم برای نمایش حاصل ضرب برابر مجموع بیت های آن دو عدد می باشد . این بیت های اضافی باعث از دست رفتن اطلاعات می شود و بایستی حاصلضرب را با دقت مضاعف ذخیره نمود . در برخی از DSP ها سرریز شدن (overflow) بسادگی با قرار دادن نتیجه برابربا بزرگترین عدد مثبت یا منفی قابل نمایش در DSP ، کنترل می شود . ولی معمولا برای اینکه کنترل بهتری روی الگوریتم وکاهش خطای آن داشته باشیم

بایستی قبل از جمع کردن، حاصلضرب را به اندازه مناسب به سمت راست شیفت دهیم تا از سرریز شدن آکومولاتور جلوگیری شود .

۵-۳-۳- مروری بر DSP های خانواده TMS320

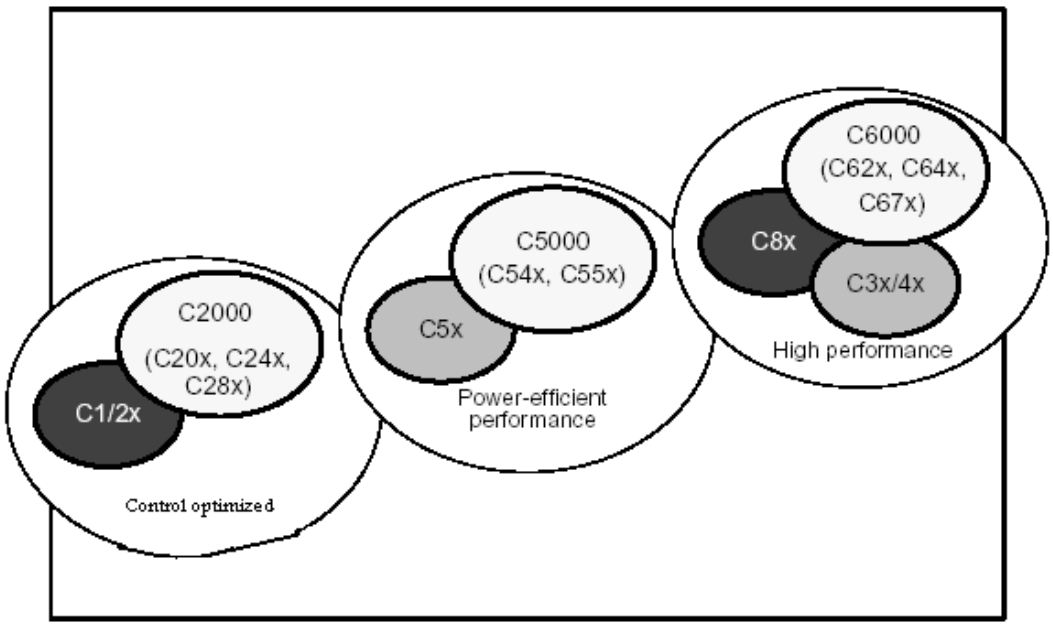
شرکت TI بعنوان پیشرو در بازار DSP ، اولین چیپ DSP را در سال۱۹۸۲ معرفی نمود. خانواده TMS320 در حال حاضر به سه گروه اصلی C2000,C5000,C6000 تقسیم بندی می شوند[16] .

گروه TMS320C6000 از نظر عملکرد و سرعت و راحتی استفاده برای برنامه نویسی سطح بالا بهینه شده اند.این گروه به نسل های ممیز ثابت C64X,C62X و ممیز شناورC67X تقسیم می شوند.

گروه TMS320C5000 از نظر عملکرد و مخصوصا توان مصرفی بهینه شده اند و برای کاربردهای موبایل و اینترنت که مشکل باتری دارند بسیار مناسب هستند.این گروه شامل دو نسل ممیز ثابت C55X,C54X می باشد.

گروه TMS320C 2000 برای کاربردهای کنترل دیجیتال در نظر گرفته شده و شامل دو نسل C28X,C24X می باشند.

در شکل زیر نمودار DSP های خانواده TMS320 نشان داده شده است.



شکل(۵-۲) : DSP های خانواده TMS320

۵-۳-۳-۱-معرفی سری TMS320C54X

DSP های سری C54X دارای سرعت و قابلیت عملیاتی بالایی هستند واز یک معماری باس پیشرفته ، CPU با سخت افزار مناسب کاربردهای خاص ، حافظه و ادوات جانبی on-chip و دستورالعمل های ویژه استفاده می کنند.

ویژگی های C54x :

- معماری پیشرفته چند باسه با یک باس برنامه ، ۳ باس داده و۴ باس آدرس
- واحد محاسبه و منطق (ALU) ۴۰ بیتی و دو آکومولاتور ۴۰ بیتی جداگانه
- ضرب کننده موازی برای عملیات ضرب/ انباشت (MAC) تک سیکلی بدون pipeline
- فضای حافظه آدرس پذیر ۱۶ بیتی(64 kword Program , 64 kword Data , 64 kword I/O)

در جدول (۵-۱) مشخصات حافظه DSP های C54x (به kword) نشان داده شده است[12].

Device	Program ROM	Program/Data ROM	DARAM†	SARAM‡
C541	20	8	5	0
C542	2	0	10	0
C543	2	0	10	0
C545	32	16	6	0
C546	32	16	6	0
C548	2	0	8	24
C549	16	16	8	24
C5402	4	4	16	0
C5410	16	0	8	56
C5420	0	0	32	168

† Dual-access RAM
‡ Single-access RAM

جدول (۵-۱)

کاربردهای زیررا می توان برای سری C54X در نظر گرفت:

- کدینک و دیکدینگ صحبت
- حذف اکو - حذف نویز
- مدولاسیون و دمدولاسیون
- فشره سازی تصویر و صدا
- رمز نگاری صحبت
- تشخیص صحبت و باز سازی صحبت

۵-۴- توسعه برنامه بلادرنگ

پیاده سازی یک نرم افزار بصورت بلادرنگ بر روی سخت افزار از اهمیت بالایی در سیستم های مخابراتی برخوردار است.در پیاده سازی بلا درنگ هزینه تجهیزات همانند پارامترهای کیفی سیستم اهمیت زیادی داشته و بایستی سعی شود که یک الگوریتم به روش بهینه پیاده سازی گردد.

در عمل پیاده سازی بلادرنگ یک الگوریتم بر روی چیپ DSP شامل مراحل زیر می شود : الف) بهبود تئوری الگوریتم. ب) آزمایش الگوریتم بوسیله شبیه سازی کامپیوتری با استفاده از یک زبان سطح بالا مانند C . ج) تبدیل کد سطح بالا به کد اسمبلی DSP مربوطه . د) آزمایش کد بلادرنگ با استفاده از ابزارهای توسعه گر موجود(مانند شبیه سازها) و ه) طراحی سخت افزار مناسب.

در قسمت (ج) برای تبدیل کد سطح بالا به کد DSP سه روش وجود دارد . روش اول، استفاده از کراس کمپایلر DSP است که زبان سطح بالای C را به اسمبلی DSP ترجمه می کند.روش دوم، برنامه نویسی دستی و مستقیم الگوریتم با استفاده از مجموعه دستورالعمل های DSP می باشد و روش سوم که عبارتست از استفاده از کراس کمپایلر و همچنین برنامه نویسی دستی در قسمتهایی که از نظر زمان اجرا محدودیت وجود دارد. از آنجا که برنامه نویسی دستی کاری مشکل و وقت گیر می باشد و مخصوصاً وقتی که اندازه برنامه بزرگ باشد آزمایش و اشکال زدایی آن وقت زیادی صرف می کند، معمولاً ازروش سوم در برنامه های بزرگ بیشتر استفاده می شود.

در قسمت (د) نتایج شبیه سازی کامپیوتری با خروجی های معادل DSP مقایسه می شوند . این کار معمولاً برای داده های آزمایش محدودی انجام می شود زیرا که پردازش مقدار زیادی داده در شبیه ساز DSP کار وقت گیر و دشواری می باشد.پس از مقایسه و درستی خروجی DSP و بررسی اجرای نرم افزار بصورت بلادرنگ آنگاه می توان از درستی پیاده سازی بلادرنگ الگوریتم اطمینان حاصل کرد.

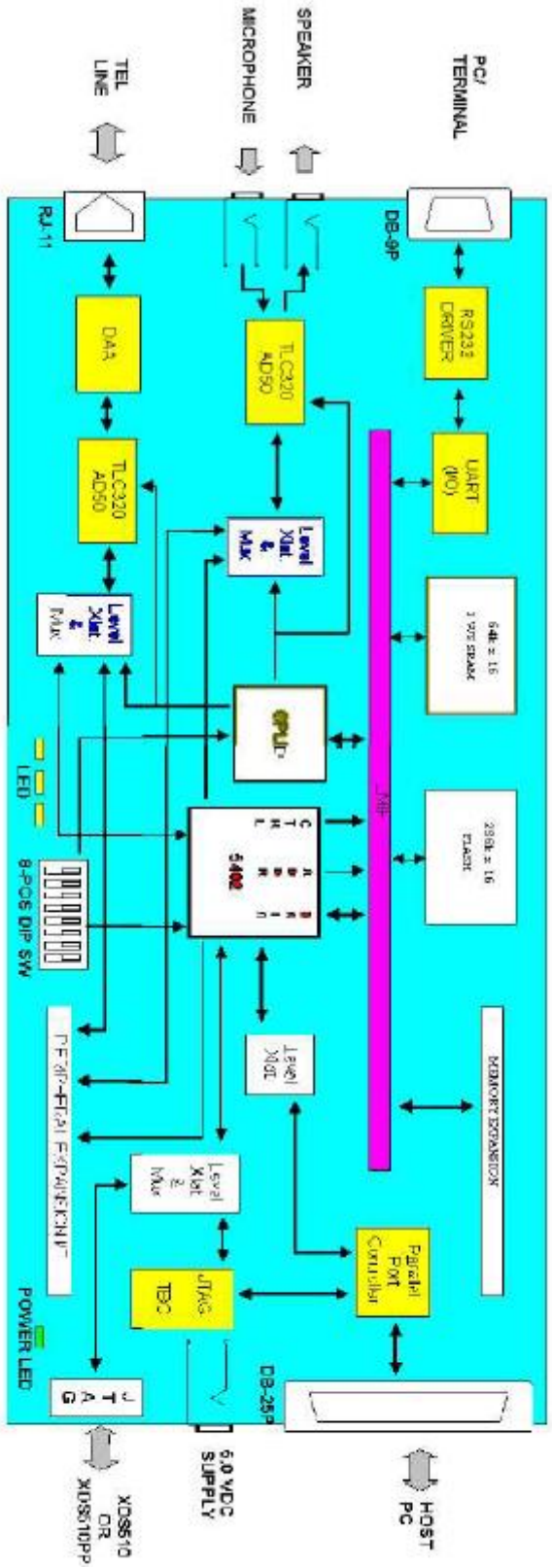
۵-۵- اجرای برنامه روی برد توسعه گر C5402 DSK

پس از آشـــنایی با پردازنده های TI ، حال نوبت به معرفی برد توســـعه گر C5402 DSK و برنامه نویســی آن می رســد . DSK امکان آزمایش و توســعه برنامه ها را بر روی پردازنده C5402 فراهم می سازد و می تواند مرجع خوبی برای طراحی سخت افزاری سیستم باشد[15] .

برد DSK شامل اجزای زیر است :

- 100 MHz VC5402 DSP
- 64 kword حافظه خارجی SRAM
- رابط صوتی میکروفن/ بلندگو
- رابط تلفن
- رابط داده ناهمزمان RS-232
- رابط JTAG برای امولیشن ورابط Host

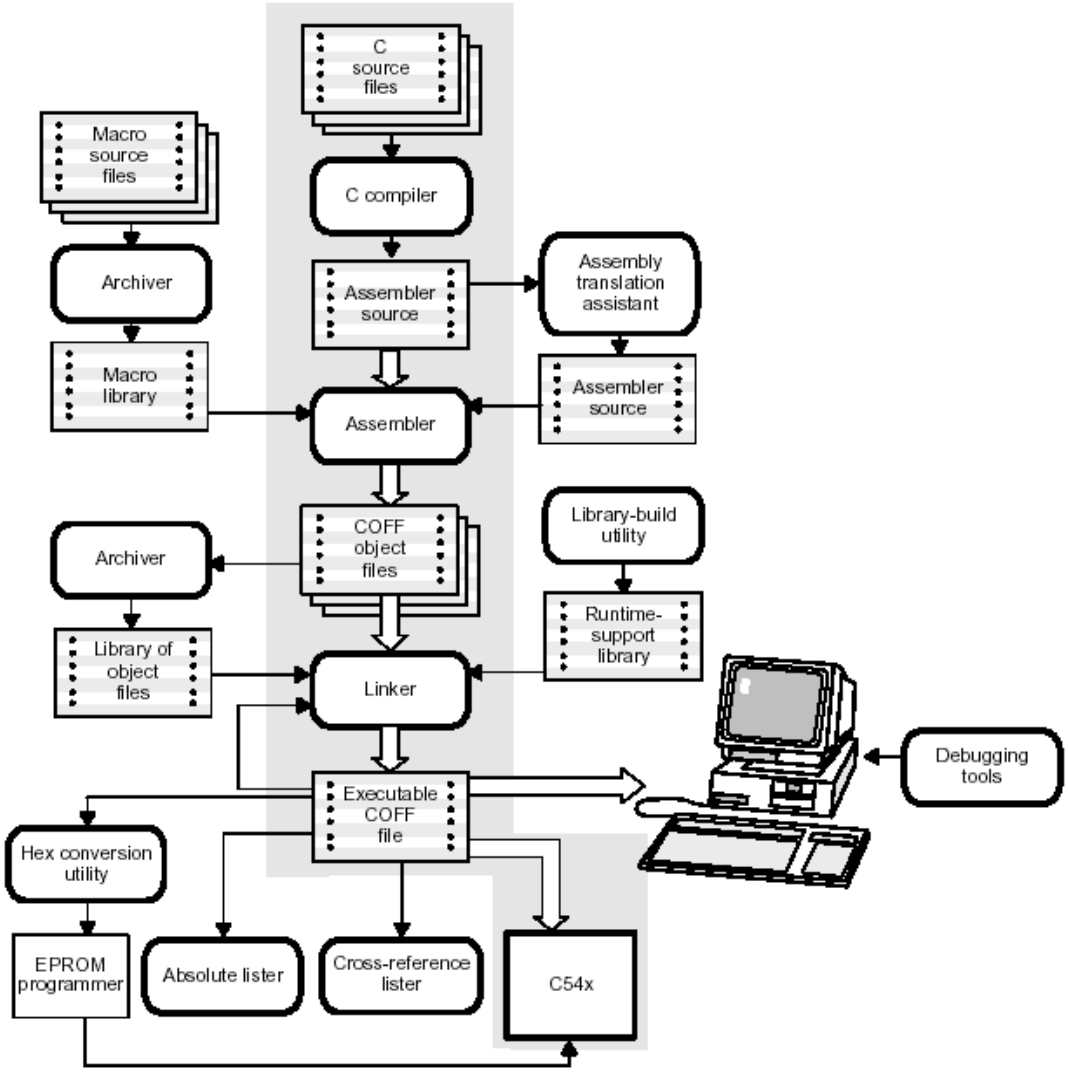
در شکل (۵-۳) بلوک دیاگرام و ارتباط های DSK نشان داده شده است .



شکل(۵-۳) : طرح بلوکی C5402 DSK

۵-۵-۱- بکارگیری ابزارهای توسعه نرم افزار

شکل (۵-۴) دیاگرام توسعه نرم افزاری C54x را نشان می دهد .بخش سایه زده مسیری را مشخص می کند که بیشتر استفاده می شود و بقیه بخش ها انتخابی هستند[13,14] .



شکل (۵-۴) : دیاگرام توسعه نرم افزاری TMS320C54x

در این قسمت برخی از ابزارهای نشان داده شده در شکل فوق را بررسی می کنیم :

- C/C++ Compiler ، کد C/C++ را به کد زبان اسمبلی ترجمه می کند .

- Assembler ، فایل های اسمبلی را به زبان ماشین بصورت یک فایل COFF Object تبدیل می نماید .
- Linker ، Object فایل های تولید شده تو سط ا سمبلر را ب صورت یک ماژول COFF Object قابل اجرا تبدیل می نماید .
- مبدل Hex ، DSP های C54x می توانند فایل COFF را بعنوان ورودی قبول کنند ولی اغلب برنامه ریز های EPROM قادر به این کارنیستند ، ازاین رو باید فایل COFF به یکی از فرمت های Tektronix, Motorola , Intel , TI-tagged تبدیل شـــود تا در برنامه ریز EPROM مربوطه ریخته شود .

۵-۵-۲- استفاده از نرم افزار CCS (Code Composer Studio)

نرم افزار CCS محصول شرکت TI ا ست و در قالب یک محیط ویژوال مانند Visual C++ ، امکان برنامه نویسی C ، ا سمبلی و تمامی ابزارهای تو سعه نرم افزاری شرح داده شده در ق سمت قبل را فراهم می آورد. این نرم افزار که بهمراه برد DSK می باشد ، از طریق پورت موازی کامپیوتر به برد DSK متصل شده و کار کنترل و مونیتورکردن آن را انجام می دهد .

همه رجیسترها و فضـای حافظه و پورت های برد DSK ، توسـط این نرم افزار کنترل و برنامه ریزی می شـوند وپس از تولید کد قابل اجرا بر روی TMS ، نرم افزار قادر اسـت پردازنده C5402 را برنامه ریزی

کند. همچنین امکانات اشکال زدایی مانند اجرای برنامه بصورت های مختلف و مشاهده متغیرها توسط CCS فراهم شده است .

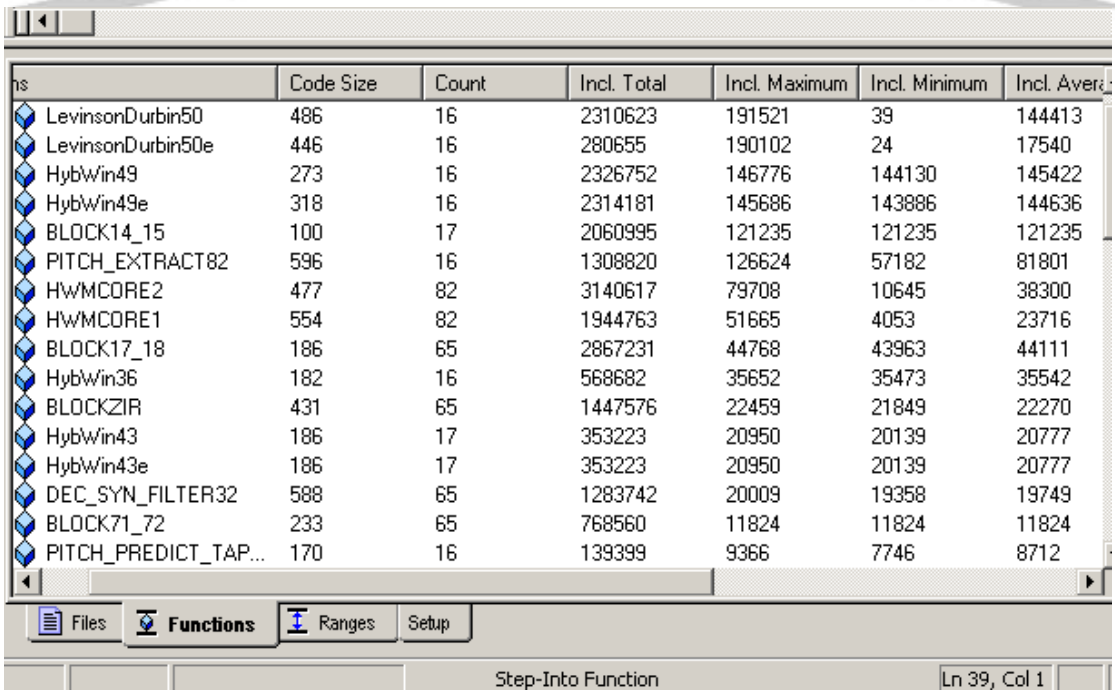
بمنظور استفاده از CCS و اجرای برنامه برروی DSK ابتدا برنامه شبیه سازی ممیز ثابت کدر G.728 را در قالب یک پروژه بنام CODEC به محیط CCS منتقل کردیم.آنگاه پروژه را بدون استفاده از optimizer ، کمپایل و لینک نموده وپس از رفع اشکالات برنامه و ساختن فایل codec.out ، آن را بر روی DSK اجرا کردیم . هدف ما در این مرحله اطمینان از صحت اجرای برنامه و اندازه گیری MIPS مورد نیاز آن بود . از اینرو بخشی از یک فایل صحبت PCM ۱۶ بیتی (در حدود ۲ ثانیه، 16 kword) را در بخشی از فضای حافظه بنام input_ ذخیره نموده و سپس اینکدر و دیکدر را در یک حلقه قرار دادیم . اینکدر نمونه های ورودی را می خواند و بصورت کد شده به دیکدر می داد و دیکدر هم آنها را باز سازی کرده و در بخشی از حافظه بنام output_ ذخیره می ساخت .

با استفاده از امکان Profiler در نرم افزار CCS می توان از عملکرد هر یک از توابع موجود در برنامه ، MIPS ومقدار حافظه مورد نیاز آنها آگاه شد . شکل (۵-۷) خروجی Profiler را برای ۶۵ دور اجرا (۳۲۵ نمونه) نشان می دهد.در ستون اول نام توابع برنامه را می توان دید که البته بدلیل محدودیت نمایش همه توابع دیده نمی شوند. ستون دوم اندازه کد هر تابع به word و ستون سوم تعداد اجرای آنرا تا کنون نشان می دهد. در ستون۴ کل سیکل دستورالعمل وستون۵ ماکزیمم سیکل دستورالعمل لازم برای اجرای هر تابع را نشان می دهد.در ستون ۶و۷ هم به ترتیب مینیمم و میانگین سیکل دستور هر تابع دیده می شود.

برای اجرای برنامه اینکدر و دیکدر بصورت دوطرفه کامل لازم است که عملیات پردازش هر دور برنامه در زمان ۵ نمونه یعنی 0.625 ms به پایان برسد و این در واقع برابر تاخیر الگوریتمی کدک کم تاخیر

G.728 می باشد. با توجه به قدرت پردازنده C5402 که برابر 100 MIPS است لازم است هر دور اجرای برنامه از ۶۲۵۰۰ سیکل دستورالعمل بیشتر نباشد. برای محاسبه تعداد سیکل دستورالعمل هر دور برنامه ، در نقطه ای که برنامه اینکدر و دیکدر به پایان می رسد یک Break Point قرار می دهیم و با استفاده از دستور RUN برنامه را تا این نقطه اجرا می کنیم. خروجی CLOCK پردازنده در این حالت تعداد سیکل دستورالعمل یک دور برنامه را نشان می دهد.

در این حالت چون از optimizer استفاده نشده ، مشاهده می شود که MIPS کلی برنامه چندین برابر قدرت C5402 (100 MIPS) است وحتی در حالت ماکزیمم به ۴۷۸۹۹۱ برای هر دور اجرا می رسد در صورتی که با توجه به قدرت C5402 ، نباید از ۶۲۵۰۰ در هر دور فراتر رود .

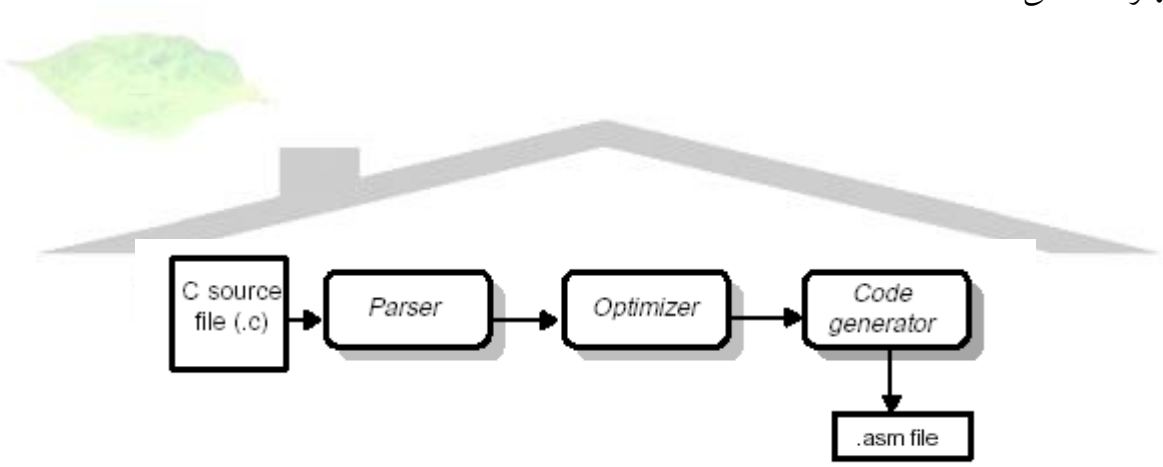


شکل (۵-۷): خروجی Profiler نرم افزار CCS

- استفاده از Optimizer

بهینه ساز باعث بهبود در سرعت اجرای برنامه وحجم کد تولید شده در کمپایلر می شود . این کار با ساده سازی حلقه ها ، مرتب کردن عبارت ها و قراردادن متغیرها در رجیسترها ممکن می شود.

کمپایلر C54x قادر به انجام بهینه سازی های مختلفی است و بهینه سازی سطح بالا در optimizer و بهینه سازی سطح پایین در code generator انجام می شود . شکل(۵-۸) جایگاه optimizer را بهتر نشان می دهد .



شکل(۵-۸): روند کمپایل یک فایل

ساده ترین راه استفاده از optimizer این است که در خط فرمان در دستورcl500 از انتخاب on- استفاده شود که n (0,1,2 & 3) ، نشان دهنده سطح بهینه سازی بصورت زیر است :

o0- :- ساده سازی روندنما

- قرار دادن متغیرها در رجیسترها

– حذف کدهای استفاده نشده

– ساده سازی جمله ها و عبارت ها

-o1 : همه بهینه سازی های o0– بعلاوه :

– حذف تخصیص های استفاده نشده

– حذف عبارت های مشترک محلی

-o2 : همه بهینه سازی های o1– بعلاوه :

– بهینه سازی حلقه ها

– حذف تخصیص های سراسری استفاده نشده

– حذف عبارت های فرعی مشترک سراسری

-o3 : همه بهینه سازی های o2– بعلاوه :

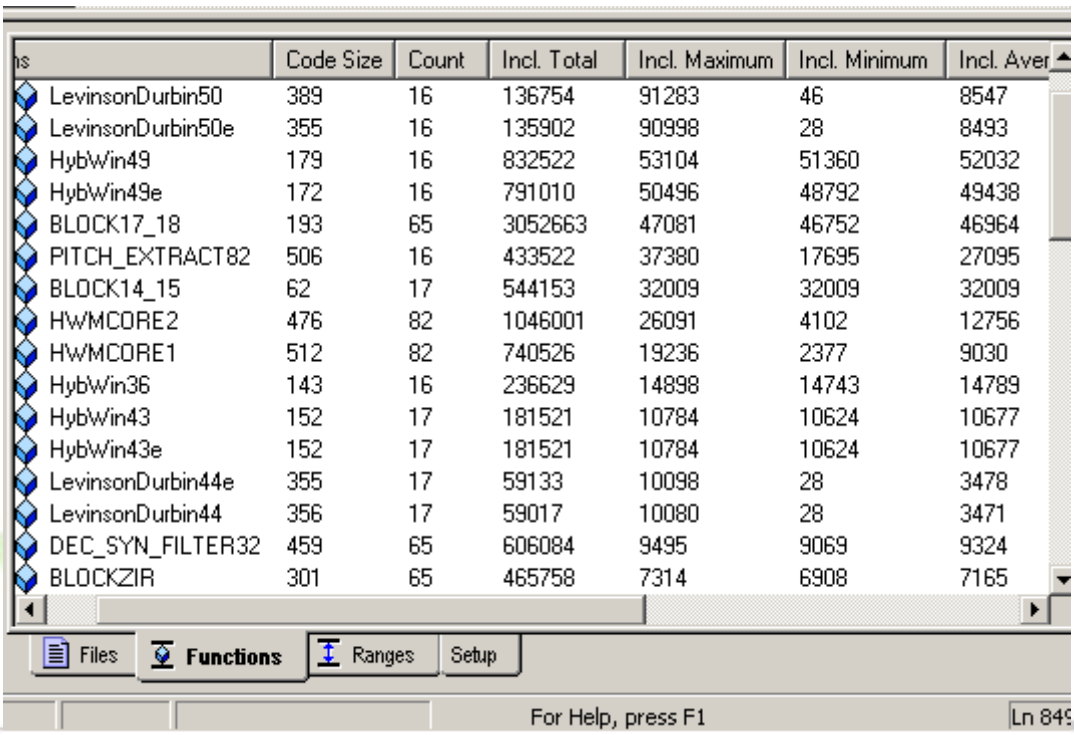
– حذف توابعی که صدا نشده

– ساده سازی توابع با مقدار برگشتی استفاده نشده

– inline کردن صدازدن توابع کوچک

– شناسایی مشخصات متغیر ها در سطح فایل

در این مرحله از پیاده سازی، پروژه CODEC را با ا ستفاده از optimizer و با حداکثر قدرت بهینه سازی o3– وبهینه سازی در سطح فایل اجرا کردیم . نتایج بد ست آمده در شکل (۵–۹) نشان می دهد که optimizer کاهش چشمگیری در MIPS و حجم کد برنامه بوجود آورده ولی هنوز هم MIPS برنامه بیش از دو برابر قدرت C5402 است و حتی در حالت ماکزیمم به ۲۳۳۸۹۲ در هر دور اجرا می رسد.



شکل (۵-۹):خروجی Profiler در حالت استفاده از Optimizer

- برنامه نویسی اسمبلی بصورت دستی

همانطور که در قسمت قبل دیدیم، کمپایلر C54x حتی با استفاده از optimizer هم نتوانست اجرای برنامه را به 100 MIPS برساند. این بدین دلیل است که در خانواده C54x، optimizer به حد کافی قوی نیست چراکه ما همین برنامه را با استفاده از optimizer پردازنده C55x کمپایل کردیم و به حدود 40 MIPS برای اجرای آن برروی C55x نیاز بود.

به هرحال باید اجرای این برنامه را به 100 MIPS برسانیم تا بتوان بصورت بلادرنگ آنرا پیاده سازی کرد. در این مرحله تنها راهی که باقی مانده اینست که بر روی توابع برنامه و MIIPS آنها و نتایج بد ست آمده از مرحله قبل بررسی کرده تا توابع و قسمت هایی که به نظر میرسد Optimizer نتوانسته خوب بهینه

سازد را بصورت دستی برنامه نویسی کنیم . واضح است که در این مرحله باید به زبان اسمبلی C54x تسلط کافی داشت تا بتوان کد اسمبلی تولید شده توسط کمپایلر C54x را بهینه کرد .

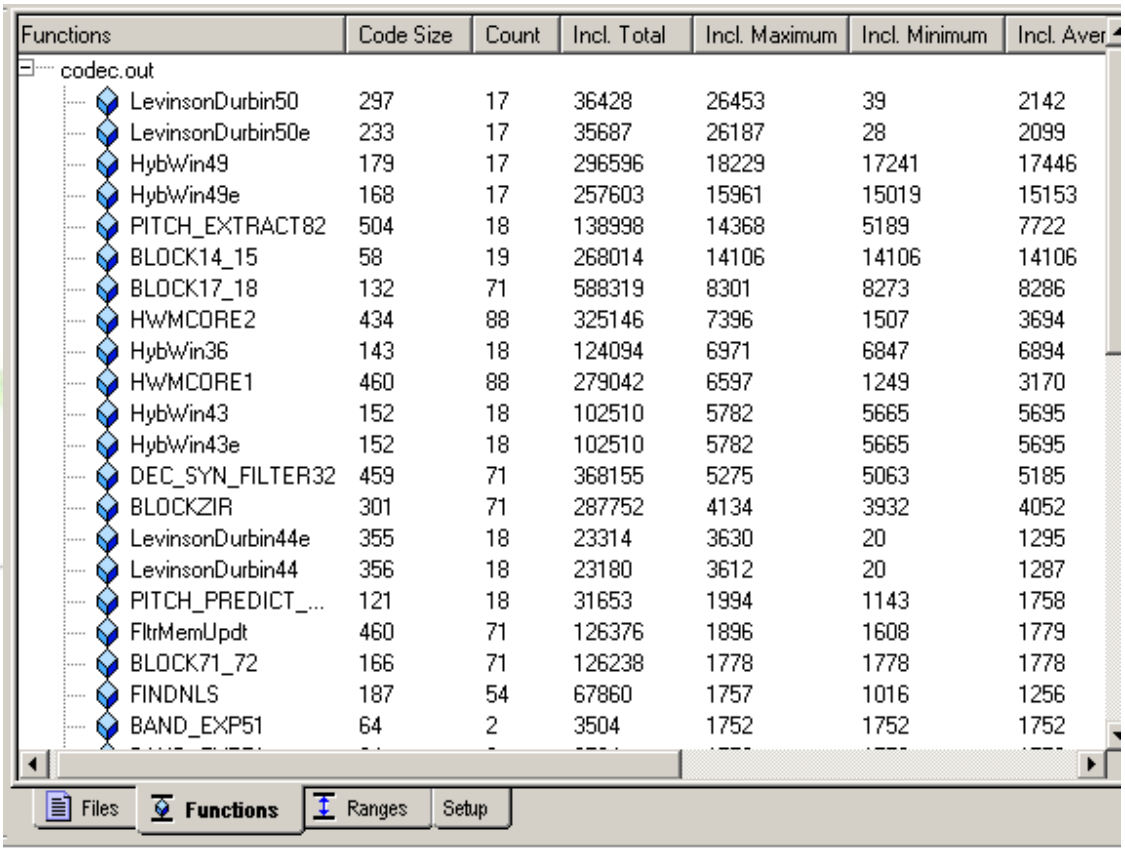
در ابتدا ملاحظه می شود که در محاسبات کورلیشن در بعضی از حلقه ها از دستور ضرب/انباشت MAC استفاده نشده و حلقه چند دستوری بوجود آمده است که می توان آنها را با استفاده از این دستور به حلقه تک سیکلی تبدیل کرد . همچنین در بعضی از حلقه های محاسبه انرژی نیز می توان از دستور تک سیکلی مجذور SQURA استفاده نمود.با انجام این اصلاحات MIPS برنامه کاهش یافت ولی هنوز فاصله زیادی با مقدار مورد نیاز ما دارد.

در این مرحله مهمترین توابع برنامه از نظر MIPS – همانطور که در شکل(۵-۹) دیده می شود –همچون LevinsonDurbin50()، HybWin49() ، Block17_18() و Block14_15() را بصورت دستی بازنویسی کردیم . در اینجا بعنوان نمونه به Block14_15() که کوچکتر است می پردازیم :

همانطور که در کد C این تابع درضمیمه (ب) دیده می شود ،تابع از ۳ حلقه تو در تو تشکیل شده است. حلقه بیرونی NCWD=128 بار ، حلقه میانی IDIM=5 بار و حلقه داخلی از ۱ تا ۵ بار اجرا می گردند. سیکل دستورالعمل این تابع بیش از ۳۲۰۰۰ است(شکل(۵-۹)). در ادامه کد اسمبلی تولید شده توسط کمپایلربا حد اکثر بهینه سازی و کد بازنویسی شده بصورت دستی در این ضمیمه آورده شده است . در بازنویسی این بلوک به این نکته توجه شده که دستورالعمل های حلقه داخلی به حد اقل برسد چرا که این حلقه در ضریب ۱۲۸×۵=۶۴۰ ضرب می شود .

نتایج این بهینه سازی و برنامه نویسی در شکل(۵-۱۰) به ترتیب صعودی بیشترین MIPS نشان داده شده است. با مقایسه نتایج این شکل و شکل (۵-۹) می توان گفت که MIPS بعضی از توابع همچون

LevinsonDurbin() به کمتراز یک سوم وبعضی دیگر همچون Block14_15() به کمتر از نصف رسیده است و نتیجه کلی اینکه هر دور اجرای برنامه که شامل اینکدر و دیکدر بصورت دوطرفه کامل (Full Duplex) می باشد ، کمتر از ۶۲۵۰۰ شده است و می توان آنرا بصورت بلادرنگ اجرا نمود .



شکل(۵-۱۰) : نتایج برنامه نویسی دستی

۵-۵-۳- نتایج پیاده سازی

در این قسمت به نتایج پیاده سازی بلادرنگ کدک G.728 می پردازیم . معمولا برای ارزیابی یک کدک صحبت از معیار های Subjective و Objective استفاده می شود. معروفترین معیار Subjective ، معیار

MOS[1] اســت که دارای ۵ ســطح از ۱ (بد)تا ۵ (عالی) می باشــد . این معیار به شــرایط آزمایش ، زبان وشــنوندگان وابســتگی دارد . معروفترین معیار Objective ، معیار SNR اســت که اعوجاج بین ورودی و خروجی سیستم را در نظر می گیرد و بصورت نسبت توان ورودی به توان خطای بین ورودی و خروجی تعریف می شود .

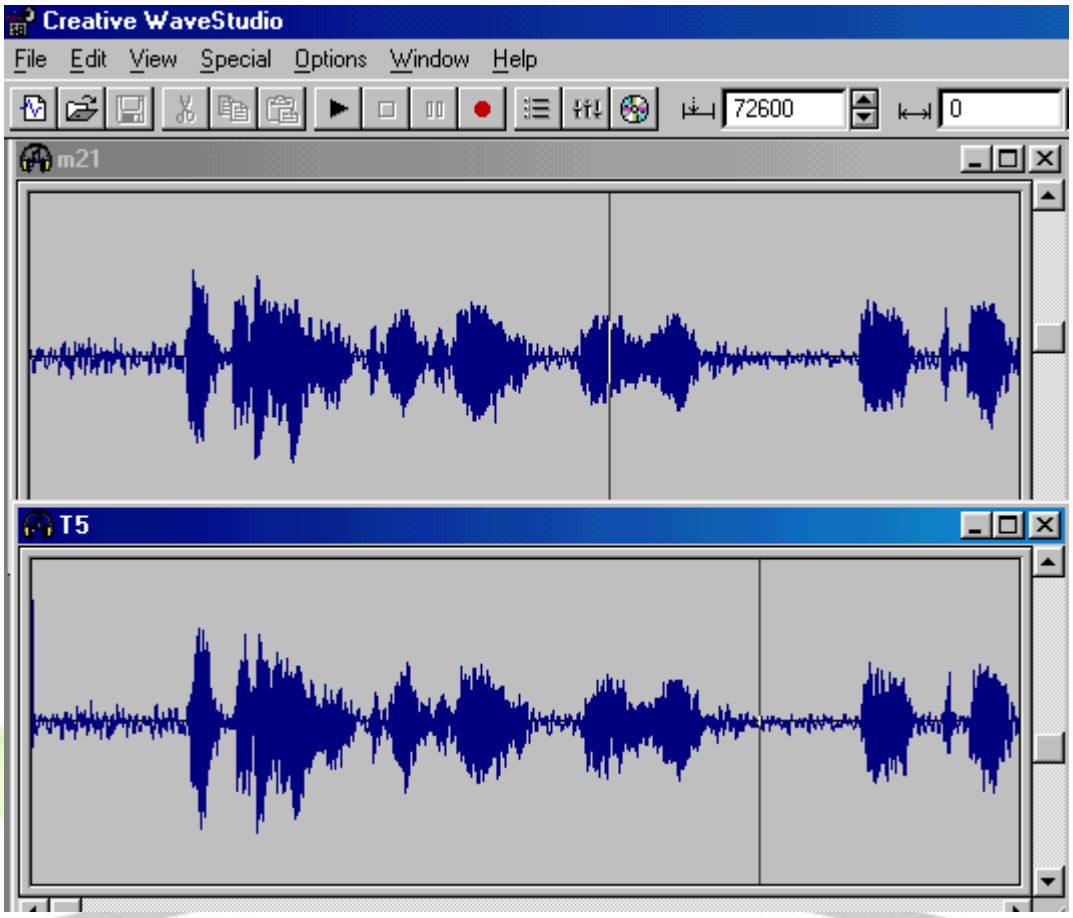
معمولا در صحبت به دلیل تفاوت انرژی قسمت های باواک و بی واک از معیار SNRseg بصورت زیر استفاده می شود :

$$SNRseg = 1/M \sum_{m=1}^{M} 10\log \frac{\sum_{n=1}^{N} S_{in}^2(n)}{\sum_{n=1}^{N} [S_{out} - S_{in}]^2}$$

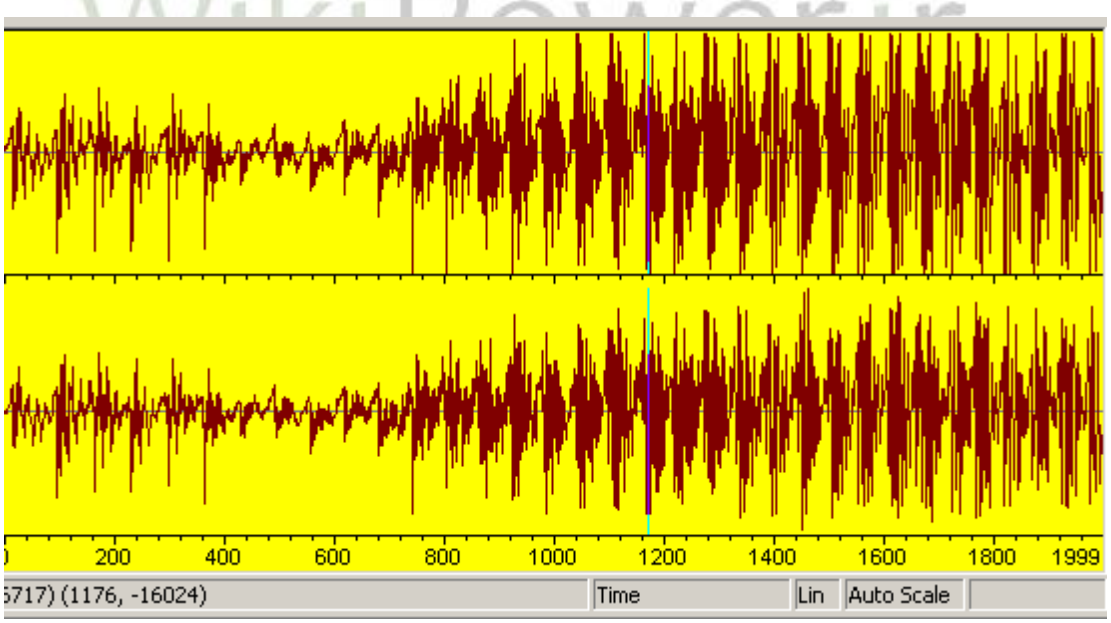
که در آن N تعداد نمونه های صحبت در یک سگمنت که معمولا ۱۲۸ تا ۲۵۶ نمونه است و M تعداد این سگمنت ها می باشد[7].

در شکل (۵-۱۱) صحبت ورودی و صحبت سنتز شده خروجی کدک حاصل از شبیه سازی ممیزثابت ودر شکل(۵-۱۲) مقایسه ورودی وخروجی در پیاده سازی بر روی C5402 نشان داده شده است.فایل صــحبت جمله " بر ســرآنم که گر ز دســت بر آید..." با گوینده مرد می باشــد.همچنین در جدول(۵-۲) مقادیر SNR محاسبه شده برای سگمنت های۲۵۶تایی وکل فایل آورده شده است .

[1] Mean Opinion Score



شکل (۵-۱۱) : مقایسه صحبت ورودی(بالا) و صحبت سنتز شده (پایین) در شبیه سازی



شکل (۵-۱۲) : مقایسه صحبت ورودی(بالا) و صحبت سنتز شده (پایین) در DSP

	SEG256	GLOBAL
SNR (dB)	17.31	17.57

جدول (۵-۲)

در مرجع [7] مقدار SNR برای این کدک از نظر تئوری 22dB محاسبه شده است هرچند که در پیاده سازی ها مثلا مرجع [17] ، SNR کلی کدک برابر 18dB بدست آمده است .

در اینجا نتایج پیاده سازی برروی پردازنده TMS320C5402 از نظر مقدار حافظه برنامه و حافظه دیتا و MIPS مورد نیاز برنامه در جدول های (۵-۳)و (۵-۴) ارائه می شوند :

Program Memory	Data Memory
8582 word	3632 word

جدول (۵-۳)

	Maximum	Average
Encoder	61.13	54.6
Decoder	38.17	32.8
Full Duplex	99.3	87.4

جدول (۵-۴)

برای مقایسه به نتایج پیاده سازی شرکت DCS بر روی C54X که به حافظه برنامه 9 kword و حافظه دیتا 2.1 kword و پردازش 37 MIPS نیاز دارد[19] ،اشاره می کنیم که البته اختلاف MIPS آن با پیاده سازی ما ناشی از برنامه نویسی اسمبلی بصورت کاملا دستی می باشد .

۵-۶- نتیجه گیری و پیشنهاد

در این گزارش به پیاده سازی بلادرنگ کدک صحبت کم تاخیر G.728 بر روی پردازنده TMS320C5402 پرداختیم. در این راه پس از بررسی روش های کدینگ صحبت ومطالعه الگوریتم کدینگ LD-CELP ، این الگوریتم را به زبان C و بصورت ممیزثابت شبیه سازی نمودیم .سپس برد توسعه گر C5402 DSK و نرم افزار CCS (Code Composer Studio) و امکانات آنرا معرفی و نحوه اجرای برنامه بر روی برد را بیان کردیم .با استفاده از نرم افزار CCS برنامه کدک به زبان C را به کد اسمبلی ترجمه و بر روی TMS اجرا نموده و دیدیم که کمپایلر C54X با حداکثر قدرت بهینه سازی هم نمی تواند برنامه را بصورت بلادرنگ اجرا نماید . از اینرو توابع مهم برنامه از نظر MIPS را بصورت دستی به زبان اسمبلی بازنویسی کردیم. در نتیجه MIPS بعضی از توابع به نصف و بعضی به یک سوم کاهش یافت و برنامه بصورت بلادرنگ قابل اجرا گردید.

بعنوان اولین پیشنهاد می توان با اعمال تغییراتی در ساختار کتاب کد LD-CELP 16 kb/s آنرا به یک کدک با نرخ بیت متغیر تبدیل نمود که جزئیات آن در Annex H توصیه نامه G.728 آورده شده است[18] . چنین کدکی با نرخ بیت متغیر برای شبکه های موبایل و اینترنت مناسب است.

پیشنهاد دیگری که در اینجا مطرح می شود ترکیب این کدک با حذف کننده اکو استاندارد G.165 است . چرا که در سیستم های انتقال صحبت بلوک های فشرده سازی و حذف اکو در کنار هم قرار می گیرند و اگر هردو در یک بلوک پیاده سازی شوند می توان از نظر MIPS و حافظه به نتایج بهتری دست یافت.

ضمـا ئـم

– ضمیمه (ب) : مقایسه برنامه نویسی **C** و اسمبلی

```
/*****************************************************
BLOCK 14,15 :SHAPE CODEVECTOR CONVOLUTION &
ENERGY CALCULATION
inputs: H,Y
output: Y2
*****************************************************/
void BLOCK14_15()
{
     int I,J,K,K1=1;
     long AA0;

     for(J=1;J<=NCWD;J++)
     {
       for(K=1;K<=IDIM;K++)
       {
          K1++;//K1=(J-1)*IDIM+K+1;
          AA0=0;
          for(I=1;I<=K;I++)
           AA0+=H[I]*Y[K1-I];

          AA0>>=14;
          TEMP[K]=AA0;
       }
       AA0=0;
       for(K=1;K<=IDIM;K++)
         AA0+=(TEMP[K]*TEMP[K]);
       AA0>>=15;
       Y2[J]=AA0;
     }
  }
```

```
/*****************************************************/
```

کد اسمبلی تولید شده توسط کمپایلر

```
/*****************************************************************/
        .sect ".text"
        .global _BLOCK14_15
        .sym _BLOCK14_15,_BLOCK14_15, 32, 2, 0
        .func 1757
        ;----------------------------------------------------------------------
        ; 1757 | void BLOCK14_15()
        ;----------------------------------------------------------------------
        ;*************************************************************
        ;* FUNCTION DEF: _BLOCK14_15 *
        ;*************************************************************
        _BLOCK14_15:
        .line 2
        .sym _I,0, 4, 1, 16
        .sym _J,1, 4, 1, 16
        .sym _K,2, 4, 1, 16
        .sym _K1,3, 4, 1, 16
        .sym _AA0,4, 5, 1, 32
        PSHM AR1
        FRAME #-6
        NOP
        .line 3
        ;----------------------------------------------------------------------
        ; 1759 | int I,J,K,K1=1;
        ; 1760 | long AA0;
        ;----------------------------------------------------------------------
        ST #1,*SP(3) ; |1759|
        .line 6
        ;----------------------------------------------------------------------
        ; 1762 | for(J=1;J<=NCWD;J++)
        ;----------------------------------------------------------------------
        SSBX SXM
        LD #128,A
        ST #1,*SP(1) ; |1762|
        SUB *SP(1),A ; |1762|
        BC L182,ALT ; |1762|
        ; branch occurs ; |1762|
        L175:
        .line 8
        ;----------------------------------------------------------------------
        ; 1764 | for(K=1;K<=IDIM;K++)
        ;----------------------------------------------------------------------
```

```
LD #5,A
ST #1,*SP(2) ; |1764|
SUB *SP(2),A ; |1764|
BC L179,ALT ; |1764|
; branch occurs ; |1764|
L176:
.line 10
;----------------------------------------------------------------------
; 1766 | K1++;//K1=(J-1)*IDIM+K+1;
;----------------------------------------------------------------------
ADDM #1,*SP(3) ; |1766|
.line 11
;----------------------------------------------------------------------
; 1767 | AA0=0;
;----------------------------------------------------------------------
LD #0,A
DST A,*SP(4) ; |1767|
.line 12
;----------------------------------------------------------------------
; 1768 | for(I=1;I<=K;I++)
;----------------------------------------------------------------------
ST #1,*SP(0) ; |1768|
LD *SP(2),A ; |1768|
SUB *SP(0),A ; |1768|
BC L178,ALT ; |1768|
; branch occurs ; |1768|
L177:
.line 13
;----------------------------------------------------------------------
; 1769 | AA0+=H[I]*Y[K1-I];
;----------------------------------------------------------------------
LD *SP(3),A
SUB *SP(0),A ; |1769|
STLM A,AR1
NOP
NOP
LD *AR1(_Y),T
MVDK *SP(0),*(AR1)
MPY *AR1(_H),A ; |1769|
DADD *SP(4),A,A ; |1769|
ADDM #1,*SP(0) ; |1769|
DST A,*SP(4) ; |1769|
```

```
LD *SP(2),A ; |1769|
SUB *SP(0),A ; |1769|
BC L177,AGEQ ; |1769|
; branch occurs ; |1769|
L178:
.line 15
;----------------------------------------------------------------------
; 1771 | AA0>>=14;
;----------------------------------------------------------------------
DLD *SP(4),A ; |1771|
SFTA A,#-14,A ; |1771|
DST A,*SP(4) ; |1771|
.line 16
;----------------------------------------------------------------------
; 1772 | TEMP[K]=AA0;
;----------------------------------------------------------------------
MVDK *SP(2),*(AR1)
LD *SP(5),A
STL A,*AR1(_TEMP)
.line 17
ADDM #1,*SP(2) ; |1773|
LD #5,A
SUB *SP(2),A ; |1773|
BC L176,AGEQ ; |1773|
; branch occurs ; |1773|
L179:
.line 18
;----------------------------------------------------------------------
; 1774 | AA0=0;
;----------------------------------------------------------------------
LD #0,A
DST A,*SP(4) ; |1774|
.line 19
;----------------------------------------------------------------------
; 1775 | for(K=1;K<=IDIM;K++)
;----------------------------------------------------------------------
ST #1,*SP(2) ; |1775|
LD #5,A
SUB *SP(2),A ; |1775|
BC L181,ALT ; |1775|
; branch occurs ; |1775|
L180:
```

```
.line 20
;----------------------------------------------------------------------
; 1776 | AA0+=(TEMP[K]*TEMP[K]);
;----------------------------------------------------------------------
MVDK *SP(2),*(AR1)
LD *AR1(_TEMP),T
MPY *AR1(_TEMP),A ; |1776|
DADD *SP(4),A,A ; |1776|
ADDM #1,*SP(2) ; |1776|
DST A,*SP(4) ; |1776|
LD #5,A
SUB *SP(2),A ; |1776|
BC L180,AGEQ ; |1776|
; branch occurs ; |1776|
L181:
.line 21
;----------------------------------------------------------------------
; 1777 | AA0>>=15;
;----------------------------------------------------------------------
DLD *SP(4),A ; |1777|
SFTA A,#-15,A ; |1777|
DST A,*SP(4) ; |1777|
.line 22
;----------------------------------------------------------------------
; 1778 | Y2[J]=AA0;
;----------------------------------------------------------------------
MVDK *SP(1),*(AR1)
LD *SP(5),A
STL A,*AR1(_Y2)
.line 24
ADDM #1,*SP(1) ; |1780|
LD #128,A
SUB *SP(1),A ; |1780|
BC L175,AGEQ ; |1780|
; branch occurs ; |1780|
L182:
.line 26
FRAME #6
POPM AR1
RET
; return occurs
.endfunc 1782,000000400h,7
```

```
/*****************************************************/
                کد اسمبلی تولید شده با برنامه نویسی دستی
/*****************************************************/

        .text
        .global _BLOCK14_15
_BLOCK14_15:
        PSHM        AR0
        PSHM        AR1
        PSHM        AR2
        PSHM        AR3
        PSHM        AR4
        PSHM        AR5
        PSHM        AR6
        frame #-2
        ssbx   sxm
        STM   #_Y2+1,AR4
        STM   #0,BK
        STM   #127,AR1
        stm     _Y,ar2
L1:
        STM #_TEMP+1,AR0
        stm #-1,ar6
        stm #4,brc

      rptb L2
        mar *ar2+
        mar *ar6+
        ldm ar2,a
        stlm a,ar5
        ldm ar6,a
        stl a,*sp(0)
        STM #_H+1,AR3
        LD #0,A
         RPT *sp(0)
         MAC *AR5-, *AR3+, A, A

        SFTA A,#-14,A
L2:      STL A,*AR0+
```

```
 STM #_TEMP+1,AR3
 LD #0,b
 rpt #4
   squra *ar3+,b
 SFTA B,#-15,B
 STL B,*AR4+
BANZ L1,*AR1-
 frame #2
 POPM AR6
 POPM AR5
 POPM AR4
 POPM AR3
 POPM AR2
 POPM AR1
 POPM AR0
RET
```

مـرا جع

1- Panos E. Papamichalis, "Practical Approaches to Speech Coding ", Prentice-Hall Inc. ,1987.

2-B.S. Atal & R.Remde "A new model of LPC excited for producing natural-sounding speech at low bit rates", Proc.ICASSP pp.614-617 1982.

3-Jason P. Woodard , "Digital Speech Coding" , Mini-Thesis , Department of Electronics & Computer Science, University of Southampton, Jun 1994.

4-J.Makhoul "Linear Prediction: A Tutorial Review " Proc. IEEE,vol 63, No.4,pp.561-580 Apr 1975.

5- Kondoz A.M," Digital Speech , Coding of low bit rate communication Systems", Chichester Wiley 2000

6- Chen,Cox & Lin,"A Low-Delay CELP Coder for the CCITT 16 kb/s Speech Coding Standard",IEEE Jour. On Selected Area in Comm.,vol.10,no.5 ,June 92.

7- L.Hanzo,A.Somerville & Jason P. Woodard, "Voice Compression and Communication",IEEE series on Digital & Mobile,2001

8- Schroeder & Atal ,"Code-Excited linear Prediction (CELP):High quality speech at very low bit rates",IEEE,ICASSP,pp.937-940,1985.

9- ITU,"Coding of Speech at 16 kbps using Low-Delay Code Excited Linear Prediction ", ITU (CCITT) Recommendation G.728, 1992.

10- ITU," G.728 Annex G , 16 kbps Fixed Point specification ", 11/94.

11- TI, "TMS320C54x Assembly language tools user guide ", www.ti.com June 2001.

12- TI, " TMS320C54x DSP reference set, volume 1: CPU", spru131, www.ti.com.

13- TI, " TMS320C54x DSP reference set, volume 2: Mnemonic instruction set", spru172, www.ti.com .

14- TI, "Code Composer user guide",spru 328, www.ti.com.

15- TI, "C5402 DSK user guide" ,www.ti.com.

16- TI," DSP product tree" , http://dspvillage.ti.com/docs/allproducttree.jhtml.

17- TCTS Lab," The LD-CELP at 16kb/s (ITU-T G.728)",Coding research group homepage ,http://tcts.fpms.ac.be/coding.htm.

18- ITU,"G.728 Annex H:variable bit rate LD-CELP operation mainly for DCME at rates less than 16 kb/s",5/99.

19- DCS G.728 C54x Vocoder, algorithm,http://wwwd.connect.ti.com/dsp/tpcat/tpcodec.nsf/SoftwareForExternal/EAB728D36C3C916E862569F200542A92.