# پیاده‌سازی سخت افزاری دیکودرکدهای LDPC برای استاندارد وایمکس

علیرضا رامز[۱]، ابومسلم جان‌نثاری[۲]

**چکیده:**

*هدف از این مقاله پیاده‌سازی یک دکودر LDPC برای استاندار وایمکس می‌باشد. ماتریس پریتی چک برای این استاندارد به طول ۲۳۰۴ و نرخ ۵/۶، با girth بزرگ و به کمک ماتریس مدل طراحی شده‌است. برای پیاده‌سازی بهینه، ماتریس H به صورت QC-LDPC در نظرگرفته شده‌است. از روش OMS به عنوان بهینه ترین الگوریتم دکودینگ برای این پیاده‌سازی استفاده کرده و بهترین مقدار $\beta$ برای این الگوریتم محاسبه شده‌است. همچنین روشی برای حل مشکل دسترسی به حافظه در پیاده‌سازی نیمه موازی ارائه شده که بر اساس آن و با استفاده از روش هم پوشانی پیام سرعت دکودر افزایش یافته‌است. برای بررسی تأثیر پارامتر های مختلف در عملکرد دکودر LDPC، پیاده‌سازی دکودر به صورت توصیف سخت افزاری به زبان verilog، روی بستر سخت افزاری FPGA انجام وتست گردید و در نهایت سنتز آن بر روی فناوری $\mu$ ۰,۱۳CMOS-TSMC با استفاده از نرم افزار Synopsis Design Complier انجام شد.*

**کلید واژه:**

*کدهای چک پریتی با چگالی کم، LDPC، پیاده‌سازی سخت افزاری، وایمکس، QC-LDPC .*

## مقدمه

زمینه ارتباطات بی سیم و انتقال اطلاعات طی سالیان اخیر رشد بسیار زیادی داشته‌است و در این بین دریافت اطلاعات به شکل صحیح بر روی کانال‌های نویزی یکی از موضوعات مهم تحقیقاتی محسوب می‌شود. تصحیح خطا عبارت است از توانایی گیرنده در بازسازی اطلاعات اولیه ارسال شده از فرستنده که پس از عبور از کانال مخدوش شده‌است. کدهای چک پریتی با چگالی کم( LDPC)[۳] که توسط گالاگر[۴] معرفی شدند[۱]، موجب تحول چشم گیری در زمینه‌ی کدهای تصحیح خطا شده‌اند، به طوری که امروزه در بسیاری از سیستم‌های مخابراتی، مانند استانداردهای IEEE 802.11n، IEEE 802.16e، IEEE 802.3an و نسل دوم استاندارد DVB و دستگاه‌های ذخیره‌سازی اطلاعات با قابلیت اطمینان بالا استفاده می‌شوند. در سال‌های اخیر تحقیقات زیادی بر روی ساختار دکودر این کدها انجام شده‌است که در یک دسته بندی کلی به ۳ دسته تقسیم می‌شوند. روش موازی[۲]، روش سری [۳]و روش نیمه موازی [۴]. از این نقطه نظر، موازی سازی ساختار و مصالحه میان هزینه سخت‌افزار و گذردهی سیستم پررنگ تر است. از چالش‌های دیگر طراحی دکودر های LDPC می‌توان از مسیریابی[۵] بسیار پیچیده، پشتیبانی از نرخ‌ها و طول‌های مختلف کد و توان مصرفی نام برد. در این مقاله دکودر LDPC برای سیستم WiMAX (IEEE 802.16e) طراحی شده‌است. برای افزایش حداکثر گذردهی و کاهش پیچیدگی سخت‌افزاری از ساختار نیمه موازی به همراه ۲ روش پیشنهادی استفاده کردیم. ۱) اصلاح نگاشت زیرماتریس‌ها ۲) روش همپوشانی پیام. در نهایت بعد از تصمیم‌گیری در مورد پارامترها، دکودر طراحی شده بعد از سنتز بر روی فناوری $\mu$ ۰,۱۳CMOS ، ۵,۶۳ میلیمتر از سطح تراشه را اشغال کرده و در فرکانس ۵۲ مگا هرتز تنها ۱۷۴ میلی وات توان مصرف می‌کند. ساختار مقاله بدین صورت خواهد بود که در بخش ۲ کدهای LDPC و دکودینگ آنها شرح مختصری داده شده‌است. بخش ۳ به تشریح طراحی دکودر اختصاص یافته‌است. در بخش ۴ نتایج پیاده‌سازی‌ها ارائه می شود و در بخش پایانی جمع بندی و نتیجه‌گیری بیان خواهد شد.

[۱] کارشناس ارشد الکترونیک،دانشگاه تربیت مدرس، alireza.ramez@modares.ac.ir

[۲] استادیار گروه الکترونیک دانشکده برق،دانشگاه تربیت مدرس

[۳] Low Density Parity Check

[۴] Gallager

[۵] routing

تاریخ دریافت: ۱۲ آبان ۱۳۹۲ تاریخ پذیرش: ۲۰ بهمن ۱۳۹۲

## کدهای LDPC

### معرفی

کدهای LDPC زیر مجموعه‌ای از کدهای بلوکی باینری هستند بنابراین یک رشته داده به طول $k$ را به یک رشته کلمه کد به طول $n$ می‌نگارند. هر گاه بردار $\underline{x}$ یک کلمه کد صحیح باشد، رابطه سندروم برای کدهای بلوکی به صورت رابطه (۱) است که در آن $H$ ماتریس پریتی چک می‌باشد.

$$H_{(n-k)\times n}\underline{x}^{T}_{1\times n}=\underline{0}_{(n-k)\times 1} \qquad (1)$$

خصوصیت بارز کدهای LDPC تنک[۶] بودن ماتریس پریتی چک آن‌ها است. منظور از تنک بودن ماتریس آن است که تعداد عناصر یک ماتریس $H$ در مقابل ابعاد آن، کسر بسیار کوچکی است. کدهای LDPC را علاوه بر ماتریس می‌توان توسط گراف نیز نمایش داد[۷] . گراف تنر گرافی دو بخشی است که در آن رئوس سمت چپ که با دوایر مشخص می‌شوند معرف متغیرها یا بیت‌های کلمه کد هستند و رئوس سمت راست که با مربع مشخص می‌شوند، معرف چک‌ها روی کلمات کد است. شکل(۱) یک ماتریس پریتی چک و گراف متناظر آن برای کدی به طول (۱۰،۵) می‌باشد. برای مثال گره‌های $v_0,v_1,v_2,v_3$ به گره چک $c_0$ متصل شده‌اند؛ چرا که در ماتریس $H$ متناظر آن عناصر $h_{00}=h_{01}=h_{02}=h_{03}$ برابر ۱ هستند.

یک حلقه در گراف تنر، به صورت یک مسیر بسته با طول $2g$، بعد از گذشتن از $g$ گره چک و $g$ گره متغیر و بدون عبور دو بار از یک لبه تعریف می‌شود. به کوچک‌ترین طول حلقه در کد girth گفته می‌شود. تنک بودن ماتریس پریتی چک کدهای LDPC باعث می‌شود که تعداد حلقه‌ها کاهش یابد و طول حلقه‌ها و طول girth کد بیشتر شود، که موجب کارایی بهتر خواهد شد.[۵]

### دکدینگ کدهای LDPC

الگوریتم دکدینگ بر پایهٔ روابط خطی بیت‌های اطلاعات و بیت‌های پریتی انجام می‌گیرد. الگوریتم معروفی که برای دکدینگ کدهای LDPC استفاده می‌شود الگوریتم انتشار گمان[۷] (BP) است[۶]. در این الگوریتم در هر مرحله، پیام‌هایی از گره‌های متغیر به گره‌های چک و بالعکس ارسال می‌شوند.

اگر خروجی گره چک را با $L(r_{ji})$ خواهیم داشت:

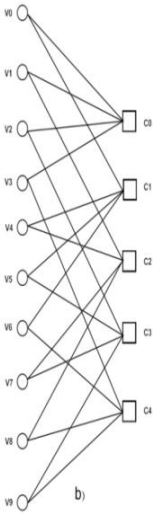
$$L(r_{ji})=\prod_{i'\in R_{j/i}}\alpha_{i'j}\cdot\varphi\left[\sum_{i'\in R_{j/i}}(\beta_{i'j})\right] \qquad (2)$$

که در آن $\varphi(x)$ به صورت زیر تعریف می‌شود:

$$\varphi(x)\equiv-\log(\tanh(\frac{1}{2}x))=\log\frac{e^{x}+1}{e^{x}-1} \qquad (۳)$$

خروجی گره متغیر در الگوریتم BP را که با $L(q_{ij})$ نشان

$$H=\begin{bmatrix}1&1&1&1&0&0&0&0&0&0\\1&0&0&0&1&1&1&0&0&0\\0&1&0&0&1&0&0&1&1&0\\0&0&1&0&0&1&0&1&0&1\\0&0&0&1&0&0&1&0&1&1\end{bmatrix}$$




شکل (۱) a) ماتریس پریتی چک یک کد LDPC به طول (۱۰،۵) b) گراف تنر متناظر ماتریس پریتی چک

دهیم به شکل زیر خواهد بود:

$$L(q_{ij})=L(c_i)+\sum_{j'\in c_{i/j}}L(r_{j'i}) \qquad (4)$$

بهترین الگوریتم دکودینگ LDPC تا به امروز، الگوریتم انتشار گمان بوده‌است. اما مشکلاتی از قبیل پیچیدگی و حساسیت به دقت محدود، باعث به وجود آمدن الگوریتم‌های جدید شد. یکی از این الگوریتم‌ها Min-Sum (MS) می‌باشد. ساختار الگوریتم MS شبیه ساختارBP است، غیر از یک تفاوت که باعث کاهش حجم محاسبات و افزایش مقاومت در مقابل نویزکوانتیزیشن می‌شود. در الگوریتم MS، تابعی که در گره چک استفاده می‌شود فقط از توابع تعیین علامت و مینیمم گیری استفاده می‌کند که باعث می‌شود معادله مورد استفاده در BP به شدت ساده‌سازی شود. بنابراین رابطه (۲) که تابع گره چک در الگوریتم BP بود به شکل زیر بازنویسی می‌شود:

---

[۶] Sparse

[۷] Belief Propagation

دهیم به شکل زیر خواهد بود:

$$L(q_{ij}) = L(c_i) + \sum_{j' \in c_{i/j}} L(r_{j'i}) \qquad (4)$$

بهترین الگوریتم دکودینگ LDPC تا به امروز، الگوریتم انتشار گمان بوده‌است. اما مشکلاتی از قبیل پیچیدگی و حساسیت به دقت محدود، باعث به وجود آمدن الگوریتم‌های جدید شد. یکی از این الگوریتم‌ها Min-Sum (MS) می‌باشد. ساختار الگوریتم MS شبیه ساختارBP است، غیر از یک تفاوت که باعث کاهش حجم محاسبات و افزایش مقاومت در مقابل نویزکوانتیزیشن می‌شود. در الگوریتم MS، تابعی که در گره چک استفاده می‌شود فقط از توابع تعیین علامت و مینیمم گیری استفاده می‌کند که باعث می‌شود معادله مورد استفاده در BP به شدت ساده‌سازی شود. بنابراین رابطه (۲) که تابع گره چک در الگوریتم BP بود به شکل زیر بازنویسی می‌شود:

$$L(r_{ji}) = \prod_{i' \in R_{j/i}} \alpha_{i'j} \cdot \min_{i' \in R_{j/i}} |\beta_{i'j}| \qquad (۵)$$

البته به علت تقریب زیاد این تابع نسبت به تابع اصلی شاهد افت عملکرد دکودر خواهیم بود. به این علت در این مقاله به سراغ الگوریتم OMS[۸] می‌رویم که نسخه بهبود یافته الگوریتم MS است، تا ضمن آنکه پیچیدگی سخت افزاری مدار را در حد مناسبی کاهش دهیم، در عملکرد دکودر هم افت کمتری مشاهده کنیم. تابع اصلاح شده برای گره چک در الگوریتم OMS به شکل زیر خواهد بود:

$$L(r_{ji}) = \prod_{i' \in R_{j/i}} \alpha_{i'j} \cdot \max(\min_{i' \in R_{j/i}} |\beta_{i'j}| - \beta, 0) \qquad (6)$$

بر اساس رابطه (۶) مقدار تمام پیام‌هایی که اندازه آن‌ها کوچک‌تر از $\beta$ باشد، در این الگوریتم برابر ۰ می‌شود. چرا که این مقادیر کم در پروسه گره متغیر کمک کمی می‌کند.

تمام الگوریتم‌های ذکر شده به دوصورت تصمیم‌گیری سخت[۹] و تصمیم‌گیری نرم[۱۰] قابل اجرا هستند. در تصمیم‌گیری سخت باروش ساده‌تری روبرو هستیم درحالیکه با روش تصمیم‌گیری نرم نتایج دکدینگ بهتری خواهیم داشت.

## طراحی دکودر

### ساختار ماتریس پریتی چک

در حالت کلی ساخت ماتریس پریتی چک را می‌توان به دو روش انجام داد، ساختن بر اساس جستجوی کامپیوتری و ساختن براساس روش‌های جبری. دسته اول به خاطرنداشتن ساختارخاص برای کدینگ بسیار پیچیده‌اند. در حالیکه کدهای با ساختار جبری می‌توانند به روش‌های مختلفی کد شوند. کدهای QC-LDPC یکی از مهم‌ترین زیر مجموعه‌های دسته دوم هستند. در قسمت دکودر در صورتی که از کد های QC-LDPC استفاده کنیم، معماری دکودر به مکانیزم ساده تری برای ساختن آدرس نیاز خواهد داشت و در نتیجه حافظه کمتری مصرف خواهد شد. ضمن آنکه دکودر این کدها می‌تواند به صورت نیمه موازی پیاده‌سازی شود[۷]. کد های QC-LDPC شامل زیر ماتریس‌های چرخشی[۱۱] هستند که به صورت

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 25 | 55 | -1 | 47 | 4 | -1 | 91 | 83 | 8 | 86 | 52 | 82 | 33 | 5 | 0 | 36 | 20 | 4 | 77 | 80 | 0 | -1 | -1 |
| -1 | 6 | -1 | 36 | 40 | 47 | 12 | 79 | 47 | -1 | 41 | 21 | 12 | 71 | 14 | 72 | 0 | 43 | 49 | 0 | 0 | 0 | 0 | -1 |
| 51 | 81 | 83 | 4 | 67 | -1 | 21 | -1 | 31 | 24 | 91 | 60 | 81 | 9 | 86 | 78 | 60 | 88 | 67 | 15 | -1 | -1 | 0 | 0 |
| 50 | -1 | 50 | 15 | -1 | 35 | 13 | 10 | 11 | 20 | 53 | 90 | 29 | 92 | 57 | 30 | 84 | 92 | 11 | 66 | 80 | -1 | -1 | 0 |

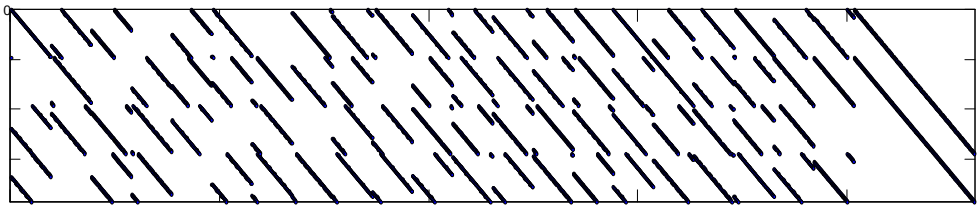
الف)

[۸] Offset Min Sum

[۹] Hard decision

[۱۰] Soft decision

[۱۱] Circulant



ب)

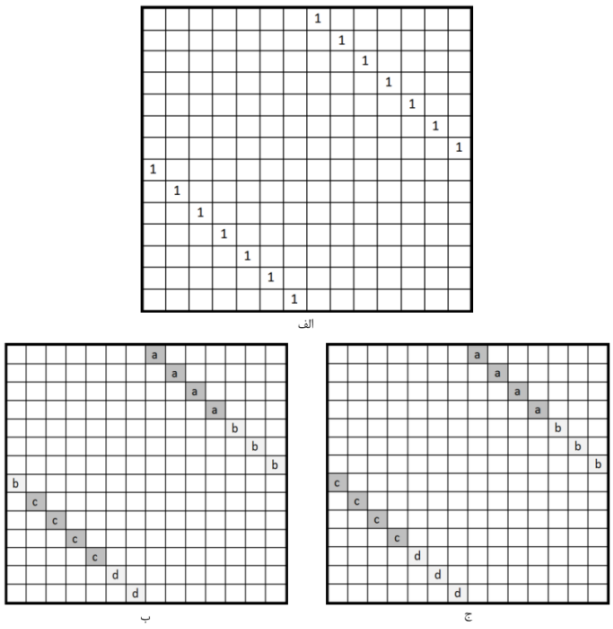
شکل (۲) الف) ماتریس مدل طراحی شده برای استاندارد وایمکس برای طول ۲۳۰۴ و نرخ ۵/۶ ب) ماتریس H گسترش یافته ماتریس مدل به طول ۲۳۰۴ و نرخ ۵/۶

افقی به هم متصل شده‌اند. هر زیر ماتریس چرخشی یک ماتریس مربعی است، به طوری که هر ردیف شیفت چرخشی یافته‌ی ردیف قبلی است و ردیف اول شیفت چرخشی یافته‌ی آخرین ردیف می‌باشد. اگر مقادیر شیفت زیر ماتریس‌ها را به صورت تصادفی بین زیر ماتریس‌ها توزیع کنیم، عملکرد کد افت خواهد کرد. بنابراین در این مقاله برای ساخت این کدها، از یک ماتریس مدل استفاده شده‌است. در این روش ابتدا یک ماتریس با ابعاد کوچک تعریف می‌کنیم و سپس آن را گسترش می‌دهیم. در این حالت ماتریس نهایی حداقل همان girth ماتریس پایه را خواهد داشت؛ ضمن آنکه مدیریت اطلاعات در دکودر آسان‌تر خواهد بود. ماتریس مدل و ماتریس پریتی چک طراحی شده برای استاندارد وایمکس به طول ۲۳۰۴ و نرخ ۵/۶ در شکل (۲) نشان داده شده‌است. در ماتریس مدل، یک مقدار غیر صفر مثبت، معرف یک ماتریس واحد شیفت یافته به آن مقدار و ۱- نشان دهنده ماتریس تمام صفر است. با تغییر ابعاد زیر ماتریس ها و ماتریس مدل به نرخ ها و طول های مختلف استاندارد وایمکس می‌توان دست یافت.

## روش‌های پیشنهادی برای طراحی دکودر LDPC

هنگام پیاده‌سازی ساختار نیمه موازی کدهای QC-LDPC به مشکل اختلال در دسترسی به حافظه بر می‌خوریم. روش استفاده شده در این مقاله برای حل این مشکل در ادامه در قالب یک مثال توضیح داده می‌شود. شکل (۳)-الف که زیر ماتریس یک کد QC-LDPC را نشان می‌دهد را در نظر بگیرید، در صورتی که بخواهیم برای این زیر ماتریس از دکودر معمولی استفاده کنیم ۱۴ بانک حافظه نیاز خواهیم داشت. برای موازی سازی ساختار می‌توانیم پردازش گره‌ها را طوری انجام دهیم که واحد پردازش گره، چند ردیف را به صورت هم زمان پردازش کند. اما در حالت معمولی این کار باعث به وجود آمدن اختلال در دسترسی به حافظه خواهد بود، چرا که درخواست دسترسی به بانک حافظه را در یک زمان، از چند جا خواهیم داشت که نهایتاً موجب کاهش سرعت شده و هدف ما از استفاده از این ساختار را مخدوش خواهد کرد. بنابراین باید روشی برای مدیریت این مسئله استفاده کنیم. زیر ماتریس نشان داده شده در شکل (۳)- ب تقسیم بندی معمولی و به ترتیب ردیف‌ها برای تخصیص چند ردیف به یک واحد برای شکل (۳)- الف خواهد بود. حال آنکه همان طور که مشاهده می‌شود در بخش پردازش ستون‌ها اختلال دسترسی به حافظه خواهیم داشت. درشکل (۳)- ج یک نگاشت پیشنهادی که باعث کاهش این اختلال می‌شود نشان داده شده‌است.

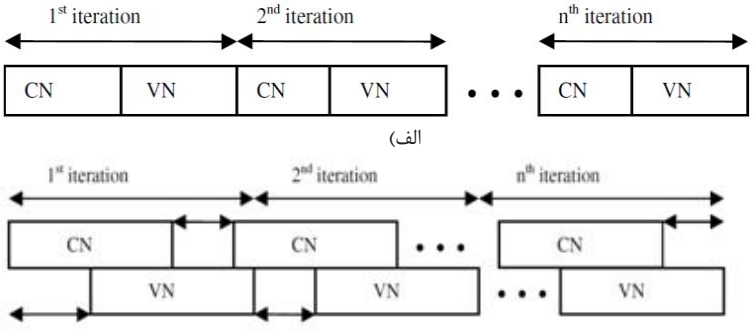
یکی از مسائل مهم در هنگام پیاده‌سازی بحث سرعت می‌باشد. در قسمت پیاده‌سازی برای افزایش سرعت پروسه دکودینگ از روش همپوشانی پیام، استفاده کرده‌ایم. در الگوریتم معمولی دکودینگ، بعد از آنکه پیام به دکودر رسید در چند کلاک عملیات مربوط به گره چک انجام شده و پیام‌ها به گره متغیر می‌روند و سپس در چند کلاک عملیات مربوطه در گره متغیر انجام می‌شود و پیام‌ها به گره چک فرستاده می‌شوند. مشکل اصلی این حالت بی استفاده ماندن دسته‌ای از گره‌ها، در هنگام انجام عملیات مجموعه گره‌های مقابل است (شکل ۴-الف). بنابراین باید به فکر راهی باشیم که قبل از اتمام انجام عملیات یک دسته گره عملیات گره‌های مقابل شروع شده باشد(شکل ۴-ب). در این حالت زمان انجام کل پروسه



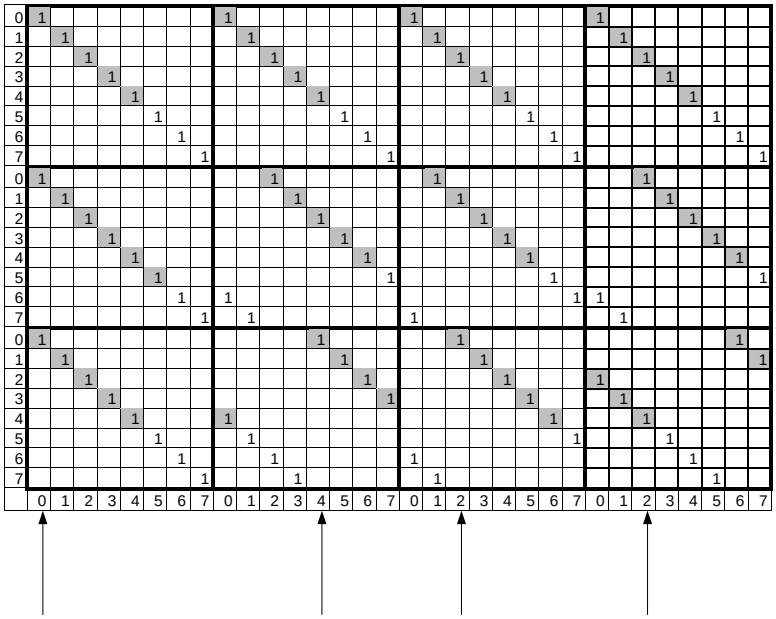

شکل (۳) الف)زیر ماتریس یک کد QC-LDPC ب) نگاشت به شکل معمولی و مشکل تداخل در دسترسی به حافظه ج)اصلاح نگاشت و حل مشکل تداخل

دکودینگ کاهش خواهد یافت.

در کدهای QC-LDPC، زمان لازم برای شروع عملیات ستون‌ها قبل از اتمام پردازش سطرها، به مقدار شیفت زیر ماتریس‌ها و به این که از چه سطری پردازش سطرها را شروع کنیم مرتبط است. ضمنا به خاطر نوع طراحی این کدها، بعد از آماده شدن یک ستون یا سطر برای پردازش، ردیف‌ها و ستون‌های بعدی آماده پردازش به ترتیب قرار خواهند گرفت، که باعث تسهیل امر طراحی خواهد شد. برای مثال به شکل (۵) که ماتریس پریتی چک یک کد QC-LDPC با اندازه زیر ماتریس ۷ است توجه کنید. اگر همه پردازش‌های مربوط به ردیف‌ها از ردیف ۰ شروع شود، بعد از انجام پردازش بر روی ۵ سطر، ستون‌هایی خواهیم داشت که آماده شروع عملیات خواهند بود. پیکان‌های زیر ماتریس نشان می‌دهد که در این زمان کدم ستون‌ها آماده پردازش می‌باشند. در گروه ستون‌های اول، نخستین ستون اطلاعات لازم را بعد از پردازش فقط یک ردیف خواهد داشت، حال آن که در گروه دوم پردازش پنج ردیف نیاز است تا ستونی با پیام‌های به هنگام شد و آماده شروع به عملیات داشته باشیم. به همین طریق برای گروه‌های دیگر نیز می‌توان این مسئله را بررسی کرد. در این مثال حداکثر تعداد ردیف لازم برای شروع پردازش ستون‌ها مربوط به گروه دوم و برابر پنج سطر می‌باشد.




شکل (۴) روش همپوشانی پیام



شکل (۵) ماتریس پریتی چک یک کد QC-LDPC

## بررسی پارامترهای موثر در دکودینگ

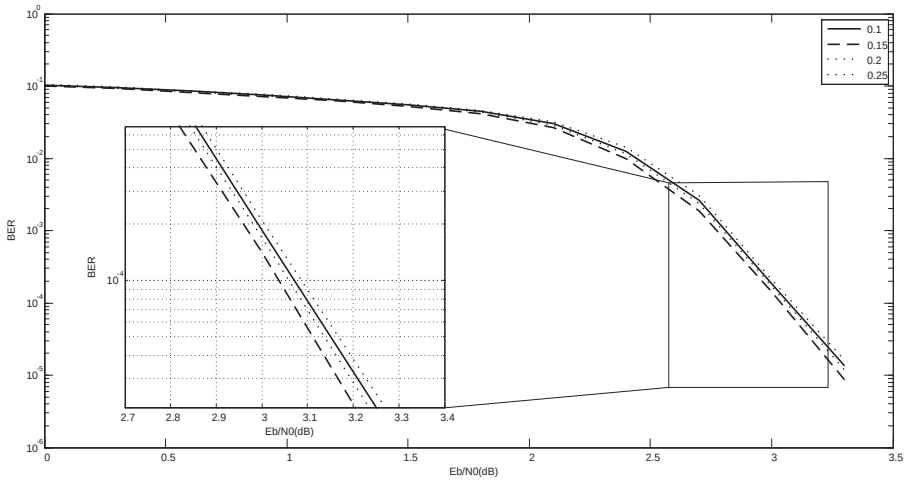
مقدار بهینه $\beta$ با توجه به [۸] بسته به ساختار کد، مقداری بین ۰,۱ و ۰,۲۵ خواهد بود. لذا برای پیدا کردن بهترین مقدار $\beta$ برای کد طراحی شده ، الگوریتم OMS با مقادیر ۰,۱،۰,۱۵،۰,۲و ۰,۲۵ برای این پارامتر شبیه‌سازی شده‌است. در شکل (۶) نتیجه شبیه‌سازی نرخ خطا بر حسب $\frac{E_b}{N_0}$، در $\beta$های مختلف نشان داده شده‌است. همان طور که در این شکل مشاهده می‌شود، بهینه‌ترین مقدار برای کد طراحی شده $\beta = 0.15$ می‌باشد.در شکل (۷) عملکرد الگوریتم‌هایBP، MSوOMS(به ازای $\beta = 0.15$ ) برای کدی که برای سیستم وایمکس طراحی کرده‌ایم، مقایسه شده‌است. در این شکل نرخ خطا برحسب $\frac{E_b}{N_0}$ برای الگوریتم‌های مختلف شبیه‌سازی شده‌است. همان طور که مشاهده می‌شود با استفاده از مقدار بهینه $\beta$الگوریتم OMS توانسته بخشی از خطای ناشی از تقریب در الگوریتم MS را جبران سازد و عملکردی نزدیکتر به الگوریتم BP داشته باشد. در ادامه شبیه‌سازی دکودر LDPC برای تعداد تکرارهای ۱۰ تا ۵۰ (با فاصله گام ۱۰) انجام شده‌است و براساس نتایج آن، احتمال خطای بیت، بر حسب $\frac{E_b}{N_0}$برای تعداد تکرارهای متفاوت در شکل (۸) آورده شده‌است، برای بخش پیاده‌سازی سخت افزاری، حداکثر تعداد تکرار ۳۰ عدد انتخاب شده‌است. البته همان طور که قبلاً نیز اشاره شد، کدهای LDPC به گونه‌ای طراحی و پیاده‌سازی می‌شوند که در صورتی که قبل از رسیدن به حداکثر تعداد تکرار، خطا صفر شود، دکودر به عملیات دیکودینگ خاتمه می‌دهد و خروجی را معتبر اعلام می‌کند.

در شبیه‌سازی‌های انجام شده روی دکودر از روش دکودینگ نرم استفاده شده‌است و اعداد به صورت Floating point به کار رفته‌اند برای پیاده‌سازی سخت‌افزاری، باید طول اعداد محدود باشد؛ لذا شبیه‌سازی‌ها باید به صورت Fixed point تکرار شود. در این بخش، کارایی کدهای QC-LDPC را در حالت Fixed point بررسی خواهیم کرد. نکته مهم دیگر تعداد بیت‌های استفاده شده برای نشان دادن LLR[۱۲]ورودی به دکودر برای پیاده‌سازی سخت افزاری است. چرا که تعداد بیت انتخاب شده برای کوانتیزیشن، مستقیماً روی پیچیدگی، سرعت و عملکرد دکودر ما تأثیر خواهد گذاشت. واضح است که با افزایش بیت‌ها عملکرد دکودر بهبود و پیچیدگی افزایش خواهد یافت. برای مقایسه و انتخاب تعداد بیت برای کوانتیزیشن، شبیه‌سازی برای ۴،۶ و ۸ بیت انجام شده‌است. در شکل (۹) شبیه‌سازی احتمال خطای بیت بر حسب $\frac{E_b}{N_0}$ در تعداد تکرار ۳۰ برای تعداد بیت‌های ذکر شده‌امده‌است. بر اساس نتایج این شبیه‌سازی در بخش پیاده‌سازی سخت افزاری، مقدار۶ بیت انتخاب شده‌است.

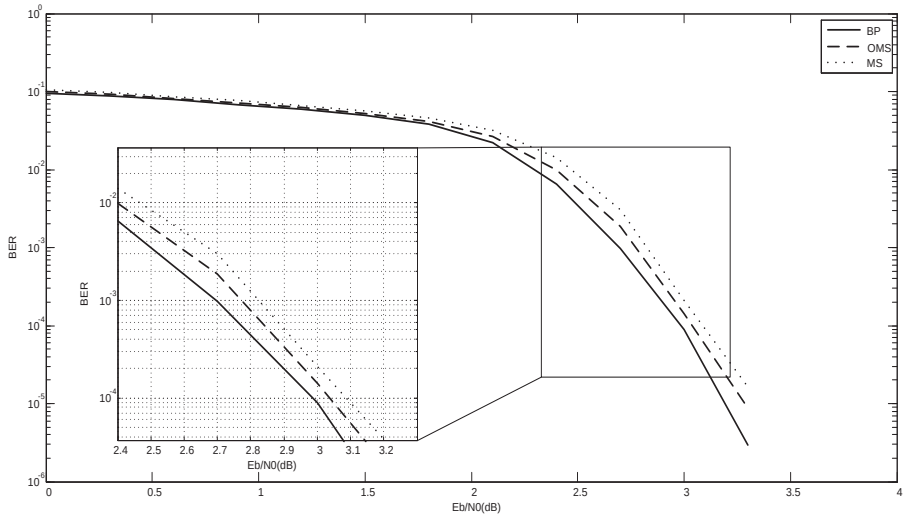
### پیاده‌سازی

همان‌طور که در بخش‌های قبلی ذکر شد، ساختار کد LDPC با استفاده از نرم افزار MATLAB ،ساخته شده و ماتریس پایه انتخاب و طراحی گردیده و با استفاده از توابع نوشته شده در نرم‌افزار MATLAB برای طول کد ۲۳۰۴ و نرخ ۵/۶ به صورت QC-LDPC، گسترش یافته‌است. در مرحله بعد

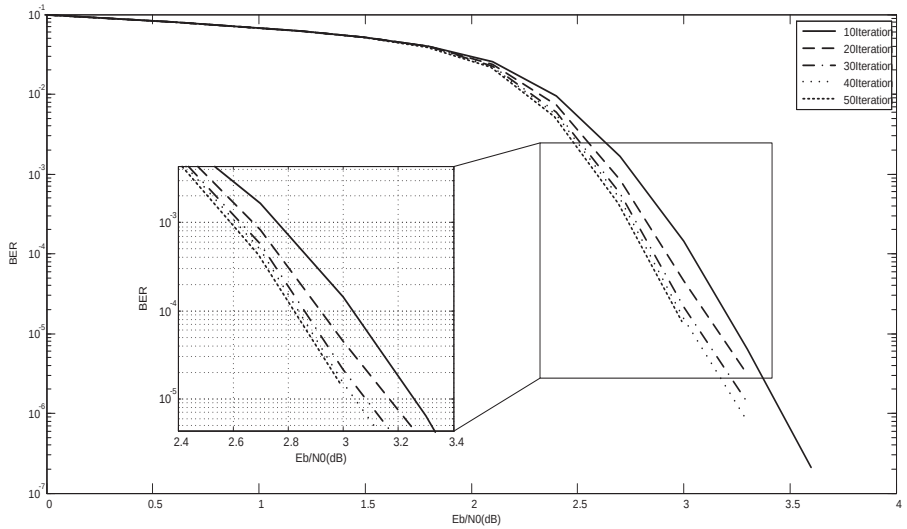
[۱۲] Log Likelihood Ratio



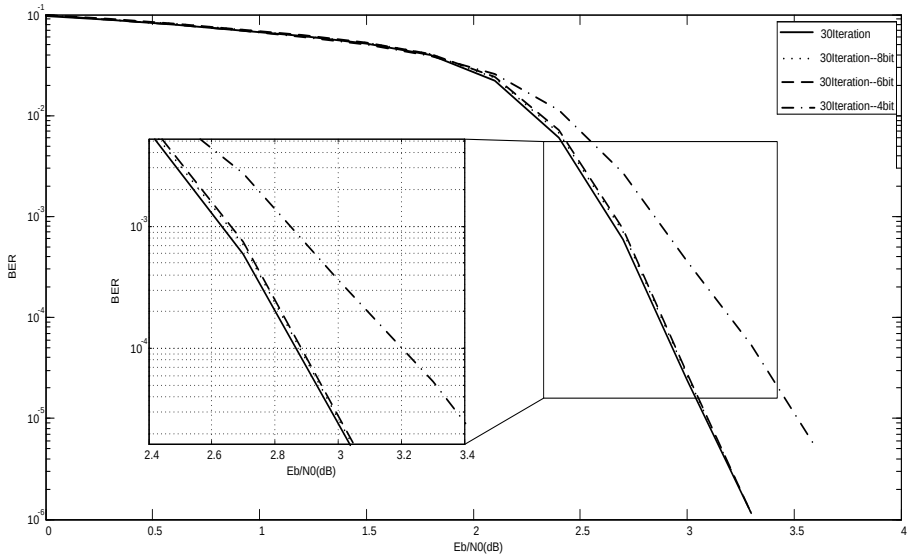

شکل (۶) احتمال خطای بیت بر حسب $\frac{E_b}{N_0}$ برای تغییرات پارامتر $\beta$ در الگوریتم OMS




شکل (۷) مقایسه احتمال خطای بیت بر حسب $\frac{E_b}{N_0}$ برای الگوریتم‌هایBP، MS و OMS




شکل (۸) احتمال خطای بیت بر حسب $\frac{E_b}{N_0}$ در تکرارهای متفاوت برای دکودر طراحی شده




شکل (۹) احتمال خطای بیت بر حسب $\frac{E_b}{N_0}$ برای تعداد بیت‌های مختلف برای دکودر طراحی شده

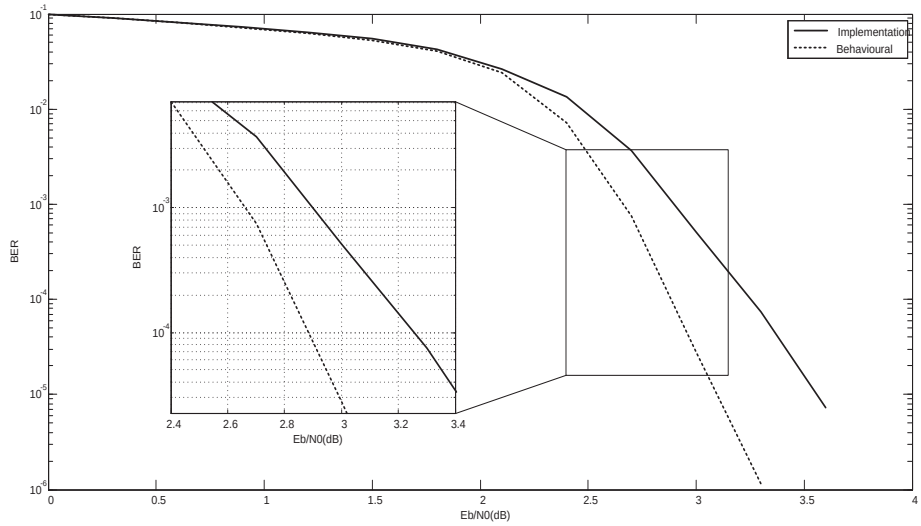
تعداد ۱۰۰۰۰ بلوک با داده‌های رندم به عنوان ورودی وارد انکودر شد و بعد از مدوله شدن با استفاده از مدولاتور BPSK خروجی آن وارد کانال AWGN شد. این عمل برای مقادیر مختلف Eb/N0 (از ۰dB تا ۵dB) انجام شده‌است. خروجی کانال که بیت‌های نویزی شده ما هستند پس از تبدیل شدن به LLR به عنوان ورودی، در حافظه ذخیره شده و برای عملیات دکودینگ به واحد دکودر تحویل می‌شوند. واحد دکودر توسط سخت افزار virtex4lx160Xilinx با پکیج FF1513 پیاده‌سازی شده‌است. از زبان verilog به عنوان زبان توصیف سخت افزاری در تمام قسمت‌ها استفاده شده‌است. از نرم افزار Xilinx ISE برای سنتز و جاسازی استفاده شده‌است. برای بخش شبیه‌سازی نیز از نرم افزار Modelsim۶,۵ استفاده شده‌است. تعداد تکرار برابر۳۰ و تعداد بیت لازم برای کوانتیزیشن ۶ بیت قرار داده شده‌است. نتایج پیاده‌سازی در جدول (۱) نشان داده شده‌است. بیشترین مقدار تأخیر در پیاده‌سازی ۴,۸۲۵نانو ثانیه است، در نتیجه حداکثر فرکانسی که دکودر در آن به درستی کار می‌کند ۵۲,۹ MHz خواهد بود.

جدول (۱) نتایج پیاده‌سازی بر روی FPGA virtex4lx160Xilinx

| Device Utilization Summary | | | |
|---|---|---|---|
| Logic Utilization | Used | Available | Utilization |
| Number of Slice Flip Flops | 33,996 | 135,168 | 25% |
| Number of 4 input LUTs | 120,817 | 135,168 | 89% |
| Number of occupied Slices | 63,956 | 67,584 | 94% |
| Number of Slices containing only related logic | 63,956 | 63,956 | 100% |
| Total Number of 4 input LUTs | 120,827 | 135,168 | 89% |
| Number of bonded IOBs | 733 | 960 | 76% |

در شکل (۱۰) منحنی احتمال خطای بیت بر حسب $\frac{E_b}{N_0}$ برای پیاده‌سازی و نیز برای شبیه‌سازی رفتاری (کوانتیزه شده با ۶ بیت و تعداد ۳۰ تکرار) که در فصل قبل به دست آمده بود، با هم مقایسه شده‌اند، مقدار خطایی که بابت پیاده‌سازی به مدار تحمیل شده‌است در در احتمال خطای $10^{-4}$به اندازه ۰,۴dB، $\frac{E_b}{N_0}$ بیشتری نیاز دارد، که نشان دهنده دقت بالای پیاده‌سازی می‌باشد.

برای مقایسه سطح تراشه و توان مصرفی با کارهای مشابه، دکودر پیشنهادی با استفاده از فناوری $\mu$-CMOS TSMC۰,۱۳ ، در نرم افزار Synopsis Design Compiler و با ولتاژ تغذیه ۱,۲ ولت سنتز شده‌است. در جدول (۲) نتایج سنتز و در جدول (۳) مقایسه این نتایج با کارهای مشابه آورده شده‌است. همانطور که مشاهده می‌شود، دکودر پیشنهادی نسبت به مراجع [۹]و [۱۰]سطح تراشه به مراتب کمتری




شکل (۱۰) منحنی احتمال خطای بیت بر حسب $\frac{E_b}{N_0}$ برای پیاده‌سازی و نیز برای شبیه‌سازی رفتاری

جدول (۲) نتایج سنتز دکودر پیشنهادی با استفاده از فناوری $\mu$ ۰،۱۳CMOS

| | | | | | |
|---|---|---|---|---|---|
| Number of ports | 740 | Combinational area | 3096450 µm² | Cell Internal Power | 95.3556 mW (55%) |
| Number of nets | 8688 | Noncombinational area | 2531758 µm² | Net Switching Power | 78.6404 mW (45%) |
| Number of cells | 787 | Net Interconnect area | 2056 µm² | Total Dynamic Power | 173.9660 mW |
| Number of references | 43 | Total area | 5630264 µm² | | |

جدول (۳) مقایسه سطح تراشه و توان مصرفی دکودر پیشنهادی با کارهای مشابه

| | دکودر پیشنهادی | [۱۱] | [۱۰] | [۹] |
|---|---|---|---|---|
| سطح تراشه | 5.63 mm² | 3.834 mm² | 52.5 mm² | 10.08 mm² |
| توان مصرفی و فرکانس کاری | 174 mW @ 52 MHz | 787 mW @ 125 MHz | 690mW @ 64 MHz | ـــــ |
| طول کد | 2304 | 576~2304 | 1024 | 1024 |
| نرخ کد | 5/6 | 1/2, 2/3, 3/4, 5/6 | 1/2 | ½ |
| ماتریسپریتی چک | QC | QC | Irregular | Regular-(۳،۶) |
| ابعاد فن‌آوری CMOS و ولتاژ تغذیه | 130nm,1.2 V | 130 nm,1.2 V | 160 nm,1.5 V | 180nm,1.8 V |

اشغال کرده‌است و با مرجع [۱۱]نیز قابل مقایسه می باشد. در مورد توان مصرفی هم، دکودر پیشنهاد شده نسبت به سایر کارها برتری دارد. البته بخش عمده‌ای از این اختلاف ها به علت تک نرخی بودن، فرکانس کاری پایین تر و ولتاژ تغذیه کمتر نسبت به کارهای دیگر می باشد. دلیل اصلی بهینه شدن مشخصات دکودر استفاده شده در این مقاله، استفاده از ساختار نیمه موازی به همراه روش‌های پیشنهادی است، که به لطف استفاده از ماتریس پریتی چک QC و پیاده‌سازی مناسب، آن را از کارهای دیگر متمایز ساخته‌است.

## نتیجه‌گیری

در این مقاله یک پیاده‌سازی سخت افزاری کارآمد برای دکودرهای کدهای LDPC برای استاندارد وایمکس با کمترین اختلاف با شبیه‌سازی رفتاری، ارائه شده‌است. از کدهای QC-LDPC به منظور کاهش پیچیدگی سخت افزاری و تطابق آن

برای پیاده‌سازی نیمه موازی استفاده کردیم و ماتریس پریتی چک را با استفاده از این کد و با استفاده از ماتریس مدل پیشنهادی، برای سیستم وایمکس با نرخ ۵/۶ و طول ۲۳۰۴ طراحی کردیم. از میان الگوریتم‌های دکودینگ، الگوریتم Min-Sum بهبود یافته را استفاده کردیم تا ضمن کاهش پیچیدگی سخت افزاری نسبت به الگوریتم انتشار گمان، از افت عملکرد سیستم به علت تقریب زدن تابع گره چک در الگوریتم Min-Sum هم جلوگیری کنیم. یکی از مشکلاتی که در هنگام پیاده‌سازی با آن مواجه شدیم اختلال در دسترسی به حافظه بود که با روش ذکر شده آن را مرتفع کردیم. برای افزایش سرعت دکودر از روش هم پوشانی پیام‌ها [۱۳] استفاده شده‌است تا دسته گره‌های مختلف بتوانند تقریباً به شکل همزمان پردازش انجام دهند. یکی دیگر از بخش‌های اصلی این مقاله بررسی پارامتر های موثر بر روی کارایی دکودر بود، جایی که نقش تعداد تکرارها در پروسه دکودینگ و هم چنین تعداد بیت کوانتیزیشن به عنوان دو عامل مهم هم در عملکرد مدار و هم در مقدار سخت افزار مورد نیاز، بررسی شده و مقدار مناسبی برای پیاده‌سازی انتخاب شد. در ادامه دکدر طراحی شده را با حداکثر دقت بر روی FPGA پیاده‌سازی کردیم و در نهایت شبیه‌سازی سنتز با استفاده از فناوری $\mu$ ۰,۱۳CMOS انجام و نتایج با کارهای مشابه مقایسه شد.

## مراجع


[1] R. G. Gallager, “Low density parity check codes,” *IRE Trans. Inform.Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[2] K. Se-Hyeon and P. In-Cheol, ”Loosely coupled memory-based decoding architecture for low density parity check codes” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, pp. 1045-1056, 2006.

[3] Y. Engling, P. Pakzad, B. Nikolic, and V. Anantharam, ”VLSI architectures for iterative decoders in magnetic recording channels” *IEEE Transactions on Magnetics*, vol. 37, pp. 748-755, 2001.

[4] C. Zhiqiang, W. Zhongfeng, Z. Xinmiao, and J. Qingwei, ”Efficient decoder design for high-throughput LDPC decoding,” *IEEE Asia Pacific Conference on Circuits and Systems, 2008. APCCAS 2008.*, 2008, pp. 1640-1643.

[5] L. Yang, H. Liu, and C.-J. R. Shi, "Construction and FPGA implementation of low-error-floor multi-rate low-density parity-check code decoders", *IEEE Trans. on Circuits and Systems-I*, vol. 53, no. 4, pp. 892-904, April 2006.

[6] ”J. Sun, "An Introduction to Low Density Parity Check (LDPC) Codes,”*WirelessCommunication Research Laboratory(WCRL), Lane Dept. of Comp. Sci. andElec. Engr., West Virginia University*, June 2003.."

[7] C. Yanni and K. K. Parhi, ”Overlapped message passing for quasi-cyclic low-density parity check codes,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, pp. 1106-1113, 2004.

[8] C. Jinghu and P. M. C. Fossorier, ”Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions,” *IEEE Global Telecommunications Conference, 2002.*, 2002, pp. 1378-1382 vol.2.

[9] K. Sungwook, G. E. Sobelman, and M. Jaekyun, ”Parallel VLSI architectures for a class of LDPC codes,” *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002.*, 2002, pp. II-93-II-96 vol.2.

[10] ”IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1," *IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004)*, pp. 0_1-822, 2006.

[11] T. Brack, M. Alles, F. Kienle, and N. Wehn, ”A Synthesizable IP Core for WIMAX 802.16E LDPC Code Decoding,” *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, 2006*, 2006, pp. 1-5.


[۱۳] Message Overlapping