بخش پنجم: پیاده سازی پروتکل G.168

بخش اول

پردازنده‌های *DSP* سری *TMS320C64x*

مقدمه

پردازنده‌هایDSP دسته‌ای از پردازنده‌های خاص می‌باشند که بیشتر برای انجام بلادرنگ پردازش سیگنالهای دیجیتال استفاده می‌شوند. این پردازنده‌ها توانائی انجام چندین عملیات همزمان در یک سیکل دستورالعمل شامل چندین دسترسی به حافظه، تولید چندین آدرس با استفاده از اشاره‌گرها و انجام جمع و ضرب سخت افزاری بطور همزمان را دارا می‌باشند و سرعت بالای آنها نیز به واسطه این ویژگیها است.

اولین پردازنده DSP بصورت تجاری در سال ۱۹۸۲ توسط شرکت TI تحت عنوان TMS320C10 به بازار ارائه شد. TMS32010 یک پردازشگر ممیز ثابت، ۱٦ بیتی با ساختار هاروارد بود.

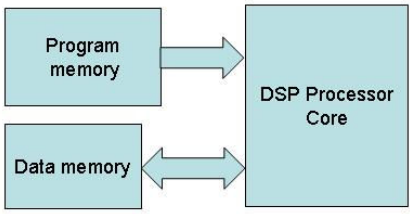
نسل دوم پردازنده‌های DSP در اوایل دهه ۱۹۹۰، توسط شرکت Motorola ارائه گردید. این پردازنده‌ها ۲٤ بیتی بوده و دارای فضای جداگانه داده و برنامه بودند.

در اواخر دهه ۱۹۹۰ دو شرکت TI و Motorola با معرفی دو پردازنده TMS320C541 و DSP56301 نسل سوم پرازنده‌های DSP را به بازار ارائه دادند. کار با ولتاژهای پائین، حافظه داخل چیپ بالاتر و توابع جدید برای پردازش‌های دیجیتال، از ویژگی‌های این نسل می‌باشد.

TMS320C6201 جزو نسل چهارم پردازنده‌های DSP می‌باشد. این پردازنده‌ها سرعت کلاک بالا و قابلیت زیادی برای موازی شدن دستورالعمل‌ها دارند.

مشخصات عمومی پردازشگرهای DSP

پردازنده‌های DSP ازساختار Harvard استفاده می‌کنند. بدین صورت که فضای حافظه شامل کدهای برنامه و داده‌ها از هم جدا می‌شوند و برای هرکدام یک مسیر داده اختصاص داده می‌شود. در ساختارهای بهبود یافته، فضای حافظه داده به چند قسمت تقسیم می‌شود و برای هر قسمت، یک مسیر دسترسی[1] اختصاص داده می‌شود. در ۲شکل (۳-۱) ساختار معماری Harvard نشان داده شده است.



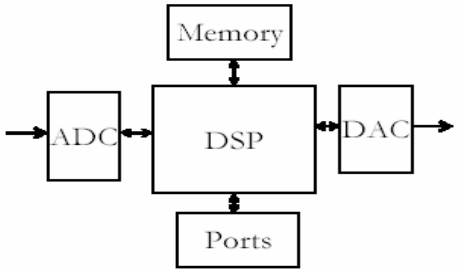
شکل (۳-۱) ساختار معماری ***Harvard***

[1] ***data path***

این پردازنده‌ها یک واحد خاص برای محاسبه آدرس حافظه، قابلیت دسترسی پی درپی به حافظه، قابلیت آدرس‌دهی بصورت غیرمستقیم (بوسیله اشاره‌گر به حافظه)، افزایش آدرس بعد از عملیات[1] و نیز آدرس‌دهی به شیوه دایره وار[2] را تامین می‌کنند[18].

علاوه بر ویژگیهای فوق این پردازنده‌ها دارای روتین‌های سرویس‌دهی وقفه[3] و اجزاء جانبی روی چیپ می‌باشد که ارتباط پردازنده را با دنیای بیرون فراهم می‌کند.

از آنجا که پردازنده‌های DSP اکثراً به منظور انجام عملیات پیچیده ریاضی برروی سیگنالهای بلادرنگ بکار می‌روند دارای سخت افزارخاصی جهت تسریع تکرار و محاسبات ریاضی پیچیده هستند[19] . در شکل (۳-۲) یک سیستم ساده دارای پردازنده DSP نشان داده شده است.



شکل (۳-۲) سیستم پردازش سیگنالهای دیجیتال ***(DSP)***

معماری پردازنده های DSP دارای قسمتهای زیر می باشد:

واحد *MAC*

اکثر پردازنده‌ها عملیات ضرب را با یکسری عملیات انتقال[4] وجمع پیاده‌سازی می‌کنند که هرکدام از آنها نیاز به یک سیکل کاری پردازنده دارند. در پردازنده‌های DSP یک سخت افزار خاص جهت انجام عملیات ضرب دریک سیکل کاری ترتیب داده شده است. در اکثر پردازنده‌های DSP پیشرفته حداقل یک واحد اجرایی[5] برای انجام عملیات ضرب در یک سیکل (و یا عملیات ضرب- جمع) وجود دارد.

دسترسی موثر به حافظه

در واقع انجام عملیات ضرب به بیش از یک سیکل نیاز دارد چرا که علاوه بر دستورالعمل ضرب احتیاج به دسترسی به داده‌های مورد نیاز وجود دارد. پردازنده‌های DSP امکان دسترسی وسیعتری به حافظه نسبت به پردازنده‌های همه منظوره[6] دارند.

---

[1] ***post increment***

[2] ***circular***

[3] ***interrupt***

[4] ***shift***

[5] ***execution unit***

[6] ***General purpose***

ضمن اینکه در پردازنده‌های نسل جدید امکان چند دسترسی به حافظه فراهم شده است و این پردازنده‌ها دارای بانک‌های داده مجزا می‌باشند که هر یک از این بانکها توسط یک مسیرخاص داده[1] قابل دسترسی می‌باشند و می‌توان در یک سیکل در آنها نوشت و یا از آنها خواند. چون حافظه مربوط به دستورالعمل‌ها (کدهای برنامه) و داده‌ها جدای از هم می‌باشند پردازنده می‌تواند در یک سیکل بطور موازی[2] به هر دو دسترسی داشته باشد. در ضمن برای بهینه سازی بهتر، بانکهای کوچکی ازحافظه، ازنوعRAM ، نزدیک پردازنده قرار می‌گیرد که این بانکها حافظه پنهان[3] نام دارند.

وقتی که یک گروه ازدستورالعمل‌ها بصورت مداوم اجرا می‌شوند ،مثلاً دریک حلقه[4]، بانک حافظه پنهان با دستورالعمل‌های مربوطه فوق پر می‌شود و مسیر مربوطه به کدهای برنامه، برای واکشی کردن داده‌ها از حافظه استفاد می‌شود.

مسیرهای دسترسی به حافظه با پهنای بالا نیاز به حمایت واحدهای سخت افزاری خاصی دارند که برای تولید آدرس بکار می‌روند. این واحدها بطور موازی با واحد اصلی اجرای کدها کار می‌کند و دسترسی به داده، در موقعیت جدید حافظه بدون اتلاف زمان جهت محاسبه آدرس جدید را قادر می‌سازد.

از مودهای مهم آدرس‌دهی می‌توان به آدرس‌دهی غیرمستقیم توسط ثبات بهمراه افزایش آدرس قبل و یا بعد از عملیات اشاره کرد.[5] افزایش یا کاهش آدرس بصورت اتوماتیک مربوط به اشاره‌گر به حافظه در الگوریتمهایی که عملیات بصورت تکراری بر روی یکسری از داده‌ها انجام می‌شود، بکارمی‌رود.

درضمن اکثر پردازنده‌های DSP، آدرس‌دهی بصورت دایره‌ای را نیز تامین می‌کنند، بدین صورت که در پردازنده این امکان وجود دارد که بصورت پی درپی به بلوکی از داده دسترسی نماید و بصورت اتوماتیک به آدرس شروع بلوک داده بازگردد. این ویژگی برای دسترسی به ضرائب فیلتر در یک فیلترFIR و پیاده‌سازی بافرهای بصورتFirst In-First out بسیار مفید می‌باشد

از مودهای آدرس‌دهی دیگر که در پردازنده‌ها بکار می‌رود می‌توان به مود آدرس‌دهی"Bit Reversed"، که مناسب برای محاسبات تبدیل فوریه است و نیز بانک‌های ثباتها برای دسترسی خاص اشاره کرد.

واحدهای اجرایی[6] مستقل از هم

پردازنده‌های DSP اکثراً دارای واحدهای اجرائی مستقلی هستند که بصورت موازی قادر هستند کار کنند. بعنوان مثال علاوه بر واحد انجام عملیات ضرب، واحد ALU برای انجام محاسبات منطقی و ریاضی وجود دارد[18] .

نمایش داده و دقت نمایش

بیشتر پردازنده‌های DSP از داده‌ها بصورت نقطه ثابت[7] بجای نمایش نقطه شناور[8]، که اکثراً در کاربردهای مهندسی بکار می‌رود، استفاده می‌کنند. درفرمت نقطه ثابت نقطه باینری (معادل با نقطه ممیز در عملیات ریاضی مبنای۱۰) در مکان ثابتی در

[1] ***data path***

[2] ***parallel***

[3] ***cache***

[4] ***Loop***

[5] ***Register indirect post(pre) increment***

[6] ***execution unit***

[7] ***fixed point***

[8] ***floating point***

طول کلمه[1] قرار گرفته است و این برخلاف نمایش به فرم نقطه شناور است که در آن اعداد به کمک توان و مانتیس نمایش داده می‌شوند و نقطه بانیری طبق ارزش توان تغییر می‌کند. نمایش نقطه شناور این امکان را میدهد که محدوده وسیعتری از داده‌ها را نمایش داد و بصورت مجازی مسأله سرریز[2] در اکثر کاربردها را برطرف کرد.

پردازنده‌های نقطه ثابت ارزانتر بوده و توان کمتری نسبت به پردازنده‌های نقطه شناور(دریک سرعت یکسان) مصرف می‌کنند. پردازنده‌های نقطه شناور نیاز به سخت افزار اضافه‌تر وپیچیده‌تری برای تحقق یافتن ساختار خود دارند.

اغلب پردازنده‌هایDSP نقطه ثابت از داده‌های بطول ١٦ بیت استفاده می‌کنند که برای اکثر کاربردهای DSP مناسب می‌باشند. پردازنده‌های با طول ٢٠ و٢٤ یا ٣٢ بیت دقت بیشتری را فراهم می‌کنند.

برای اطمینان یافتن از اینکه در داده‌های نقطه ثابت کیفیت کافی می‌باشد، اغلب پردازنده‌های DSP دارای یک سری انباشتگر[3] هستند تا نتایج چندین عملیات ضرب را در خود نگه دارند. این ثبات‌ها از دیگر ثباتها وسیعترند(طول بزرگتری دارند)، تابتوانند اعداد بزرگتری را نمایش دهند و از ایجاد سرریز جلوگیری کنند. ضمناً اغلب پردازنده‌ها دارای واحدهایی برای مشخص کردن اشباع در عملیات ریاضی و واحدهای شیفت دهنده می‌باشند.

حلقه‌های بدون بالا سری[4]

اکثر الگوریتمهای پردازش سیگنال دارای حلقه‌های فراوانی می‌باشند. پردازنده‌های DSP برای تحقق حلقه‌ها دارای ساختار خاصی هستند. اغلب یک دستورالعمل تکرار فراهم شده است که به برنامه نویس این امکان را میدهد تا بدون صرف کردن سیکل اضافی, برای به روز کردن و آزمایش کردن شمارەگر حلقه و یا پرش به ابتدای حلقه، یک حلقه را پیاده سازی کند این مشخصات تحت عنوان " zero-overhead loop "نامیده می‌شود.

اجزاء جانبی

جهت سهولت در دریافت و ارسال سیگنالها، پردازنده‌ها شامل واسطه‌های سریال و موازی هستند و مکانیسم خاصی برای ساده کردن عملیات دارند. بعنوان نمونه وقفه‌های دارای کمترین بالاسری و واحد دسترسی مستقیم به حافظه[5] فراهم شده‌اند تا اینکه انتقال و تبادل داده‌ها با کمترین مداخله پردازنده صورت گیرد. همچنین وقفه‌های با اولویت‌های[6] مختلف و قابل ماسک کردن فراهم شده‌اند و در هنگام وقفه‌ها مشخصات و وضیعت پردازنده بصورت اتوماتیک ذخیره می‌شوند.

دستورالعمل‌های خاص[7]

دستورالعمل‌های پردازنده‌ها با در نظر گرفتن دو هدف زیر طرح شده‌اند:

- ازسخت افزار مربوطه بیشترین استفاده صورت بگیرد.
- کمترین فضای حافظه برای ذخیره کردن برنامه‌ها نیاز باشد.

---

[1] ***word***

[2] ***overflow***

[3] ***accumulator***

[4] ***overhead***

[5] ***Direct Memory Access (DMA)***

[6] ***priority***

[7] ***instruction set***

با برآورده کردن هدف اول، در یک دستورالعمل می‌توان چندین عملیات را انجام داد. بعنوان مثال واکشی یک یا دو داده از حافظه بطور موازی با عملیات ریاضی، توسط واحد اجرای مربوطه‌اش انجام می‌شود.

برای تحقق هدف دوم طول دستورالعمل‌ها[1] کوتاه نگه داشته می‌شود و لذا ثباتهای مورد نیاز برای واکشی کدهای برنامه از حافظه محدود می‌شوند. درضمن پردازنده‌ها شامل یکسری بیتهای نشان دهنده وضعیت پردازنده هستند که ویژگی‌های عملکرد پردازنده رانشان می‌دهند؛ مثلاً بیتهایی برای نشان دادن اشباع[2] عملیات و یا گردکردن[3] نتایج.

بهبودکارآیی پردازنده‌های *DSP* معمولی

یک ایده جهت افزایش کارآیی پردازنده‌ها، افزودن واحدهای اجرایی موازی می‌باشد. در عمل اکثراً واحدهای ضرب کننده و جمع کننده اضافه می‌شوند. علاوه بر این تغییرات سخت افزاری، یکسری دستورالعمل جدید نیز ارائه می‌شوند که از امتیازات اضافه شدن واحدهای سخت افزاری استفاده می‌کنند. بدین ترتیب با افزایش موازی کاری[4] در پردازنده‌های نسل جدید می‌توان کارهای بیشتری دریک سیکل انجام داد. این پردازنده‌ها عموماً مسیر داده پهن‌تری را دارا هستند تا در آنها امکان دسترسی داده بیشتری در هر سیکل فراهم شود. هم چنین این پردازنده‌ها از دستورالعمل‌های با طول بزرگتری استفاده می‌کنند تا امکان توصیف انجام عملیات[5] موازی در یک دستورالعمل فراهم شود.

افزایش پیچیدگی سخت افزار باعث افزایش توان مصرفی و قیمت پردازنده می‌شود. فراهم کردن این ویژگی که پردازنده بتواند دستورالعمل‌های بیشتری را درگروههای موازی منتشر و اجرا کند، باعث می‌شود تا توان مصرفی به اندازه پردازنده‌های نسل قبل (معمولی) باقی بماند و در هزینه ساخت آنها صرفه‌جویی شود. لذا با استفاده از دستورالعمل‌های ساده عملیات رمز گشایی و اجرا ساده‌تر و سرعت اجرای برنامه‌ها بالاتر می‌شود[20] .

شرکت TI اولین شرکتی بود که از رهیافت فوق[6] در پردازنده‌های C6000 استفاده کرد. اکنون چهار شرکت اصلی سازنده پردازنده‌ها، یعنی TI، Analog Devices ، Motorola و Lucent از معماری انتشار چندگانه[7] در پردازنده‌های نسل جدید استفاده می‌کنند[22,21] .

دو کلاس ازپردازنده‌هایی که ازخاصیت فوق استفاده می‌کنند، تحت عناوین VLIW[8], SuperScaler شناخته می‌شوند. این دو معماری به جز در نحوه گروه‌بندی دستورالعمل‌ها برای اجرای موازی، کاملاً شبیه به هم هستند.

اکثر پردازنده‌های معماری انتشار چندگانه ازکلاس VLIW هستند. این کلاس از واحدهای اجرایی زیادی استفاده می‌کند و هر واحد اجرایی دستورالعمل‌های مخصوص به خود را دارا است [23]. شکل (۳-۳) یک واحد اجرایی بهمراه مسیر داده‌های مربوطه را نشان می‌دهد.

---

[1] Opcode

2 saturation

3 rounding

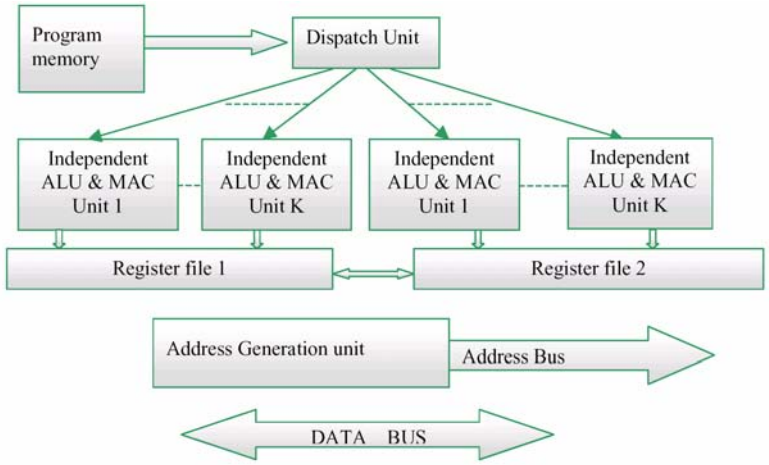
4 parallelism

5 operation

[6] multi-issue

[7] multi-issue

[8] Very Long Instruction Word



شکل (۳-۳) بلوکهای اصلی پردازنده بهبود یافته

پردازنده‌های با معماری VLIW اکثراً تعداد زیادی مسیر برای دسترسی به حافظه داده و تغذیه واحدهای اجرایی مختلف را دارا هستند.

این پردازنده‌ها اغلب از توان مصرفی بالاتری نسبت به پردازنده‌های معمولی استفاده می‌کنند.

ساختار *SIMD*[1]

SIMD به تنهایی یک معماری نمی‌باشد، بلکه تکنیکی است که می‌تواند در هر معماری بکار رود و کارآیی پردازنده را در بعضی از الگوریتمها افزایش دهد. بدین ترتیب که به پردازنده این امکان را می‌دهد که چندین نوع مثال از یک عملیات را بصورت موازی با داده‌های مختلف اجرا کند.

مثلاً یک دستورالعمل ضرب از نوع SIMD می‌تواند دو یا چندین ضرب را بر روی داده‌های مختلف ورودی و در یک سیکل (بصورت موازی) اجرا کند. این نوع عملیات درکاربردهای پردازش سیگنال و مالتی مدیا بسیار بکار می‌رود.

پردازنده‌های شرکت TI

شرکت TI از جمله شرکتهای مطرح در ساخت پردازنده‌های DSP می‌باشد. DSPهای این شرکت شامل پردازنده‌های نقطه ثابت[2]، نقطه شناور[3] و چندتایی[4] می‌باشد. این خانواده عموماً برای پردازش بلادرنگ سیگنالها طرحی شده است. در سال 1982 شرکت TI اولین پردازنده نقطه ثابت خود را با عنوان TMS320C10 تولید کرد. در حال حاضر خانواده TMS320 شامل سریهای زیر می‌باشد:

- سری ***C1x*** ، ***C2x***، ***C27x*** ، ***C5x*** ، ***C54x*** و ***C55x*** پردازنده‌های نقطه ثابت
- سری ***C4x*** و ***C3x*** پردازنده‌های نقطه شناور

[1] ***Single Instruction Multiple Data***

[2] ***Fixed Point***

[3] ***Floating Point***

[4] ***Multiprocessor***

- سری ***C8x*** پردازنده‌های ***Multiprocessor***

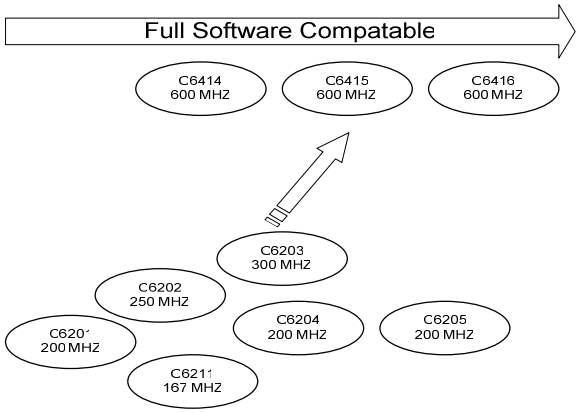
در ضمن نسل جدید پردازنده‌های این خانواده با نام TMS320C6000 با قابلیتها و توانائی‌های بسیار بالا برای کاربردهای پردازش سیگنال به بازار ارائه شده است.

پردازنده‌های سری *C6x*

اولین سری پردازشگرهای C6000 در سال ۱۹۹۷ ارائه شده است. در این سری، پردازشگرهای C62x و C64x از نوع fixed point و سری C67x از نوع floating point می‌باشند.

این سری با قابلیت اجرای حداکثر ٤٨٠٠ میلیون دستورالعمل در ثانیه[1] و داشتن کامپایلر موثر زبان C امکانات بسیار زیادی را فراهم می‌کند. توانایی بسیار بالا، استفاده راحت‌تر و قیمت مناسب باعث شده که برای کاربردهای چند کاناله و چند منظوره بسیار ایده‌آل باشند[21].

مقایسه‌ای بین سرعت پردازنده‌های مختلف این سری صورت گرفته است.شکل (۳-٤) در



شکل (۳-۴) مقایسه سرعت پردازنده‌های سری ***C6x***

مشخصات کلی پردازندهای سری*C6x*

سری C6000 این قابلیت را دارد که در هر سیکل تا ۸ دستورالعمل ۳۲ بیتی را اجرا کند. واحد CPU در این سری شامل دو بانک ثبات همه منظوره A وB ، با ۳۲ بیت طول، و ۸ واحد کاری[2] (دو ضرب کننده و شش ALU[3]) می‌باشد. در ضمن این سری دارای ابزار برنامه نویسی بهینه شده شامل کامپایلر قدرتمند زبان C، بهینه‌ساز زبان اسمبلی(برای ساده کردن برنامه‌نویسی به زبان اسمبلی)، شبیه‌ساز و اشکال زدای کدهای برنامه می‌باشد.

مشخصات سری C6x را می‌توان بصورت زیر خلاصه کرد:

- واحد CPU با دستورالعمل‌های طویل[4] ، شامل دو ضرب کننده و شش واحد ریاضی

[1] ***MIPS***

[2] ***Function unit***

[3] ***Arithmetic Logic Unit***

[4] ***VLIW***

- اجرای حداکثر هشت دستورالعمل در هر سیکل که باعث افزایش قابلیت ۱۰ برابر این پردازنده نسبت به پردازنده‌های دیگر می‌شود.
- بسته‌بندی[1] دستورالعمل؛ باعث می‌شود حجم کد هنگامی که هشت دستورالعمل بصورت سریالی یا موازی با هم اجرا می‌شوند یکسان بماند و نیز توان مصرفی کاهش یابد.
- اجرای همه دستورالعمل‌ها بصورت شرطی؛ باعث می‌شود در عملیات پرش کردن[2] وقت کمتری صرف شود و نیز اجرای موازی دستورالعمل‌ها افزایش یابد.
- اجرای کدها در واحدهای کاری مستقل از هم (افزایش اجرا شدن موازی کدها)
- حافظه با ساختار قوی برای دسترسی داده بصورت ۸ یا ١٦ یا ۳۲ بیت در کاربردهای مختلف
- وجود انجام عملیات ریاضی بصورت ٤٠ بیتی که دقت بالاتری را در کاربردهای صوتی و دیگر کاربردهای پیچیده فراهم می‌کند.
- اشباع کردن و نرمالیزه کردن؛ که در عملیات ریاضی بسیار مفید می‌باشند.
- ضرب کردن اعداد صحیح ۳۲ بیت در ۳۲ بیت و فراهم کردن نتیجه بصورت ۳۲ یا ٦٤ بیتی
- انطباق پینهای پردازنده‌های نقطه ثابت و نقطه شناور
- دسترسی به حافظه خارجی از طریق EMIF [3]

در ضمن سری C64x علاوه بر قابلیتهای فوق مشخصات زیر را نیز دارا می‌باشد:

- افزایش تعداد ثباتها
- گسترش عرض مسیر داده[4]
- افزایش واحدهای سخت افزاری

معماری پردازنده‌های *C6x*

شکل (۳-۵) بلوک دیاگرام پردازندهای این خانواده رانشان می‌دهد. سریC6x شامل حافظه برنامه قرارگرفته داخل چیپ[5] می‌باشدکه می‌تواند بصورت حافظه پنهان نیز استفاده شود. البته سریهای مختلف شامل حجم‌های متفاوتی ازحافظه می‌باشند. اجزا جانبی ازقبیل پورت سریال و موازی و رابط دسترسی به حافظه خارجی[6] همراه چیپ موجود می‌باشند.

واحدCPU پردازشگر مرکزی شامل قسمتهای زیراست[21]:

- واحد واکشی[7] کد برنامه

---

[1] ***Packing***

[2] ***Branch***

[3] ***External Memory Interface***

[4] ***Data Path***

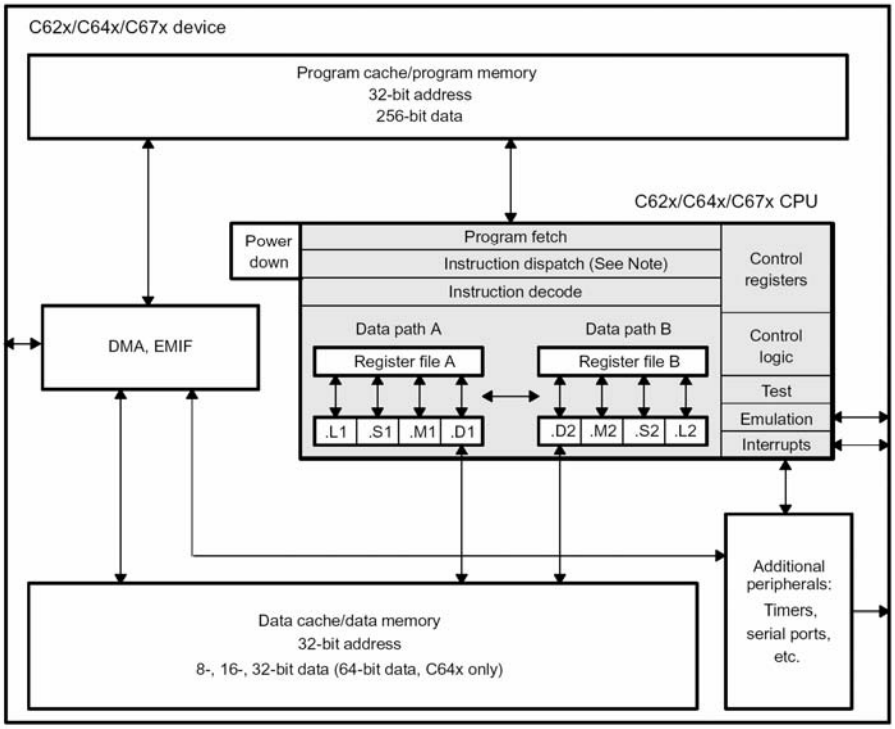
[5] ***internal***

[6] ***EMIF***

[7] ***fetch***

- واحد پخش کردن[1] دستورالعمل‌ها و دسته بندی
- واحد رمزگشایی
- واحد اجرای دستورالعمل
- بانکهای ثبات ۳۲ بیتی کنترلی
- واحد کنترل وقفه و...

بانکهای ثبات همه منظوره شامل دو بانکA و B می‌باشند. بانک A شامل ثباتهای A0 تا A15 و بانک B نیز شامل ثباتهای B0 تا B15 هستند(در سریC64X بانکها ۳۲ تایی هستند). این ثباتهای همه منظوره، می‌توانند برای ذخیره داده، اشاره‌گر آدرس به داده و عملیات شرطی استفاده شوند برای داده‌های با سایز بزرگتر از۳۲ بیت (مثلاً٤٠ بیت) داده‌ها در یک جفت ثبات ذخیره می‌شوند. بدین صورت که ۳۲ بیت کم ارزش در ثبات با شماره زوج و ۸ بیت با ارزش در ثبات بعدی با شماره فرد ذخیره می‌شوند (مثلا A1:A0 برای ذخیره ٤٠ بیت داده بکار می‌رود).



شکل (۳-۵) بلوک دیاگرام پردازنده‌های ***C6000***

واحدهای کاری[2]

هشت واحد کاری درسریC6x به دوگروه چهارتایی تقسیم می‌شوند.

هر واحد کاری در یک مسیر داده تقریباً برابر با واحد کاری در مسیر داده گروه دیگر است. جزئیات واحدهای کاری در ۲جدول (۱-۲) آمده است.

[1] ***dispatch***

[2] ***function unit***

هر واحد دارای پورت خاص خود برای نوشتن در بانک ثبات‌هاست. واحدهای گروه ۱ به بانک ثبات A و واحدهای گروه ۲ به بانک ثبات B دسترسی دارند. هر واحد کاری دارای دو پورت برای خواندن[1] داده‌ها می‌باشد[21].

جدول (۱-۲) توصیف واحدهای کاری در پردازنده‌های ***TMS320C6000***

Functional Unit	Fixed-Point Operations	Floating-Point Operations
.L unit (.L1, .L2)	32/40-bit arithmetic and compare operations 32-bit logical operations Leftmost 1 or 0 counting for 32 bits Normalization count for 32 and 40 bits	Arithmetic operations DP → SP, INT → DP, INT → SP conversion operations
.S unit (.S1, .S2)	32-bit arithmetic operations 32/40-bit shifts and 32-bit bit-field operations 32-bit logical operations Branches Constant generation Register transfers to/from control register file (.S2 only)	Compare Reciprocal and reciprocal square-root operations Absolute value operations SP → DP conversion operations
.M unit (.M1, .M2)	16 x 16 multiply operations	32 X 32-bit fixed-point multiply operations Floating-point multiply operations
.D unit (.D1, .D2)	32-bit add, subtract, linear and circular address calculation Loads and stores with 5-bit constant offset Loads and stores with 15-bit constant offset (.D2 only)	Load doubleword with 5-bit constant offset

مسیر متقاطع[2] در ستونهای ثباتها

هر واحد کاری بطور مستقیم در بانک مربوط به خود می‌نویسد و می‌خواند. درضمن هر واحد کاری به بانک ثبات مقابل خود ازطریق مسیرهای متقاطع دسترسی دارد (مسیر1X برای واحدهایA و مسیر2X برای واحدهای B).

[1] ***read***

[2] ***cross path***

مسیرهای بارگذاری[1] وذخیره کردن[2] درحافظه

پردازنده‌هایC6x دارای یک مسیر ۳۲ بیتی برای بارگذاری داده ازحافظه در ثباتها می‌باشد. مسیر LD1 برای بانک ثبات A و مسیر LD2 برای بانک ثبات B بکارمی‌رود.

مسیرهای آدرس داده[3]

میسرهای آدرس داده معین (DA1,DA2) که به واحد D. متصل شده‌اند، امکان تولید آدرس داده را فراهم می‌کنند. مسیر آدرس DA1 و دیگر مسیرهای داده برای واحدهای A ، بصورت T1 نمایش داده می‌شود. همینطور مسیر آدرس DA2 و دیگر مسیرهای داده مربوطه برای واحدهای B با T2 نمایش داده می‌شود.

بنابراین T1 شامل مسیر آدرس DA1 و مسیرهای داده LD1 و ST1 می‌باشد.

مثلاً در دستورالعمل بارگذاری بصورت رو به‌رو : $LDW \quad .D1T2 \quad *A0[3],B1$

از واحد T1 برای تولید آدرس ولی از LD2 و DA2 برای مسیر داده به بانک ثبات B استفاده می‌کنند [19].

ساختار حافظه درپردازنده‌های *C6x*

پیکربندی حافظه داخلی[4] بین پردازنده‌های سری C6x متغیر می‌باشد. ولی همه‌ی پردازنده‌ها شامل قسمتهای زیر هستند[24]:

حافظه برنامه و داده داخلی، اجزاء جانبی[5] داخلی، دسترسی به حافظه خارجی ازطریق واسط مربوط(EMIF)

پردازنده‌های سری*C620X,C670X*

این پردازنده‌ها شامل حافظه‌های داده و برنامه جدا از هم هستند. حافظه برنامه داخلی می‌تواند به فضای قابل آدرس‌دهی توسط پردازنده نگاشت شود و یا بصورت حافظه پنهان عمل کند.

یک مسیر با پهنای ۲۵٦ بیت برای دسترسی پردازنده به دستوراالعمل‌های پی درپی درحافظه (۸ دستورالعمل ۳۲ بیتی) برای افزایش کارآیی فراهم شده است.

حافظه داده داخلی به بانک‌های با عرض ۱٦بیت تقسیم می‌شوند و کنترلر حافظه داده داخلی امکان دسترسی پردازنده را به بلوکهای حافظه داده فراهم می‌کند.

پردازنده‌های سری*C621x,C671x,C64x*

در معماری این پردازنده‌ها حافظه‌های پنهان داده و برنامه با سطح‌های مختلف وجدا از هم فراهم شده‌اند. حافظه داده و برنامه سطح یک (L1P,L1D) جدا از هم می‌باشند و درفضای آدرس‌دهی حافظه نگاشت نشده‌اند. این حافظه همیشه فعال است و تنها توسط پردازنده (هر دو واحد کاری A,B) قابل دسترسی می‌باشد.

---

[1] load

[2] store

3 data address path

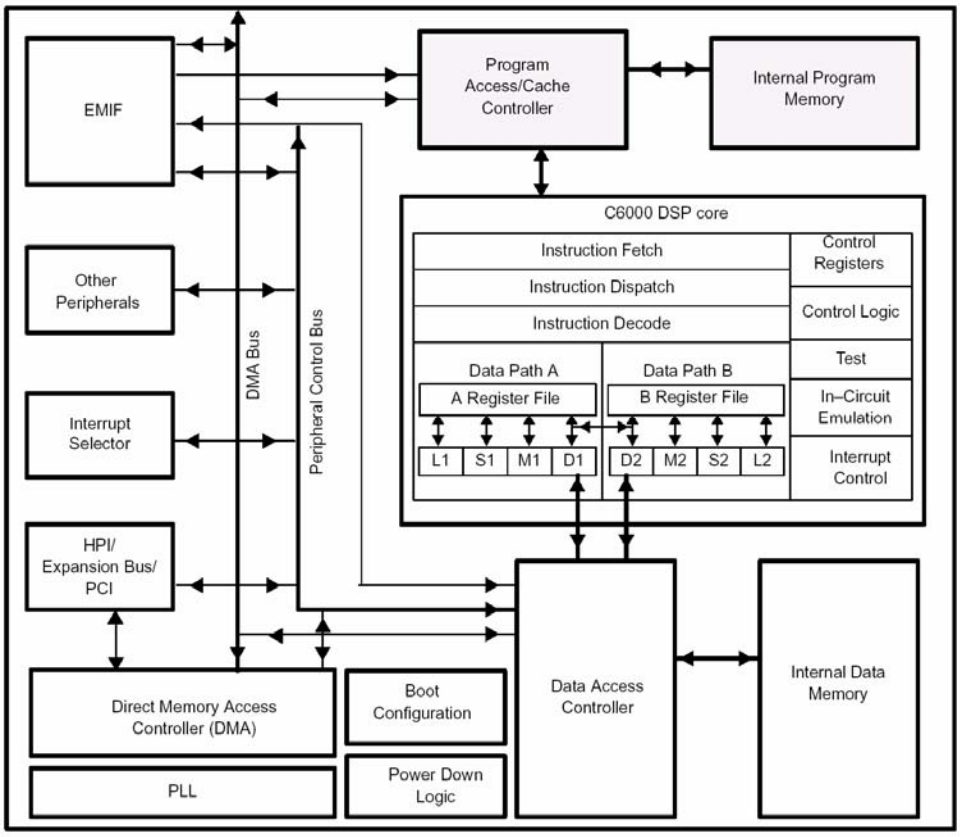
[4] internal

[5] peripheral

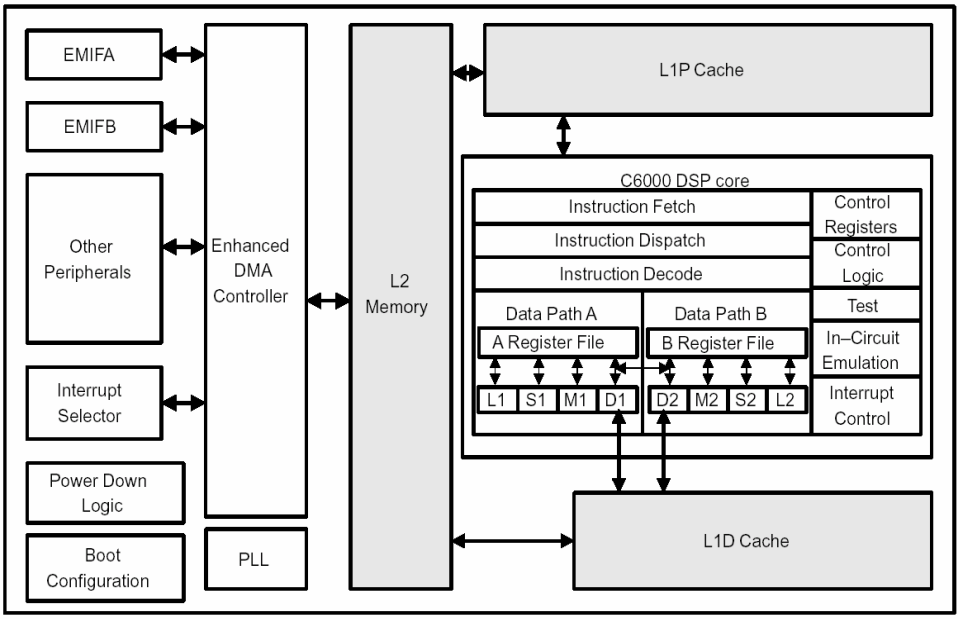
یک مسیر با عرض۲۵٦ بیت از پردازنده به حافظه پنهان برنامه سطح یک(L1P) برای دسترسی به ۸ دستورالعمل ۳۲ بیتی فراهم شده است.

اگر داده و یا برنامه مربوطه در حافظه پنهان سطح یک یافت نشود تقاضا به کنترل کننده با سطح دو (L2) پاس داده می‌شود. حافظه پنهان سطح دو توسط پردازنده و (E)DMA قابل دسترسی می‌باشد. این حافظه از نوع RAM می‌باشد و می‌تواند بصورت فضای قابل آدرس‌دهی توسط پردازنده (شامل کدهای برنامه یا داده یا بصورت حافظه پنهان و یا ترکیبی از این دو) نگاشت شود[24].

بلوک دیاگرام حافظه و سایر اجزاء مربوطه پردازنده‌های سریC6x در شکل (۳-٦) و شکل (۳-۷) نشان داده شده است. [19].



شکل (۳-۶) بلوک دیاگرام حافظه درپردازنده‌های سری***C620X,C670X***



شکل (۳-۷) بلوک دیاگرام حافظه در پردازنده‌های سری *C621x,C671x,C64x*

اجزاء جانبی پردازنده‌های سری *TMS320C6000*

اجزاء جانبی قابل دسترس توسط کاربر از طریق یکسری ثباتهای کنترلی، نگاشت شده در فضای حافظه، پیکربندی می‌شوند. کنترل کننده باس اجزاء جانبی الویت‌دهی دسترسی به اجزاء روی چیپ را به عهده دارد. بلوک دیاگرام اجزاء جانبی در پردازنده‌های C64x در شکل (۳-۷) آمده است.

کاربردهای اصلی اجزاء جانبی موجود در پردازنده بصورت زیر خلاصه شده‌اند[24]:

- **کنترل کننده DMA**

این کنترل کننده، داده‌ها را بین محدوده‌های قابل آدرس‌دهی، بدون دخالت CPU، انتقال می‌دهد. کنترل کننده DMA چهار کانال قابل برنامه‌ریزی و یک کانال کمکی پنجم را دارا می‌باشد.

- **کنترل کننده EDMA**

کنترل کننده EDMA کارهایی شبیه کنترل کننده DMA انجام می‌دهد. در سری C64x کنترل کننده EDMA ، دارای ٦٤ کانال قابل برنامه‌ریزی، شامل فضایی شبیه RAM برای ذخیره کردن نحوه انتقالات، می‌باشد.

- **پورت سریال بافر شده چند کاناله(MCBSP[1])**

این پورت، بر اساس پورتهای سریال استاندارد در سریهای C5000 وC2000 می‌باشد و می‌تواند نمونه‌های سریالی بافر شده را بصورت اتوماتیک، به کمک کنترل کننده DMA یا EDMA ، در حافظه ذخیره کند. در ضمن این پورت قابلیتهای چند کاناله سازگار با استانداردهای E1 و T1 و امثال آن را دارد. پورت سریال خواص زیر را نیز دارا می‌باشد:

- ارتباط دو طرفه

[1] Multi Channel Buffered Serial Port

- ثبات‌های داده بافر دوگانه، که امکان دریافت پیوسته داده را فراهم می‌کند.
- فریم بندی و پالس ساعت مستقل برای دریافت و ارسال
- قابلیت اتصال مستقیم به کدکننده‌های صنعتی استاندارد، مبدل‌های آنالوگ به دیجیتال(A/D) و دیجیتال به آنالوگ(D/A).
- ارسال و دریافت چند کاناله، حداکثر تا ۱۲۸ کانال
- امکان انتخاب اندازه داده شامل داده‌های ۸ ، ۱۲ ، ۱٦ ، ۲۰ ، ۲٤ و ۳۲ بیتی
- فشرده کردن و نافشرده کردن A-law و μ-law
- تنظیم پلاریته برای فریم سنکرون کننده و پالس ساعت
- قابلیت تولید فریم سنکرون و پالس ساعت بصورت درونی

- **انتخاب‌کننده وقفه[1]**

اجزاء جانبی پردازنده‌های سری C6000 وقفه‌های مختص به خود را تولید می‌کنند. انتخاب‌کننده وقفه، این امکان را به کاربر می‌دهد که وقفه‌هایی را که نیاز دارد انتخاب کند. در ضمن می‌توان پلاریته وقفه‌های خارجی را تغییر داد.

مزایای سری *C64x*

C64x از ساختار VelociTI.2 استفاده می‌کند. مهمترین مزایای این سری عبارتند از:

- **Register File Enhancement**

تعداد رجسیترها در این سری دو برابر شده است. در C62x، 32 رجسیتر ۳۲ بیتی وجود دارد در حالی که در C64x، 64 رجسیتر ۳۲ بیتی وجود دارد. همچنین علاوه بر ثباتهای A1،A2، B0، B1، B2 که در C62x برای دستورات شرطی استفاده می‌شد، A0 نیز می‌تواند استفاده گردد. در C62x، فرمتهای ۱٦، ۳۲ و ٤۰ بیتی پشتیبانی می‌شوند در C64x علاوه بر فرمتهای فوق، فرمتهای ۸ و ٦٤ بیتی نیز پشتیبانی می‌گردند.

- **Data Path Extension**

واحد D. می‌تواند در یک کلاک عملیات خواندن و نوشتن در حافظه(Load & Store) را برای یک داده ٦٤ بیتی(Double Word) انجام دهد. در C62x این توانایی وجود ندارد و در C67x نیز تنها Load امکان پذیر است. همچنین در C62x، D. نمی‌تواند از ساختار Cross-path استفاده کند ولی در این سری این امکان برای D. نیز وجود دارد.

- **Advance Instruction Packing**

در C62x، ۸ دستور در هر کلاک fetch می‌شود. همچنین در C62x لازم بود بعد از بعضی از دستورها NOP اضافه کرد. در C64x از ساختار VelociTI.2 استفاده می‌شود. در این سری دیگر NOP لازم نیست و در نتیجه حجم کد برنامه کاهش می‌یابد.

- **Packed Data Processing**

در C64x دستورات زیادی وجود دارند که می‌توانند بر روی ٤ داده ۸ بیتی و یا دو داده ۱٦ بیتی کار کنند.

- **Additional Function Unit Hardware**

تغییرات سخت افزاری روی آنها اجرا شده است. این تغییرات باعث اضافه .D, .S, .L, .M برای افزایش توانایی واحدهای شدن تواناییهای زیر شده است.

[1] Interrupt Selector

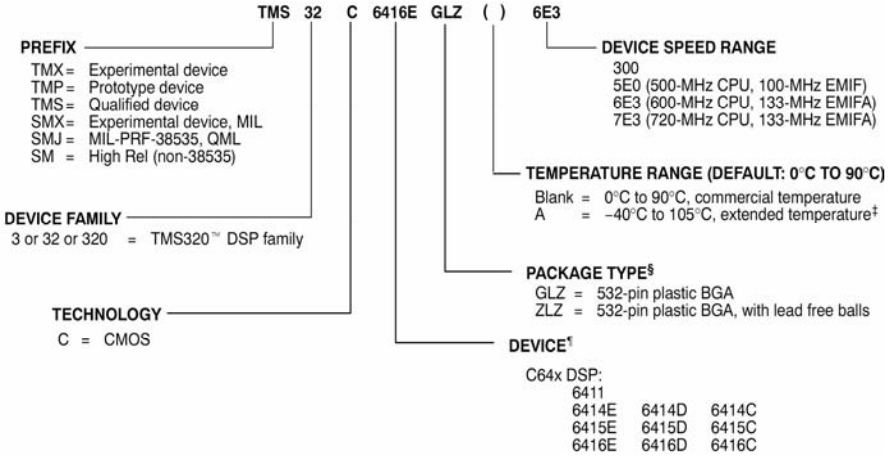
- هر واحد M. می‌تواند دو ضرب ۱۶×۱۶ یا چهار ضرب ۸×۸ را در هر کلاک انجام دهد.
- واحدهای D.، می‌توانند بر روی Word یا Double Word عملیات خود را انجام دهند.
- واحدL.، می‌تواند بایتها را شیفت دهد. M. می‌تواند عملیات شیفت bi-directional را انجام دهد. S. نیز میتواند هر دو کار فوق را انجام دهد.
- L. می‌توان چهار تفریق ۸ بیتی را در یک زمان انجام دهد.
- دستورات جدیدی مانند SHFL، DEAL و GMPY4 توانایی شمارش بیت و چرخش در واحد M. باعث افزایش عملکرد این پردازشگر برای الگوریتم‌های مبتنی بر بیت، مانند پردازش تصویر و رمزنگاری شده است.

- **Increased Orthogonality**

- واحدD.، می‌تواند مانند S. و L. عملیات منطقی ۳۲ بیتی را انجام دهد.
- واحدهای L. و D. می‌توانند یک ثابت ۵ بیتی و S. می‌تواند یک ثابت ۱۶ بیتی را از حافظه بخوانند.
- در C62x تنها اگر یکی از پارامترهای یک اپراتور طولانی باشد (تعداد بیت زیاد داشته باشد) و جواب نیز طولانی باشد، عملیات امکان پذیر است. ولی در C64x اگر هر دو پارامتر طولانی و نتیجه نیز طولانی باشد در یک سیکل عملیات انجام می‌شود.
- از مزایای سری C6416، اضافه شدن دو پردازشگر همراه[1] برای دیکد کردن ویتربی[2] و توربوکد[3] می‌باشد. این توانایی باعث محبوبیت زیاد این سری در پروژه‌های مخابرات دیجیتال گردیده است.
- همچنین علاوه بر اینکه تمام دستورات C62x در C64x قابل اجرا است، در سری C64x دستوراتی وجود دارد که بر روی ۸ بیت یا ۱۶ بیت قابل اجرا است مثلاً دستور چهار ضرب ۸×۸ را در یک دستور انجام می‌دهد.

پردازنده C6416 از خانواده پردازنده‌های TMS320C6000

مجموعه لغات این خانواده از پردازنده‌ها آورده شده است.شکل (۳-۸) در



شکل (۳-۸) مجموعه لغات پردازنده سری ***C64x***

[1] ***Co-Processor***

[2] ***Viterbi Co-Processor***

[3] ***Turbo Co-Processor***

. از ویژگی‌های این پردازنده:[25] نقطه ثابت می‌باشدDSP پردازنده با قابلیت بالا و جزء پردازنده‌هایC6416

- هر سیکل آن ۲ ، ۱/٦۷ یا ۱/۳۹ نانو ثانیه طول می‌کشد(کلاک‌های۵۰۰، ٦۰۰ یا ۷۲۰ مگاهرتز).
- قابلیت اجرای هشت دستورالعمل در یک سیکل را دارد و لذا توانایی در حد ٤۰۰۰، ٤۸۰۰ یا ۵۷٦۰ دستورالعمل در ثانیه[1] را دارا می‌باشد.
- دستورالعمل‌های آن با پردازنده‌های سری ***C62x*** سازگار می‌باشد.
- پینهای سخت افزاری آن با پردازنده‌های ***C6415*** و ***C6414*** سازگار می‌باشد.
- ***CPU*** آن از تکنیک ***VLIW*** استفاده می‌کند
- شامل هشت واحد کاری[2] مستقل به صورت زیر می‌باشد:
  - شش واحد ریاضی منطقی[3] که هر کدام در هر سیکل می‌تواند عملیات‌های ۳۲ ، ۱٦ یا ۸ بیتی را انجام دهد.
  - دو واحد ضرب کننده که در هر سیکل می‌توانند ضرب ۱٦×۱٦ بیت با نتیجه ۳۲ بیتی یا ۸×۸ بیت با نتیجه ۱٦ بیتی را انجام دهند.
- شامل ٦٤ ثبات ۳۲ بیتی همه منظوره هستند.
- دارای دو واسط ٦٤ و ۱٦ بیتی برای دسترسی به حافظه بیرونی.
- دارای چیپ ۵۳۲ پین و از نوع ***BGA*** [4]
- دارای مولد کلاک ***PLL*** [5] برای ضرب کردن فرکانس پالس کلاک ورودی(***6***× یا ***12***×)
- از واحدی به نام بسته‌بندی دستورالعمل[6] استفاده می‌کند که حجم کد را کاهش می‌دهد.

دستور العمل‌های برنامه[7] خواص زیر را دارا هستند:

- قابلیت آدرس‌دهی بصورت مضربی از بایت (کار با داده‌های ۸ ، ۱٦ ، ۳۲ و ٦٤ بیتی).
- قابلیت نرمالیزه کردن و اشباع کردن نتیجه عملیات و آشکار کردن سر ریز.

ساختار حافظه در این پردازنده بصورت زیر است:

- دارای حافظه پنهان برنامه سطح اول ***(L1P)*** با ظرفیت ۱٦ کیلو بایت و از نوع ***Direct Mapped***
- دارای حافظه پنهان داده سطح اول ***(L1D)*** با ظرفیت ۱٦ کیلو بایت و از نوع ***Set-Associative***
- دارای حافظه درونی سطح دوم ***(L2)*** با ظرفیت۱ مگا بایت و قابل استفاده بصورت ***RAM*** یا حافظه پنهان[8].

از اجزاء جانبی این پردازنده به موارد زیر می‌توان اشاره کرد:

- واحد دسترسی به حافظه گسترش یافته ***(EDMA)*** و شامل ٦٤ کانال مستقل.

---

[1] ***MIPS***

[2] ***Function unit***

[3] ***ALU***

[4] Ball Grid Array

[5] ***Phase Lock Loop***

[6] Instruction Packing

[7] Instruction Set

[8] ***Cache***

- پورت ***HPI*** بصورت ١٦ یا ٣٢ بیتی.
- واسط ***PCI*** ٣٢ بیتی و با فرکانس ***32 MHZ***.
- سه پورت سریال بافر شده چند کاناله***(MCBSP)***
- سه تایمر ٣٢ بیتی همه منظوره
- ١٦ پین همه منظوره
- پورت مخصوص اشکال‌زدایی سخت افزاری[1] سازگار با استاندارد ***IEEE1149.1***

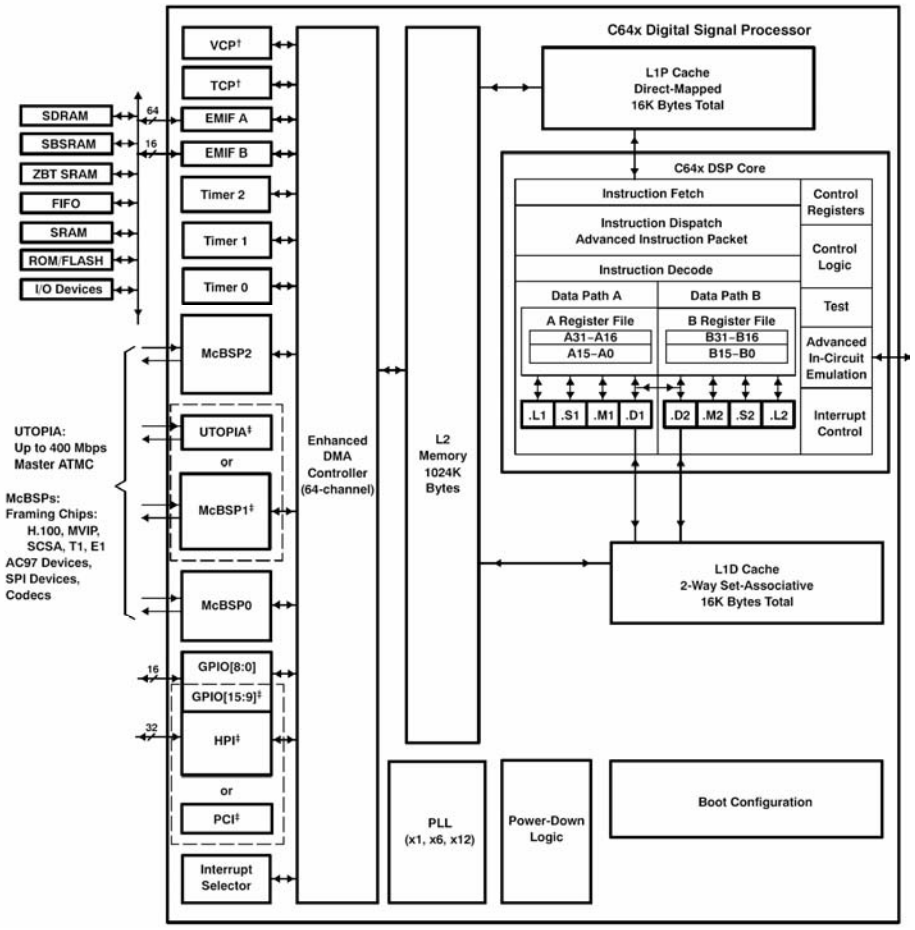
شکل (٣-٩) بلوک دیاگرام این پردازنده و اجزاء جانبی آنرا نشان می‌دهد.

واحد مرکزی پردازنده[2] دو دسته واحد کاری دارد. هر دسته دارای چهار واحد و یک بانک ثبات می‌باشد. دسته اول شامل واحدهای L1. ، S1. ، D1. و M1. و دسته دوم شامل واحدهای L2.، S2. ، D2. وM2. می‌باشد. هر بانک ثبات شامل ٣٢ ثبات ٣٢ بیتی همه منظوره می‌باشد. در کل ٦٤ ثبات ٣٢ بیتی همه منظوره وجود دارد. هر دسته می‌تواند به بانک ثباتهای طرف مقابل از طریق یک مسیر متقاطع[3] دسترسی یابد. همه دستورالعمل‌ها بر روی ثباتها عمل می‌کنند و دو واحد آدرس‌دهی داده (واحدهای D1. وD2.) مسئول انتقال داده‌ها بین ثباتها و حافظه هستند. آدرس داده محاسبه شده توسط واحد D. برای بارگذاری و ذخیره داده‌های درون بانکهای ثباتها به حافظه بکار می‌رود.

[1] ***JTAG***

[2] ***CPU***

[3] cross path



شکل (۳-۹) واحدهای کاری پردازنده ***C6416*** و اجزاء جانبی آن

واحد D. داده‌ها را بصورت ۸ ، ۱۶، ۳۲ یا ٦٤ بیت می‌تواند در یک سیکل بارگذاری و یا ذخیره کند. در ضمن دستورات بارگذاری و ذخیره غیر ترتیبی[1] در واحد D. این امکان را فراهم می‌کند که بتوان به کلمه‌ها[2] و کلمه‌های دوتایی[3] در هر محدوده بایتها دسترسی داشت. پردازنده C64x انواع مختلف مدهای آدرس‌دهی غیر مستقیم بصورت خطی و دایره‌ای را دارا می‌باشد. همه دستورالعمل‌ها بصورت شرطی[4] اجرا می‌شوند و می‌توانند به هر کدام از ٦٤ ثبات دسترسی یابند. دو واحدکاری M. انجام عملیات ضرب را بر عهده دارند. این ضربها می‌توانند بصورت ۸ * ۸ ، ۱٦ * ۱٦ و حتی ۱٦ * ۳۲ بیتی در یک سیکل کاری پردازنده انجام شوند. واحدهای کاری L. و S. انجام عملیات ریاضی، منطقی و پرش[5] را در هر سیکل کاری برعهده دارند.

[1] ***non-aligned***

[2] ***Word***

[3] ***double word***

[4] ***conditional***

[5] ***branch***

وقتی یک بسته واکشی[1] دستورالعمل ۲۵۶ بیتی از حافظه برنامه واکشی شود، عملیات پروسس آغاز می‌شود. دستورالعمل‌های ۳۲ بیتی، بر حسب وجود عدد ۱ در بیت کم ارزش ، به هم متصل می‌شوند و به واحدهای کاری مخصوص نسبت داده می‌شوند. دستورالعمل‌هایی که با هم بصورت همزمان (موازی با هم) اجرا می‌شوند یک بسته اجرایی را تشکیل می‌دهند. در هر بسته واکشی از یک تا هشت بسته اجرایی می‌تواند وجود داشته باشد. بسته‌های اجرایی بر حسب واحد کاری مربوط به خود پخش می‌شوند و تا وقتی که همه بسته‌های اجرایی از بسته واکشی فعلی پخش نگردند بسته واکشی جدیدی از حافظه واکشی نمی‌شود[25].

نگاشت حافظه در پردازنده *C6416*

۲جدول (۱-۲) محدوده آدرسهای حافظه در پردازنده C6416 را نشان می‌دهد. حافظه درونی همواره از آدرس0 شروع می‌شود و می‌تواند هم به عنوان حافظه برنامه و هم به عنوان حافظه داده به کار رود. محدوده آدرس حافظه بیرونی از آدرس 0x6000 0000 hex، برای پورت EMIFB و از آدرس 0x8000 0000 hex برای پورت EMIFA، شروع می‌شود.

ابزار بهبود کد برنامه و سخت افزار

نرم افزار شبیه ساز Code Composer Studio ، شامل ویرایشگر و کامپایلر زبان C/C++، ابزار اشکال زدای برنامه و نرم افزار ارتباط بلادرنگ با پردازنده (DSP/BIOS) برای این سری پردازنده استفاده می‌شود. امولاتور XDS[2] سازگار با سیستم اشکال‌زدای سخت‌افزار پردازنده‌های C6x نیز می‌تواند به بوردهای EVM[3] متصل شود [25].

[1] ***fetch***

2 Extended Development System

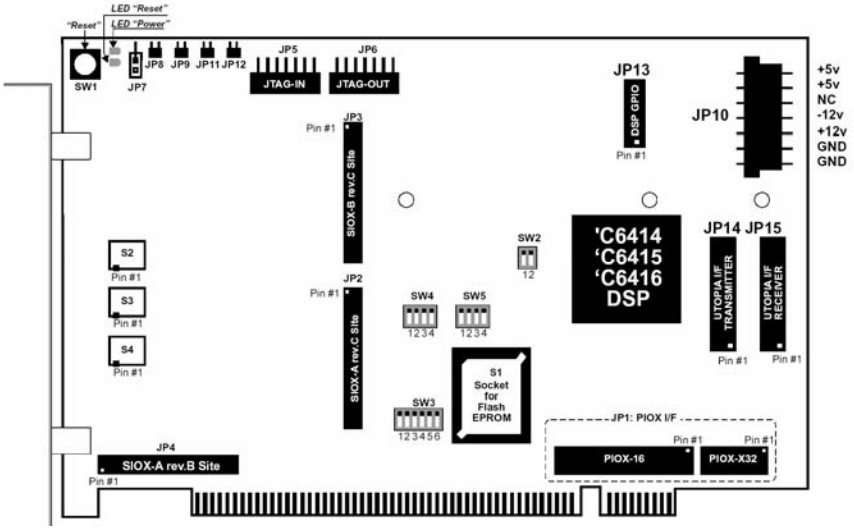
3 Evaluation Module

جدول (۱–۲) نگاشت حافظه در پردازنده***TMS320C6416***

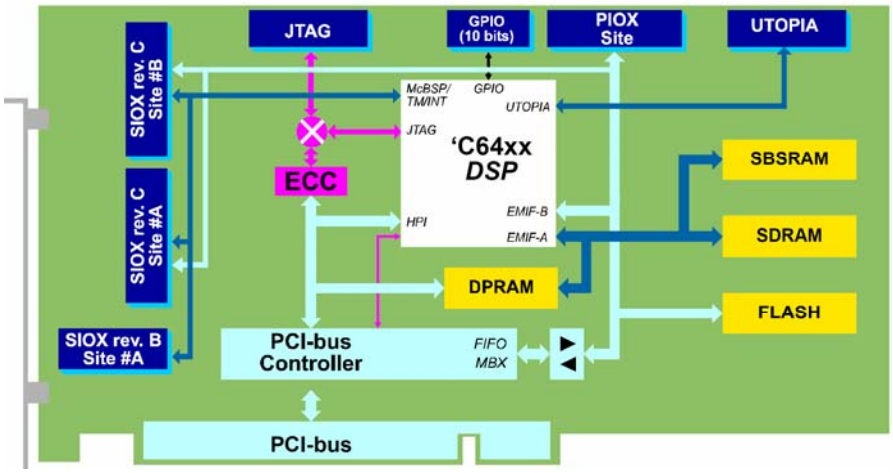
MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	1M	0000 0000 – 000F FFFF
Reserved	23M	0010 0000 – 017F FFFF
External Memory Interface A (EMIFA) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	256K	0184 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	256K	019C 0000 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
McBSP 2 Registers	256K	01A4 0000 – 01A7 FFFF
EMIFB Registers	256K	01A8 0000 – 01AB FFFF
Timer 2 Registers	256K	01AC 0000 – 01AF FFFF
GPIO Registers	256K	01B0 0000 – 01B3 FFFF
UTOPIA Registers (C6415 and C6416 only)†	256K	01B4 0000 – 01B7 FFFF
TCP/VCP Registers (C6416 only)‡	256K	01B8 0000 – 01BB FFFF
Reserved	256K	01BC 0000 – 01BF FFFF
PCI Registers (C6415 and C6416 only)†	256K	01C0 0000 – 01C3 FFFF
Reserved	4M – 256K	01C4 0000 – 01FF FFFF
QDMA Registers	52	0200 0000 – 0200 0033
Reserved	736M – 52	0200 0034 – 2FFF FFFF
McBSP 0 Data	64M	3000 0000 – 33FF FFFF
McBSP 1 Data	64M	3400 0000 – 37FF FFFF
McBSP 2 Data	64M	3800 0000 – 3BFF FFFF
UTOPIA Queues (C6415 and C6416 only)†	64M	3C00 0000 – 3FFF FFFF
Reserved	256M	4000 0000 – 4FFF FFFF
TCP/VCP (C6416 only)‡	256M	5000 0000 – 5FFF FFFF
EMIFB CE0	64M	6000 0000 – 63FF FFFF
EMIFB CE1	64M	6400 0000 – 67FF FFFF
EMIFB CE2	64M	6800 0000 – 6BFF FFFF
EMIFB CE3	64M	6C00 0000 – 6FFF FFFF
Reserved	256M	7000 0000 – 7FFF FFFF
EMIFA CE0	256M	8000 0000 – 8FFF FFFF
EMIFA CE1	256M	9000 0000 – 9FFF FFFF
EMIFA CE2	256M	A000 0000 – AFFF FFFF
EMIFA CE3	256M	B000 0000 – BFFF FFFF
Reserved	1G	C000 0000 – FFFF FFFF

مشخصات بورد TORNADO-P6416 DSP

از بورد TORNADO-P6416 برای پیاده‌سازی کد کننده صحبت G.729 استفاده شده است. معماری کلی بورد TP6416 شامل پردازنده DSP در شکل (۳-۱۰) و شکل (۳-۱۱) آورده شده است[26].



شکل (۳-۱۰) وضعیت فیزیکی اجزاء بورد ***TP6416***



شکل (۳-۱۱) اجزا کلی در بورد ***TP6416***

از مزایای بورد TP6416 می‌توان اشاره کرد به موارد زیر [26]:

- حجم حافظه زیاد روی بورد و درون پردازنده برای کد برنامه و داده
- بسط‌هایی[1] برای ارتباط با بورد ***daughter-card*** و دریافت داده
- ارتباط با پردازنده میزبان از طریق باس ***PCI*** با سرعت ***133 Mbyte/sec***
- امکان ارتباط چند طرفه پردازنده میزبان متصل به باس ***PCI*** و ***DSP*** به کمک حافظه دارای دو پورت ***DPRAM*** و پورت ***HPI***

[1] expansion

- کار در حالت تکی و شامل مدار نگهبان ساعت[1]
- امکان اشکال زدایی سخت افزاری به کمک امولاتور همراه بورد و یا از طریق امولاتورهای شرکت ***TI*** و بهمراه نرم افزار ***Code Composer***
- شامل پردازنده سری ***TMS320C6416*** با فرکانس کلاک ***600 MHZ*** و توانایی ***4800 MIPS*** می‌باشد.

از طریق پورت HPI میزبان می‌تواند به همه فضای DSP از قبیل حافظه درونی و ثباتها دسترسی یابد. یک سری کانکتور بصورت I/O سریال (SIOX)[2] برای دریافت داده‌های صحبت و صدا در کاربردهای بلادرنگ وجود دارد. یک کانکتور نیز بصورت موازی (PIOX) برای کاربردهای پردازش سیگنال با سرعت بیشتر و ارتباط با A/D و D/Aها فراهم شده است. در حالت کار بصورت Stand-alone پردازنده از حافظه EPROM راه اندازی می‌شود و نیازی به قرار گرفتن در باس PCI ندارد و ولتاژ تغذیه را از یک کانکتور جداگانه دریافت می‌کند. برای اشکال زدایی سخت افزاری، پورت JTAG می‌تواند به امولاتورهای XDS شرکت TI ارتباط داشته باشد و یا از چیپ امولاتور همراه بورد (ECC) [3] استفاده کند.

نگاشت حافظه در بورد *TP6416*

بورد TP6416 دارای حافظه‌هائی به شرح زیر است:

- ***1Mbyte*** حافظه از نوع ***DPRAM***[4] برای برنامه و داده در انتقالات بین پردازنده و باس ***PCI***.
- ***4 Mbyte*** حافظه از نوع ***SBSRAM*** [5] برای داده و برنامه.
- ***128 Mbyte*** حافظه از نوع ***SDRAM*** [6] برای داده و برنامه.
- ***1 Mbyte*** حافظه پاک نشدنی ***(EPROM/FLASH)*** برای کدهای راه انداز سیستم.

دسترسی دارد.جدول (۱-۳) به حافظه‌های بیرونی مطابق آدرسهای TP6416 در بورد C6416پردازنده

[1] watchdog timer
[2] ***Serial I/O Expansion***
[3] emulation controller chip
[4] dual-port static RAM
[5] synchronous burst static RAM
[6] synchronous dynamic RAM

جدول (۱–۳) نگاشت حافظه در بورد ***TP6416***

memory area of TMS320C64xx DSP	DSP address range (in bytes)	valid data bits	value at DSP RESET	Access mode	wait states	EMIF CE area and mode
On-board (DSP off-chip) memories						
SBSRAM	80000000H ..800FFFFFH (128Kx64) 80000000H ..801FFFFFH (256Kx64) 80000000H ..803FFFFFH (512Kx64)	D0..D31	-	r/w	6ws	EMIF-A CE-0 SBSRAM mode
DPRAM	90000000H ..9007FFFFH (128Kx32) 90000000H ..900FFFFFH (256Kx32)	D0..D31	-	r/w	6ws	EMIF-A CE-1 SBSRAM mode
DPRAM: *DPRAM_HM_RQ* register (PCI-to-DSP interrupt request via DPRAM) (256Kx32 DPRAM only)	000FFFFCH (256Kx32)	D0..D31	-	r/w	6ws	
DPRAM: *DPRAM_MH_RQ* register (DSP-to-DSP interrupt request via DPRAM) (256Kx32 DPRAM only)	000FFFF8H (256Kx32)	D0..D31	-	r/w	6ws	
SDRAM	A0000000H ..A1FFFFFFH (4Mx64) A0000000H ..A7FFFFFFH (16Mx64)	D0..D31	-	r/w	1ws	EMIF-A CE-2 SDRAM mode
FLASH/EPROM	64000000H ..641FFFFFH (1Mx8)	D0..D7 @W16	-	r/w @A8 @A16 (with write disable)	FWS	EMIF-B CE-1 ASYNC ROM mode
DSP on-chip memory and peripheral registers						
DSP on-chip RAM	00000000H ..000FFFFFH	D0..D31	-	r/w	-	-
DSP on-chip peripheral registers	01800000H ..02000033H	D0..D31	-	r/w	-	-

راه اندازی شدن[1] بورد *TP6416*

بورد TP6416 این امکان را فراهم می‌کند که پردازنده C6416 را در پیکربندیهای مختلف، برای کاربردهای متفاوت راه اندازی کرد. وضعیت راه اندازی DSP از طریق یکسری سوئیچها و جامپرهای روی بورد و نیز از طریق برنامه نرم افزاری روی کامپیوتر میزبان تعیین می‌گردد. وقتی که پردازنده در حالت Stand alone کار می‌کند، یعنی در شکاف PCI قرار نگرفته باشد یا در شکاف PCI باشد ولی گزینه M_SA_MODE در ثبات HIF_CONTROL_RG برابر با ۱ باشد، از طریق سوئیچ ۲ روی بورد تنظیمات لازم برای راه اندازی مشخص می‌شود(جدول (۱–٤)). وقتی که پردازنده در وضعیت ارتباط با کامپیوتر میزبان باشد، یعنی در شکاف PCI قرار گرفته باشد و گزینه M_SA_MODE در ثبات HIF_CONTROL_RG برابر صفر باشد، راه‌اندازی پروسسور DSP از طریق کامپیوتر میزبان تعیین می‌شود.

جدول (۱–٤) چگونگی راه اندازی پروسسور ***DSP*** در بورد ***TP6416***

[1] ***Boot***

'C64xx DSP operation mode	'C64xx DSP Bootmode	Description	HIF_DSP_BMODE_RG register bits		on-board SW2 switch	
			DSP_BMODE-0	DSP_BMODE-1	SW2-1	SW2-2
Host mode	NO BOOT	No boot process.	0	0	x	x
	HPI BOOT	Boot from HPI port.	1	0	x	x
	FLASH BOOT	Boot from on-board 8-bit FLASH/EPROM memory.	0	1	x	x
Stand-alone mode	NO BOOT	No boot process.	x	x	ON	ON
	HPI BOOT	Boot from HPI port.	x	x	OFF	ON
	FLASH BOOT	Boot from on-board 8-bit FLASH/EPROM memory.	x	x	ON	OFF

در وضعیت بدون راه اندازی(NO BOOT) پردازنده بعد از بازنشانی، از آدرس 0000 به اجرای کدها می‌پردازد. این وضعیت اغلب در هنگام اشکال‌زدایی سخت افزاری، از طریق امولاتور JTAG استفاده می‌شود.

توسط یک برنامه با مقادیر درست پُر شوند. EMIF بایستی ثباتهای کنترلی TP6416برای عملکرد صحیح اجزاء روی بورد مراجعه کرد. البته یکسری C64x می‌توان به برگه مشخصات پردازنده EMIFبرای اطلاعات کامل در مورد ثباتهای کنترلی و مثالهای دیگر توسط شرکت سازنده ارائه شده است.EMIFبرنامه نمونه برای تنظیم ثباتهای کنترلی

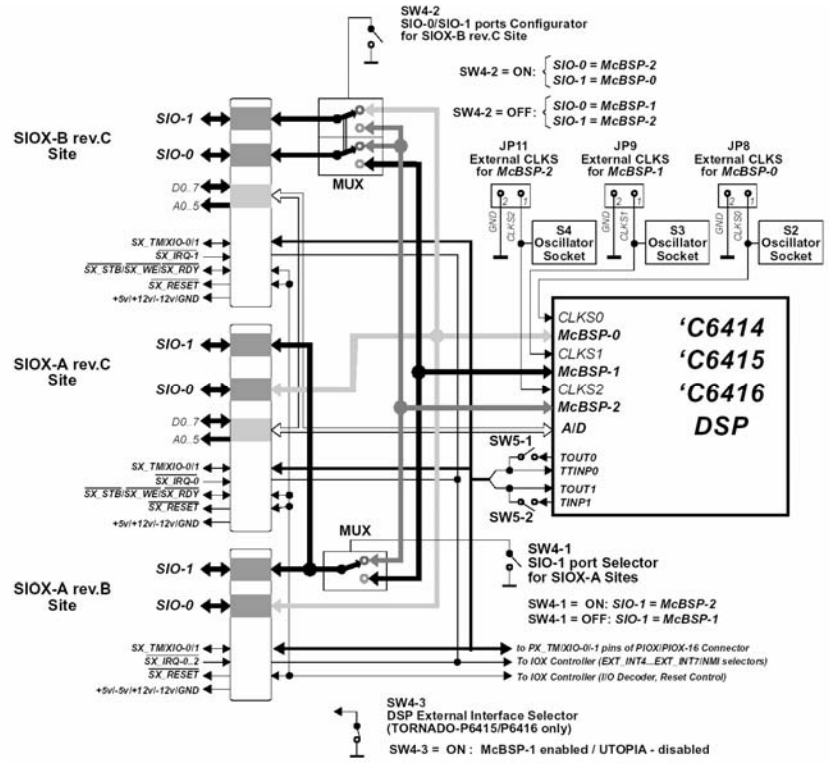
سایت ارتباط *SIOX*

بورد TP6416 دو سایت (محل) A وB، برای ارتباط سریال به منظور انتقال داده دارد. این سایتها شامل پورتهای سریال[1] پردازنده DSP، تایمرهای پردازنده، پینهای وقفه خارجی، و پینهایی برای تغذیه بورد daughter card وجود دارد. علاوه بر دو سایت فوق، سایت C SIOX نیز شامل دو پورت سریال پردازنده، تایمر، وقفه‌های خارجی و داده‌های موازی[2] پردازنده می‌باشد. از دیگر سایتهای روی بورد می‌توان به سایت PIOX[3]، بصورت ۱۶ و ۳۲ بیتی برای دسترسی به I/Oها بطور موازی، اشاره کرد. دیاگرام مربوط به سایتهای SIOX در شکل (۳-۱۲) آمده است.

[1] ***MCBSP***

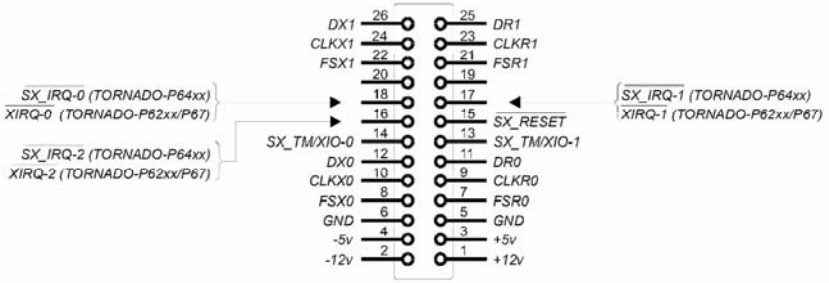
[2] DSP parallel data

[3] Parallel I/O Expansion



شکل (۳-۱۲) دیاگرام ارتباطی سایت ***SIOX*** به پردازنده ***DSP*** در بورد ***TP6416***

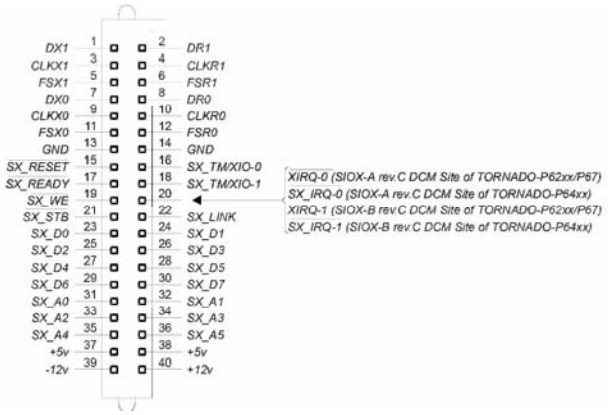
سایت SIOX ارتباط با پورتهای 0، 1 و 2 را فراهم می‌کند. بدین ترتیب که پورت سریال ۱ می‌تواند بصورت واسط UTOPIA[1] نیز عمل کند. بخش ۱ در سایت مذکور (SIO-1) می‌تواند به پورت سریال ۱ یا ۲ (از طریق سوئیچ SW4-1 تعیین می‌شود) متصل شود. وضعیت پینهای سایت A و B در شکل (۳-۱۳) آورده شده است



شکل (۳-۱۳) وضعیت پینها در سایت ***SIOX A/B***

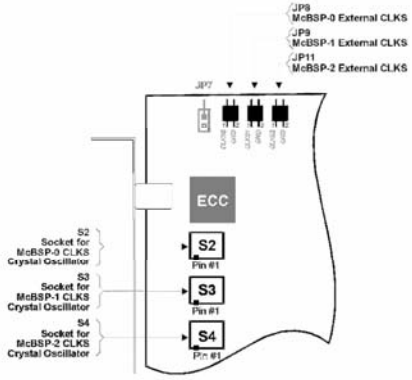
بخش C سایت SIOX نیز شامل کانکتور دارای۵۰ پین برای ارتباطات سریال می‌باشد. وضعیت پینهای این سایت نیز در شکل (۳-۱٤) آورده شده است.

[1] ***Universal Test & Operations Physical Interface for ATM***



شکل (۳-۱۴) وضعیت پینها در بخش ***C*** سایت ***SIOX***

در بورد TP6416 برای پورت سریال می‌توان پالس کلاک را بصورت خارجی نیز اعمال کرد. این سیگنالها به عنوان CLKS_0 و CLKS_1 و CLKS_2 نامیده می‌شوند. برای اعمال کردن این سیگنالها از یک منبع خارجی برای پورت سریال صفر از جامپر 8 (JP8)، برای پورت سریال یک از JP9 و برای پورت سریال ۲ از JP10 استفاده می‌شود. هم چنین می‌توان از یک نوسانساز کریستالی در مکانهای (Socket) با نام S2 و S3 و S4 برای تولید سیگنال کلاک استفاده کرد. وضعیت پینهای فوق در شکل (۳-۱۵) آمده است.



شکل (۳-۱۵) طریقه اعمال کلاک خارجی به پورت سریال پردازنده ***DSP***

برای استفاده از کلاک خارجی در پورت سریال به هنگام برنامه نویسی باید مقادیر لازم در ثباتهای کنترلی پورت سریال از قبیل PCR[1]، RCR[2]، XCR[3] و ... قرار بگیرد.

امولاتور بورد *TP6416*

بورد TP6416 شامل ابزار امولاتور برای اشکال‌زدایی سخت‌افزاری پروسسور DSP و برنامه‌های نرم‌افزاری روی پردازنده می‌باشد. برای امولاتور می‌توان از ابزار زیر استفاده کرد:

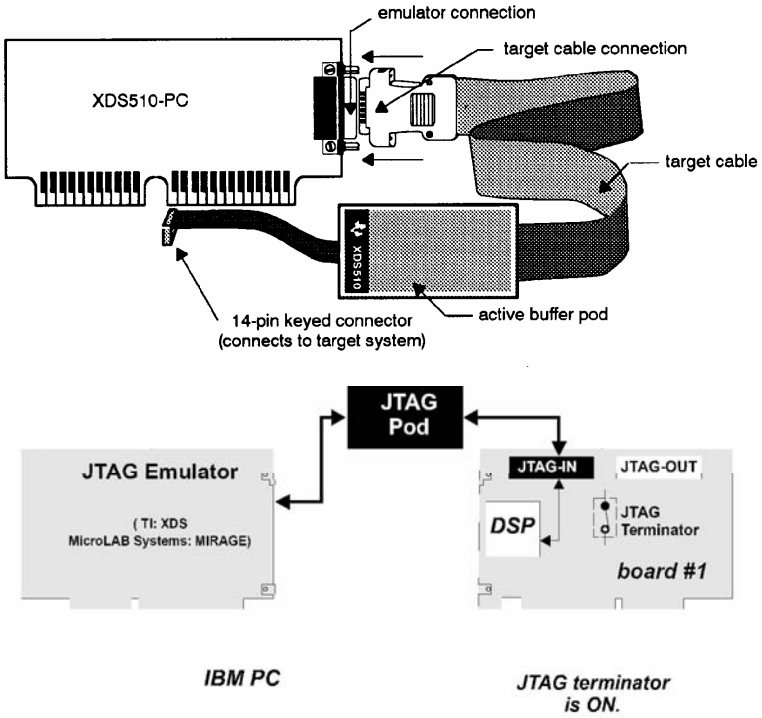
[1] ***Pin Control Register***

[2] ***Receive Control Register***

[3] ***Transmit Control Register***

- استفاده از امولاتورهای ***XSD*** شرکت ***TI*** (***XDS510*** و ***XDS560*** و ...) که به کانکتور ***JTAG-IN*** روی بورد اتصال می‌شوند.
- استفاده از چیپ امولاتور روی بورد (***ECC***)
- استفاده از نرم افزار ***Code Composer*** که همراه ابزار امولاتور اجرا شده باشد.

مسیر JTAG برای اتصال به امولاتور خارجی در شکل (۳-۱۶) آورده شده است. کانکتور JTAG-IN بایستی به امولاتور JTAG خارجی متصل شود. پورت JTAG-OUT نیز برای وسیله JTAG دیگری شامل مسیر مشابه JTAG بورد TP6416 استفاده می‌شود. زمانی که امولاتور خارجی از طریق پورت JTAG-IN به بورد TP6416 متصل شده باشد بایستی ترمینال JTAG، توسط سوئیچ SW4-4، ON شده باشد.



شکل (۳-۱۶) اتصال امولاتور ***JTAG*** خارجی به بورد ***TP6416***

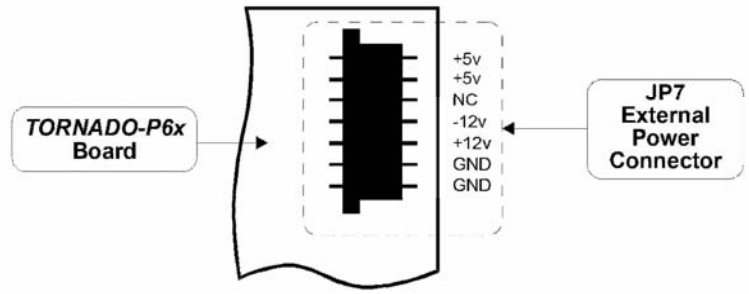
راه اندازی بورد *TP6416* به هنگام اتصال به امولاتور خارجی

وقتی که بورد TP6416 می‌خواهد در شکاف کامپیوتر قرار داشته باشد، ابتدا کامپیوتر را خاموش کرده و بورد TP6416 را در شکاف PCI قرار می‌دهیم. قبل از اینکار، بایستی سوئیچ‌ها و جامپرهای روی بورد برای کاربرد مورد نظر به درستی تنظیم شده و کانکتورهای روی بورد از قبیل سایتهای SIOX و دیگر پورتها به اجزا خارجی متصل شده باشند. از سوئیچهای مهم روی بورد می‌توان به موارد زیر اشاره کرد:

- سوئیچ ***SW2*** مربوط به وضعیت ***Bootmode***
- سوئیچ ***SW4-3*** برای تعیین اجزاء خارجی(***UTOPIA*** یا ***McBSP-1***)
- سوئیچ ***SW4-1*** و ***SW4-2*** برای تنظیم درست پورت ***SIOX-A***

- سوئیچ ***SW4-4*** برای تنظیم درست ترمینال ***JTAG***

اگر بورد در شکاف کامپیوتر قرار می‌گیرد، سیستم عامل کامپیوتر بایستی Windows98 باشد و اگر در مود Stand-alone کار می‌کند، تغذیه خارجی همانند شکل (۳-۱۷) فراهم می‌شود.



شکل (۳-۱۷) وضعیت پینها در کانکتور مربوط به تغذیه خارجی

هنگامی که بورد TP6416 در شکاف PCI قرار دارد هم می‌تواند در مود Host و هم در مود Stand-alone کار کند. این کار از طریق یکسری برنامه نرم افزاری که خود سازنده بورد فراهم کرده است و قابل اجرا در محیط Windows98 هستند، تعیین می‌شود.

پس از روشن کردن کامپیوتری که بورد TP6416 روی آن قرار داده شده است، باید قبل از اجرای نرم‌افزار Code Composer مربوط به ابزار امولاتور، پروسسور DSP از حالت بازنشانی[1] خارج و در مود اجرا[2] قرار بگیرد. برای قرار دادن پروسسور DSP در مودهای مختلف از برنامه TP6CC.EXE، فراهم شده توسط سازنده بورد استفاده می‌کنیم. در برنامه فوق گزینه‌های زیر وجود دارد:

- $TP6CC.EXE \quad -r$: برای بازنشانی پروسسور
- $TP6CC.EXE \quad -hcsa1$: برای قرار دادن پردازنده در مودStand-alone
- $TP6CC.EXE \quad -ex$: برای فعال کردن پورت و اتصال به امولاتور خارجی JTAG-IN
- $TP6CC.EXE \quad -hcr0$ یا $TP6CC.EXE \quad -cr0$ برای قرار دادن پردازنده در حالت اجرا
- $TP6CC.EXE \quad -?$ برای آگاهی از سایر گزینه‌ها

از دیگر برنامه‌های فراهم شده برای کار با بورد TP6416 برنامه TP6COFF.EXE می‌باشد که برای بارگذاری کردن فایلهای برنامه حاصل از خروجی ابزار اسمبلر (فایل OUT.) به درون حافظه روی بورد و حافظه درون چیپ DSP بکار می‌رود. البته هنگامی که از نرم افزارCode Composer بهمراه امولاتورXDS510 استفاده می‌شود [27] ، به کمک گزینه Load از منوی File در نرم افزار، می‌توان فایل OUT. را به درون حافظه بار گذاری کرد[26].

[1] ***Reset***

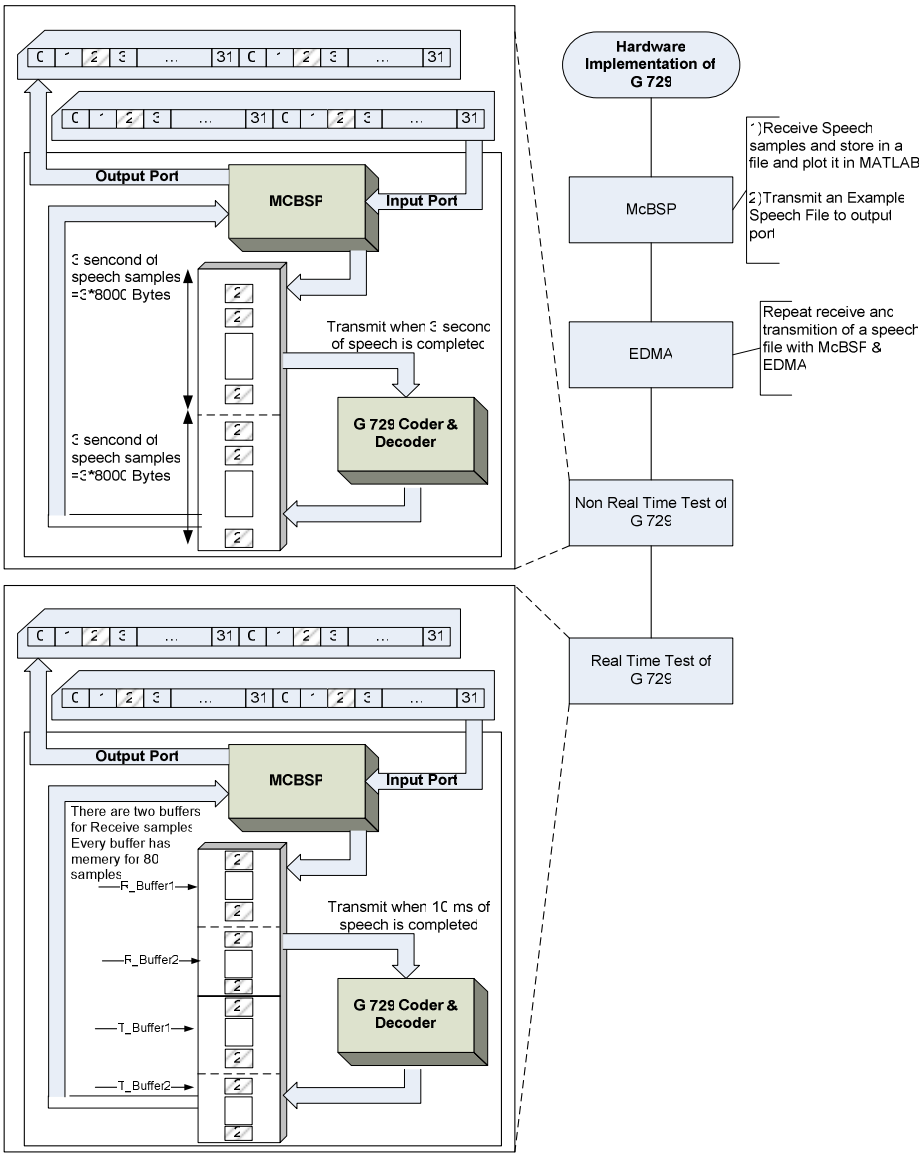
[2] ***Run***

بخش دوم

پیاده‌سازی ارتباط چندکاناله بر روی بورد ***DSP***

مقدمه

۲شکل (۳-۱) مراحل پیاده‌سازی بر روی بورد را نشان می‌دهد. طبق شکل فوق ابتدا راه‌اندازی McBSP و EDMA برای دریافت و ارسال داده‌ها و سپس دو تست غیر بلادرنگ و بلادرنگ انجام شده است. در ادامه این بخش مراحل فوق به طور کامل توضیح داده می‌شود.



شکل (۳-۱) مراحل پیاده‌سازی بر روی بورد

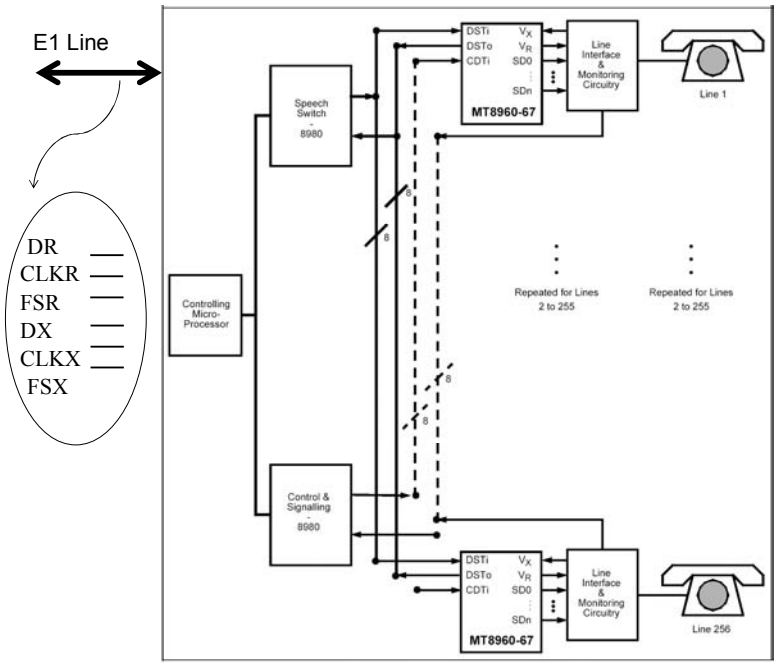
راه اندازی بورد DSP و امولاتور

برای انجام پردازشهای لازم روی رسانه ، از یک بورد DSP قابل قرار گرفتن درشکاف PCI به نام TornadoP6416 استفاده شده است[26]. این بورد شامل پردازنده TMS320C6416 و با فرکانس کلاک 600MHz کار می‌کند. برای اشکال‌زدایی و بارگذاری برنامه‌ها درون پردازنده از امولاتور XDS510 قابل اتصال در شکاف ISA کامپیوتر استفاده می‌شود[27].

پس از فراهم کردن تغذیه بورد TP6416 (از طریق قرارگرفتن درشکاف PCI) و ارتباط دادن پورت JTAG[1] به امولاتور، کامپیوتر را روشن می‌کنیم. پردازنده DSP بایستی ابتدا در مود تکی[2] و سپس در مود اجرا[3] قرار بگیرد. برای اینکار از نرم‌افزارهای داده شده توسط سازنده بورد TP6416 استفاده می‌شود.

در تنظیمات نرم‌افزار Code Composer، پیکربندی امولاتور XDS510 را انتخاب می‌کنیم. نرم‌افزار Code Composer بهمراه امولاتور، قابلیت اجرای خط به خط کدهای برنامه، مشاهده محتویات حافظه بورد DSP و ثباتها را فراهم می‌کند.

در این پروژه نمونه‌های صحبت از یک ترانک[4] دیجیتال شامل کارت لاین و دیگر مدارات لازم برای درست کردن خط E1 استفاده می‌شود. در این ترانک از IC به شماره MT8960 از شرکت zarlink برای نمونه‌برداری صدا با فرکانس 8KHz و تبدیل نمونه‌ها به فرمت فشرده‌سازی A-law استفاده می‌شود. همچنین از یک پردازنده و بافرهای موجود برای مالتی پلکس زمانی و تولید رشته داده E1 استفاده می‌شود. دیاگرام کلی مدارات در شکل (۳-۲) آورده شده است.



شکل (۳-۲) ساختار ترانک دیجیتال

[1] ***Joint Test Action Group (IEEE 1149.1 Standard)***

[2] ***Stand- alone***

[3] ***Run***

[4] ***Trunk***

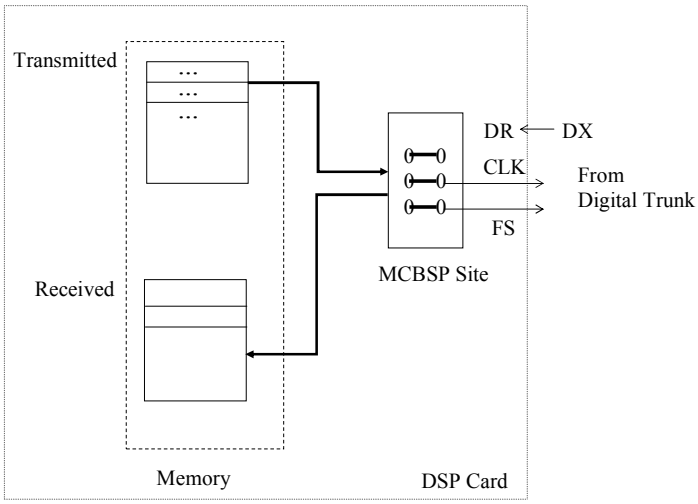
سه رشته سیم برای دریافت و سه رشته سیم برای ارسال داده‌ها استفاده می‌شوند. در هر سه رشته سیم، پالس کلاک، پالس سنکرون کننده و رشته داده‌ها وجود دارد. در رشته‌های داده‌های E1، ۳۲ کانال داده هشت بیتی وجود دارد. از آنجا که سیگنال صحبت با فرکانس۸ کیلوهرتز نمونه‌برداری شده، لذا فرکانس پالس کلاک ترانک 32×8×8000=2.048MHz است. پالس سنکرون کننده نیز چون ابتدای هر فرم را نشان می‌دهد فرکانس 8KHz دارد.

راه‌اندازی و تنظیم پورت سریال

پردازنده C6416 واقع بر بورد TP6416 دارای سه پورت سریال چند کاناله[1] می‌باشد. از آنجا که هر پورت می‌تواند همزمان هم در مود ارسال و هم در مود دریافت کار کند، پس در این پروژه تنها از پورت شماره صفر استفاده شده است. برای راه-اندازی پورت سریال و نیز معتبر بودن پالس‌های کلاک و سنکرون کننده، به ترتیب زیر عمل می‌کنیم:

رشته سیم‌های مربوط به پالس کلاک و پالس سنکرون کننده درگیرنده را به سایت SIOX واقع بر بورد TP16 و در قسمت مربوط به پینهای پورت شماره صفر متصل می‌کنیم[26]. پین گیرنده (DR0) و پین فرستنده (DX0) را با یک سیم بهم متصل می‌کنیم. سپس یکسری داده واقع در یک بلوک، به نام Transmitted را از طریق پورت سریال و پین (DR0) با فرکانس 2.048MHz و پالس سنکرون 8KHz فرستاده و در پین DX0 این داده‌ها دریافت و در بلوک Recived قرار می‌گیرند.

روندنمای[2] مربوط به این تست در شکل (۳-۳) آورده شده است.



شکل (۳-۳) جابجایی داده در حافظه توسط پورت سریال

برای درست کردن بلوکهای received, transmited به زبان اسمبلی مطابق شکل (۳-٤) عمل می‌کنیم.

```
.global _transmited,_received        ; Define variable as global
        .data
_transmited:                          ; Define lable
        .word 12345678h   ; Define word as Data
        .word 98765432h
```

[1] ***MCBSP***

[2] ***flowchart***

```
                .word 11111111h
                .word 22222222h
                .word 33333333h
                .word 1abcdef2h
                .word 77777777h
                .word 66666666h
                .word 55555555h
                .word 44444444h
                .word 54321543h
                .word 12341234h

_received:
                .space 200              ; reserve space on memory
```

شکل (۳-۴) درست کردن بلوک داده در حافظه

چون در روتین وقفه، پورت سریال فعال می‌شود، لذا نیاز به تنظیم جدول وقفه‌ها و تنظیم روتین‌های سرویس‌دهی وقفه می‌باشد. برای درست کردن جدول وقفه‌ها، یک فایل اسمبلی مطابق شکل (۳-۵) درست می‌کنیم. در روتین‌های مربوط به سرویس‌دهی به پورت سریال، یک پرش[1] به تابع مربوطه صورت می‌گیرد.

```
; Interrupt vector table

        .global _c_int00, _c_ int4, _c_ int5
        .sect "vec"

reset:
        mvkl  _c_int00,b1
        mvkh  _c_int00,b1
        b     b1
        nop
        nop
        nop
        nop
        nop

nmi:
        b     nmi
        nop
        nop
        nop
        nop
        nop
        nop
        nop

rsrv2:
```

[1] ***Jump***

```
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop

rsrv3:
        b rsrv3
        nop
        nop
        nop
        nop
        nop
        nop
        nop

int4:
        mvkl  _c_int4,b1
        mvkh  _c_int4,b1
        b     b1
        nop
        nop
        nop
        nop
        nop

int5:
        mvkl  _c_ int5,b1
        mvkh  _c_ int5,b1
        b     b1
        nop
        nop
        nop
        nop
        nop
```

شکل (۳-۵) تنظیم جدول وقفه‌ها

در زبان اسمبلی توسط عبارت global متغیرها را بصورت عمومی تعریف می‌کنیم و برای متغیرهایی که در فایل اسمبلی استفاده می‌شوند از "_" قبل از نام آنها استفاده می‌کنیم.

برای آنکه بتوان برنامه‌نویسی و تنظیمات منوط به پورت سریال، وقفه‌ها و دسترسی به ثباتها را در زبان C انجام داد، از فایلهای سرباره regs.h, mcbsp.h, inter.h استفاده می‌کنیم. برای تنظیم پورت سریال تابعی به نام init_mcbsp( ) تعریف می‌کنیم. در بدنه این تابع، تنظیمات مربوط به پورت سریال از قبیل پلاریته پالس کلاک و سنکرون کننده، سایز داده‌ها،

چگونگی تولید وقفه توسط پورت سریال و ... انجام می‌شود[24]. شکل (۳-۶) قسمتی از بدنه این تابع را نشان می‌دهد. در این تابع به ثباتهای کنترلی پورت سریال دسترسی داریم و مقادیر مناسب در آنها بارگذاری می‌شود.

پس از آن تابع set_interrupt( ) را برای تنظیم وقفه‌ها تعریف می‌کنیم. در بدنه این تابع فعال کردن وقفه‌ها، نگاشت وقفه‌های پورت سریال به وقفه‌های دلخواه و دیگر کارهای مربوطه انجام می‌شود (شکل (۳-۷)). در تنظیمات پورت سریال تعیین شده است که پورت سریال با دیدن پالس سنکرون کننده فریم وقفه بدهد. این وقفه، به وقفه شماره ٦ پردازنده نگاشت شده است. در بدنه سرویس روتین وقفه ٦، گیرنده و فرستنده هر دو فعال می‌شوند و نحوه تولید وقفه توسط پورت سریال تغییر می‌کند. پس از فعال شدن گیرنده و فرستنده در صورت خالی شدن بافر فرستنده (ثبات DXR) و یا پر شدن بافر گیرنده (ثبات DRR) وقفه تولید می‌شود. بدنه مربوط به روتینهای سرویس‌دهی وقفه در شکل (۳-۸) آمده است.

```
void init_mcbsp(void)
{
 /* PCR setup*/
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 1, CLKXP, 1);   //rising adge
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 1, FSXP, 1);    //active low
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 0, CLKXM, 1);
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 0, FSXM, 1);

 /* XCR setup */
 LOAD_FIELD (MCBSP_XCR_ADDR(0), SINGLE_PHASE, XPHASE, 1);
 LOAD_FIELD (MCBSP_XCR_ADDR(0), WORD_LENGTH_8, XWDLEN1,
XWDLEN1_SZ);
 LOAD_FIELD (MCBSP_XCR_ADDR(0), 0x1F, XFRLEN1, XFRLEN1_SZ);
 LOAD_FIELD (MCBSP_XCR_ADDR(0), 0, XDATDLY, XDATDLY_SZ);

 /* PCR setup*/
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 1, CLKRP, 1);//rising adge
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 1, FSRP, 1);//active low
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 0, CLKRM, 1);
 LOAD_FIELD (MCBSP_PCR_ADDR(0), 0, FSRM, 1);
 /* RCR setup */
 LOAD_FIELD (MCBSP_RCR_ADDR(0), 0, RPHASE, 1);
 LOAD_FIELD (MCBSP_RCR_ADDR(0), WORD_LENGTH_8, RWDLEN1,
RWDLEN1_SZ);
 LOAD_FIELD (MCBSP_RCR_ADDR(0), 0x1F, RFRLEN1, RFRLEN1_SZ);
 LOAD_FIELD (MCBSP_RCR_ADDR(0), 0, RDATDLY, RDATDLY_SZ);
 /* SPCR setup */
 LOAD_FIELD (MCBSP_SPCR_ADDR(0),0x02, XINTM, XINTM_SZ);
 LOAD_FIELD (MCBSP_SPCR_ADDR(0),0x02, RINTM, RINTM_SZ);
   return;
}
```

شکل (۳-۶) تنظیم ثباتهای کنترلی پورت سریال

```
void set_interrupts(void)
{
 intr_init();
```

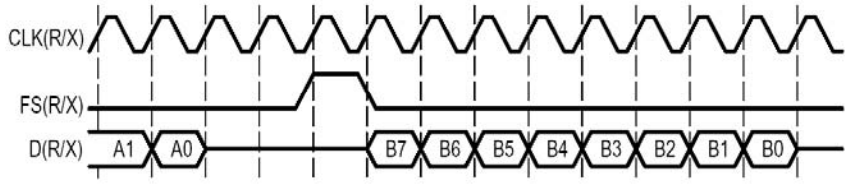
```
 INTR_MAP_RESET();

 INTR_ENABLE(CPU_INT_NMI);   // Enable NMIE
 INTR_GLOBAL_ENABLE();       // Set GIE in CSR

 intr_map(CPU_INT5,ISN_RINT0);
 intr_map(CPU_INT6,ISN_RINT0);
 intr_map(CPU_INT7,ISN_XINT0);
 intr_map(CPU_INT4,ISN_XINT0);

 INTR_ENABLE(4);
 INTR_ENABLE(5);

 return;
}
```

شکل (۳-۷) تنظیم کردن وقفه‌ها

```
...
 *temp_Received=MCBSP_READ(0);      // receiver interrupt routine
 temp_Received++;
...
  MCBSP_WRITE(0,*temp_Transmitted);  // transmitter interrupt routine
 temp_Transmitted++;
...
```

شکل (۳-۸) برنامه روتین وقفه گیرنده و فرستنده

در پورت سریال بایستی مشخص شود که پالس کلاک و پالس سنکرون کننده فریم از خارج اعمال می‌شوند و اینکه پالس کلاک با لبه بالا رونده فعال می‌باشد(پهنای پالس سنکرون کننده فریم گرفته شده از ترانک دیجیتال باریک می‌باشد و تنها در لبه بالا رونده قابل مشاهده است). در شکل (۳-۹) وضعیت داده و پالسهای کلاک و سنکرون کننده نشان داده شده است.



شکل (۳-۹) وضعیت داده و پالس‌های کلاک و سنکرون کننده

جدول وقفه‌ها باید در آدرس 000hex قرار بگیرد. در فایل دستورات لینکر(cmd)که به پروژه اضافه می‌شود، طرز قرار گرفتن جدول وقفه‌ها مشخص می‌گردد. شکل (۳-۱۰) نمونه‌ای از فایل cmd برای نگاشت صحیح جدول وقفه‌ها و دیگر کدها در حافظه را نشان می‌دهد.

با تنظیم برنامه به روش فوق، داده‌های بلوک transmited در بلوک received به درستی کپی می‌شوند.

```
MEMORY
{
   VECTORS  : o = 0h          , l = 200h                  // interrupts table
   onchipDM : o = 00000300h   , l = 6ffffh
```

```
    onchipPM : o = 0007ffffh  , l = 3ffffh
    SBSRAM   : o = 80000000h  , l = 3fffffh
    DPRAM    : o = 90000000h  , l = 0fffffh
    SDRAM    : o = 0A0000000h  , l = 7fffffffh
    EEPROM   : o = 64000000h  , l = 1fffffh
}
SECTIONS
{
    vec    :> VECTORS                          // mapping interrupts into tables
   .data   :> onchipDM
  .bss     :> onchipDM
   .text   :> onchipPM
   .cinit  :> onchipDM
   .stack  :> onchipDM
   .far    :> onchipDM
   .cio    :> onchipDM
   .const  :> onchipDM
}
```

شکل (۳-۱۰) فایل دستورات لینکر برای نگاشت کدها در حافظه

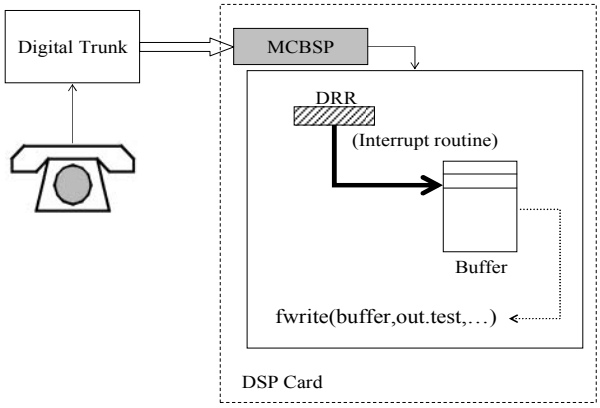
برای آگاهی از تنظیم درست جدول وقفه می‌توان در برنامه سرویس‌دهی وقفه به پورت سریال، نقطه انفصال قرار داد و ملاحظه کرد که اجرای برنامه به آن قسمت وارد می‌شود. دقت کنید که زمانی که بورد در حال اجرای برنامه می‌باشد و با وسائل جانبی از قبیل پورت سریال و واحد DMA کار می‌کند گذاشتن نقاط انفصال در روتین وقفه و کدهای برنامه باعث می‌شود زمان حقیقی را از دست بدهیم و در اجرای برنامه دچار مشکل شویم، لذا پس از اطمینان از درست بودن کدها بایستی نقاط انفصال برداشته شوند.

پس از تنظیمات لازم پورت سریال و وقفه‌ها در تابع ( )init_mcbsp و تابع ( )set_interrupt، با رویت اولین پالس سنکرون کننده، پورت سریال وقفه شماره ۵ می‌دهد. وقفه پورت سریال بعنوان گیرنده فعال می‌شود. پس از آن با پر شدن بافر گیرنده (ثبات DRR) وقفه شماره ٤ صادر می‌شود. در این وقفه داده از بافر گیرنده در بافر receive_buffer ریخته می‌شود.

دریافت صحبت از ترانک دیجیتال و ذخیره در فایل

در این قسمت برنامه‌ای نوشته می‌شود که نمونه‌های صحبت را از پورت سریال دریافت و پس از ذخیره در یک فایل، توسط کامپیوتر پخش می‌کند تا از کیفیت ضبط شدن صدا اطمینان حاصل شود.

روند نمای چگونگی ارتباط اجزاء و عملکرد پردازنده در شکل (۳-۱۱) آورده شده است. طبق شکل حدود ٤ ثانیه از صحبت، شامل 32000=8000×4. نمونه ۸ بیتی در یک بافر ذخیره و سپس در فایل out.test نوشته می‌شود.



شکل (۳-۱۱) ضبط چند ثانیه صحبت در فایل

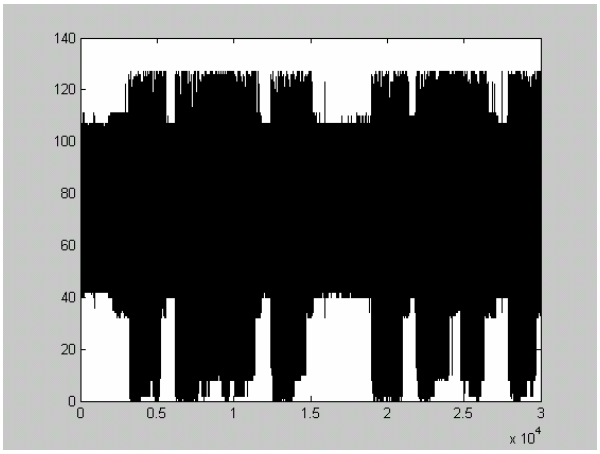
در این برنامه، از پورت سریال شماره صفر فقط بعنوان گیرنده استفاده می‌شود پس از دریافت 32000 نمونه صحبت، توسط دستور fwrite( ) نمونه‌های ذخیره شده در فایل نوشته می‌شود.

در این برنامه نمونه‌های صحبت از کانال شماره ۲ خط E1 دریافت می‌شود. برنامه‌های مربوط به تنظیم کردن پورت سریال برای مود چند کاناله در شکل (۳-۱۲) آورده شده است.

```
...
LOAD_FIELD (MCBSP_RCR_ADDR(0), 0x1F, RFRLEN1, RFRLEN1_SZ);
...
/* MCR setup - Receiver */
LOAD_FIELD (MCBSP_MCR_ADDR(0),1, RMCM, 1);
LOAD_FIELD (MCBSP_MCR_ADDR(0), 0, RPABLK, RPABLK_SZ); /* ch0-15 */
LOAD_FIELD (MCBSP_MCR_ADDR(0), 0, RPBBLK, RPBBLK_SZ); /* ch16-31 */

/*Enable channel  #2*/
SET_REG(MCBSP0_RCER, 0x4);
...
```

شکل (۳-۱۲) تنظیم کانالها در پورت سریال

پس از ذخیره ٤ ثانیه از صحبت در فایل، می‌توان آنرا با نرم‌افزار MATLAB آنالیز کرد. در شکل (۳-۱۳) پلات مربوط به نمونه‌های ذخیره شده، رسم شده است. همانطور که مشاهده می‌شود داده‌ها ما بین اعداد ۰ و ۱۲۸+ قرار گرفته‌اند.



شکل (۳-۱۳) پلات ٤ ثانیه صحبت ضبط شده

فشرده‌سازی و نافشرده‌سازی *A-law* به کمک نرم‌افزار در پردازنده *C6416*

پس از دریافت نمونه‌های یک فریم ۱۰ میلی ثانیه‌ای، آنها را بایستی به نمونه‌های ١٦ بیتی خطی تبدیل و سپس به کد کننده G.729 اعمال کرد. در هنگام ارسال نیز خروجی واکد کننده بایستی به ۸ بیتی A-law تبدیل و سپس به ترانک دیجیتال فرستاده شود. در توصیه نامه G.711 فرمت فشرده‌سازی A-law عنوان گردیده است[38].

کدهای برنامه این توابع درشکل (۳-١٤) آورده شده است. تابع ( )int2alaw یک ورودی linear بصورت ۱۳ بیتی که به ١٦ بیت گسترش علامت داده شده[1] است را دریافت و یک خروجی فشرده شده ۸ بیتی برمی‌گرداند. این پیاده‌سازی برای پردازنده C64x با دستورالعمل‌های نقطه ثابت تنظیم شده است.

تابع ( )alaw2int نیز بر عکس تابع فوق، یک ورودی ۸ بیتی A-law را گرفته و خروجی ۱۳ بیتی خطی شده و گسترش علامت داده شده به ١٦ بیت را برمی‌گرداند.

```
unsigned short alaw2int(unsigned char log){
        unsigned char sign, segment;
        unsigned short temp, quant;
        temp=log^0xD5;
        sign=(temp&0x80)>>7;
        segment=(temp&0x70)>>4;
        quant=temp&0x0F;
        quant<<=1;
        if(!segment)
            quant+=1;
        else{
            quant+=33;
            quant<<=segment-1;
        }
        if(sign)
          return (-quant);
        else
          return quant;
}
```

```
unsigned char int2alaw(short linear){
        char segment;
        unsigned char i, sign,quant;
        unsigned short output, absol, temp;
        temp=absol=abs(linear);
        sign=(linear >= 0) ? 1:0;
        for(i=0;i<16;i++){
            output=temp&0x8000;
            if(output)break;
            temp<<=1;
        }
```

[1] ***Sign–intended***

```
        segment=11-i;
        if(segment<=0){
            segment=0;
           quant=(absol>>1)&0x0F;
        }
        else
           quant=(absol>>segment)&0x0F;
        segment<<=4;
        output=segment+quant;
        if(absol>4095) output=0x7F;
        if(sign)
          return output^=0xD5;
        else
          return output^=0x55;
}
```

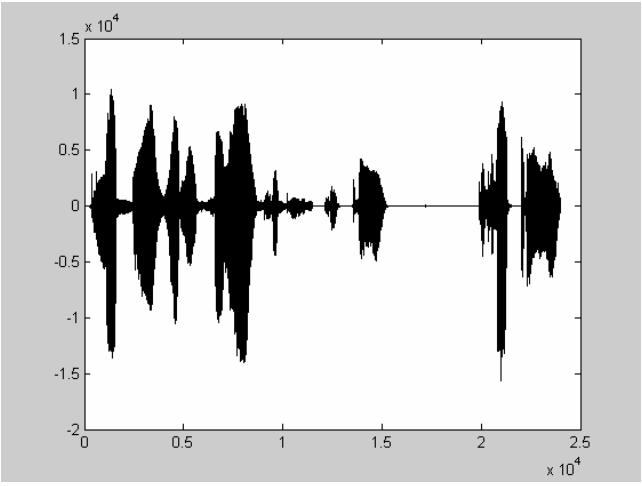
شکل (۳-۱۴) کدهای توابع فشرده و نافشرده‌سازی ***A-law***

تعداد سیکلهای مصرف شده دو تابع فوق در جدول (۱-۵) آورده شده است.

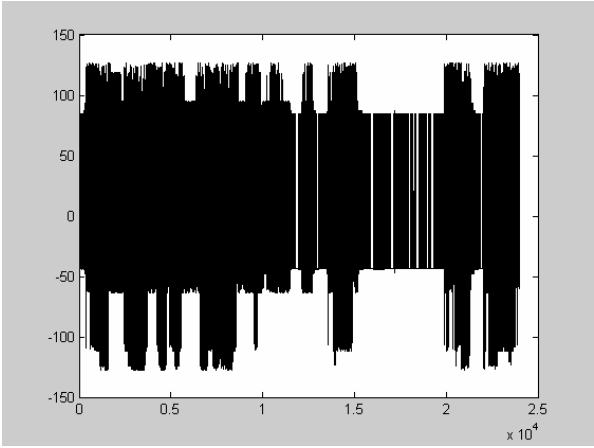
جدول (۱-۵) تعداد سیکل مصرفی توابع فشرده و نافشرده‌سازی ***A-law***

تابع	تعداد سیکل مصرف شده
int2alaw()	***192 Cycles=1.5 MCPS***
alaw2int()	***19 Cycles=0.15 MCPS***

شکل (۳-۱۵) پلات ۳ ثانیه از صحبت فایل speech-in که حاوی نمونه‌های ۱۶ بیتی خطی است و شکل (۳-۱۶) خروجی تابع ( )int2alaw که حاوی نمونه‌های ۸ بیتی A-law شده است، را نشان می‌دهند. همانطور که ملاحظه می‌شود پس از A-law شدن، نمونه‌های با دامنه کمتر، با دامنه بزرگتری نشان داده می‌شود.



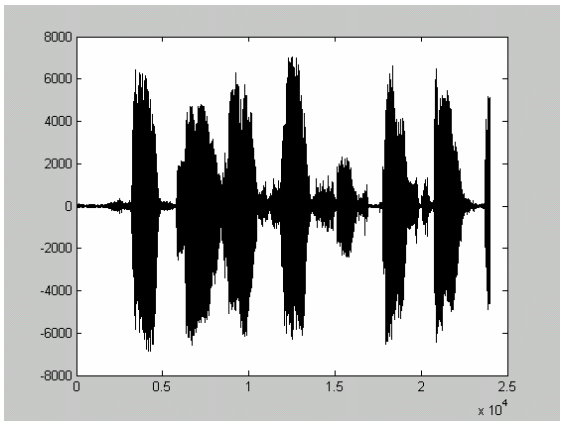
شکل (۳-۱۵) پلات ۳ ثانیه از فایل ***Speech.in***



شکل (۳-۱۶) پلات***A-law*** شده فایل ***speech.in***

در تابع alaw2int() باید نمونه‌ها بصورت متقارن بین اعداد ۱۲۸- و ۱۲۸+ باشند و گرنه کیفیت صحبت خروجی تابع alaw2int( ) به شدت خراب می‌شود. بنابراین نمونه‌های دریافت شده از ترانک بایستی پس از انتقال[1] و مقیاس شدن بین ۱۲۸- و۱۲۸+، به تابع alaw2int() وارد شوند.

نمونه‌های صحبت دریافت شده از ترانک دیجیتال برای ٤ ثانیه از صحبت در شکل (۳-۱۳) پس از انتقال و مقیاس شدن به تابع alaw2int( ) وارد شده‌اند. خروجی در شکل (۳-۱۷) نشان داده شده است. پس از خطی شدن نمونه‌های دریافتی، می‌توان آنها را برای کد شدن به تابع g729e _exec( ) اعمال کرد.

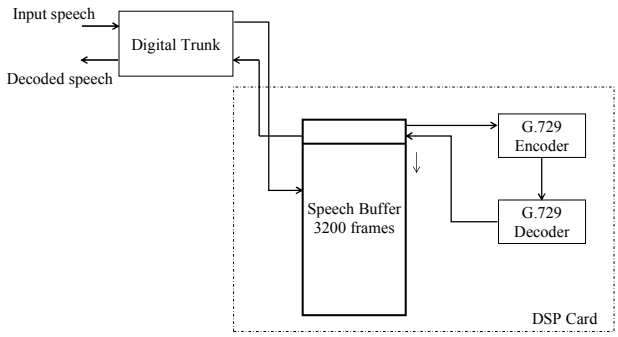


شکل (۳-۱۷) صحبت خطی شده شکل (۳-۱۳)

آزمایش غیر بلادرنگ پیاده‌سازی کد و واکد کننده G.729

[1] ***Shift***

برای اطمینان از معتبر بودن کدهای بهینه شده و عملکرد صحیح اجزاء سخت‌افزاری یک آزمایش غیر بلادرنگ همانند شکل (۳-۱۸) ترتیب داده شده است.



شکل (۳-۱۸) پیاده‌سازی غیر بلادرنگ کد و واکد کننده ***G.729***

در این آزمایش، حدود ٤ ثانیه صحبت از ترانک دیجیتال و بوسیله پورت سریال دریافت و در حافظه پردازنده ذخیره می‌شوند. این نمونه‌ها پس از خطی شدن به کد کننده و سپس به واکد کننده G.729 وارد می‌شوند. خروجی واکد کننده G.729 در حافظه نوشته می‌شود. پس از آن نمونه‌های بازسازی شده، A-law شده و به ترانک دیجیتال فرستاده می‌شوند. این آزمایش بصورت غیربلادرنگ می‌باشد، بدین صورت که شخص ابتدا از گوشی تلفن اول حدود ٤ ثانیه صحبت می‌کند و پس از چند ثانیه، صحبت بازسازی شده خود را از گوشی تلفن دوم می‌شنود. شکل (۳-۱۹) توابع کلی استفاده شده در این آزمایش را نشان می‌دهد.

```
...
 R_Buffer=(char**)malloc(2*sizeof(char*));
 T_Buffer=(char**)malloc(2*sizeof(char*));
 R_Buffer[0]=(char*)malloc(80*sizeof(char));
 R_Buffer[1]=(char*)malloc(80*sizeof(char));
 T_Buffer[0]=(char*)malloc(80*sizeof(char));
 T_Buffer[1]=(char*)malloc(80*sizeof(char));
 //*************************************
hmem=g729e_alloc();
G729e_INIT(hmem);
 hmemd=g729d_alloc();
G729d_INIT(hmemd);
//*************************************
 init_mcbsp();
 set_interrupts();
printf("START \n");
 while(!count1)                  // count1 will be True in interrupt service routine after 300
frames
 {
        if( R_Temp!=R_Check)       // R_Check changes in IST after receiving one frame
        {
                for(i=0;i<80;i++)    //converting into linear and filling buffer
                {
       law_speech=(R_Buffer[!R_Check][i]-64)*2;
```

```
        linear_speech=alaw2int(law_speech);
        linear_speech<<=3;
        *buffer_read_fill=linear_speech;
        buffer_read_fill++;
                }
               R_Temp=R_Check;
        }
}
printf(" ENOUGH \n ");
fwrite(buffer_write_fill, sizeof(Word16), 24000,f_speech); // writing buffer into file
//*********************  Doing G.729 Codec
frame=0;
while(frame++<300)
 {
    INTR_GLOBAL_DISABLE();
    for(i=0;i<80;i++)
     {
         samples[i]=*buffer_read;
         buffer_read++;
     }

   G729e_exec(hmem,samples,serial);
   G729d_exec(hmemd,serial, synthsamples);

   for(i=0;i<80;i++)
   {
      *buffer_write=synthsamples[i];
       buffer_write++;
   }
   INTR_GLOBAL_ENABLE();
 }
//************************
 for(i=0;i<80;i++)
  {
                linear_speech=*buffer_write_fill;
                buffer_write_fill++;
                linear_speech>>=3;
                law_speech=int2alaw(linear_speech);
                T_Buffer[0][i]=(law_speech>>1)+64;
  }
 for(i=0;i<80;i++)
  {
                linear_speech=*buffer_write_fill;
                buffer_write_fill++;
                linear_speech>>=3;
                law_speech=int2alaw(linear_speech);
                T_Buffer[1][i]=(law_speech>>1)+64;
  }
```

```
INTR_ENABLE(5);
INTR_ENABLE(7);
//*****************************sending to Trunk
while(!count1)
{
        if( R_Temp!=R_Check)
        {
                for(i=0;i<80;i++)
                {
        linear_speech= *buffer_write_fill;
        buffer_write_fill++;
        linear_speech>>=3;
        law_speech=int2alaw(linear_speech);
        T_Buffer[!R_Check][i]=(law_speech>>1)+64;
                }
                R_Temp=R_Check;
    }
}
printf(" FINISHED \n ");
...
```

شکل (۳-۱۹) قسمتی از کدهای برنامه آزمایش غیر بلادرنگ

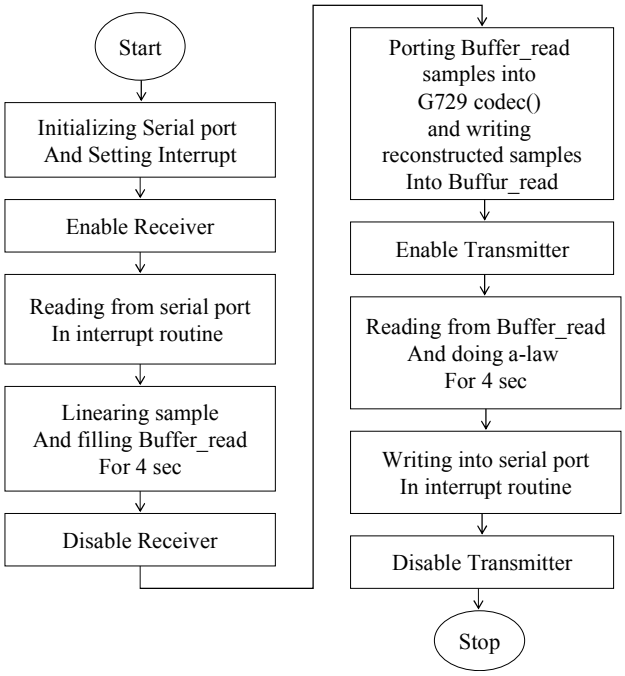
برای دریافت و ارسال نمونه‌ها بصورت فریم‌های۱۰ میلی ثانیه‌ای از بافرهای دوگانه[1] استفاده شده است. درگیرنده از R-Buffer[1], R-Buffer[0] که آرایه‌هایی از نوع هشت بیتی[2] و به طول ۸۰ نمونه هستند و در فرستنده نیز T-Buffer[1], T-Buffer[0] استفاده شده است. در این حالت، زمانی که نمونه‌ها در یک بافر ذخیره می‌شوند، CPU بافر دیگر را پردازش می‌کند و پس از پر شدن به بافر دوم سوئیچ می‌کند(تکنیک ping-pong).

کدهای برنامه کد کننده G.729 بصورت فایل کتابخانه‌ای در آمده‌اند به این ترتیب صدا زدن توابع کد کننده راحت‌تر و از صرف زمان برای کامپایلر کردن دوباره کدها جلوگیری می‌شود.

نکته مهمی که در اینجا بایست به آن توجه داشت استفاده از کلمه Volatile می‌باشد. زمانی که از متغیرهای عمومی در سرویس روتین‌های وقفه و برنامه‌های اصلی دیگر (Main برنامه) استفاده می‌شوند و یا اگر بخواهیم یک مقداری در محل خاص از حافظه پردازنده قرار بگیرد (مثلاً مقداردهی یک ثبات کنترلی) و از گزینه‌های بهینه‌سازی کامپایلر همانند ُo3-ُ استفاده گردد، بایستی قبل از تعریف متغیر مفروض از کلمه Volatile استفاده شود. علت این امر این است که وقتی از گزینه‌های بهینه‌سازی استفاده می‌گردد، کامپایلر متغیرها را به ثباتهای همه منظوره پردازنده نسبت می‌دهد و دسترسی به حافظه را محدود می‌کند. لیکن در موارد خاص مقدار متغیر مورد نظر بایستی در مکانی از حافظه پردازنده نوشته شود و لذا استفاده از کلمه Volatile ضروری می‌باشد. شکل (۳-۲۰) روند نمای آزمایش غیر بلادرنگ کد کننده G.729 را نشان می‌دهد.

[1] ***dual – buffer***

[2] ***char***

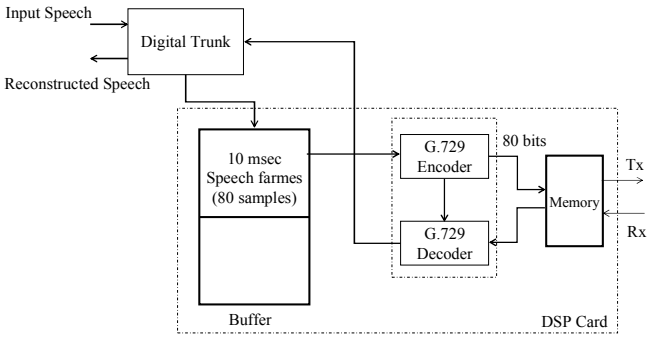


شکل (۳-۲۰) روندنمای آزمایش غیربلادرنگ

در آزمایش فوق برای سرویس‌دهی پورت سریال از وقفه دادن استفاده شده است که البته روش فوق برای کاربردهای بلادرنگ و همچنین کاربردهای چند کاناله مفید نمی‌باشد.

آزمایش بلادرنگ کدهای برنامه G.729 بر روی بورد

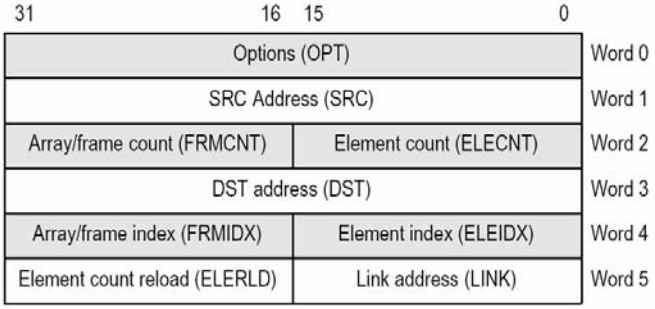
ملزومات لازم برای آزمایش بلادرنگ در شکل (۳-۲۱) آمده است.



شکل (۳-۲۱) آزمایش واقعی بلادرنگ ***G.729***

همانطور که ذکر شد برای سرویس‌دهی پورت سریال استفاده از وقفه دادن به CPU کار درستی نمی‌باشد، زیرا با دریافت هر نمونه صحبت به CPU وقفه اعمال می‌شود و خصوصاً در کاربردهای چندکاناله که تعداد نمونه‌ها بسیار زیاد می‌شود، به CPU بطور پی در پی و با فاصله زمانی بسیار کوتاه وقفه وارد می‌شود و از کارآیی پردازنده کم می‌شود. راه حلی که در این پروژه استفاده شده است استفاده از کنترل کننده (E)DMA برای سرویس‌دهی به پورت سریال می‌باشد[24].

پورت سریال با دریافت هر نمونه به کنترل کننده EDMA رویداد[1] می‌دهد. برای هر کانال، ٢٤ بایت لازم است تا پارامترهایی از قبیل آدرس مبدأ و مقصد داده‌ها، تعداد پارامترهایی که باید کپی شوند، سایز داده‌ها، چگونگی اعمال وقفه و ... مشخص شوند. دقت کنید که در EDMA بر خلاف DMA، هر وسیله جانبی، با یک رویداد منحصر به فرد که تولید می‌کند به یک کانال EDMA مربوط می‌شود. پارامترهای کنترلی کانالهای EDMA، در فضای مشخصی از حافظه به نام Parameter RAM قرار می‌گیرند. مثلاً پورت سریال شماره صفر با پر شدن بافر گیرنده به کانال شماره ١٣ رویداد می‌دهد و پارامترهای کنترلی آن در آدرس 01A0 0138hex تا 01A0 014Fhex قرار می‌گیرند. شکل (٣-٢٢) فضای پارامترهای کنترلی را نشان می‌دهد.

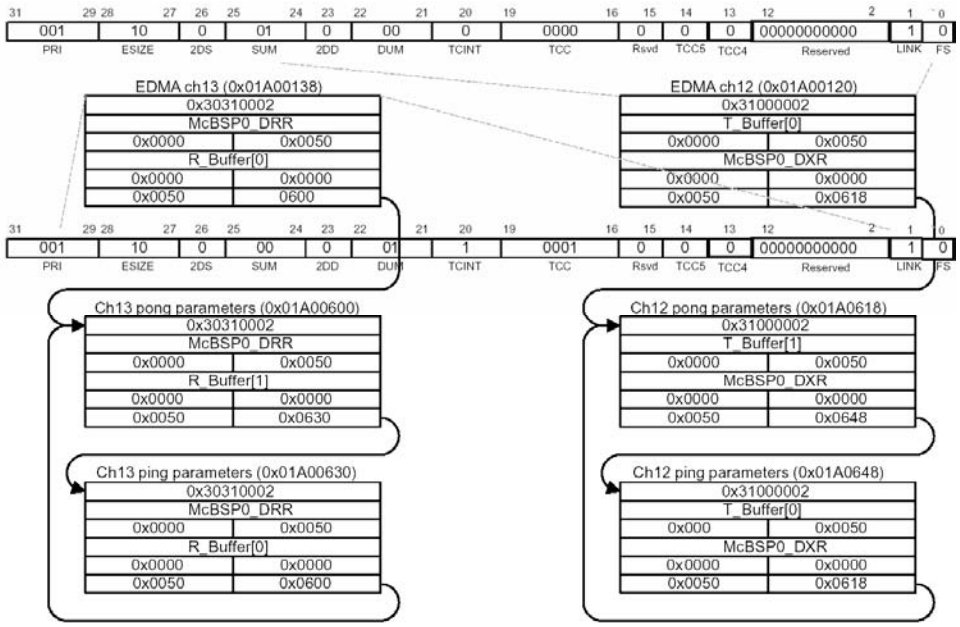


شکل (٣-٢٢) فضای پارامترهای کنترلی کانال ***EDMA***

کنترل کننده EDMA دارای یکسری ثبات کنترلی برای دریافت رویدادها و فعال کردن آنها می‌باشد. اطلاعات کامل در مورد کنترل کننده EDMA، انواع انتقالات و تنظیمات آن در پردازنده‌های سری C6000 در مرجع[24,39] توضیح داده شده است.

پارامترهای کنترلی فضای Parameter RAM برای گیرنده و فرستنده پورت سریال شماره صفر (کانال ١٢ و ١٣) برای بافر کردن Ping-Pong در شکل (٣-٢٣) آورده شده است.

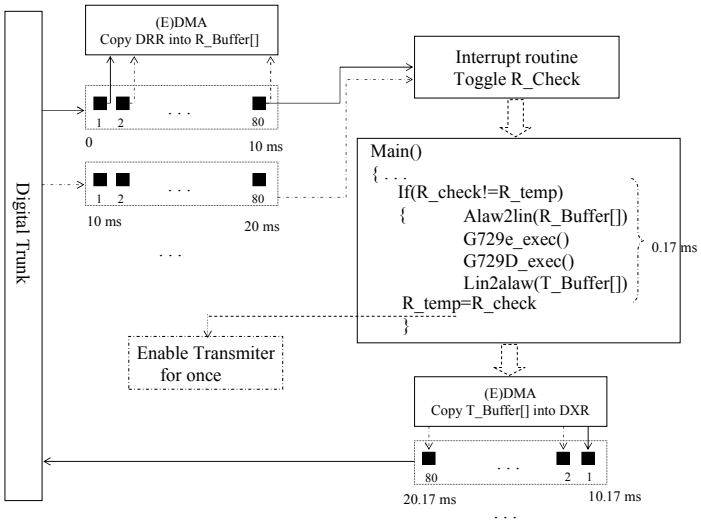
[1] ***Event***



شکل (۳-۲۳) تنظیمات ***EDMA*** در بافر کردن ***Ping-Pong***

کنترل کننده EDMA نمونه‌های صحبت دریافت شده از پورت سریال را در بافر مربوطه کپی می‌کند و پس از دریافت یک فریم ۱۰ میلی ثانیه‌ای به پردازنده وقفه می‌دهد. سپس نمونه‌های بعدی را در بافر دیگری کپی و بهمین ترتیب سوئیچ می‌کند(تکنیک بافر کردن ping–pong). برای این کار از دو بافر R-Buffer[0] و R-Buffer[1] درگیرنده و R-Buffer[0] و R-Buffer[1] در فرستنده، برای ذخیره نمونه‌های صحبت استفاده شده است. هر کدام از بافرهای فوق حاوی ۸۰ نمونه صحبت هستند. در ضمن برای اطلاع از پر شدن بافرها و سوئیچ کردن CPU به بافر دیگر از یکسری پرچم عمومی به نام‌های R-check , T-check استفاده شده است.

روند نمای شکل (۳-۲٤) چگونگی راه‌اندازی آزمایش بلادرنگ کد کننده G.729 را نشان می‌دهد.



شکل (۳-۲۴) روندنمای آزمایش بلادرنگ ***G.729***

شروع عملیات بدین صورت است که ابتدا پورت سریال با دیدن اولین پالس فریم سنکرون کننده وارد سرویس روتین وقفه می‌شود، در آنجا بعنوان گیرنده فعال می‌گردد و وقفه مربوطه غیر فعال می‌گردد. پورت سریال با دریافت هر نمونه صحبت به کنترل کننده EDMA ، رویداد می‌دهد و آن نیز نمونه‌ها را مرتب در بافرR-Buffer[0] می‌چیند. با پر شدن بافر R-Buffer[0] کنترل کننده EDMA به بافر R-Buffer[1] سوئیچ می‌کند و به CPU وقفه می‌دهد. در سرویس روتین وقفه مزبور، R-check تغییر می‌کند و لذا CPU می‌فهمد که فریم جدید صحبت دریافت شده است. پردازش بافر R-Buffer[0] بایستی قبل از کامل شدن بافر R–Buffer[1] تمام شده باشد.

پس از تکمیل پردازش اولین فریم صحبت (R-Buffer[0]) خروجی واکد کننده G.729 در T-Buffer[0] ریخته می‌شود و پورت سریال با دیدن پالس فریم سنکرون کننده وارد سرویس روتین وقفه شده، در آن بعنوان فرستنده فعال می‌گردد و وقفه مزبور غیر فعال می‌گردد و کنترل کننده EDMA به فرستنده نیز سرویس‌دهی می‌کند و نمونه‌های بازسازی شده صحبت از T-Buffer[0] (یا T-Buffer[1]) را به فرستنده پورت سریال کپی می‌کند.

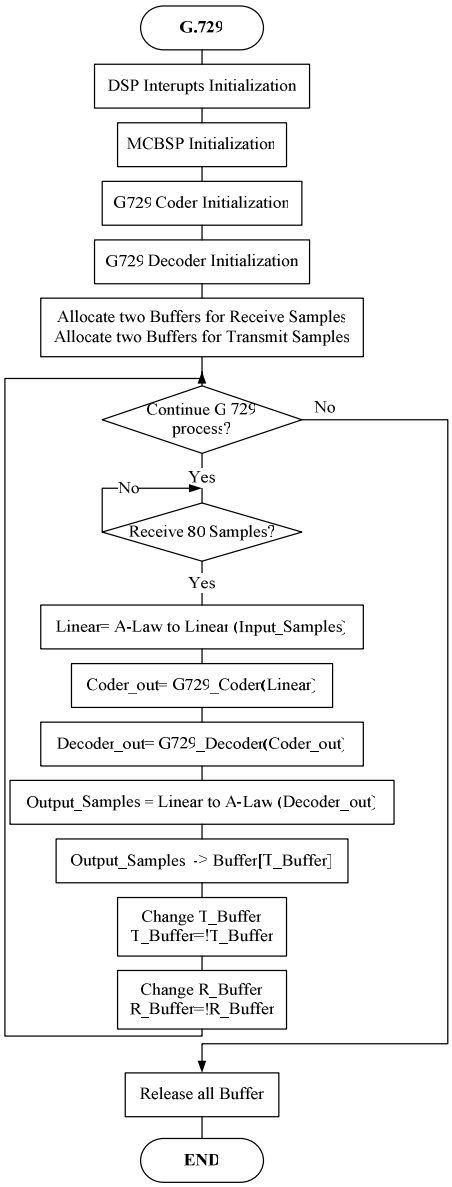
عملیات فوق بهمین ترتیب بطور مداوم تکرار می‌شود و هنگامی که از یک گوشی تلفن صحبت شود صدای بازسازی شده G.729 از گوشی تلفن دیگر بصورت بلادرنگ قابل شنیدن می‌باشد. شکل (۳-۲۵) قسمتی از کدهای برنامه مربوط به توابع main() و سرویس روتین وقفه آورده است.

```
...
EDMA_init();
init_mcbsp();
set_interrupts();
EDMA_rxStart( );
while(!count1)
{
 if( R_Temp!=R_Check)
 {
      for(i=0;i<80;i++)
      {
      law_speech=(R_Buffer[!R_Check][i]-64)*2;
      linear_speech=alaw2int(law_speech);
      linear_speech<<=3;
      samples[i]=linear_speech;
      }
      //************************************************************
        G729e_exec(hmem,samples,serial);
        G729d_exec(hmemd,serial, synthsamples);
      //************************************************************
      for(i=0;i<80;i++)
      {
            linear_speech=synthsamples[i];          //linear_speech= samples[i];
            linear_speech>>=3;
            law_speech=int2alaw(linear_speech);
            T_Buffer[!R_Check][i]=(law_speech>>1)+64;
      }
      if (!First)
      {
```

```
            First=1;
            INTR_ENABLE(7);
        }
        R_Temp=R_Check;
    }
}
...
```

شکل (۳-۲۵) قسمتی از کدهای پردازش بلادرنگ ***G.729***

مراحل اجرای کدهای برنامه در آزمایش بلادرنگ در شکل (۳-۲۶) نشان داده شده است.



شکل (۳-۲۶) مراحل اجرای کدهای برنامه در آزمایش بلادرنگ

آزمایش بلادرنگ کدهای G.729 بصورت چندکاناله

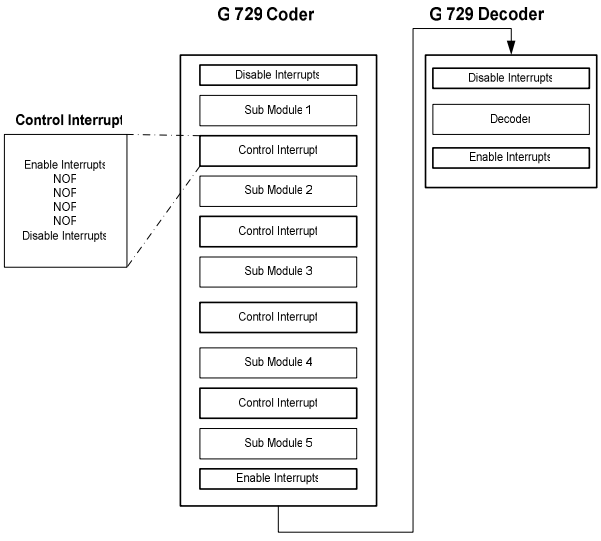
همانطور که در بخش قبل ذکر شد تا قبل از پر شدن یکی از بافرهای دریافت بایستی پردازش بافر قبلی تمام شده باشد، زیرا کنترل کننده EDMA پس از پر شدن یک بافر به بافر دیگر سوئیچ می‌کند و در اینجا تنها دو بافر وجود دارند که بصورت ping-pong کار می‌کنند. در حالت چند کاناله در ۱۰ میلی ثانیه بایستی پردازش داده‌های ٦٤ کانال صحبت تمام شود. برای آزمایش چند کاناله از آنجا که در عمل ٦٤ کانال داده معتبر وجود نداشت و نیز امکان شنیدن صحبت بازسازی شده کانال‌های متفاوت مقدور نبود، پس از کامل شدن یک فریم صحبت کد کننده G.729 به تعداد ٦٤ بار صدا زده شده است.

پس از انجام این آزمایش صحبت بازسازی شده G.729 با کیفیت مطلوب قابل شنیده شدن می‌باشد. اما هنگامی که تعداد بار صدا زدن‌های کد کننده G.729 بیش از عدد فوق می‌شود کیفیت صدا بازسازی شده کاهش می‌یابد. این بدین معنی است که زمان صرف شده برای پردازش کدها بیش از ۱۰ میلی ثانیه شده است، لذا یکسری از فریم‌های صحبت از دست رفته است. این آزمایش نشان می‌هد که بهینه‌سازی کدهای G.729 برای پردازش ٦٤ کانال مناسب می‌باشد.

اعمال وقفه در وسط اجرای برنامه

در حین اجرای کدهای برنامه G.729 وقفه‌های پردازنده غیر فعال می‌شوند. از آنجا که پردازش یک فریم صحبت در کد کننده G.729 برابر 8.6MCPS طول می‌کشد و پردازنده C6416 با کلاک 600MHz کار می‌کند، حداکثر به مدت 14.3 میکرو ثانیه وقفه‌ها غیر فعال هستند.

در حالت چند کاناله بودن زمان غیر فعال بودن وقفه‌ها بسیار زیاد می‌شود. برای اینکه زمان غیر فعال بودن وقفه‌ها کمتر گردد، کدهای برنامه به ۵ زیر و احد[1] تقسیم شده‌اند. در انتهای هر زیر واحد مقادیر لازم در حافظه ذخیره می‌شوند و بین دو زیر واحد وقفه‌ها فعال می‌گردند. در هر کدام از این زیر واحدها یکی از عملیات‌های الگوریتم G.729 از قبیل محاسبه ضرائب LP، کوانتیزه کردن ضرائب و امثال آن انجام می‌شود. با زیر واحد کردن و دسته‌بندی کدهای برنامه، حداکثر زمان غیر فعال بودن وقفه‌ها در کدکننده ٤٠ میکرو ثانیه و در واکد کننده ۳۵ میکرو ثانیه می‌باشد(شکل (۳-۲۷)).



[1] ***Sub module***

شکل (۳-۲۷) اعمال وقفه در وسط اجرای برنامه

فصل سوم

کدکننده‌های صحبت و استاندارد *G.729*

مقدمه

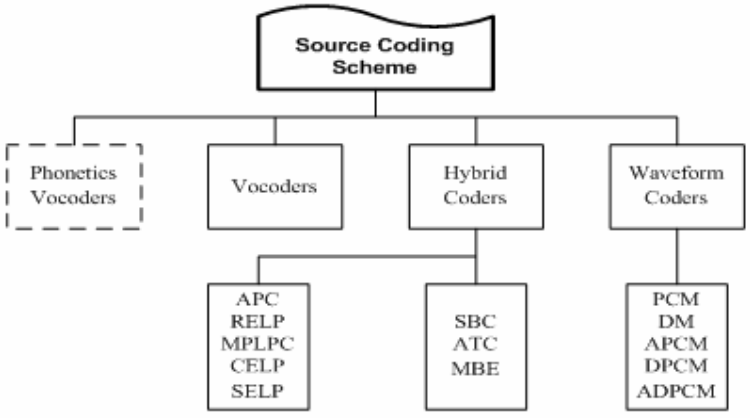
امروزه در عصر ارتباطات و گسترش روزافزون استفاده از شبکه های تلفن، موبایل و اینترنت در جهان و محدودیت پهنای باند در شبکه های مخابراتی، کدکردن و فشرده‌سازی صحبت امری اجتناب ناپذیر است.

کیفیت صحبت و نرخ بیت دو عامل اساسی هستند که بطور مستقیم با هم درگیر می‌باشند. با کاهش نرخ بیت از کیفیت صحبت کاسته می‌شود و با افزایش آن کیفیت صحبت بهبود می‌یابد. در سیستم‌هایی که به شبکه تلفن متصل می‌شوند، صحبت کد شده باید دارای کیفیت بالائی باشد و با استانداردهای ITU[1] مطابقت داشته باشد. اما برای سیستم‌های محدود مانند شبکه‌های تجاری خصوصی و سیستم‌های نظامی، ممکن است عامل نرخ بیت مهمتر از کیفیت بالای صحبت باشد.

یکی دیگر از ویژگیهای سیستم‌های کدکردن صحبت، تأخیر کدینگ می‌باشد 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 نمونه‌های صحبت) و تاخیر مربوط به ارسال و انتقال می‌باشد. تاخیر اندک در سیستم‌های مخابراتی باعث کاهش اثر اکو و افزایش کیفیت صدا می‌شود. چون کاهش تاخیر الگوریتمی و تاخیر زمان دریافت و ارسال تا حدودی غیر ممکن است، پس باید برای پیاده‌سازی بلادرنگ تنها عامل تاخیر محاسباتی مدنظر قرار بگیرد.

روشهای کدکردن

روشهای کدکردن صحبت را می‌توان به چند دسته اصلی که در۲شکل (۳-۱) نشان داده شده است، تقسیم‌بندی نمود. از این میان، سه دسته اصلی که با خط پر نشان داده شده‌اند موضوع تحقیقاتی بیشتری هستند. در این روشها سیگنال صحبت آنالیز شده، افزونگی‌های آن حذف می‌شود و بخش های غیر زائد صحبت به روشی کد می‌شود که بعد از بازسازی از نظر شنیداری قابل قبول باشد.



شکل (۳-۱) دسته بندی روشهای کدینگ

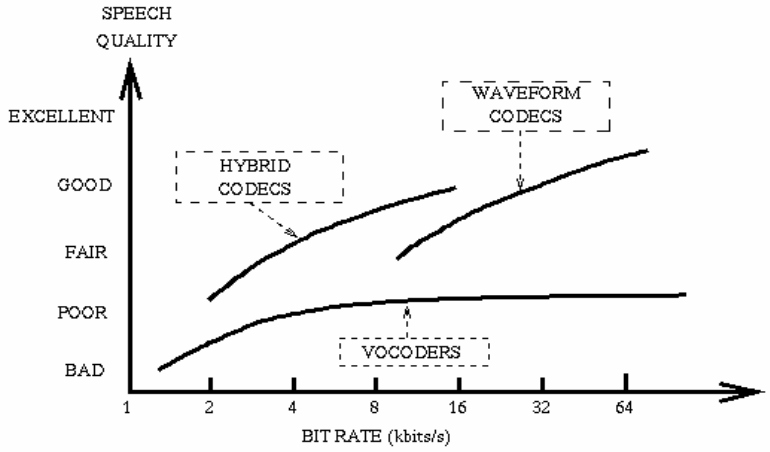
[1] ***International Telecommunication Union (CCITT)***

کدکننده های شکل موج[1] نوعا نرخ بیت بالایی دارند ولی به دلیل کیفیت مناسب و سادگی پیاده‌سازی مورد توجه قرار می‌گیرند. همچنین این کدکننده ها می‌توانند هر شکل موجی را در باند صوتی قبول کنند و صرفا مختص صحبت نیستند. از آنجا که در این کدکننده ها کدینگ به صورت نمونه به نمونه انجام می‌شود، عملکرد آنها همانند کوانتیزاسیون بوسیله نسبت سیگنال به نویز[2] اندازه‌گیری می‌شود***[1]***. در این کدکننده ها، شکل کلی سیگنال صحبت حفظ می‌شود.

کدکننده های صوتی[3] در نرخ بیت‌های خیلی پایین کار می‌کنند و صحبت را از طریق سنتز بازسازی می‌کنند. بر خلاف کد کننده های شکل موج، در این کد کننده ها شکل موج اصلی صحبت حفظ نمی‌شود. یک کدکننده صوتی از یک آنالیزکننده و یک سنتزکننده تشکیل شده است. آنالیزکننده از صحبت اصلی یک دسته پارامتر را که نمایش دهنده مدل تولید صحبت هستند، استخراج نموده و آنها را ارسال می‌کند و در گیرنده، صحبت با استفاده از این پارامترها بازسازی می‌شود***[2]***.

کد کننده های مختلط ترکیبی از تکنیک‌های صوتی و شکل موج را بکار می‌گیرند و صحبت با کیفیت خوب را در نرخ بیت‌های میانی ارائه می‌دهند***[3]***.

در شکل (۳-۲) کیفیت صحبت بر حسب نرخ بیت برای سه دسته اصلی کدینگ یعنی کدینگ شکل موج، کدینگ صوتی و کدینگ مختلط نشان داده شده است.



شکل (۳-۲) مقایسه کیفیت صحبت روشهای کدینگ صحبت

اتحادیهٔ بین المللی مخابرات[4] یک سری استاندارد در سری G.7xx را برای کدکننده‌های صحبت معرفی کرده است. جدول (۱-٦) این استانداردها بر اساس نرخ بیت، تاخیر و کیفیت مقایسه کرده است.

جدول (۱-٦) استانداردهای کدینگ صحبت

[1] ***Waveform Coders***

[2] ***SNR***

[3] ***Vocoders***

[4] ***ITU-T***

Year	*1972*	*1984 (1990)*	*1992*	*1995*	*1996*	*1995*	*1995*
ITU Std	*G.711*	*G.721 (G.726)*	*G.728*	*G.729*	*G.729A*	*G.723.1 (6.3)*	*G.723.1 (5.3)*
Bit-rate (kbit/s)	*64*	*32 (16,24,32,40)*	*16*	*8*	*8*	*6.3*	*5.3*
Technique	*PCM*	*ADPCM*	*LD-CELP*	*CS-ACELP*	*CS-ACELP*	*MP-MLQ*	*ACELP*
Quality (MOS)	*4.3*	*4.1*	*4.0*	*3.9*	*~ 3.7*	*~ 3.9*	*~ 3.7*
Delay (ms) (frame + look-ahead)	*0.125*	*0.125*	*0.625*	*10 + 5*	*10 + 5*	*30 + 7.5*	*30 + 7.5*

معرفی سیگنال صحبت

صحبت در اثر دمیدن هوا از ریه‌ها به سمت حنجره و فضای دهان تولید می‌شود. در طول این مسیر در انتهای حنجره، تارهای صوتی[۱] قرار دارند. فضای دهان را از بعد از تارهای صوتی، لوله صوتی[۲] می‌نامند. در تولید برخی اصوات، تارهای صوتی کاملاً باز هستند و مانعی بر سر راه عبور هوا ایجاد نمی‌کنند که این اصوات را اصطلاحاً اصوات بی واک[۳] می‌نامند. در دسته دیگر اصوات، تارهای صوتی مانع خروج طبیعی هوا از حنجره می‌گردند 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] می‌گویند.

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 توسط لوله صوتی شکل می‌گیرد و بسته به شکل لوله، پدیده تشدید در فرکانس‌های خاصی رخ می‌دهد که به آن فرکانس‌های تشدید فرمنت[۵] می‌گویند.

از آنجا که شکل لوله صوتی برای تولید اصوات مختلف، متفاوت است پس فرمنت‌ها برای اصوات گوناگون با هم فرق می‌کنند. با توجه به اینکه صحبت یک فرآیند متغییر با زمان است پس پارامترهای تعریف شده فوق اعم از فرمنت‌ها، پریود Pitch و مد صحبت به طور نامنظمی از باواک به بی واک و بالعکس در طول زمان تغییر می‌کند. لوله صوتی، همبستگی های زمان-کوتاه، در حدود 1ms، درون سیگنال صحبت را در بر می‌گیرد. بخش مهمی از کار کدکننده‌های صوتی مدل کردن لوله صوتی به صورت یک فیلتر زمان-کوتاه می‌باشد. چون شکل لوله صوتی نسبتاً آهسته تغییر می‌کند، تابع انتقال این فیلتر مدل کننده هم نیاز به تجدید[۶]، معمولاً در هر 20ms یکبارخواهد داشت.

[1] ***Vocal Cords***

[2] ***Vocal Tracts***

[3] ***Unvoiced***

[4] ***Voiced***

[5] ***Formant***

[6] ***Update***

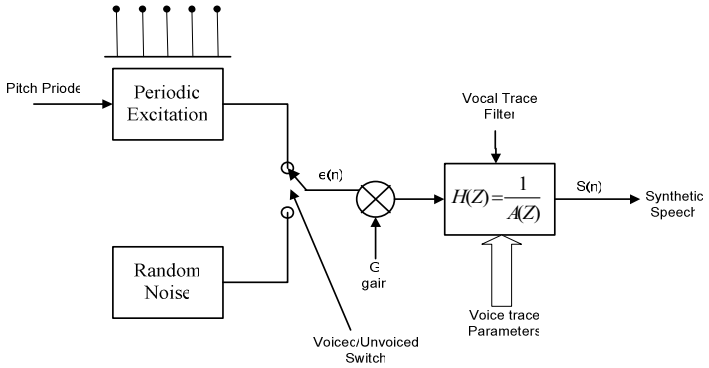
بطورکلی سیگنال صحبت دارای افزونگی[1] زیادی است که ناشی از عوامل ذیل هستند:

- وابستگی‌های زمان-کوتاه: این وابستگی‌ها عمدتاً به کندی تغییرات صحبت با زمان و ساختار نسبتاً منظم فرمنت‌ها مربوط می‌شوند.
- وابستگی‌های زمان- بلند: عمدتاً از طبیعت نیمه متناوب اصوات با واک و تغییرات آرام پریود Pitch ناشی می‌شوند.
- تابع 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 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***[4].***

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

مدل سازی پیشگویی خطی

روش کدینگ پیشگویی خطی(LPC[2]) مبتنی بر مدل تولید صحبت در کدکننده های صوتی می‌باشد(شکل (۳-۳)). برای استفاده از مدل لازم است که معلوم شود سیگنال با واک است یا بی‌واک و اگر با واک است پریود Pitch محاسبه گردد. در تحلیل LPC ، لوله صوتی به صورت یک فیلتر دیجیتال تمام قطب در نظر گرفته می‌شود***[5].***



شکل (۳-۳) مدل تولید صحبت در ***LPC***

[1] ***Redundancy***

[2] ***Linear Predictive Coding***

با شرکت دادن بهره G در این فیلتر داریم:

$$H(Z) = \frac{G}{1 + a_1 Z^{-1} + ... + a_p Z^{-p}} = \frac{S(Z)}{E(Z)} \qquad (۱-۳)$$

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:

$$S(n) = Ge(n) - a_1 S(n-1) - \Lambda \ - a_p S(n-p) \qquad (۲-۳)$$

به عبارت دیگر هر نمونه صحبت به صورت ترکیب خطی از نمونه‌های قبلی قابل بیان است و این دلیل نام گذاری کدینگ پیشگویی خطی می‌باشد***[6]***.

پنجره کردن سیگنال صحبت

روش LPC هنگامی دقیق است که به سیگنالهای ایستان[1] اعمال شود، یعنی به سیگنالهایی که رفتار آنها در زمان تغییر نمی‌کند. هر چند که این موضوع 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 فریم‌ها شبه ایستان هستند. این قسمت بندی با ضرب کردن سیگنال صحبت S(n)، در سیگنال پنجره W(n) انجام می‌شود. معروف‌ترین انتخاب برای پنجره، پنجره همینگ[2] به صورت زیر است:

$$W(n) = 0.54 - 0.46 \cos\frac{2\pi n}{N} \quad , \quad 0 \le n \le N-1$$
$$W(n) = 0 \quad , \quad otherwise \qquad (۳-۳)$$

N طول پنجره بر حسب نمونه‌های سیگنال صحبت است که عموماً در محدوده 320-160 انتخاب می‌گردد.

انواع دیگری از پنجره‌ها مانند پنجره Hamming و Kaiser نیز وجود دارند***[7]***.

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

پیش تاکید سیگنال صحبت

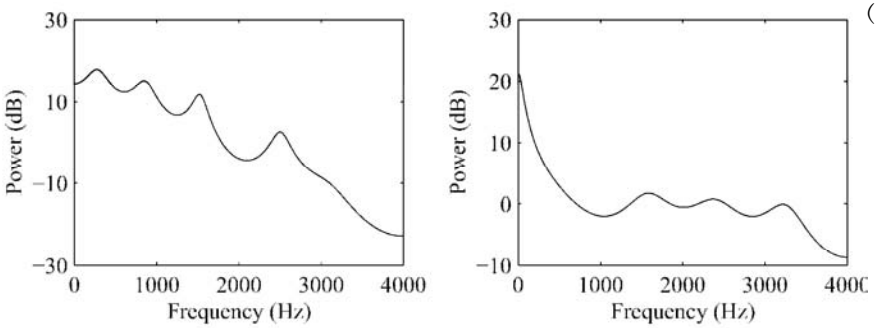
شکل (۳-٤) یک توزیع طیفی نمونه سیگنال صحبت را برای اصوات باواک و بی واک نشان می‌دهد. با توجه به افت طیف در فرکانس‌های بالا و ضعیف بودن آنها در طیف صحبت، تحلیل LPC در فرکانس‌های بالا عملکرد ضعیفی خواهد داشت. برای

[1] ***Stationary***

[2] ***Hamming***

تقویت مؤلفه‌های فرکانس بالای صحبت، آن را از یک فیلتر بالا گذر با تابع انتقال $1-az^{-1}$ که فیلتر پیش تاکید نامیده می‌شود، عبور می‌دهیم. 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:

$$S'(n)=S(n)-0.9375S(n-1) \qquad (۴-۳)$$



شکل (۳-۴) : پوشش طیفی نمونه اصوات باواک و بی واک

تخمین پارامترهای LPC

اگر تخمین S(n) از روی نمونه های قبلی باشد، به صورت زیر باشد

$$\hat{S}(n)=-a_1S(n-1)-\Lambda\ -a_ps(n-p) \qquad (۵-۳)$$

ضرایب a_i را چنان تعیین می‌کنیم که خطای

$$\sum_n\left[S(n)-\hat{S}(n)\right]^2 \qquad (۶-۳)$$

روی همه نمونه‌های موجود حداقل گردد *[5]*. این مینیمم‌سازی ما را به معادلات خطی زیر می‌رساند:

$$\begin{aligned}
&a_1r(0)+a_2r(1)+\Lambda\ +a_pr(p-1)=-r(1)\\
&a_1r(1)+a_2r(1)+\Lambda\ +a_pr(p-2)=-r(2)\\
&\qquad\qquad \mathrm{M}\\
&a_1r(p-1)+a_2r(p-2)+\Lambda\ +a_pr(0)=-r(p)
\end{aligned} \qquad (۷-۳)$$

و یا در فرم ماتریسی $R.a=-r$ در معادلات فوق تعریف زیر را داریم:

$$r(i)=r(-i)=\sum_{n=0}^{N-i-1}S(n)S(n+i) \qquad (۸-۳)$$

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 معکوس‌پذیر است و در نتیجه همواره می‌توانیم جوابی به صورت $a=-R^{-1}.r$ داشته باشیم[5,6] .

در روش کواریانس 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:

$$r(i,j)=\sum_{n} S(n+i)S(n+j) \qquad (۹-۳)$$

جواب دستگاه معادلات فوق را می‌توان با یکی از روش های کلاسیک آنالیز عددی مثل حذف گوسی بدست آورد. اما چون R یک ماتریس Toeplitz است می‌توان از روشی مؤثر بنام روش تکرار Durbin سود برد که بصورت زیر ضرائب فیلتر را تولید می‌کند:

$$\begin{aligned}
&E(0)=r(0)\\
&for \quad i=1,\ldots,p\\
&\quad K_i=-\frac{r(i)+a_1^{(i-1)}r(i-1)+\Lambda+a_i^{(i-1)}r(1)}{E(i-1)},\\
&\quad a_i^{(i)}=K_i\\
&\quad for \quad j=1,\ldots,i-1 \qquad a_j^{(i)}=a_j^{(i-1)}+K_i a_{i-j}^{(i-1)}\\
&\quad E(i)=(1-K_i^2)E(i-1)
\end{aligned} \qquad (۱۰-۳)$$

که در آن i, $a_j^{(i)}, j=1,\Lambda$، 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:

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

روش تکرار Durbin پارامترهای $K_i, i=1,\Lambda\ ,p$ را که ضرایب انعکاس نامیده می‌شوند و E(p) را بدست می‌دهد که مربع بهره پیشگویی G و مورد نیاز فیلتر سنتز می‌باشد[5,6,7]:

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

و چون داریم :

$$E(p)=(1-K_1^2)(1-K_2^2)\Lambda\ (1-K_p^2)r(0) \qquad (۱۱-۳)$$

می‌توانیم به جای r(0)،E(p) 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 Ki یا PARCOR (برای PARtial CORrelation) نقش مهمی در تحلیل LPC دارند و دارای خواص زیر هستند:

- ضرایب انعکاس K_i معادل با ضرایب فیلتر ai هستند . به عبارت دیگر می‌توان K را به a و برعکس

تبدیل کرد :

K به a :

$$\begin{aligned}
&a_i^{(i)}=K_i\\
&a_j^{(i)}=a_j^{(i-1)}+K_i a_{i-1}^{(i-1)} \qquad i=1,\Lambda\ ,p\\
&\qquad\qquad\qquad\qquad\qquad\quad j=1,\Lambda\ ,i-1
\end{aligned} \qquad (۱۲-۳)$$

a به K:

$$K_i = a_i^{(i)}$$
$$a_j^{(i-1)} = \frac{a_j^{(i)} - a_i^{(i)} a_{i-j}^{(i)}}{1 - K_i^2} \qquad i = p, \Lambda, 1 \qquad j = 1, \Lambda, i-1 \qquad (۱۳-۳)$$

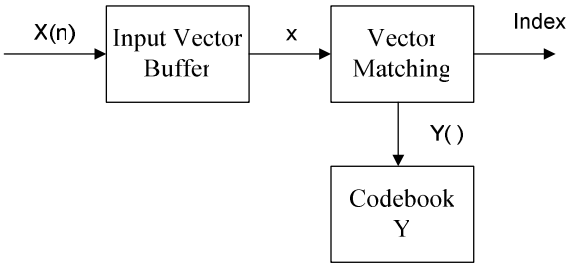
- برای یک فیلتر پایدار یعنی یک فیلتر LPC که همه قطب های آن داخل دایره واحد باشد داریم:

$$-1 < K_i < 1 \quad , \quad i = 1, \Lambda, p \qquad (۱٤-۳)$$

که این شرط بسیار مهمی است چرا که با اطمینان از اینکه Ki بین 1– و 1+ است 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 نیستند که پایداری فیلتر را تضمین نمایند و کوانیتزاسیون ai ها می‌تواند موجب ناپایداری شود[5].

چندی کردن بردار[1]

در چندی کردن بردار که خصوصاً در پردازش تصویر کاربرد دارد به جای بردار $\underline{X} = [x_1, x_2, ..., x_N]^T$ که مولفه‌های آن مقادیر دامنه یک سیگنال زمان- گسسته هستند یکی از بردارهای $\underline{Y_i} = [Y_{1i}, Y_{2i}, ..., Y_{Ni}]^T$، $1 \le i \le L$ انتخاب می‌شود. مجوعه بردارهای $\underline{Y_i}$ را کتاب کد و هرکدام از این بردارها را یک کلمه کد و L را اندازه کتاب کد می‌نامند. در شکل (۳-۵) بلوک دیاگرام ساده آن نشان داده شده است. فایده V.Q این است که بجای ارسال نمونه اصلی اندیس نمونه که حجم کمتری دارد ارسال می‌شود. هر چقدر اعضای کتاب کد بیشتر باشد، دقت عملیات بهبود یافته ولی زمان مقایسه بیشتر و در نتیجه تاخیر الگوریتم نیز بیشتر می‌شود***[8].***



شکل (۳-۵) بلوک دیاگرام یک ***V.Q***ساده

کد کننده‌های LPC با تحریک کد[2]

[1] ***Vector Quantization***

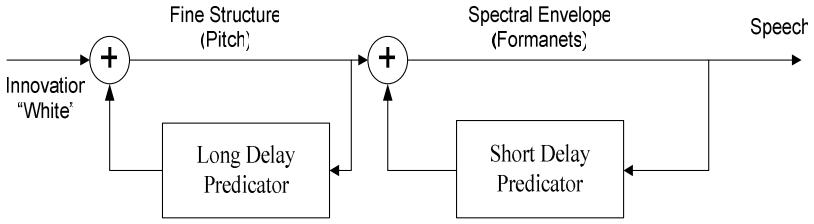
[2] Code excited LPC (CELP)

روش پیشگویی خطی با تحریک کد[1] در سال ۱۹۸۵ توسط Schroeder & Atal معرفی شد و تا کنون چندین استاندارد مهم کدینگ صحبت بر اساس CELP تعریف شده‌اند***[9]***

در این کدکننده‌ها مدل خاصی برای سیگنال تحریک در نظر گرفته نمی‌شود، بلکه بجای آن تعداد قابل توجهی دنباله تحریک در کتاب کد ذخیره می‌گردد و هنگام تحلیل یک قطعه از سیگنال صحبت با جستجو در کتاب کد، مناسب‌ترین دنباله تحریک برای آن قطعه پیدا و کد اشاره کننده به این دنباله تحریک ارسال می‌شود. لازم به ذکر است که نحوه کد کردن سیگنال مانده در CELP نوعی کوانتیزاسیون برداری است.

اعضای کتاب کد تحریک در مرحله آنالیز یکی پس از دیگری به فیلتر تمام قطب LPC اعمال می‌گردند. سپس سیگنال بازسازی شده با صحبت اصلی مقایسه شده، انرژی خطای حاصل به عنوان معیار در نظر گرفته می‌شود. طی یک جستجوی کامل در کتاب کد، نهایتاً دنباله تحریکی برگزیده می‌شود که کمترین انرژی خطا را دار باشد. اعضای کتاب کد تحریک با روشهای متنوعی قابل دستیابی هستند. کتاب کدهای تصادفی (کتاب کدی که کلمات کد آن به صورت تصادفی انتخاب می‌گردند) از نظر طراحی محدیت کمتری نسبت به کتاب کدهای یقینی دارند، اما از نظر کیفیت حد پائین‌تری دارند.

سنتز کننده صحبت در یک کدکننده CELP از دو عدد فیلتر بازگشتی متغیر با زمان، که هریک شامل یک پیشگویی کننده در حلقه فیدبک است، تشکیل شده است[9](شکل (۳-٦)).



شکل (۳-۶) مدل سنتز صحبت با پیشگویی تاخیر کوتاه وبلند

پیشگویی کننده تاخیر کوتاه، ١٦ ضریب دارد و به کمک روشهای آنالیز ضرایب LP، در هر ۱۰ میلی ثانیه، محاسبه می‌شوند. پیشگویی خطی با تاخیر بلند[2]، ۳ ضریب دارد و براساس مینیمم کردن خطای مجذور مربعات، پس از پیشگویی گام صدا، محاسبه می‌شود.

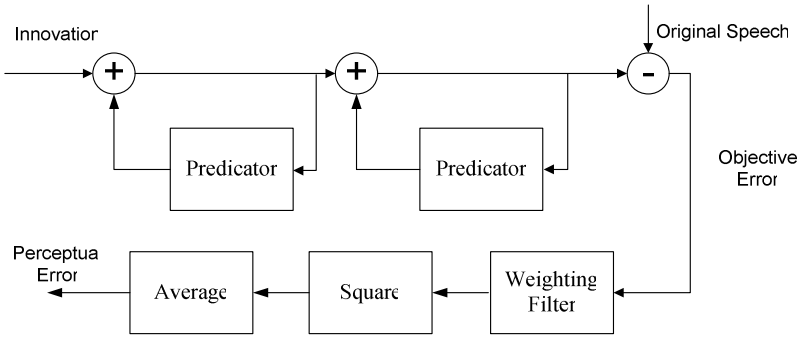
پروسه انتخاب سیگنال تحریک

، را بررسی می‌کنیم. 8khz برای این منظور یک بلوک کوچک ۵ میلی ثانیه از سیگنال صحبت، نمونه برداری شده با فرکانس نشان داده شده است.شکل (۳-۷) این بلوک شامل ٤٠ نمونه صحبت می‌باشد. نحوه انتخاب بهترین ترتیب در

هر عضو کتاب کد، ٤٠ نمونه از سیگنال تحریک را فراهم می‌کند. هر نمونه از سیگنال تحریک با یک ضریب دامنه، که برای بلوک ۵ میلی ثانیه ثابت است و در ابتدای هر بلوک reset می‌شود، مقیاس شده است. این نمونه‌ها بترتیب از دو فیلتر بازگشتی عبور داده می‌شوند[9,10].

[1] ***CELP***

[2] ***Pitch***



شکل (۳-۷) بلوک دیاگرام پروسه انتخاب سیگنال تحریک

نمونه‌های صحبت بازسازی شده در خروجی فیلتر دوم با نمونه‌های صحبت اصلی مقایسه می‌شوند و یک سیگنال خطا حاصل می‌شود. سیگنال خطا از یک فیلتر خطی عبور داده می‌شود تا فرکانسهایی که در آنها خطا کمتر اهمیت دارد تضعیف شده و فرکانسهایی که در آنها خطا بیشتر اهمیت دارد تقویت شوند. تابع انتقال این فیلتر وزنی بصورت زیر است[9].

$$W(Z)=\frac{1-\sum_{k=1}^{p}a_k.Z^{-k}}{1-\sum_{k=1}^{p}a_k.\alpha^k.Z^{-k}} \qquad (۱۵-۳)$$

که در آن a_k ضرائب پیشگویی تاخیر کوتاه (ضرائب LP)، p=16 و α پارامتری برای کنترل کردن وزندهی خطا، بصورت تابعی از فرکانس، می‌باشد. یک مقدار مناسب برای α بدین صورت است:

$$\alpha = e^{-2\pi.100/f_s} \qquad (۱۶-۳)$$

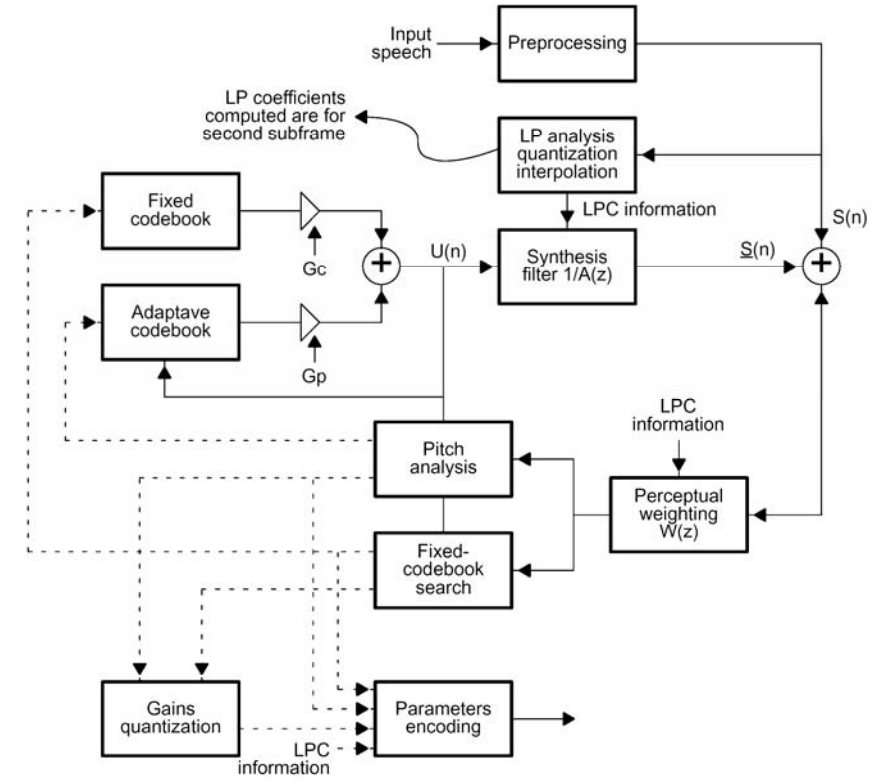
که f_s فرکانس نمونه برداری می‌باشد.خطای مجذور مربعات با مجذور کردن و میانگین‌گیری نمونه‌های خطا در خروجی فیلتر وزندهی، برای هر بلوک ۵ میلی ثانیه محاسبه می‌شود. سپس بهترین سیگنال تحریک برای هر بلوک، بطوری که خطای وزندهی مینییم گردد، جستجو می‌شود. در ضمن ضریب گین نمونه‌های سیگنال تحریک[1] از کتاب کد، طوری که خطای وزنی برای هر بلوک مینیمم شود، محاسبه می‌شود.

توصیف الگوریتم CS-ACELP

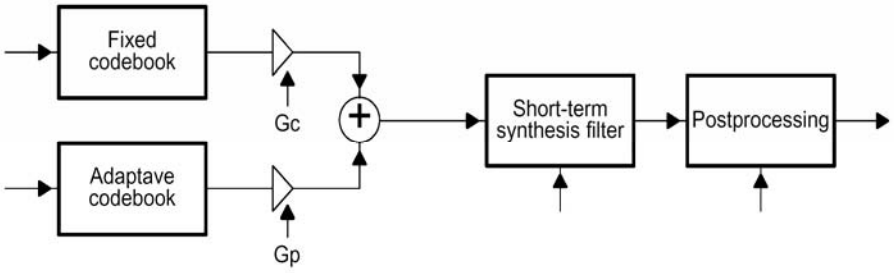
در شکل (۳-۸) و شکل (۳-۹) بلوک دیاگرامهای مربوط به کدر و دیکدر الگوریتم CS-ACELP نشان داده شده است. این کدکننده بر روی سیگنالهای صحبت نمونه برداری‌شده با فرکانس 8000Hz عمل می‌نماید. قسمت کد کننده الگوریتم، سیگنالهای صحبت ورودی را به صورت فریمی و زیر فریمی پردازش می‌نماید. طول هر فریم 10 ms بوده و شامل دو زیر فریم 5ms ای می‌باشد.

استفاده از زیر فریمها برای بدست آوردن دقیقتر پارامترهای pitch و گین و کاهش محاسبات لازم برای جستجوهای مربوط به کتاب کدها مفید می‌باشد[11-15].

[1] ***code words***



شکل (۳-۸) بلوک دیاگرام قسمت کدر در استاندارد ***G.729***



شکل (۳-۹) بلوک دیاگرام قسمت ***decoder*** در استاندارد ***G.729***

اعمالی که در کدکننده انجام می‌شود بر دو نوع می‌باشند. نخست اعمالی که در هر فریم یک بار انجام می‌شوند که شامل pre-processing، تجزیه و تحلیل پیش بینی خطی[1]، کوانتیزه کردن و جستجوی pitch به صورت حلقه باز می‌باشد. دوم اعمالی که در هر زیرفریم یک بار انجام می‌شوند که شامل جستجوی pitch به صورت حلقه بسته (کتاب کد تطبیقی) و جستجوی کتاب کد ثابت می‌باشد.

[1] ***Linear Prediction***

همان طور که در شکل (۳-۸) مشاهده می‌شود اولین عملی که بر روی سیگنالهای ورودی انجام می‌شود pre-processing می‌باشد. این مرحله شامل دو عمل signal scaling و high-pass filtering می‌باشد. پس از آن تجزیه و تحلیل پیش‌بینی خطی انجام می‌شود. در این مرحله پس از انجام محاسبات مربوط به windowing و auto-correlation ضرایب فیلتر LP با استفاده از الگوریتم Levinson-Durbin محاسبه شده سپس این ضرایب به ضرایب جفت طیفی خطی[1] تبدیل شده و با عمل کوانتیزه‌سازی برداری دو مرحله‌ای و تخصیص ۱۸ بیت به آنها، کوانتیزه می‌شوند. ضرایب فیلتر LP به صورت کوانتیزه شده و کوانتیزه نشده در زیرفریم دوم استفاده می‌شوند. در حالی که در زیرفریم اول انترپولاسیون ضرایب فیلتر LP مورد استفاده قرار می‌گیرد. برای هر زیرفریم، سیگنال تحریک فیلتر ترکیبی LP به کمک یک کتاب کد تطبیقی و یک کتاب کد ثابت بدست می‌آید. بردار مربوط به کتاب کد تطبیقی برای بیان کردن مؤلفهٔ متناوب سیگنال تحریک و بردار مربوط به کتاب کد ثابت برای نشان دادن اصوات غیر آوایی در سیگنال تحریک می‌باشد[14].

کتاب کد تطبیقی با استفاده از روشی دو مرحله‌ای مورد جستجو قرار می‌گیرد. در مرحلهٔ اول برای هر فریم، بر اساس سیگنال صحبت وزن دار یک تأخیر pitch به صورت حلقه باز تخمین زده می‌شود. سپس اندیس کتاب کد تطبیقی و گین مربوطه در یک جستجوی حلقه بسته حول تأخیر pitch حلقه باز یافت می‌شوند. سیگنالی که نظیر آن باید یافت شود با عنوان سیگنال هدف، به وسیلهٔ فیلتر کردن باقیماندهٔ LP با فیلتر ترکیبی وزن دار محاسبه می‌شود. سیگنال هدف با برداشته شدن قسمت مربوط به کتاب کد تطبیقی، در جستجوی کتاب کد ثابت مورد استفاده قرار می‌گیرد. گین‌های کتاب کدهای تطبیقی و ثابت با هفت بیت و با استفاده از یک کتاب کد با ساختار مزدوج، به صورت برداری کوانتیزه می‌شوند. دو بردار کتاب کد بدست آمده میزان شده با گین‌های مربوط به خود، با یکدیگر جمع شده تا تحریک لازم برای فیلتر ترکیبی LP را بسازند. سیگنالهای صحبت تولید شده به نحوی ساخته می‌شوند که کمترین خطا و اعوجاج نسبت به سیگنالهای اولیه بدست آید. تخصیص بیت به فریمهای 10 ms ای در الگوریتم CS-ACELP درجدول (۱-۷) نشان داده شده است[15].

جدول (۱-۷) تخصیص بیت برای هر فریم در استاندارد ***G.729***

Parameter	Codeword	Subframe1	Subframe 2	Total per frame
Line Spectrum pairs	*L0,L1,L2,L3*	–	–	18
Adaptive-Codebook Delay	*P1,P2*	8	5	13
Pitch-Delay Parity	*P0*	1	–	1
Fixed-Codebook Index	*C1,C2*	13	13	26
Fixed-Codebook Sign	*S1.S2*	4	4	8
Codebook gains (stage 1)	*GA1, GA2*	3	3	6
Codebook gains (stage 2)	*GB1, GB2*	4	4	8
Total				**80**

در قسمت واکد کننده الگوریتم، ابتدا پارامترهای ارسالی از رشته بیتهای دریافتی استخراج می‌شوند. سپس برای هر فریم 5ms ای سیگنال تحریک با جمع کردن دو بردار کتاب کد میزان شده با گین‌های مربوطه ساخته می‌شود. آنگاه سیگنال صحبت با فیلتر کردن سیگنال تحریک از طریق فیلتر ترکیبی LP ساخته می‌شود. سیگنال صحبت بازسازی شده در قسمت post-

[1] ***Line Spectral Pair***

processing برای بهبود کیفیت از چند فیلتر عبور داده می‌شود و در انتها دو عمل signal scaling و low-pass filtering بر روی سیگنالهای صحبت بازسازی شده انجام می‌شود.

استاندارد G.729

استاندارد G.729 بر اساس الگوریتم(CS-ACELP[1]) می‌باشد. این استاندارد در مارس سال ۱۹۹۶ توسط اتحادیهٔ بین المللی مخابرات ارائه شد[11].

مشخصات اصلی این استاندارد عبارتند از : نرخ دادهٔ کم(8 kbit/s)، تأخیر الگوریتمی نسبتاً پایین(10 ms)، پیچیدگی محاسباتی کاهش یافته نسبت به الگوریتمهای بر اساس CELP و کیفیت خروجی مناسب. چند ماه پس از ارایهٔ استاندارد G.729 ضمیمهٔ A آن (G.729A) با مشخصهٔ اساسی پیچیدگی محاسباتی کاهش یافته و سازگار بودن نرخ بیت با استاندارد G.729 به وسیلهٔ ITU-T ارائه گردید[16]. در سالهای بعد ضمایمی دیگر نیز به این استاندارد اضافه گردید که مهمترین آنها به شرح زیر می‌باشند:

G.729B : حاوی بلوکهای محاسباتی لازم برای فشرده سازی بیشتر سیگنالهای ورودی است هنگامی که سکوت در سیگنالهای ورودی برقرار است. بلوکهای توصیف شده در G.729B با اضافه شدن به G.729 و G.729A امکان فشرده-سازی بیشتر سیگنالهای صوتی را فراهم می‌نمایند.

G.729C : حاوی کدهای شبیه‌سازیG.729 و G.729A به صورت floating-point می‌باشد.

G.729D : همان G.729 ولی با نرخ 6.4 kbit/s و کیفیتی کمی پایین تر از G.729 می‌باشد.

G.729E : همان G.729 ولی با نرخ 11.8 kbit/s با قابلیت استفاده در فشرده‌سازی سیگنالهای صحبت همراه با نویز زمینه و حتی موسیقی می‌باشد.

ضمیمه G.729A

در مقایسه با استاندارد G.729، G.729A نیاز به قدرت محاسباتی کمتری دارد ولی در عوض کیفیت صحبت بازسازی شده توسط آن به خوبی استاندارد G.729 نمی‌باشد. البته هر دو استاندارد از نظر رشته‌های بیتی و عملیاتی سازگار با یکدیگر می‌باشند و بعنوان نمونه بسته‌های G.729 می‌توانند با واکدکننده G.729A واکد شوند و برعکس[17]. در جدول (۱-۸) این دو کد کننده بر حسب معیار MOS[2] با یکدیگر مقایسه شده‌اند[17]. محدوده معیار MOS از ۰ تا ۵ می‌باشد.

جدول (۱-۸) مقایسه ***G.729*** و ***G.729A*** بر حسب معیار ***MOS***

[1] ***Conjugate-Structure Algebraic-Code-Excited Linear-Prediction***

[2] Mean Opinion Score

Vocoder	MOS Result (In Clean Conditions)
ITU G.729	4.125
ITU G.729A	3.7

توصیف کلی کدکننده G.729A بسیار شبیه به کدکننده G.729 می‌باشد. تفاوت اصلی بین این دو کد کننده در نحوه کوانتیزه کردن ضرائب LP و درون‌یابی آنها، وزندهی ادراکی، آنالیز گام صحبت وجستجوی کتاب کدهای ثابت و وفقی می‌باشد. در جدول (۱-۹) توابع کدهای برنامه کدکنندهG.729A با کدکننده G.729 مقایسه شده‌اند[15,16].

جدول (۱-۹) مقایسه توابع کدکننده ***G.729A*** با ***G.729***

Encoder Functions		Common Functions		Decoder Functions	
Function Name	Comparison with G.729	Function Name	Comparison with G.729	Function Name	Comparison with G.729
ACELP_Code_A	few diff.	ClearOverflow	same	agc	new
Autocorr	same	Copy	same	D_lsp	same
Az_lsp	few diff.	Gain_predict	same	Dec_gain	same
Chebps_10	same	Gain_update	same	Dec_lag3	same
Chebps_11	same	Get_lsp_pol	same	Decod_ACELP	same
Cor_h	same	GetOverflow	same	Decod_ld8a	few diff.
Cor_h_X	same	Int_qlpc	same	Gain_update_erasure	same
Corr_xy2	same	Inv_sqrt	same	Lsp_iqua_cs	same
D4i40_17_fast	different	Log2	same	Pit_pst_filt	new
Dot_product	new	Lsf_lsp2	same	Post_Filter	new
Enc_lag3	same	Lsp_Az	same	Post_Process	same
Encod_ld8a	different	Lsp_expand_1_2	same	Preemphasis	new
G_pitch	same	Lsp_get_quant	same	Random	same
g729a_encode	similar	Lsp_prev_compose	same	g729a_decode	similar
g729a_encode_initialize	similar	Lsp_prev_extract	same	g729a_decode_initialize	similar
Gbk_presel	same	Lsp_prev_update	same		
Get_wegt	same	Lsp_stability	same		
Lag_window	same	Pow2	same		
Levinson	same	Pred_lt_3	same		
Lsp_expand_1	same	Qua_lsp	same		
Lsp_expand_2	same	Residu	same		
Lsp_get_tdist	same	Set_zero	same		
Lsp_last_select	same	Syn_filt	same		
Lsp_lsf2	same	Weight_Az	same		
Lsp_pre_select	same				
Lsp_qua_cs	same				
Lsp_select_1	same				
Lsp_select_2	same				
Parity_Pitch	same				
Pitch_fr3_fast	new				
Pitch_ol_fast	new				
Pre_Process	same				
Qua_gain	same				
Qua_lsp	same				
Relspwed	same				
test_err	same				
update_exc_err	same				

شکل (۳-۱۰) روندنمای کد کردن سیگنال صحبت در استاندارد G.729A را نشان می‌دهد. کد کننده صحبت G.729 به پنج زیر واحد[1] تقسیم می‌شود که عبارتند از:

[1] ***Submodule***

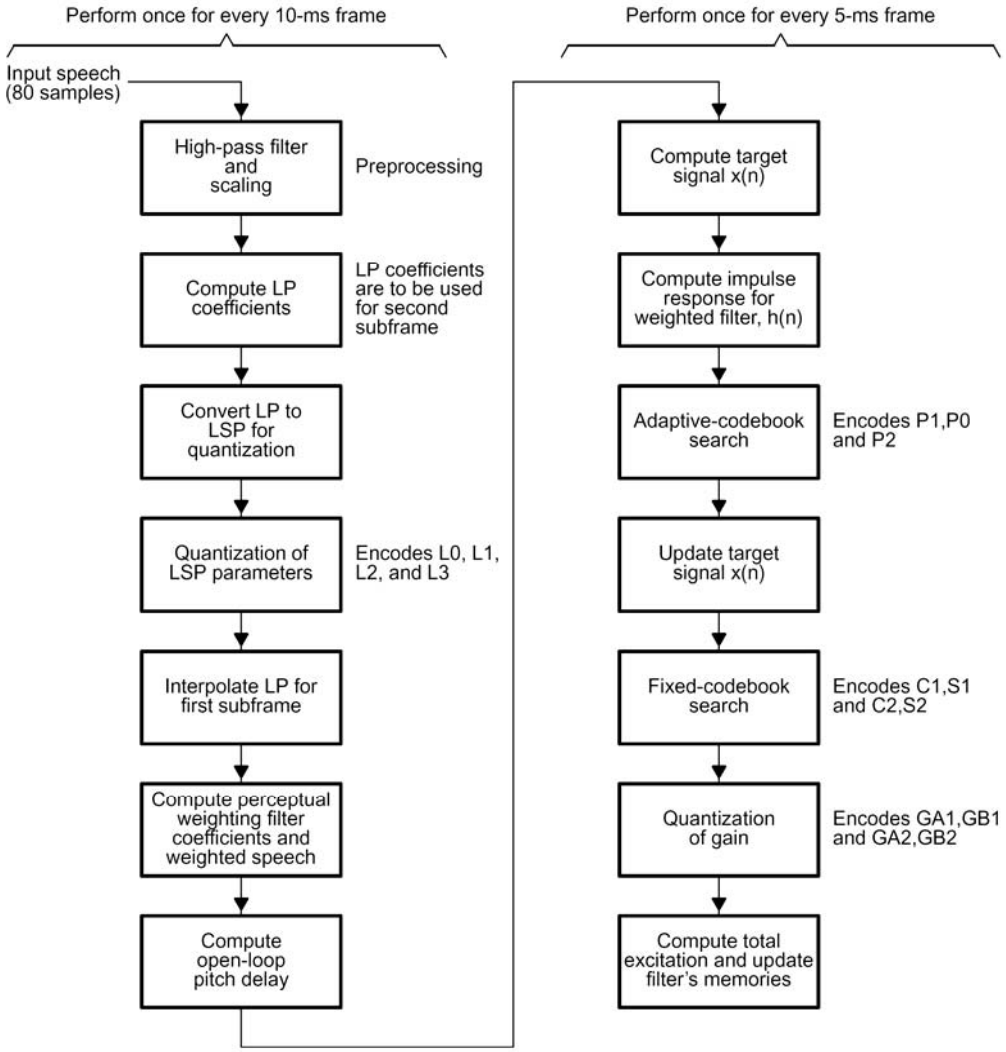
- زیر واحد اول شامل توابع( )Pre_Process، ( )Autocorr، ( )Lag_window، ( )Levinson و ( )Az_lsp برای پردازش اولیه[1]، آنالیز ضرائب LP و تبدیل ضرائب LPC به LSP می‌باشد.
- زیر واحد دوم شامل تابع ()Qua_lsp برای کوانتیزه کردن ضرائب LSP می‌باشد.
- زیر واحد سوم شامل توابع ()Int_qlpc ، ()Int_lpc، ()Weight_Az،()Residu، ()Syn_filt و()Pitch_ol برای درون‌یابی پارامترهای LPC، وزندهی نمونه‌های صحبت و جستجو گام صحبت حلقه باز می‌باشد.
- زیر واحد چهارم از توابع () Pitch_fr3، ()Enc_lag3، ()Pred_lt_3، ()Convolve و()G_pitch برای عملیات جستجوی گام کسری حلقه بسته و جستجوی کتاب کد وفقی استفاده می‌کند.
- زیر واحد پنجم از توابع ()ACELP_Codebook، ()Corr_xy2، ()Qua_gain و()Syn_filt برای جستجوی کتاب کد و به روز کردن ضرائب فیلتر بهره می‌برد.

زیر واحدهای ۱ تا ۳ مربوط به پروسس فریم هستند و بایستی یک بار در هر فریم انجام شوند. زیر برای پروسس زیر فریم هستند و بایستی دوبار در هر فریم تکرار شوند. واحد های ٤ و ٥ واکدکننده G.729A به دو زیر واحد تقسیم می‌شود:

- زیر واحد اول مربوط به آشکارکردن پارامترهای فریم می‌باشد و بایستی یک بار در هر فریم صدا زده شود.
- زیر واحد دوم مربوط به آشکارکردن پارامترهای زیر فریم، سنتز کردن صحبت و فیلتر کردن نهایی[2] می‌باشد. زیر واحد دوم بایستی دوبار در هرفریم تکرار شود.

[1] ***Pre process***

[2] ***Post-filtering***



شکل (۳-۱۰) روندنمای کد کردن در استاندارد ***G.729A***

فصل چهارم

پیاده سازی پروتکل G.728

مقدمه

این بخش نحوه پیاده‌سازی یکی از استاندارد های کدینگ صحبت – کدینگ صحبت به روش پیشگویی خطی کد تحریک با تأخیر کم ، استاندارد G.728 – را به صورت ممیز ثابت شرح می‌دهد.

این کدک، صحبت با کیفیت بسیار خوب را در نرخ بیت 16 kbps ارائه می دهد.

از این کدک می توان در شبکه‌هایی که به تأخیر زیاد حساس هستند، مانند شبکه‌های ماهواره‌ای یا موبایل استفاده کرد. همچنین یکی از استانداردهایی است که برای انتقال صحبت در سیستم‌های کنفرانس تصویری به کار می‌رود. از دیگر کاربردهای آن کدینگ صحبت در شبکه‌های است که اطلاعات را به صورت بسته ای انتقال می‌دهند.

پس از برنامه نویسی وشبیه سازی کدهای C الگوریتم G.728 با استفاده از نرم افزار CCS برنامه به زبان اسمبلی ترجمه می شود و روشهای بهینه سازی به منظور افزایش سرعت اجرا و کاهش اندازه کد تولید شده توابع بر روی آن صورت می گیرد.

پس از انجام بهینه سازیهای لازم برخی از توابعی که تاثیر زیادی در زمان اجرای برنامه دارند به صورت دستی به زبان اسمبلی باز نویسی می شوند تا شرایط بلادرنگ بودن را فراهم آورند.

شرح استاندارد G.728

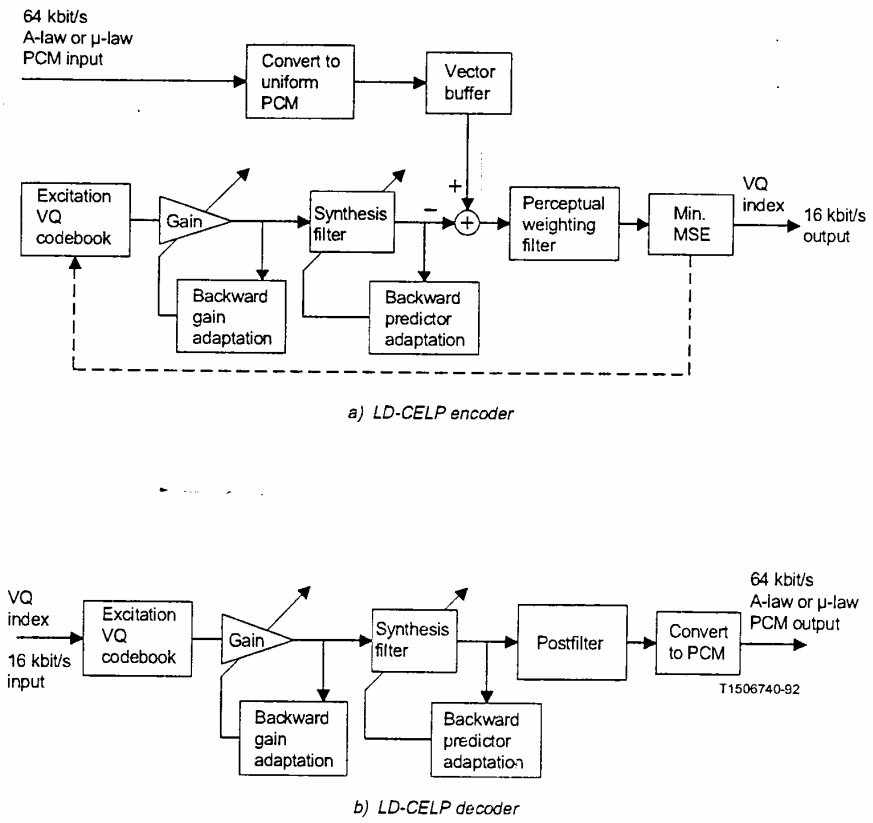
بنابر تعریف، تأخیر یک کدک صحبت زمان بین ورودیک نمونه به اینکدرو خروج نمونه متناظر آن از دیکدر است با فرض اینکه خروجی اینکدر به طور مستقیم به ورودی دیکدر داده شود. این تأخیر برای یک کدر مرکب نوعاً بین 50ms تا 100ms است. این تاخیر زیاد برای شبکه‌های حساس به تأخیر مشکلاتی را ایجاد می‌کند. بنا براین ITU-T در سال ۱۹۸۸ مشخصات فنی یک کدک مرکب 16kbps را منتشرکرد. در جدول (٤-١)مشخصات مهم این کدک آورده شده است. در سال ۱۹۹۲، الگوریتم یک کدک CELP تطبیقی پسرو توسعه یافته توسط آزمایشگاههای بل که کلیه مشخصات خواسته شده را برآورده می‌کرد به نام G.728 استاندارد شد. از کاربردهای رایج این کدک می توان انتقال صحبت برروی شبکه‌های انتقال داده به صورت بسته‌ای مانند اینترنت وDSL و کنفرانس ویدئویی را نام برد.

Parameter	CCITT requirements	CCITT objective
Coding Delay	$\leq$ 5 ms	$\leq$ 2 ms
Quality at $BER^* = 0$	Distortion < 4 qdu	
Quality at $BER^* = 10^{-3}$	Not Worse than G.721**	
Quality at $BER^* = 10^{-2}$	Not Worse than G.721	
Tandeming	3 Asynchronous Tandem with Distortion $\leq$ 14 qdu	Synchronous Tandem without Distortion Accumulation
Signalling Tones	DTMF	
Music		No Annoying Effects
At Lower Rates		Graceful Degradation
Complexity		As Low As Possible

جدول(۴-۱)- مشخصات یک کدر 16kbps

کلیات LD-CELP(Low delay Code Excited Linear Prediction):

ویژگی اساسی کدکهای CELP که جستجوی کتاب کد به شیوه آنالیز با ترکیب است در کدک LD-CELP حفظ شده است. اما کدک LD-CELP با استفاده از تطبیق پسرو پیشگویی کننده‌ها و بهره، نیاز به ارسال اطلاعات جانبی مربوط به ضرایب فیلترها و بهره را مرتفع می‌سازد. به علت اینکه در ورودی اینکدر تنها به بافر کردن ۵ نمونه متوالی صحبت احتیاج است؛ تأخیر الگوریتمی کدک 0.625ms است. روزآمد کردن ضرایب فیلتر ترکیب از طریق آنالیز LPC و روزآمد کردن بهره از طریق اطلاعات نهفته در تحریک چندی شده قبلی انجام می‌گیرد. شکل (٤-١)اینکدر و دیکدر LD-CELP را نشان می‌دهد.



شکل (۴-۱) بلوک دیاگرام کدر ودیکدر LD-CELP

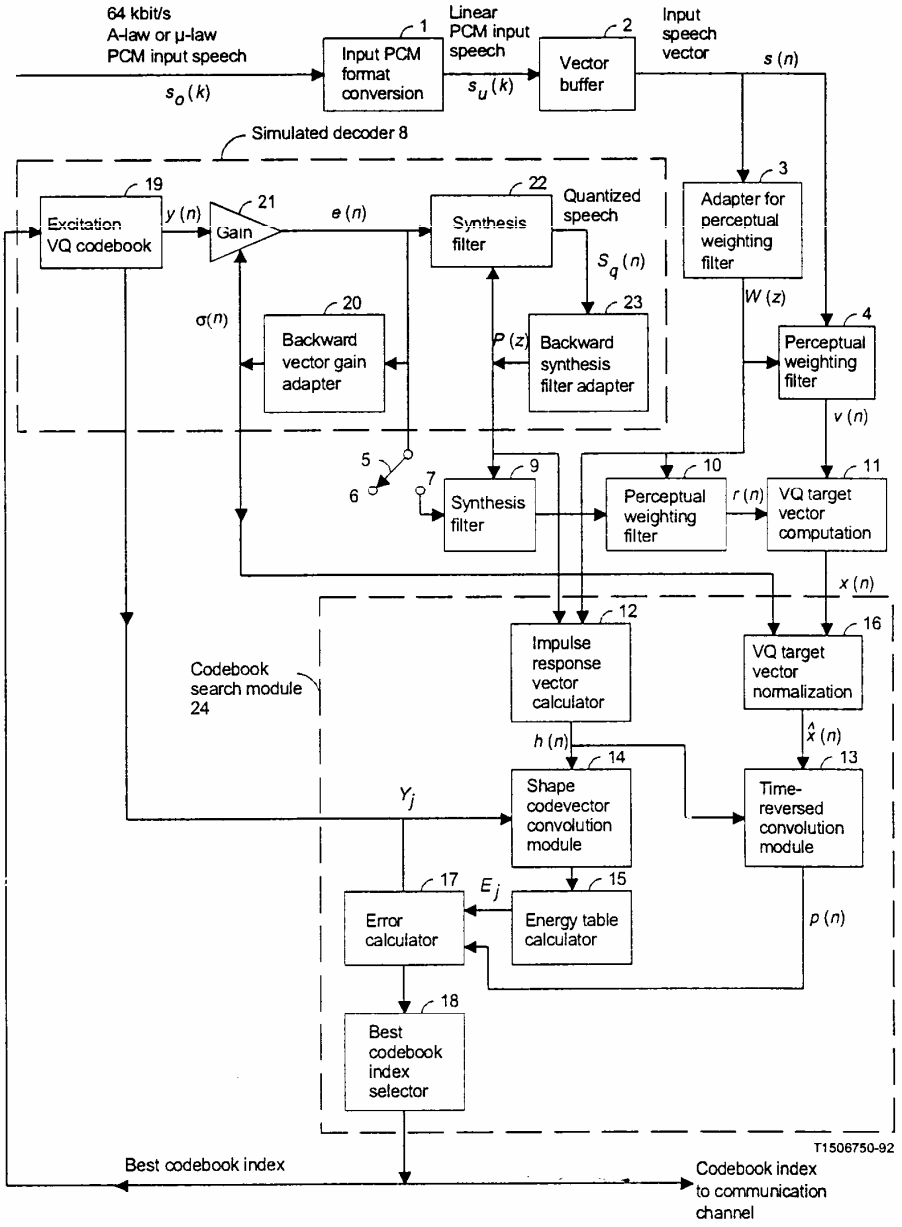
در اینکدر سیگنال ورودی بعد از تبدیل PCM نوع A-Law یا نوعμ-law به PCM یکنواخت، به بلوکهای تشکیل شده از ۵ نمونه متوالی سیگنال تقسیم می‌شود. برای هر بلوک ورودی، اینکدر همه ١٠٢٤ کلمه کد موجود در کتاب کد را از واحد مقیاس دهی بهره و از فیلتر ترکیب عبور می‌دهد. لازم به ذکراست که هر کلمه کد را می‌توان یک بردار ۵ مولفه‌ای تصور کرد. سپس اینکدر از میان این ١٠٢٤ بردار سیگنال چندی شده، یکی را که حداقل متوسط مربع خطای وزن داده شده‌ی فرکانسی

آن نسبت به سیگنال ورودی مینیمم است، انتخاب کرده و اندیس ۱۰ بیتی مربوط به آن را به دیکدر ارسال می‌کند. در این لحظه بردار کد انتخاب شده از واحد مقیاس دهی بهره و فیلتر ترکیب نیز عبور داده می‌شود تا حافظه فیلتر به منظور کد کردن بردار ورودی بعدی آماده باشد. ضرایب فیلتر ترکیب و بهره، به ترتیب از روی سیگنال چندی شده قبلی و سیگنال تحریک مقیاس دهی شده توسط بهره، روزآمد می‌گردند.

با رسیدن هر اندیس ۱۰ بیتی به دیکدر، کلمه کدمتناظر با آن از کتاب کد استخراج و از یک واحد مقیاس دهی بهره و یک فیلتر ترکیب عبور داده می‌شود تا بردار سیگنال دیکدر شده مربوط به آن اندیس حاصل گردد. به منظور افزایش کیفیت شنیداری بردار سیگنال دیکد شده به یک فیلتر پایانی (Postfilter) تطبیقی داده می‌شود. ضرایب فیلتر مذکور به طور متناوب، با استفاده از اطلاعات موجود در دیکدر تعیین می‌شوند. در آخر ۵ مولفه بردار خروجی فیلترپایانی به ۵ نمونه PCM از نوع A-Law یا μ-Law تبدیل می‌شوند.

اینکدر:

شکل(٤-٢)اینکدر LD-CELP را با جزییات بیشتری نشان می‌دهد. اگر چه این بلوک دیاگرام از نظر ریاضی معادل بلوک دیاگرام شکل(٤-١)است، از نظر محاسباتی، برای پیاده سازی کاراتر است.



شکل (۴-۲) بلوک دیاگرام اینکدر LD-CELP

در ادامه این متن:

۱-در باره‌ی هر متغیری که صحبت می‌شود، k اندیس نمونه برداری در نظر گرفته شده و فاصله زمانی نمونه‌ها 125μs است.

۲- یک گروه از ۵ نمونه متوالی یک سیگنال مفروض، یک بردار از آن سیگنال نامیده می‌شود. برای مثال ۵ نمونه متوالی صحبت یک بردار صحبت و ۵ نمونه سیگنال تحریک را یک بردار تحریک را بوجود می‌آورند.

۳- اندیس هر بردار با n نمایش داده می‌شود که با اندیس نمونه برداری k متفاوت است.

٤- چهار بردار متوالی یک دوره تطبیق یا یک فریم را تشکیل می‌دهند.

اندیس بردار کد انتخاب شده توسط اینکدر تنها اطلاعاتی است که به طور صریح به دیکدر ارسال می‌شود و همانطور که گفته شد ۳ نوع دیگر پارامترهای مورد نیاز در دیکدر یعنی بهره، ضرایب فیلتر ترکیب و ضرایب فیلتر وزنی کیفیت به روش تطبیق پسرو از سیگنالهای مقدم بر بردار فعلی سیگنال تعیین می‌گردند. بهره برای هر بردار روزآمد می‌شود در حالیکه ضرایب فیلتر ترکیب و فیلتر وزنی کیفیت در هر دوره تطبیق یا فریم یکبار روزآمد می‌شوند. با آنکه دنباله‌ای که در هر دوره تطبیق پردازش می‌شود از چهار بردار (۲۰ نمونه) تشکیل یافته، اندازه بافر تنها ۵ نمونه است که تأخیر یکطرفه کمتراز 2ms را ممکن می‌سازد.

در زیر هر کدام از بلوکهای اینکدر را شرح می‌دهیم.

- بلوک تبدیل فرمت PCM ورودی:

این بلوک سیگنال ورودی PCM از نوع A-Law یا μ-Law ، $S_o(k)$ را به سیگنال یکنواخت PCM، $S_u(k)$ تبدیل می‌کند.

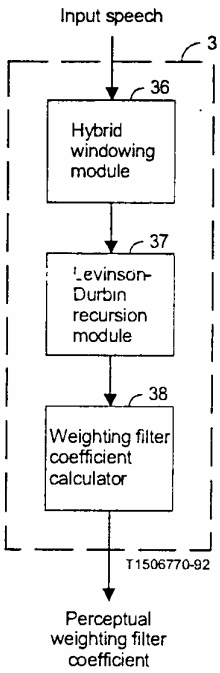
- بلوک بافر:

این بلوک ۵ نمونه متوالی سیگنال صحبت، $S_u(5n)$ ، $S_u(5n+1)$ ، $S_u(5n+2)$ ، $S_u(5n+3)$ ، $S_u(5n+4)$ را ذخیره و بردار پنج بعدی $S(n)=\{S_u(5n),...,S_u(5n+4)\}$ را تشکیل می‌دهد.

- بلوک تطبیق دهنده فیلتر وزنی کیفیت:

این بلوک ضرایب فیلتر وزنی کیفیت را از روی سیگنال چندی نشده ورودی و با آنالیز LPC تعیین می‌کند. روزآمد کردن این ضرایب یکبار در هر دوره تطبیق و در زمان بردار سوم دوره انجام می‌شود. در فاصله زمانی بین روزآمد شدنها این ضرایب ثابت هستند.

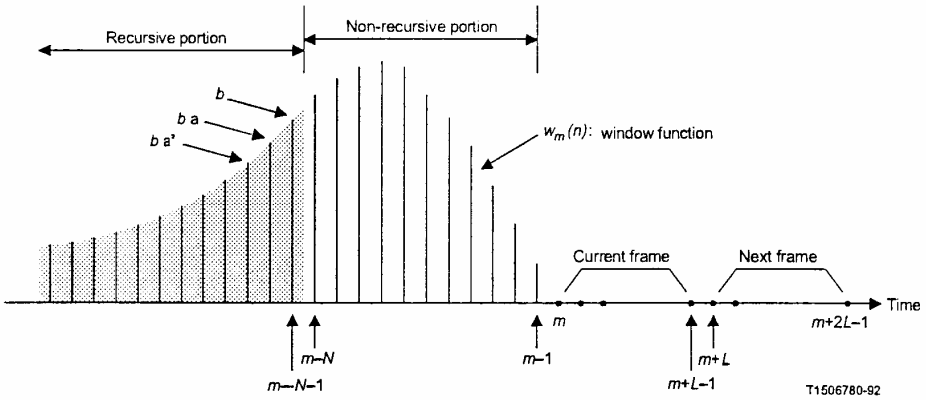
شکل(٤-۳) زیر بخشهای این بلوک را نشان می‌دهد.



شکل (۳-۴) بلوک تطبیق دهنده فیلتر وزنی کیفیت

ابتدا بردار صحبت ورودی وارد یک مدول پنجره‌گر مرکب می‌شود که پنجره‌ای را بر بردارهای قبلی صحبت قرار می‌دهد و ۱۱ ضریب اتوکرولیشن سیگنال پنجره شده را در خروجی بدست می‌دهد. اکنون مدول بازگشتی، Levinson-Durbin، این ضرایب اتوکرولیشن را به ضرایب پیشگویی کننده تبدیل و مدول محاسبه گر ضرایب فیلتر وزنی کیفیت را با استفاده از آنها، محاسبه می‌کند. در زیر جزییات عملکرد هرکدام از این مدولها آمده است.

اگر فرض کنیم که آنالیز LPC یکبار برای هر L نمونه سیگنال انجام می‌شود یا به عبارت دیگر دوره تطبیق کنونی از نمونه‌های $S_u(m)$، $S_u(m+1)$،... $S_u(m+L-1)$ تشکیل یافته باشد، پنجره باید به تمام نمونه‌های سیگنال که اندیس آنها از m کوچکتر است اعمال گردد. پنجره مورد استفاده دارای دو بخش بازگشتی و غیر بازگشتی است. اگر N نمونه در بخش غیر بازگشتی پنجره موجود باشد، نمونه‌های $S_u(m-1)$، $S_u(m-2)$،...، $S_u(m-N)$ بوسیله بخش غیر بازگشتی پنجره و نمونه‌های واقع در سمت چپ $S_u(m-N-1)$ بوسیله بخش بازگشتی پنجره وزن داده می‌شوند. شکل ٤-٤ پنجره و بخشهای بازگشتی و غیر بازگشتی و وضعیت قرار گرفتن پنجره نسبت به دوره تطبیق کنونی نمایش می‌دهد.



شکل(۴-۴) بخشهای مختلف پنجره هایبرید

در زمان m، تابع پنجره مرکب $W_m(k)$ به صورت

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

تعریف می‌شود و سیگنال پنجره شده برابر با:

$S_m(k)=S_u(k)W_m(k)$

است. برای یک آنالیز LPC از مرتبه M، باید M+1 ضریب اتوکرولیشن یعنی $R_m(i)$ را برای i=0,1,2,…,M محاسبه کرد. i امین ضریب اتوکرولیشن برای دوره تطبیق کنونی می‌تواند به صورت

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

که در آن

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

است. $r_m(i)$ **جز** بازگشتی $R_m(i)$ و باقیمانده جز غیر بازگشتی آن است. جمع محدود بخش غیر بازگشتی برای هر دوره تطبیق محاسبه می‌گردد. در حالیکه $r_m(i)$ به صورت بازگشتی محاسبه می‌شود.

اگر فرض کنیم که تمام $r_m(i)$ های دوره تطبیق کنونی را محاسبه و ذخیره کرده‌ایم و می‌خواهیم به دوره تطبیق بعدی برویم که از نمونه $S_u(m+L)$ آغاز می‌شود. بعد از اینکه پنجر L نمونه به سمت راست انتقال داده شد، سیگنال جدید پنجره شده برای دوره تطبیق بعدی

$$S_{m+L}(k)=S_u(k)W_{m+L}(k)=\begin{cases} S_u(k)f_m(k)\alpha^L & k\le m+L-N-1 \\ -S_u(k)Sin[c(k-m-L)] & m+L-N\le k\le m+L-1 \\ 0 & k\ge m+L \end{cases}$$

خواهد بود. جزء بازگشتی $R^{(i)}_{m+L}$ به صورت زیر می‌تواند نوشته شود.

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

بنابراین می توان $r_{m+L}(i)$ را به صورت بازگشتی از روی $r_m(i)$ محاسبه کرد. سپس ضریب اتوکرولیشن $R_{m+L}(i)$ مطابق با رابطه زیر تعیین می‌گردد.

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

پارامترهای مدول پنجره مرکب در شکل(٤-٤)،

$$M = 10, L = 20, N = 30 \alpha = \left(\frac{1}{2}\right)^{1/40}$$

هستند.

هنگامی که روند محاسبه ۱۱ ضریب اتوکرولیشن به پایان رسید، عمل اصلاح نویز سفید(White Noise Correction) انجام می‌شود. این کار با افزایش اندک انرژی $R(o)$ صورت می‌پذیرد.

$$R(o) \leftarrow \left(\frac{257}{256}\right) R(o)$$

اصلاح نویز سفید برای پر کردن فرورفتگیهای موجود در طیف به منظور کاهش بازه تغییر آن و بهبود عملکرد فرآیند بازگشتی Levinson-Durbin. در مدول بعدی اعمال می‌شود.

اکنون با استفاده از ضرایب اتوکرولیشن اصلاح شده، مدول Levinson-.Durbin ضرایب پیشگویی کننده از مرتبه یکم تا مرتبه دهم به شیوه بازگشتی بدست می آید. اگر $a_j^{(i)}$ ، jامین ضریب پیشگویی کننده مرتبه i ام باشد آنگاه فرآیند بازگشتی Levinson-.Durbin با روابط زیر می‌تواند توصیف شود.

$$E(o) = R(o)$$

$$k_i = -\frac{R(i) + \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)}{E(i-1)}$$

$$a_i^{(i)} = k_i$$

$$a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)} \qquad 1 \le j \le i-1$$

$$E(i) = (1 - k_i^2)E(i-1)$$

بعد از اینکه چهار معادله اخیر برای i=1,2,...,10 ارزیابی شدند، جواب نهایی

$$q_i = a_i^{(10)} \qquad 1 \le i \le 10$$

خواهد بود. بعد از این مرحله مدول محاسبه‌گر ضرایب فیلتر وزنی کیفیت ضرایب این فیلتر را مطابق با روابط زیر محاسبه می‌کند.

ضرایب صورت تابع تبدیل فیلتر $= NUM_i = q_i \gamma_1^i = q_i (0.6)^i$

ضرایب مخرج تابع تبدیل فیلتر $= DENUM_i = q_i \gamma_2^i = q_i (0.9)^i$

- فیلتر وزنی کیفیت:

فیلتر وزنی کیفیت استفاده شده در LD-CELP به شکل کلی فیلترهای شکل دهنده نویز بسیار شبیه است. این فیلتر شکل طیف سیگنال خطا را به گونه ای که اثر شنیداری آن کاهش یابد تغییر می دهد. تابع تبدیل این فیلتر به صورت

$$W(z) = \frac{1 + \sum_{i=1}^{10} NUM_i z^{-i}}{1 + \sum_{i=1}^{10} DENUM_i z^{-i}}$$

است که NUM_i و $DENUM_i$ به همان نحوی که در بالا ذکر شد، بوسیله پیشگویی کننده این فیلتر برای هر دوره تطبیق محاسبه و جایگزین می‌شوند. در اینکدر ۲ فیلتر وزنی کیفیت، بلوکهای ٤ و ۱۰، به کار رفته است. بردار صحبت ورودی، $S(n)$، از این فیلتر عبور داده می‌شود و نتیجه بردار صحبت وزن داده شده، $V(n)$ ، است. باید توجه داشت که به جز در آغاز پیاده سازی الگوریتم و مقداردهی اولیه، حافظه فیلتر را (مقادیر نگهداری شده در واحدهای تأخیر فیلتر) نباید صفر کرد.

آزمایشهای ITU-T نشان می‌دهد که برای سیگنالهای غیر صحبت مطلوب است که فیلتر وزنی کیفیت غیر فعال شود یعنی $W(z)=1$. این امر با صفر قراردادن γ_1 و γ_2 در روند محاسبه ضرایب فیلتر مقدور است.

- فیلتر ترکیب:

همانطور که در شکل (٤-۲) نشان داده شده است، دو فیلتر ترکیب، بلوکهای ۹ و ۲۲، در استاندارد وجود دارند که ضرایب آنها کاملاً یکسان هستند. هر دو فیلتر بوسیله تطبیق دهنده پسرو فیلتر ترکیب، بلوک ۲۳، روزآمد می‌شوند و هر کدام از این فیلترهای ترکیب، فیلترهای تمام قطب مرتبه ۵۰ هستند که از یک حلقه فیدبک با یک پیشگویی کننده LPC مرتبه ۵۰ در شاخه فیدبک تشکیل یافته‌اند. تابع تبدیل این فیلترها

$$F(z) = \frac{1}{1 - P(z)}$$

است که در آن $P(z)$ تابع تبدیل یک پیشگویی کننده مرتبه ۵۰ است.

بعد از آنکه بردار صحبت وزن داده شده، $V(n)$، بدست آمد پاسخ ورودی صفر، $r(n)$ ، فیلتر وزنی کیفیت و فیلتر ترکیب محاسبه می‌شود. برای انجام این کار کلید ۵ را باز کرده یعنی آن را به گره ٦ متصل می‌کنیم. به این ترتیب سیگنالی که از گره 7 به فیلتر ۹ میرود صفر خواهد بود. اکنون عملیات فیلترکردن را برای مدت ۵ نمونه با ورودی صفر انجام داده و خروجی فیلتر وزنی کیفیت پاسخ ورودی صفر مطلوب است.

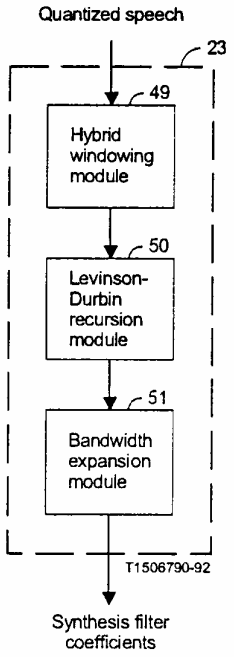
باید توجه داشت که به استثنای بردار صحبت بلافاصله بعد از مقدار دهی اولیه حافظه فیلترهای ۹ و ۱۰ به طور کلی دارای حافظه غیر صفر می‌باشند. بنابراین $r(n)$ هم در حالت کلی غیر صفر است، اگر چه ورودی صفر باشد. در حقیقت $r(n)$ پاسخ فیلترهای ۹ و ۱۰ به بردارهای تحریک قبلی مقیاس دهی شده با بهره است.

- بلوک محاسبه‌گر سیگنال خطا:

این بلوک بردار پاسخ ورودی صفر r(n) را از بردار صحبت وزن داده شده v(n) کم می‌کند حاصل برداری است که برای جستجوی کتاب کد در چندی کردن برداری استفاده می‌گردد.

- تطبیق دهنده پسرو فیلتر ترکیب:

وظیفه این بلوک روزآمد کردن ضرایب فیلترهای ترکیب ۲۲ و ۹ است. این بلوک صحبت چندی شده (سنتز شده) را در ورودی گرفته و مجموعه‌ای از ضرایب فیلتر ترکیب را در خروجی تولید می‌کند. در شکل ٤-۵ مدولهای داخلی این بلوک نشان داده شده است.



شکل ۴-۵- بلوک تطبیق دهنده پسرو فیلتر ترکیب

عملکرد مدولهای Hybridwind.49 و Levinson- Durbin 50 به جز در موارد زیر، به عملکرد مدولهای همنامشان در تطبیق دهنده فیلتر وزنی کیفیت کاملاً شباهت دارد.

۱- سیگنال ورودی، به جای صحبت چندی نشده، سیگنال صحبت چندی شده است.

۲- مرتبه Levinson-Durbin، به جای مرتبه ۱۰ ، از مرتبه ۵۰ است.

۳- پارامترهای پنجره مرکب متفاوت است. در مدول ٤۹، N=۳۵ و $\alpha = \left(\frac{3}{4}\right)^{1/40}$ هستند.

به منظور بهبود عملکرد در برابر خطاهای کانال، ضرایب محاسبه شده در مدول Levinson-.Durbin 50 تغییر داده می‌شوند تا پهنای باند قله‌های طیف فرکانسی LPC حاصل، قدری عریض تر شوند. این عمل را مدول گسترش پهنای باند ۵۱ مطابق با رابطه زیر انجام می‌دهد.

$$a_i = \lambda^i \hat{a}_i = \left(\frac{253}{256}\right)^i \hat{a}_i \quad i = 1.2,...,50$$

بعد از انجام گسترش پهنای باند، پیشگویی کننده اصلاح شده تابع تبدیل

$$P(z) = -\sum_{i=1}^{50} a_i z^{-i}$$

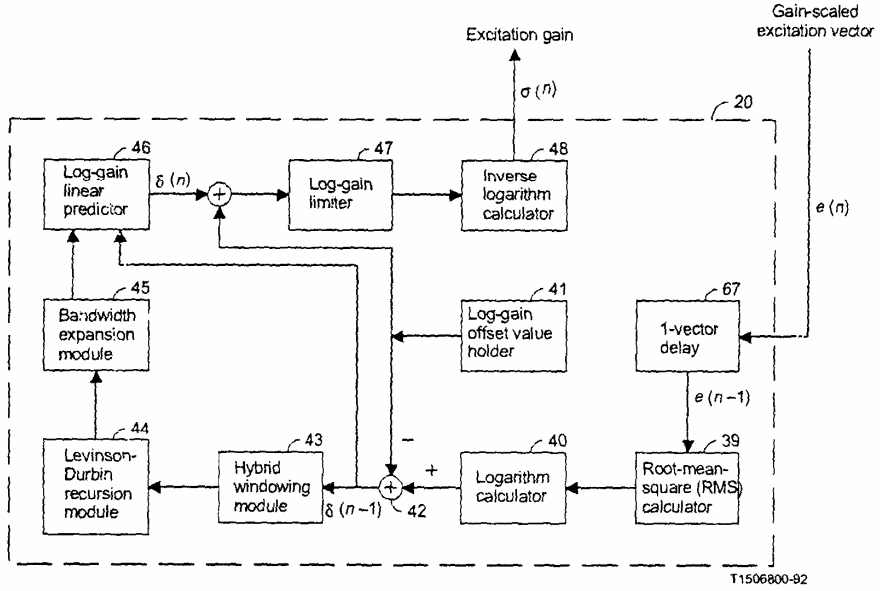
را خواهد داشت. ضرایب اصلاح شده به بلوکهای ۲۲ و ۹ و هم چنین به بلوک ۱۲ برای محاسبه پاسخ ضربه در فرآیند جستجوی اندیس بهترین بردار کد داده می‌شود. همانگونه که ذکر شد تابع تبدیل فیلترهای ترکیب به صورت

$$F(z) = \frac{1}{1 - P(z)} = \frac{1}{1 + \sum_{i=1}^{50} a_i z^{-i}}$$

است.

- تطبیق دهنده پسرو بهره:

تطبیق دهنده پسرو بهره، بلوک ۲۰، بردار تحریک مقیاس دهی شده با بهره، $e(n)$ ، را در ورودی می‌گیرد و بهره بردار تحریک، $\sigma(n)$، را در خروجی تولید می‌کند. اساساً این بلوک، بهره بردار $e(n)$ را از روی بهره بردارهای $e(n-1)$ و $e(n-2)$ و... با استفاده از پیشگویی خطی تطبیقی در حوزه لگاریتمی محاسبه می‌کند. شکل ٤-٦ جزییات این بلوک را نشان می‌دهد.



شکل ۴-۶- بلوک تطبیق دهنده پسرو بهره

- ساختار کتاب کد:

مدول جستجوی کتاب کد، بلوکهای ۱۲ تا ۱۸ را شامل می‌شود. این مدول ۱۰۲٤ بردار کد موجود در کتاب را جستجو می‌کند. و اندیس بهترین برداری را که نزدیک ترین بردار صحبت چندی شده به بردار صحبت ورودی است، مشخص می‌کند. برای کاهش پیچیدگی عملیات جستجو کتاب کد ۱۰۲٤ تایی به دو کتاب کد کوچکتر تقسیم می‌شود یکی از آنها کتاب کد

شکل(Shape code book) است که دارای ۱۲۸ بردار کد مستقل است و اندیس هر کدام با یک عدد باینری ۷ بیتی نمایش داده می‌شود و دیگری کتاب کد بهره(Gain code book) که از ۸ عدد اسکالر که قرینه هم هستند تشکیل یافته است. بدیهی است که هر کدام از این اعداد بایک اندیس ۳ بیتی می‌توانند نمایش داده شوند که ۲ بیت برای دامنه و بیت سوم مربوط به علامت است. بردار کد مطلوب از ضرب بهترین کلمه کد شکل انتخابی و بهترین سطح بهره انتخابی بدست می‌آید.

- اصول جستجوی کتاب کد:

ابتدا مدول جسجو هر کدام از ۱۰۲٤ کلمه کد را توسط بهره تحریک کنونی، $\delta(n)$، مقیاس دهی می‌کند. بعد آنها را یک به یک از فیلتر با تابع تبدیل $H(z)=F(z)W(z)$ که ترکیب سری فیلتر ترکیب و فیلتر وزنی کیفیت است، عبور می‌دهد. حافظه فیلتر هر بار که کلمه کد جدیدی به آن داده می‌شود با صفر مقدار دهی می‌شود.

فیلتر کردن بردار کدهای موجود در کتاب کد می‌تواند با ضرب ماتریسی بیان شود. اگر γ_j ، j امین بردار کد موجود در کتاب کد شکل، g_i، i امین سطح بهره، $\delta(n)$ بهره تحریک فعلی و $\{h(n)\}$ دنباله پاسخ ضربه فیلتر سری $H(z)$ باشد آنگاه خروجی فیلتر

$$\overline{x_{ij}} = \underline{H}\delta(n)g_i y_j$$

خواهد بود که در آن

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

است. در واقع مدول جستجوی کتاب کد بهترین ترکیب i و j را که خطای زیر را مینی‌مم می‌کند، را مشخص می‌نماید.

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

که $\hat{x}(n) = \dfrac{x(n)}{\sigma(n)}$ نرمالیزه شده بردار خطا است. با بسط سمت راست معادله بالا عبارت زیر بدست می‌آید.

$$D = \sigma^2(n)[\| \hat{x}(n) \|^2 - 2g_i \hat{x}^T(n)\underline{H}y_j + g_i^2 \| \underline{H}y_i \|^2]$$

از آنجایی که جملات $\| \hat{x}(n) \|^2$ و $\delta^2(n)$ در هنگام جستجو ثابت هستند. مینی‌مم کردن D معادل مینی‌مم کردن

$$= -2g_i p^T(n) y_i + g_i^2 E_j \quad \hat{D}$$

است که در آن

$$P(n) = \underline{H}^T \hat{x}(n)$$

و

$$E_j = \| \underline{H} y_j \|^2$$

هستند. E_j انرژی j امین بردار کد شکل فیلتر شده و مستقل از بردار اختلاف نرمالیزه شده، $\hat{x}(n)$، است. از طرفی y_j ثابت هستند. بنابراین E_j ها نیز در یک دوره تطبیق ثابت هستند و بعد از روزآمد شدن فیلترها می‌توان E_jهای مربوط به دوره تطبیق فعلی را محاسبه و ذخیره نمود. این کار به کاهش پیچیدگی محاسبات کمک می‌کند.

همچنین دو آرایه

$b_i = 2g_i$

و

$c_i = g_i^2$

به علت ثابت بودن g_i می‌توانند از قبل محاسبه و ذخیره شوند.

اکنون $\hat{D}$ را می‌توان به صورت

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

بیان کرد که در آن

$$P_j = p^T(n) y_i$$

لازم به ذکر است که هنگامی که جداول E_j ، b_i و c_i محاسبه شوند، به منظور تعیین $\hat{D}$ تنها به محاسبه $P_j = p^T(n) y_j$ احتیاج است. $P_j = p^T(n) y_j$، در یک دوره تطبیق، فقط وابسته به j است. به این ترتیب فرآیند جستجو برای هر بردار کد شکل i اندیس بهترین بهره را مشخص می‌کند. هنگامی که i , j تعیین شدند دنبال هم قرار گرفته و خروجی مدل جستجو که یک اندیس ۱۰ بیتی است را تشکیل می‌دهند.

- شبیه سازی دیکدر در اینکدر:

اگر چه اینکدر بهترین اندیس کتاب کد را انتخاب و به دیکدر ارسال کرده است در این مرحله چند کار دیگر به منظور آماده‌سازی اینکدر برای کد کردن بردارهای ورودی بعدی باید انجام گیرد. ابتدا اندیس تعیین شده به کتاب کد تحریک، بلوک ۱۹، داده می‌شود تا بهترین بردار کد متناظر یعنی $y(n) = g_{i_{min}} y_{j_{min}}$ از آن استخراج گردد. این بردار توسط بلوک ۲۱ با بهره تحریک کنونی، $\delta(n)$ ، مقیاس دهی شده سپس از فیلتر ترکیب ۲۲ به منظور بدست آوردن بردار صحبت چندی شده فعلی، $S_q(n)$ ، عبور داده می‌شود. باید توجه داشت که بلوکهای ۱۹ تا ۲۳ ، دیکدر را شبیه سازی می‌کنند و با فرض عدم وجود خطا در انتقال، $S_q(n)$ ، همان بردار صحبت چندی شده‌ای است که در دیکدر برای اندیس ارسالی بدست خواهد آمد. تطبیق دهنده پسرو فیلتر ترکیب از بردار $S_q(n)$ برای روزآمد کردن ضرایب این فیلتر استفاده می‌کند. به طریق مشابه، تطبیق دهنده بهره ، بردار تحریک مقیاس دهی شده، $e(n)$ ، را در روزآمد کردن ضرایب پیشگویی کننده خطی بهره به کار می‌برد.

کاردیگری که قبل از کدکردن بردار صحبت بعدی باید انجام گیرد، روزآمد کردن حافظه فیلترهای ترکیب، بلوک ۹، و فیلتر وزنی کیفیت، بلوک ۱۰ ، است. به این منظور ابتدا حافظه فیلترها را که از عملیات محاسبه پاسخ ورودی صفر به جا مانده است، ذخیره کرده و سپس حافظه این دو فیلتر برابر صفر قرار داده می‌شود. اکنون کلید ۵ بسته می‌شود، یعنی به گره ۷ متصل می‌شود و بردار تحریک مقیاس دهی شده، $e(n)$ ، از دو فیلتر می‌گذرد. چون $e(n)$ تنها ۵ مولفه دارد و حافظه فیلترها نیز صفر است، تعداد ضرب و جمعهای لازم برای یک دوره تناوب ۵ نمونه‌ای تنها از صفر تا ٤ می‌رود. این امر باعث کاهش زیادی در پیچیدگی می‌شود، زیرا اگر حافظه فیلترها غیر صفر باشد تعداد ضرب و جمعهای لازم برای هر نمونه ۷۰ خواهد شد. اکنون حافظه‌ای را که ذخیره کرده بودیم به حافظه ناشی از فیلتر کردن $e(n)$ اضافه می‌شود. اکنون حافظه فیلترها برای محاسبه پاسخ ورودی صفر در کدینگ بردار صحبت بعدی آماده است.

پس از روزآمد کردن حافظه، ۵ عنصر بالای حافظه فیلتر ترکیب ۹، دقیقاً برابر با بردار صحبت چندی شده، $S_q(n)$ ، است. بنابراین می‌توان بلوک ۲۲ را حذف کرد و $S_q(n)$ را از حافظه روزآمد شده فیلتر ترکیب، بلوک ۹ بدست آورد. این به معنای صرفه جویی ۵۰ ضرب و جمع دیگر برای هر نمونه است.

عملیات توضیح داده شده در بالا، کارهایی را که برای کدکردن یک بردار صحبت لازم است، بیان می‌کند. برای کد کردن کل شکل موج صحبت باید عملیات فوق برای هر بردار صحبت تکرار شود.

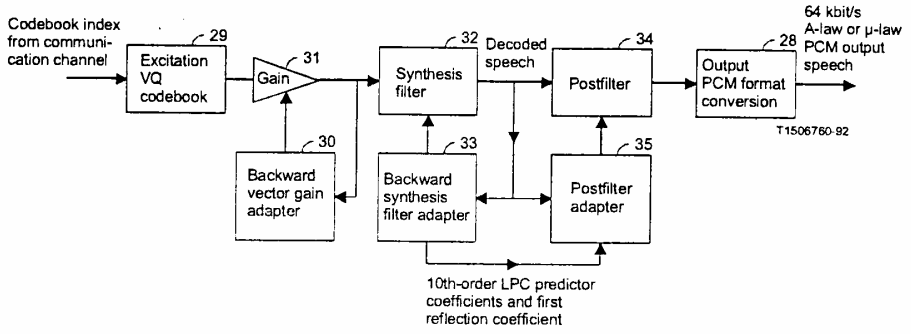
- همزمانی و سیگنالینگ:

در توصیف عملیات اینکدر در بالا فرض شده بود که دیکدر مرزهای اندیسهای دریافت شده را می‌داند و هم چنین می‌داند که در چه هنگامی روزآمد کردن فیلترهای ترکیب و پیشگویی کننده خطی بهره باید صورت پذیرد. در عمل این اطلاعات می‌تواند با اضافه کردن بیتهای بیشتری، یعنی با نرخ بیت بالاتر از 16kbps، صورت پذیرد. اما در بسیاری از کاربردها لازم ا ست که بیتهای همزمانی یا علامت دهی (سیگنالینگ) بخشی از همان نرخ بیت 16kbps باشد. این امر به صورت زیر می‌تواند محقق شود.

اگر فرض شود که برای هر N بردار صحبت یک بیت همزمانی باید استفاده گردد. بنابراین برای هر N امین بردار صحبت تنها نیمی از کتاب کد شکل را جستجو کرده و یک اندیس بردار کد شکل ٦ بیتی را تولید کرد. به این ترتیب به ازای هر N بردار صحبت ورودی یک بیت برای همزمانی یا علامت دهی در اختیار داریم. البته اینکدر باید بداند که برای N امین بردار صحبت تنها نیمی از کتاب کد شکل را جستجو کند و از اندیس کد N امین بردار صحبت یک بیت کاسته شود.

دیکدر:

در شکل ٤-٧بلوک دیاگرام کاملی از دیکدر آمده است. در ذیل عملکرد هر کدام از بلوکها توصیف می‌شود.



شکل ۴-۷- بلوک دیاگرام دیکدر

- کتاب کد تحریک:

این بلوک حاوی یک کتاب کد (تشکیل یافته از کتاب کدهای شکل و بهره) کاملاً یکسان با بلوک ۱۹ در اینکدر است. این بلوک اندیس بهترین بردار کد را دریافت و بردار کد متناظر با آن را که در اینکدر انتخاب شده بود در خروجی تولید می‌کند.

- مقیاس دهی بهره:

این بلوک بردار تحریک مقیاس دهی شده را با ضرب هرکدام از مولفه‌های بردار، y(n)،در بهره،σ(n) ،محاسبه می‌کند.

- فیلتر ترکیب دیکدر:

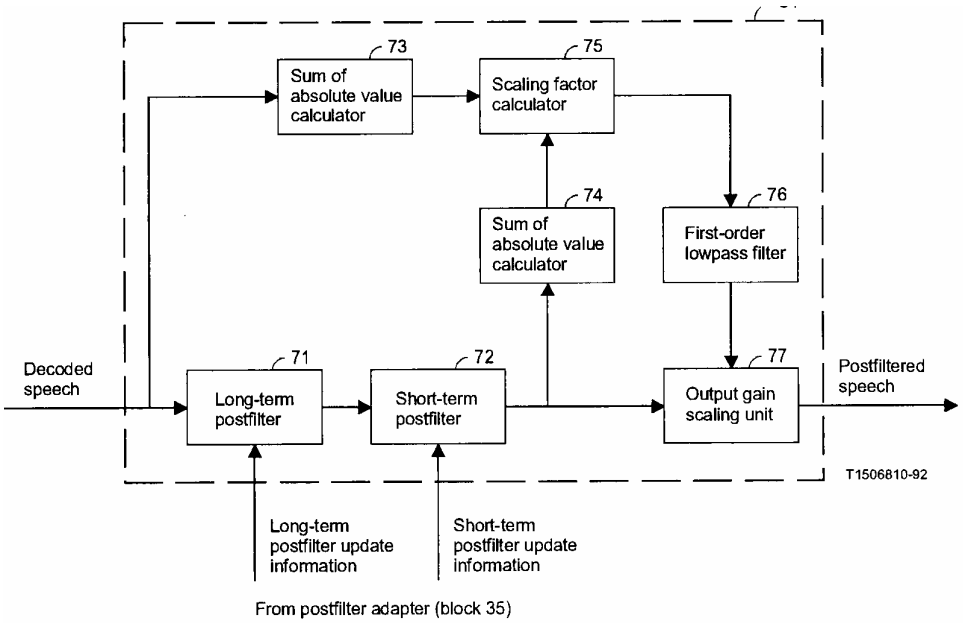
این فیلتر با فرض عدم رویداد خطا در انتقال، دارای همان تابع تبدیل فیلتر ترکیب در اینکدراست. این بلوک با فیلتر کردن بردار تحریک مقیاس دهی شده با بهره، بردار صحبت چندی شده، $s_d(n)$، را تولید می‌کند. لازم به ذکر است که به منظور اجتناب از هر انباشتگی خطاهای گرد کردن در فرایند دیکد کردن، گاهی مطلوبست که دقیقاً فرایندی را که برای تولید $S_q(n)$ در دیکدر انجام داده شده، در اینجا نیز صورت پذیرد. به این معنا که اگر در اینکدر $S_q(n)$ از روزآمد کردن حافظه فیلتر ترکیب ۹ بدست آمده است، آنگاه دیکدر نیز $S_d(n)$ را باید با جمع کردن پاسخ ورودی - صفر و پاسخ حالت- صفر فیلتر ترکیب ۳۲ محاسبه کند.

- تطبیق دهنده‌های پسرو فیلتر ترکیب و بهره:

عملکرد این دو بلوک کاملاً شبیه به بلوکهای همنامشان در اینکدر می‌باشد.

-فیلتر پایانی:

این فیلتر صحبت دیکد شده را به منظور افزایش کیفیت شنیداری فیلتر می‌کند. اجزاء داخلی این بلوک در شکل ۴-۸ نشان داده شده است. فیلتر پایانی از ۳ بخش اصلی تشکیل یافته است. این ۳ بخش فیلتر پایانی بلند- مدت ۷۱، فیلتر پایانی کوتاه - مدت ۷۲ و واحد بهره خروجی ۷۷ هستند. چهار مدول دیگر موجود در شکل بهره را برای استفاده در بلوک ۷۷ محاسبه می‌کنند.



شکل۴-۸- فیلتر پایانی

فیلتر پایانی بلند - مدت ۷۱ که گاهی فیلتر پایانی pitch نیز نامیده می‌شود، یک فیلتر شانه‌ای است که قله‌های طیف در ضریبهایی از فرکانس اصلی (یا فرکانس pitch) واقع شده‌اند. معکوس فرکانس pitch ، دوره تناوب pitch نام دارد. دوره تناوب pitch می‌تواند از روی صحبت دیکد شده و بوسیله یک تشخیص دهنده pitch استخراج گردد. اگر p دوره تناوب pitch (بر حسب تعداد نمونه) باشد، تابع تبدیل فیلتر پایانی بلند - مدت به صورت

$$H_L(z) = g_L(1 + b\,z^{-p})$$

است که پارامترهای g_L ، b و p یکبار در هر دوره تطبیق، روزآمد می‌گردند. این کار در زمان سومین بردار دوره تطبیق صورت می‌پذیرد.

فیلتر پایانی کوتاه- مدت ۷۲ از ترکیب متوالی یک فیلتر IIR مرتبه دهم و یک فیلتر FIR مرتبه اول تشکیل یافته است. فیلتر IIR مرتبه ۱۰، مولفه های فرکانسی بین فرکانسهای فرمنت را تضعیف می‌کند. در حالیکه فیلتر FIR مرتبه اول سعی در جبران tilt فرکانسی در پاسخ فرکانسی فیلتر IIR دارد.

اگر فرض شود که $\tilde{a}_i$ $i=1,2,...,10$ ضرایب پیشگویی کننده مرتبه دهم LPC محاسبه شده توسط آنالیز LPC بر روی صحبت دیکد شده باشد و k_L اولین ضریب بازگشت بدست آمده با همان آنالیز باشد. آنگاه $\tilde{a}_i$ و k_L را می‌توان به عنوان بخشی از اطلاعات جانبی فراهم آمده توسط آنالیز LPC مرتبه ۵۰ ، بلوک ۵۰ در شکل (٤-٥) به شمار آورد. تنها کاری که باید انجام گیرد اینست که روند تکرار الگوریتم Levinson Durbin.50 را در مرتبه دهم متوقف کرده و $\tilde{a}_i$ و k_L را کپی کرده و Levinson-Durbin50 را از مرتبه ۱۱ تا ۵۰ ادامه داد.

تابع تبدیل فیلتر پایانی کوتاه - مدت

$$H_s(z) = \frac{1 - \sum_{i=1}^{10} \bar{b}_i z^{-i}}{1 - \sum_{i=1}^{10} \bar{a}_i z^{-i}} [1 + \mu z^{-1}]$$

است که درآن

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

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

و

$$\mu = (0.15) k_L$$

است. ضرایب $\tilde{a}_i$، $\bar{b}_i$ و μ نیز یکبار در هر دوره تطبیق روزآمد می‌شود که این امر در اولین بردار فریم انجام می‌شود، یعنی به محض اینکه $\tilde{a}_i$ محاسبه شد.

به طور کلی هنگامی که صحبت دیکد شده از فیلترهای فیلتر پایانی کوتاه- مدت و بلند - مدت عبور می‌کند، دیگر صحبت فیلتر شده همان سطح توان صحبت دیکد شده فیلتر نشده را ندارد. برای جلوگیری از کاهش زیاد در توان سیگنال لازم است که از فرآیند اتوماتیک کنترل بهره استفاده شود. این کار را بلوکهای ۷۳ تا ۷۷ انجام می‌دهند.

بلوک ۷۳، بردار دیکد شده صحبت، $S_d(n)$، را گرفته و مجموع قدرمطلق مولفه‌های آن را محاسبه می‌کند. بلوک ۷٤ نیز دقیقاً همین عمل را برای بردار خروجی فیلتر پایانی کوتاه- مدت، $S_f(n)$ ، انجام می‌دهد. محاسبه گربهره ۰٫۷۵ خروجی بلوک ۷۳ را بر خروجی بلوک ۷٤ تقسیم کرده تا بهره بردار خروجی بدست آید. تابع تبدیل فیلتر پایین گذر مرتبه اول بلوک ۷٦ به صورت

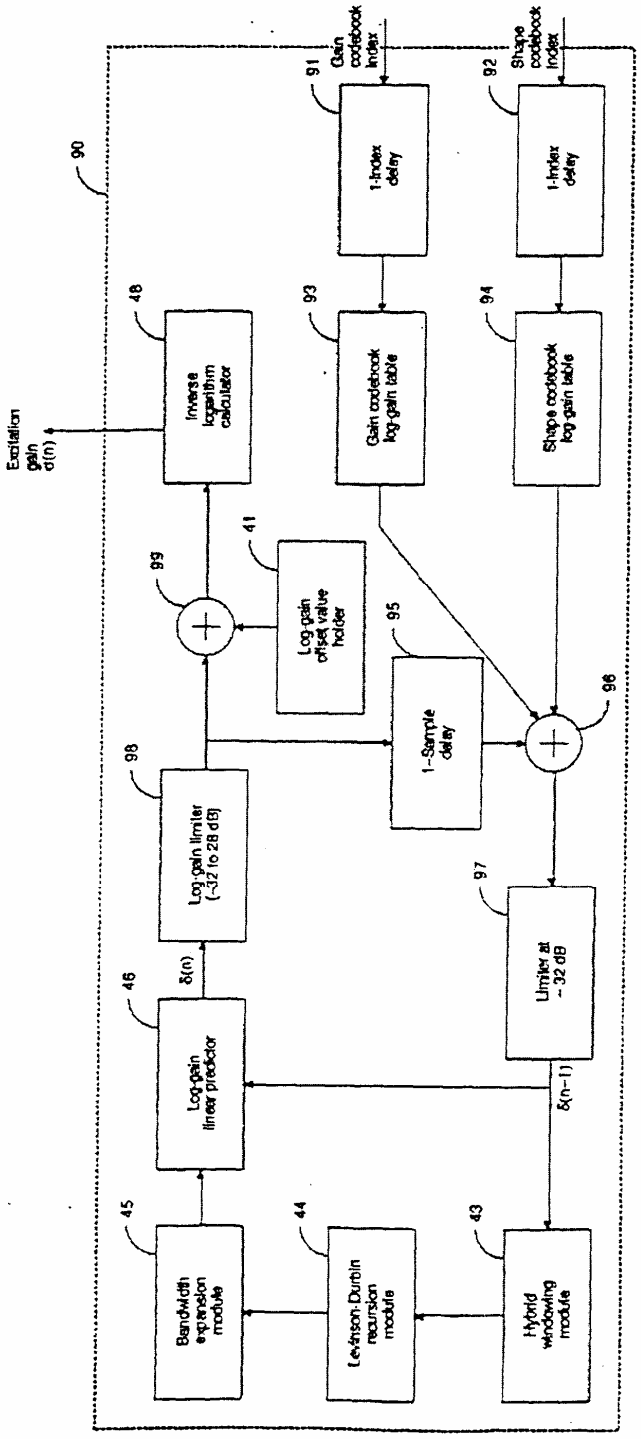
$$\frac{0.01}{1 - 0.99 z^{-1}}$$

است. خروجی بلوک ۷۵ به بلوک ۷۶ داده می‌شود و سپس بهره فیلتر شده برای مقیاس دهی نمونه به نمونه خروجی پست فیلتر کوتاه- مدت، $S_f(n)$، استفاده می‌شود. اگر از بلوک ۷۶ استفاده نشود هر ۵ نمونه $S_f(n)$ با یک بهره مقیاس دهی شده و این امر موجب ایجاد اثر پله‌ای در صحبت خروجی می‌شود.

آزمایشهای ITU-T نشان داده‌اند که عملکرد کدر برای سیگنالهای غیر صحبت هنگامی که فیلتر پایانی غیر فعال باشد بهبود می‌یابد.

- تطبیق دهنده فیلتر پایانی:

وظیفه این بلوک محاسبه‌ي ضرایب فیلترهای پایانی کوتاه- مدت و بلند- مدت و دوره تناوب pithc از روی ضرایب پیشگویی کننده مرتبه دهم و ضریب بازگشت، k_L ، است . شکل۴-۹ جزییات بیشتری از این بلوک را نشان می‌دهد.


90
Excitation gain σ(n)
Gain codebook index
Shape codebook index
91
92
1-index delay
1-index delay
48
93
94
Inverse logarithm calculator
Gain codebook log-gain table
Shape codebook log-gain table
41
99
Log-gain offset value holder
95
96
1-Sample delay
98
Log-gain limiter (-32 to 28 dB)
97
Limiter at - 32 dB
46
σ(n)
Log-gain linear predictor
σ(n-1)
45
44
43
Bandwidth expansion module
Levinson-Durbin recursion module
Hybrid windowing module

شکل ۴-۹

خروجی بلوک ۹۷ دراین شکل از نظر ریاضی معادل با خروجی بلوک ٤٢ درشکل ٤-٦ است. به همین دلیل بلوکهای ٤٣ تا ٤٦ کاملاً شبیه به بلوکهای همنوعشان در شکل ٤-٦ است. عملکرد محدود کننده بهره ۹۸ نیز شبیه به عملکرد بلوک ٤٧ است. اما بازه مجاز بهره به جای 0 تا ٦٠ دسی بل از ٣٢- تا ٢٨ دسی بل است. خروجی محدود کننده توسط بلوک جمع کننده ۹۹ با مقدار 32dB جمع شده و توسط محاسبه گر لگاریتم معکوس، بلوک ٤٨ ، که مشابه همنوعش در شکل ٤-٦ است به حوزه خطی برگردانده می شود.

مزیتهای این روش بر روش قبلی عبارتند از:

۱- در DSPهای ممیز ثابت محاسبه لگاریتم تعداد زیادی از سیکلهای ماشین را به خود اختصاص می‌دهد. در اینجا نیاز به محاسبه لگاریتم با قراردادن جدولهای $10Log_{10}P[y_i]$ و $10Log|gi|$ رفع گردیده است.

۲- هنگامی که از DSP ممیز ثابت استفاده می‌شود این روش نتایج عددی دقیقتری را فراهم می‌آورد.

البته این روش به 132=4+128 کلمه اضافی در حافظه نسبت به قبل دارد و ورودی آن نیز به جای e(n)، اندیسهای بهترین سطح بهره و بردار کد شکل انتخاب شده i و j است.

-ثابتها ، متغیرهای داخلی وروندنمای برنامه:

جداول ٤-٢ و ٤-٣ به ترتیب ثابتها و متغیرهای داخلی بکار رفته در برنامه و فرمت و محدوده اندیس هر کدام را نشان می‌دهند.

جدول (۴-۲) ثابتهای موجود در برنامه

نام ثابت	مقدار	توضیح
AGCFAC	١٦٢٢٠	فاکتور کنترل کننده سرعت AGC
KPDELTA	٦	انحراف مجاز از پریود PITCH قبلی
KPMIN	٢٠	پریود PITCH مینیمم (بر حسب تعداد نمونه)
KPMAX	١٤٠	پریود PIITCH ماگزیمم (بر حسب تعداد نمونه)
LPC	٥٠	مرتبه فیلتر ترکیب
LPCW	١٠	مرتبه فیلتر وزنی کیفیت
LPCLG	١٠	مرتبه پیشگویی کننده گین
NFRSZ	٢٠	اندازه(تعداد نمونه های) یک فریم
NONR	٣٥	تعدادنمونه های بخش غیربازگشت پنجره هایبرید برای فیلتر ترکیب
NONRW	٣٠	تعدادنمونه های بخش غیربازگشت پنجره هایبرید برای فیلتر وزنی کیفیت
NPWSZ	١٠٠	اندازه پنجره استفاده شده درآنالیز PITCH
NUPDATE	٤	دوره تناوب روز آمد شدن پیشگویی کننده ها بر

حسب تعداد بردار		
تعدادنمونه های بخش غیربازگشت پنجره هایبرید برای پیشگویی کننده گین	۲۰	**NONRLG**
تعداد کلمه-کد های موجود در کتاب-کد شکل	۱۲۸	**NCWD**
تعداد سطوح گین موجود در کتاب-کد گین	۸	**NG**
تعداد نمونه های موجود صحبت در بردار ورودی	۵	**IDIM**
مقدار off-set گین	۳۲×۵۱۲	**GOFF**
آستانه TAP برای خاموش کردن پست فیلتر- بلند	۹۸۳۰	**PPFTH**
فاکتور کنترل کننده صفر پست فیلتر بلند-مدت	۹۸۳۰	**PPFZCF**
آستانه TAP برای جایگزین کردنPITCH اصلی	۲۶۲۱٤	**TAPTH**
فاکتور کنترل کننده جبران tilt طیفی	٤۹۱۵	**TILTF**
بزرگترین عدد صحیح ۳۲ بیتی	۳۲٦٤۷	**MAXINT**

توضیح	فرمت نمایش ممیزثابت	تعداد عنا صر آرایه	نام متغیر
ضرایب فیلتر ترکیب	Q14	LPC+1	**A**
ضرایب مخرج فیلتر پایین گذر 1kHz	Q13	3	**AL**
ضرایب مخرج پست فیلتر کوتاه-مدت	Q14	11	**AP**
ضرایب فیلتر LPC مرتبه ۱۰	Q13	11	**APF**
بافر موقت برای ضرایب فیلتر ترکیب	Q13/Q14/Q15	LPC+1	**ATMP**
ضرایب مخرج فیلتر وزنی کیفیت	Q14	LPCW+1	**AWP**
ضرایب صورت فیلتر وزنی کیفیت	Q14	LPCW+1	**AWZ**
بافر موقت برای ضرایب فیلتر وزنی کیفیت	Q13/Q14/Q15	LPCW+1	**AWZTMP**
ضرایب صورت پست فیلتر کوتاه-مدت	Q14	11	**AZ**
ضرایب پست فیلتر بلند-مدت	Q16	۱	**B**
ضرایب صورت فیلتر پایین گذر1kHz	Q19	۴	**BL**
باقیمانده پیشگویی LPC	Q1	۲۴۰	**D**
نمونه برداری از باقیمانده پیشگوییLPC (۱از۴)	Q1	۶۰	**DEC**
حاصلضرب بردار تحریک و گین	15b BFL	IDIM	**ET**
بردار عریض کننده پهنای باند فیلتر ترکیب	Q14	LPC+1	**FACV**
بردار عریض کننده پهنای باند پیشگویی کننده گین	Q14	LPCLG	**FACGPV**
دو برابر سطوح گین در کتاب کد گین	Q12	NG	**G2**

گین تحریک خطی	SFL	1	**GAIN**
متوسط دوسطح گین مجاور	Q13	NG-1	**GB**
فاکتور مقیاس دهی درپست فیلتربلند-مدت	Q14	۱	**GL**
جمله حاصلضرب در پست فیلتر بلند-مدت	Q16	۱	**GLB**
ضرایب پیشگویی کننده گین در حوزه لگاریتم	Q14	LPCLG+1	**GP**
آرایه موقت برای ضرایب پیشگویی خطی گین	Q13/Q14/Q15	LPCLG+1	**GPTMP**
سطوح گین	Q13	NG	**GQ**
مربع سطوح گین	Q11	NG	**GSQ**
حافظه پیشگویی کننده گین در حوزه لگاریتم	Q9	LPCLG	**GSTATE**
بافر موقت برای گین	Q9	4	**GTMP**
بردار پاسخ ضربه F(z)×W(z)	Q13	IDIM	**H**
اندیس بهترین کلمه-کد که به گیرنده ارسال می شود	Q0	۱	**ICHAN**
شمارنده بردارهای صحبت در هر فریم(۱ تا ۴)	Q0	۱	**ICOUNT**
اندیس ۳-بیتی بهترین گین	Q0	۱	**IG**
پرچم ILL-CONDITIONING در فیلترترکیب	Q0	۱	**ILLCOND**
پرچم ILL-CONDITIONING در پیشگوی گین	Q0	۱	**ILLCONDG**
پرچم ILL-CONDITIONING در پست فیلتر	Q0	۱	**ILLCONDP**
پرچم ILL-CONDITIONING در فیلتر وزنی	Q0	۱	**ILLCONDW**
اشاره گر آدرس باقیمانده پیشگویی LPC	Q0	۱	**IP**
اندیس ۷-بیتی بهترین کلمه-کد شکل	Q0	۱	**IS**
پریود PITCH فریم کنونی و فریم قبلی	Q0,Q0	۱و۱	**KP,KP1**

توضیح	فرمت نمایش ممیز ثابت	تعداد عناصر آرایه	نام متغیر
لگاریتم گین	Q9	۱	**LOGGAIN**
حافظه بخش FIR فیلتر پایین گذر	Q1	۳	**LPFFIR**
حافظه بخش IIR فیلتر پایین گذر	Q1	۳	**LPFIIR**
پرچم دقت روند بازگشتیDurbin برای ATMP	Q0	۱	**NLSATMP**
پرچم دقت روند بازگشتیDurbin برایAWZTMP	Q0	۱	**NLSAWZTMP**
پرچم دقت روند بازگشتی Durbin برای GPTMP	Q0	۱	**NLSGPTMP**
تعداد شیفتهای به چپ ET	Q0	۱	**NLSET**
تعداد شیفتهای به چپ گین خطی	Q0	۱	**NLSGAIN**

تعداد شیفتهای به چپ REXP	Q0	۱	**NLSREXP**
تعداد شیفتهای به چپ REXPLG	Q0	۱	**NLSREXPLG**
تعداد شیفتهای به چپ REXPW	Q0	۱	**NLSREXPW**
تعداد شیفتهای به چپ SB	Q0	۲۱	**NLSSB**
تعداد شیفتهای به چپ ST	Q0	۱	**NLSST**
تعداد شیفتهای به چپ STATELPC	Q0	۱۱	**NLSSTATE**
تعداد شیفتهای به چپ STTMP	Q0	۴	**NLSSTTMP**
بردار کرولیشن در مدول جستجوی کتاب-کد	Q7	IDIM	**PN**
تپ (tap) پیشگویی کننده PITCH	Q14	۱	**PTAP**
ضرایب اتوکرولیشن	BFL	۱۱	**R**
ضرایب انعکاس	Q15	۱	**RC**
بافر موقت برای ضریب انعکاس اول	Q15	۱	**RC1**
بخش بازگشتی اتوکرولیشن (فیلتر ترکیب)	BFL	LPC+1	**REXP**
بخش بازگشتی اتوکرولیشن(پیشگویی کننده گین)	BFL	LPCLG+1	**REXPLG**
بخش بازگشتی اتوکرولیشن (فیلتر وزنی)	BFL	LPCW+1	**REXPW**
بافر موقت برای ضرایب اتوکرولیشن	BFL	LPC+1	**RTMP**
بردار صحبت ورودی(PCM یکنواخت)	15b Q2	IDIM	**S**
بافر ۱۰۵ نمونه کوانتیزه شده قبلی صحبت	14b BFL	۱۰۵	**SB**
بافر گین های قبلی در حوزه لگاریتم	Q9	۳۴	**SBLG**
ذافر ۶۰ نمونه قبلی صحبت ورودی	Q2	۶۰	**SBW**
فاکتور مقیاس دهی پست فیلتر قبل از عبور از فیلتر پایین گذر	SFL	۱	**SCALE**
فاکتور مقیاس دهی پست فیلتر بعد ازعبور از فیلتر پایین گذر	Q14	۱	**SCALEFIL**
بافر صحبت دیکد شده	Q0	IDIM	**SD**
بردار صحبت پست فیلتر شده	Q2	IDIM	**SPF**
بردار کنترل کننده قطب پست فیلترکوتاه-مدت	Q14	۱۱	**SPFPCFV**
بردار کنترل کننده صفر پست فیلترکوتاه-مدت	Q14	۱۱	**SPFZCFV**
بردار صحبت ورودی(U-Law/ A-Law)	byte	۱	**SO**
بردارصحبت کوانتیزه شده	13b Q0	۲۴۰	**SST(past)**
حافظه فیلتر ترکیب	15b Q2	IDIM	**SST(current)**

توضیح	فرمت نمایش ممیز ثابت	تعداد عناصرآرایه	نام متغیر
بردار صحبت کوانتیزه شده	14b BFL	IDIM	**ST**
حافظه فیلتر ترکیب	14b SBFL	LPC	**STATELPC**

حافظه فیلتر LPC معکوس	Q2	۱۰	**STLPCI**
بافر پنجره هایبرید مربوط به فیلتر وزنی	15b Q2	۴×IDIM	**STMP**
بافر پنجره هایبرید مربوط به فیلترترکیب	14b SBFL	۴×IDIM	**STTMP**
حافظه بخش تمام صفرپست فیلتر کوتاه-مدت	Q2	۱۰	**STPFFIR**
حافظه بخش تمام قطب پست فیلترکوتاه مدت	Q2	۱۰	**STPFIIR**
نمونه صحبت ورودی (PCM یکنواخت)	Q2	1	**SU**
مجموع قدرمطلقهای مقادیر صحبت پست فیلتر شده	Q2	1	**SUMFIL**
مجموع قدرمطلقهای مقادیر صحبت دیکد شده	Q2	1	**SUMUNFIL**
خروجی فیلتر وزنی	Q2	IDIM	**SW**
بردار تفاضل(خروجی بلوک۱۱)که بر گین تقسیم شده است.	BFL	IDIM	**TARGET**
آرایه چکنویس که به عنوان فضای کار موقت استفاده می شود.	*	IDIM	**TEMP**
ضریب جبران tilt پست فیلترکوتاه-مدت	Q14	۱	**TILTZ**
حافظه فیلتر وزنی(بلوک۴)بخش تمام صفر	Q2	LPCW	**WFIR**
حافظه فیلتر وزنی(بلوک۴)بخش تمام قطب	Q2	LPCW	**WIIR**
تابع پنجره هایبرید برای فیلتر ترکیب	Q15	105	**WNR**
تابع پنجره هایبرید برای پیشگویی کننده گین	Q15	۳۵	**WNRLG**
تابع پنجره هایبرید برای فیلتر وزنی	Q15	60	**WNRW**
بردار کنترل کننده قطب در فیلتر وزنی	Q14	LPCW+1	**WPCFV**
آرایه چکنویس	#	105	**WS**
بردار کنترل کننده صفر در فیلتر وزنی	Q14	LPCW+1	**WZCFV**
آرایه ای که کتاب-کد شکل درآن قرار میگیرد.	Q11	IDIM×NCWD	**Y**
انرژی کد شکلهایی که کانولوشن شده اند.	Q5	NCWD	**Y2**
پاسخ ورودی صفر	15b Q2	IDIM	**ZIR**
حافظه فیلتر وزنی(بلوک۱۰)بخش تمام صفر	15b Q2	LPCW	**ZIRWFIR**
حافظه فیلتر وزنی(بلوک۱۰)بخش تمام قطب	15b Q2	LPCW	**ZIRWIIR**

SFL : فرمت ممیز شناور اسکالر

BFL : فرمت ممیز شناور بلوکی

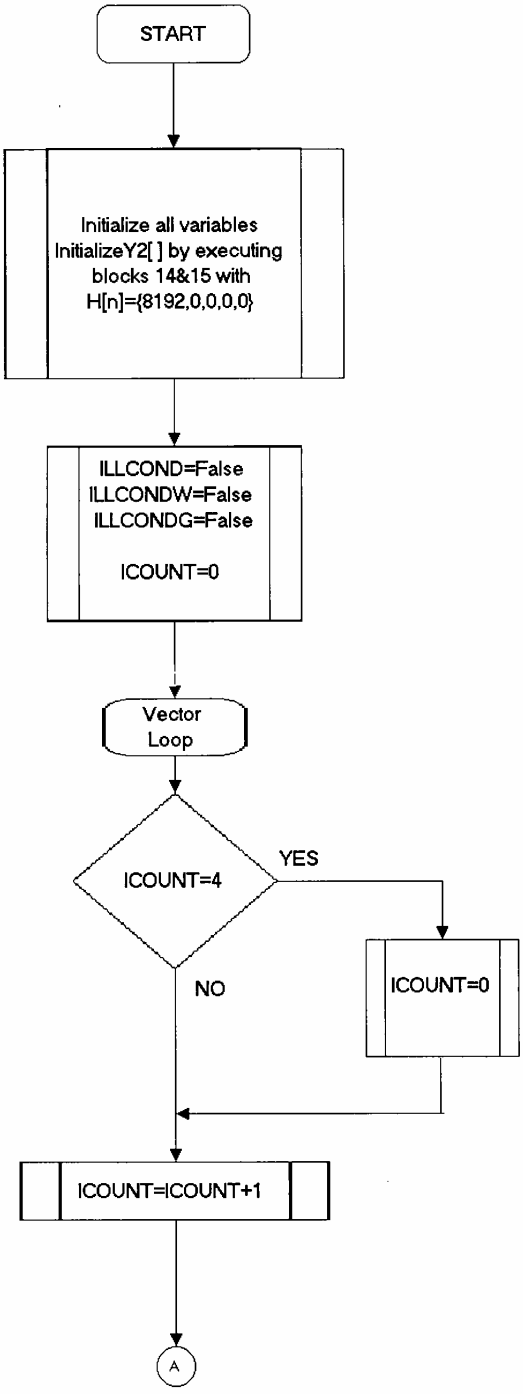
Qx : فرمت Qx

14B یا 15b : نشان دهنده دقت 14بیت یا ۱۵ بیت است. بقیه متغیر ها دارای دقت ۱۶ بیتی هستند.

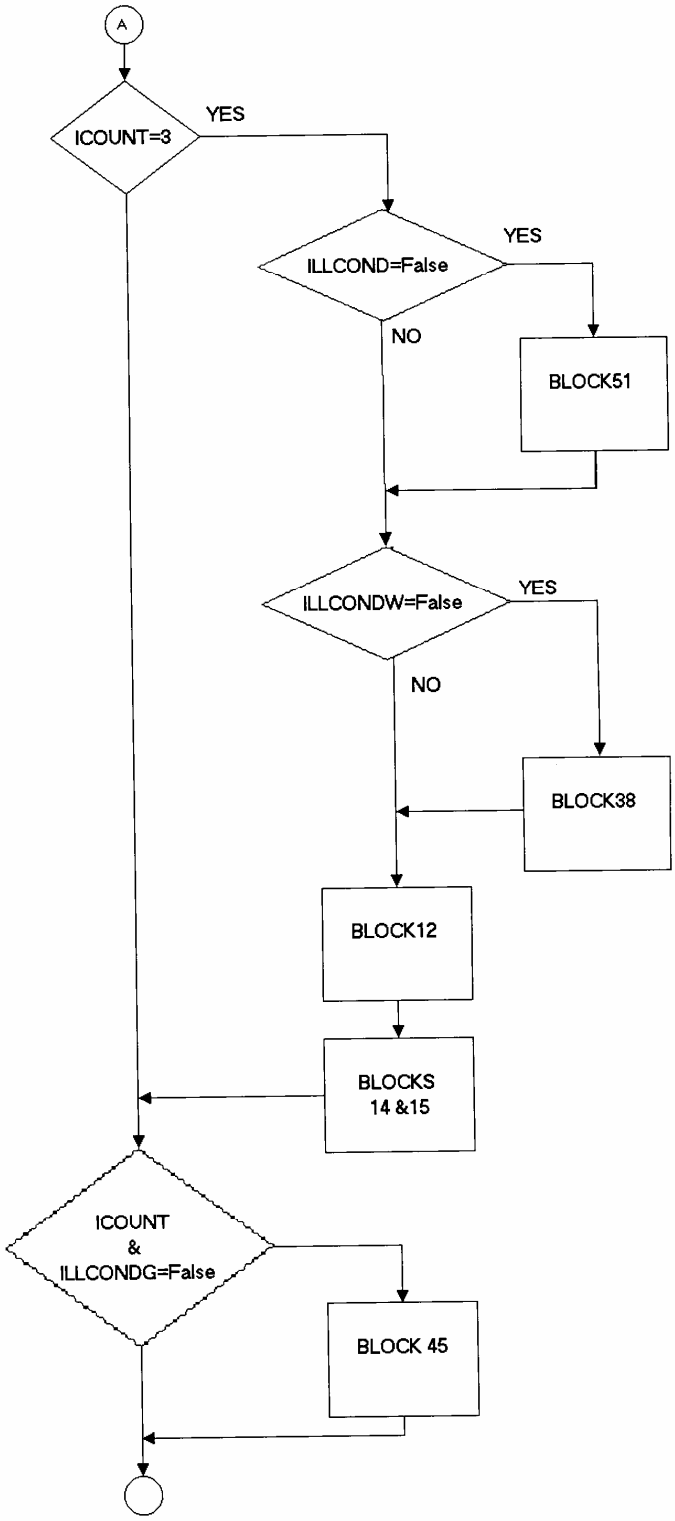
: بستگی به مکان استفاده از آرایه چکنویس دارد وممکن است BFL یا Qx باشد.

- روندنمای برنامه:

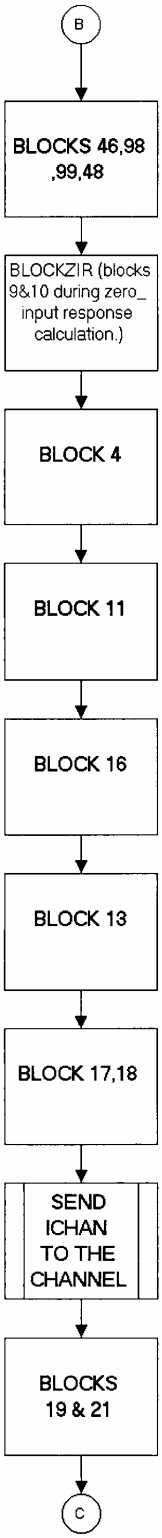
روند نماهایی که از اینکدر و دیکدر در زیر رسم شده است، براساس بخش G.7 استاندارد G.728-Anne×G تهیه شده است. در بخش ذکر شده الگوریتم برنامه‌های main اینکدر و دیکدر بیان شده است. بنابر روندنماهای رسم شده در این بخش تنها یکی از ترتیبهای ممکن برای اجرای بلوکها است و ترتیبهای بسیاری برای اجرای بلوکها وجود دارد که نتیجه‌ی مطلوبی دارند.



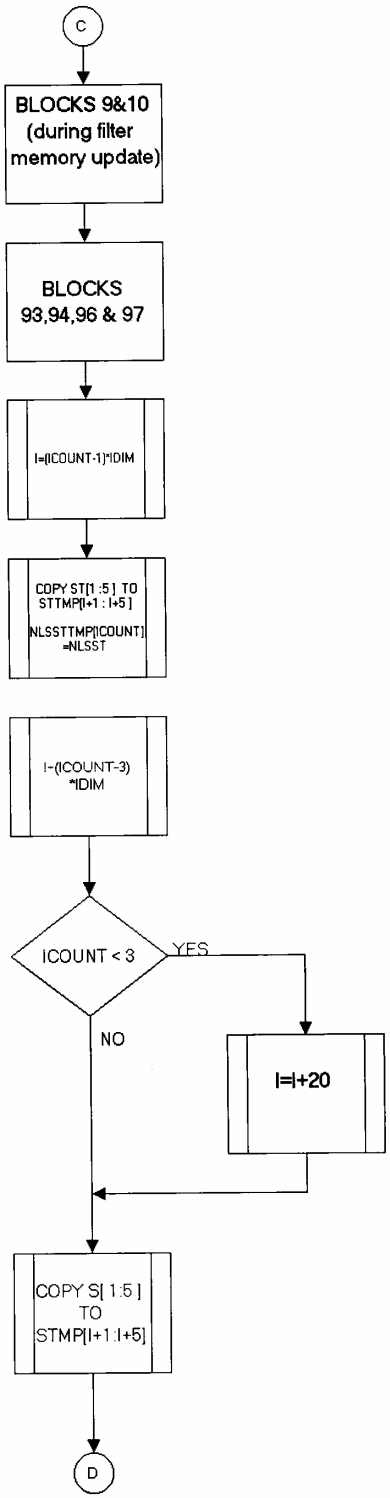
شکل ۴-۱۰- روند نمای اینکدر



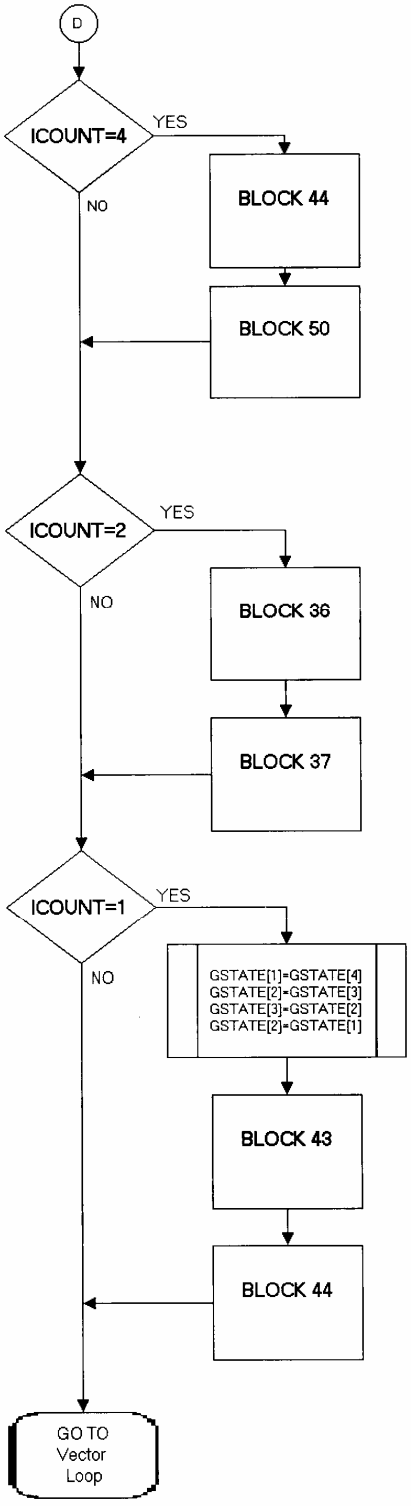
شکل ۴-۱۰- روند نمای اینکدر(ادامه)



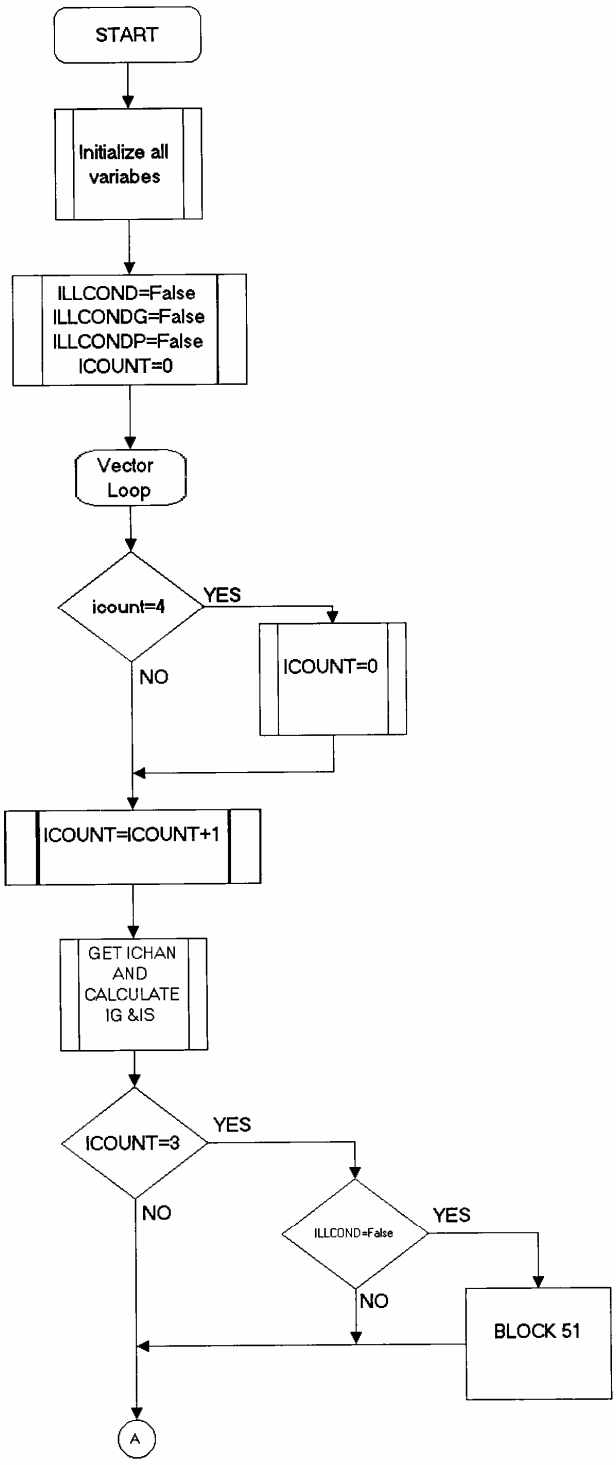
شکل ۴-۱۰- روند نمای اینکدر(ادامه)



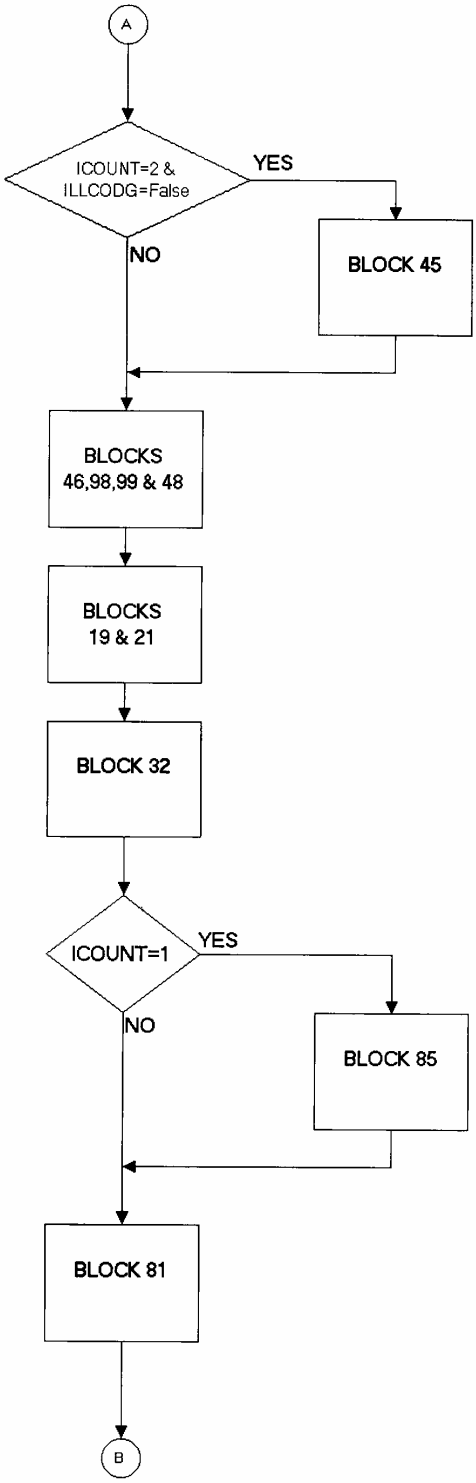
شکل ۴-۱۰- روند نمای اینکدر(ادامه)



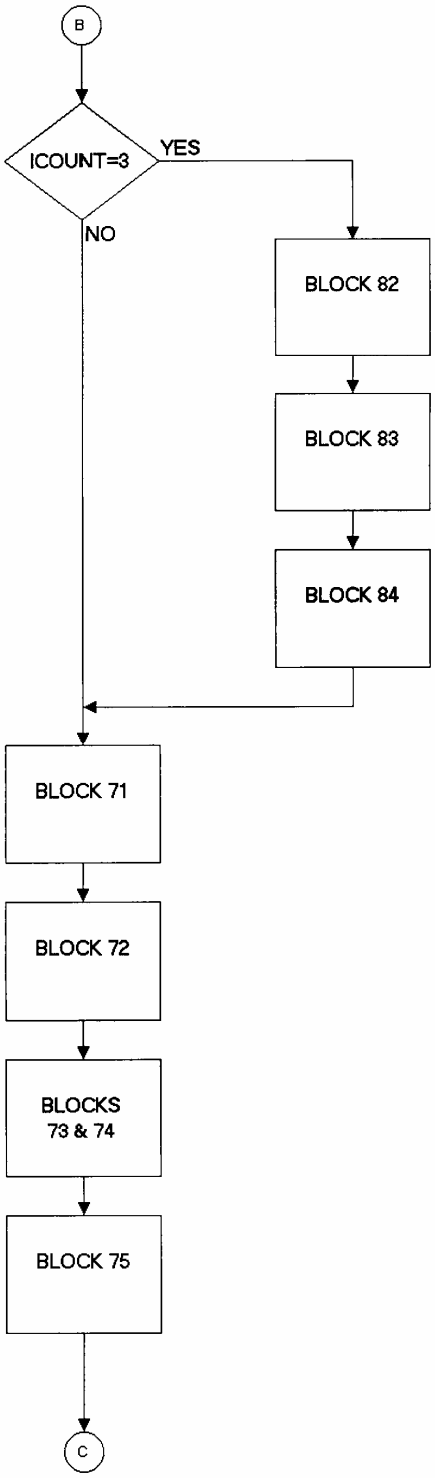
شکل ۴-۱۰- روند نمای اینکدر(ادامه)



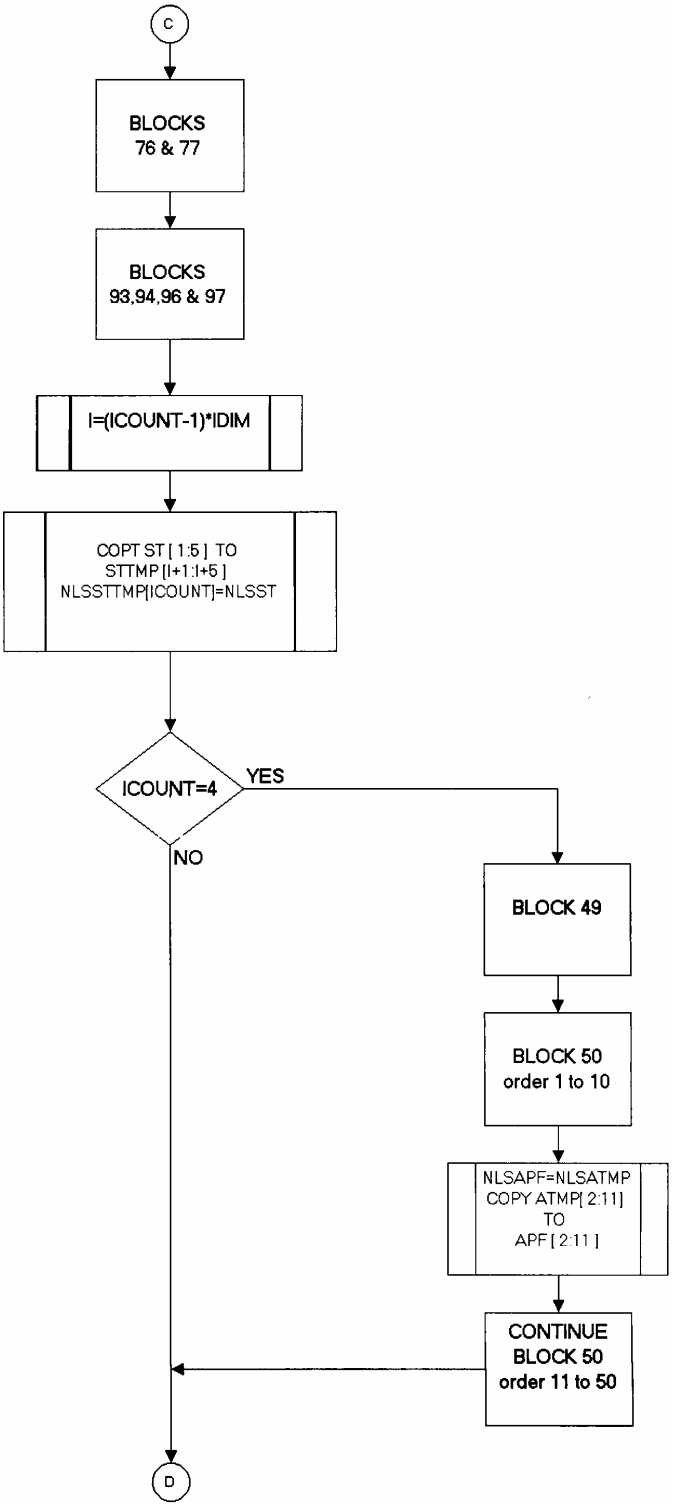
شکل۴-۱۱- روند نمای دیکدر



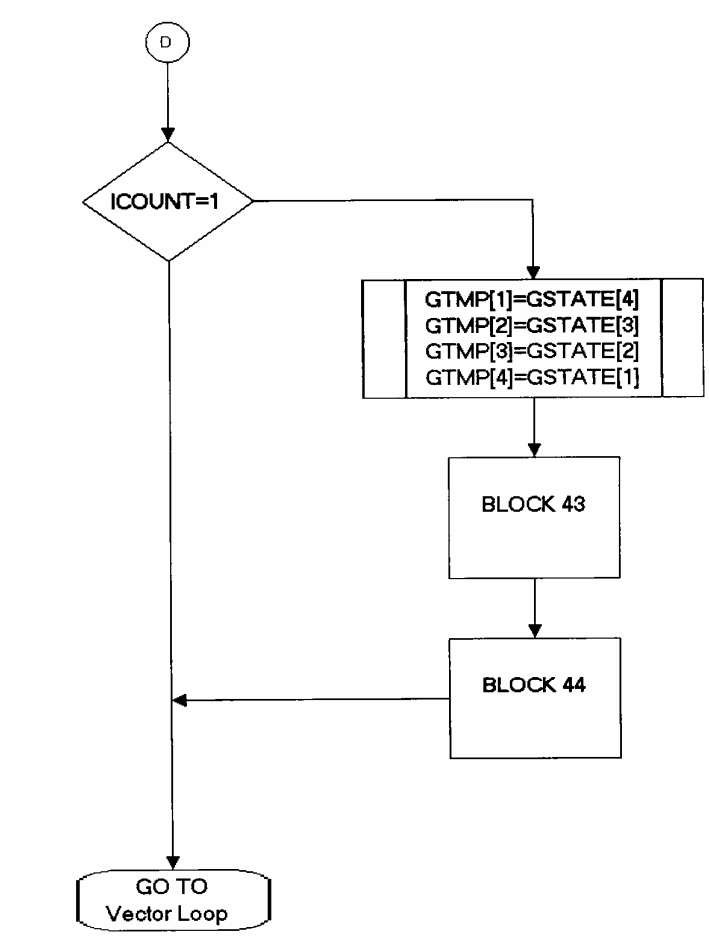
شکل۴-۱۱- روند نمای دیکدر(ادامه)



شکل۴-۱۱- روند نمای دیکدر(ادامه)



شکل۴-۱۱- روند نمای دیکدر(ادامه)



شکل۴-۱۱- روند نمای دیکدر(ادامه)

-بررسی صحت پیاده سازی ممیز ثابت استاندارد G.728 :

فایلهای داده ای که ITU-T برای بررسی صحت پیاده سازی ممیز ثابت ارائه کرده است به شرح زیر هستند:

...IN : در روند تستها این فایلها به عنوان سیگنال ورودی اینکدر استفاده می شوند.

...INCW*G : این فایلها خروجی مرجع اینکدر هستند که با خروجی واقعی که برنامه تولید کرده است مقایسه می شوند.

...CW: این فایلها به عنوان ورودی دیکدر استفاده می شوند.

..OUTA*G: این فایلها خروجی مرجع دیکدر هستندو باخروجی واقعی که دیکدر در حالیکه فیلتر پایانی غیر فعال است، تولید می کند مقایسه می شوند.

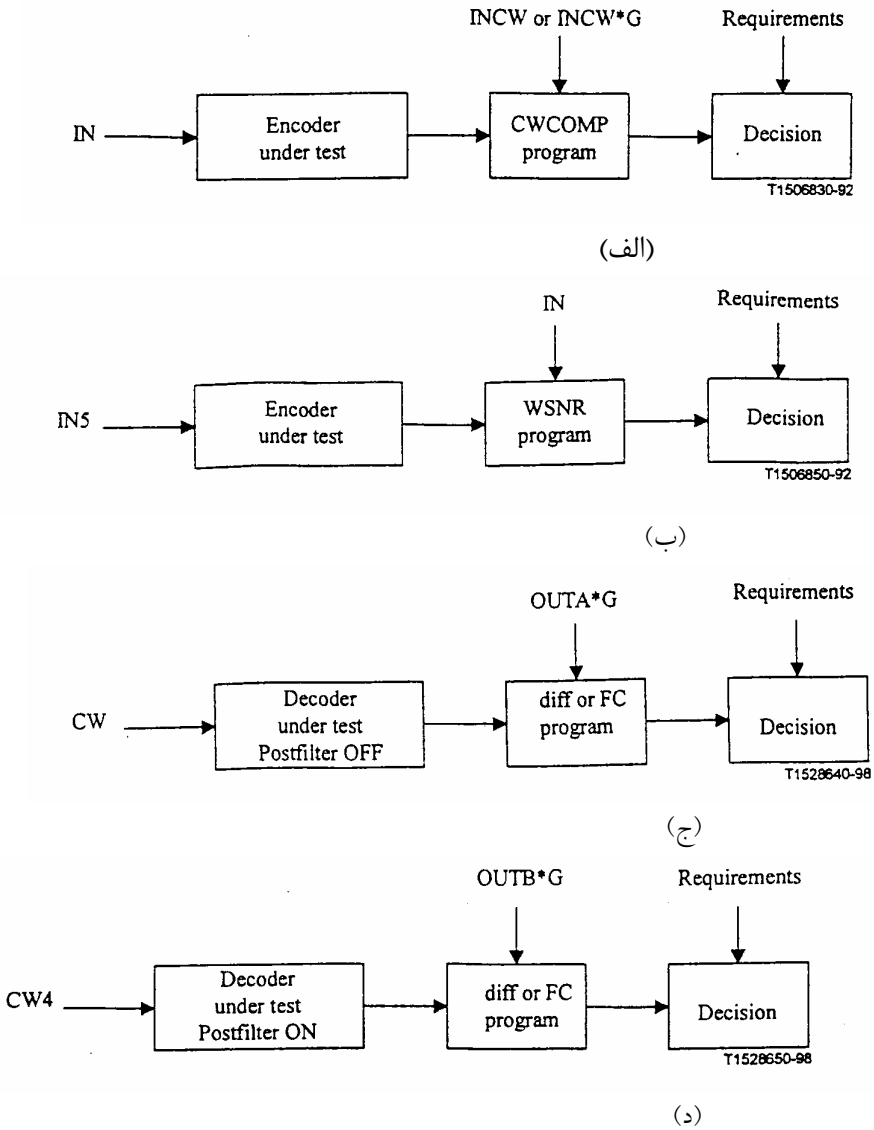
...OUB*G : این فایلها خروجی مرجع دیکدر هستند و با خروجی واقعی که دیکدر درحالیکه فیلتر پایانی فعال است تولید می کند مقایسه می شوند.

کلیه فایلهای مذکور دارای فرمت باینری (.BIN) هستند .

برنامه های مورد استفاده در تست نیز به صورت زیر می باشد.

CWCOMP : این برنامه دو فایل باینری را مقایسه می کند و تعداد اختلاف و محل بروز اولین اختلاف را نشان میدهد.

WSNR : این برنامه SNR را به صورت لگاریتمی برای هر بردار(نمونه متوالی صحبت) محاسبه می کند .
FC: دستور FC در سیستم عامل MS-DOS دو فایل را مقایسه می کند و اختلاف یا عدم اختلاف آنها را نمایش می دهد(نحوه استفاده از این دستور به صورت FC/B FILE1 FILE2 است) هستند .بلوکهای دیاگرامهای شکل٤-١٢ نحوه انجام این تستها و جدول (٤-٤)شرایطی را که باید دراین قسمتها محقق شوند نمایش می دهند.



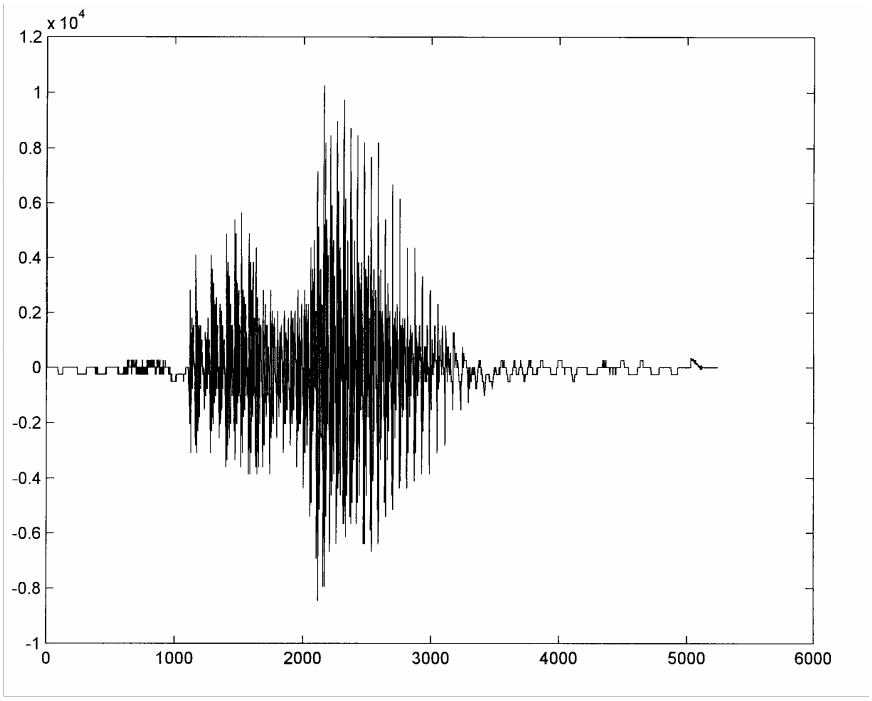
شکل٤-١٢ بلوک دیاگرام تستهای ITU-T

جدول (٤-٤)-شرایط مطلوب حاصل از انجام تستها

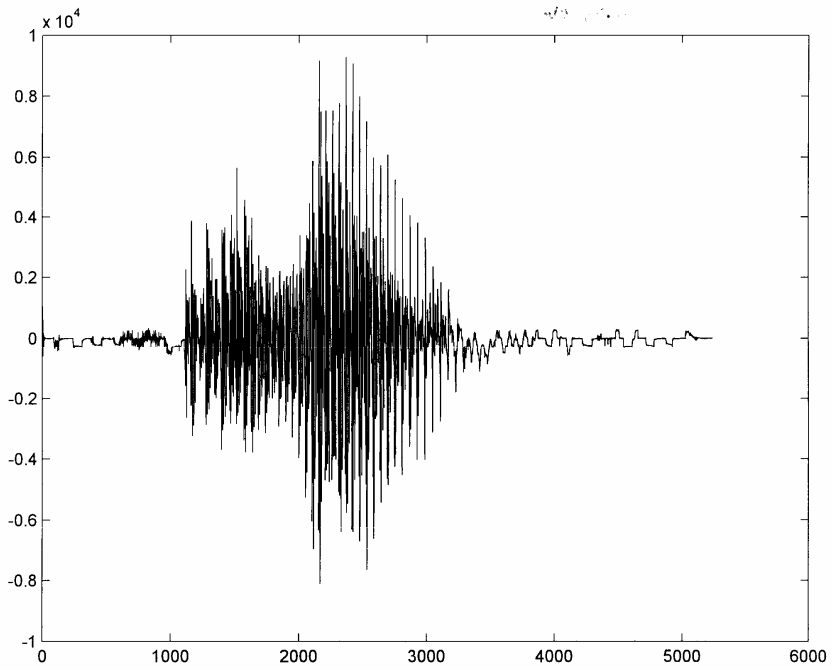
Input signal	Output signal	Requirement
IN1	INCW1G	0 different codewords detected by CWCOMP
IN2	INCW2G	0 different codewords detected by CWCOMP
IN3	INCW3G	0 different codewords detected by CWCOMP
IN4	INCW4G	0 different codewords detected by CWCOMP
IN5	INCW5G	0 different codewords detected by CWCOMP
IN6	INCW6G	0 different codewords detected by CWCOMP

علاوه بر تستهای فوق به منظور BIT-EXACT کردن برنامه قطعه ای کوتاه از صحبت واقعی بـه عنـوان ورودی ومقـادیر تمام متغیرهای داخلی برنامه برای این ورودی توسط ITU-T فراهم آمده است .ما کلیه تستهای فوق الذکر را برای این پروژه انجام داده و نتیجه تستها برآورده شدن تمام شرایط مطلوب خواسته شده در جدول (٤-٤)بود.

شکل ٤-١٣ نمایش سیگنال PCM ناشی از کلمه سلام که توسط یک گوینده مرد بیان شده است را نشان می دهد. این فایـل به عنوان ورودی به اینکدر داده شده و شکل موج خروجی متناظر دیکدر در شکل ٤- ١٤آمده است.



شکل (۴-۱۳) سیگنال ورودی به اینکدر



شکل (۴-۱۴) سیگنال خروجی از دیکدر

حال با استفاده از معیار SNR که اعوجاج بین ورودی و خروجی سیستم را در بر می گیرد و به صورت نسبت تـوان ورودی به توان خطای بین ورودی و خروجی تعریف می شود به ارزیابی این کدک می پردازیم.

معمولا در صحبت به دلیل تفاوت انرژی قسمت های با واک وبی واک از معیار SNRseg به صورت 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 و M تعداد این سگمنتها می باشد.با استفاده از فرمول بالا SNRseg برای این کدک در حدود 9 dB بدست آمد.