تسریع عملکرد کدهای توربو توسط شاخه های موازی کدبردار و کدگذار

فاطمه افقه
دانشگاه صنعتی خواجه نصیر
f_afghah@yahoo.com

مهرداد اردبیلی پور
دانشگاه صنعتی خواجه نصیر
mehrdad@eetd.kntu.ac.ir

ابوالفضل راضی
شرکت ارتباطات سیار
abolfazl.razi@gmail.com

چکیده- تاخیر طولانی در عملیات کدبرداری کدهای توربو، یکی از مهمترین محدودیت های استفاده از این کدها در کاربردهای عملی به شمار می آید. در این مقاله به بررسی و شبیه سازی روش پیاده سازی کدبردار و کدگذار توربو توسط شاخه های موازی می پردازیم. در این شیوه، با تقسیم شاخه اصلی کدبردار و کدگذار توربو به چندین شاخه فرعی موازی و تقسیم نمودن فریم اطلاعات بین این زیرشاخه ها، امکان انجام مستقل عملیات هر زیرشاخه، توسط یک پردازشگر مجزا فراهم می گردد. نتایج شبیه سازی این مقاله نشان می دهد که این روش علاوه بر اینکه موجب کاهش چشمگیر زمان کدگذاری و کدبرداری کدهای توربو، تقریبا متناسب با عکس تعداد شاخه ها، می گردد، دارای عملکردی نزدیک و قابل قیاس با کدهای توربو عادی می باشد.

کلید واژه- تاخیر کدبرداری، کد های توربو، کدبردار موازی توربو، کدگذار موازی توربو.

۱- مقدمه

در سال ۱۹۴۸، شانون در مقاله ای به طور نظری نشان داد که یک سیستم مخابراتی با توان، پهنای باند و رفتار تخریبی کانال مشخص، دارای ظرفیت مشخص می باشد، به طوری که اگر نرخ داده از ظرفیت سیستم کمتر باشد، کدهایی وجود خواهند داشت که با استفاده از آن ها می توان اطلاعات را با خطای بسیار ناچیز از طریق سیستم مخابراتی منتقل نمود[۱]. تا اوایل دهه ۹۰، تلاش های دانشمندان در زمینه یافتن کدهای قابل پیاده سازی ای که دارای عملکرد نزدیک به حد شانون باشند، به نتیجه مطلوبی نرسیده بود، تا اینکه در سال ۱۹۹۳، Berrou و گروهش،[۲] موفق به معرفی کدهای توربو شدند که با داشتن تفاوت عملکرد تنها در حدود 0.7 dB با حد ظرفیت شانون، امید تازه ای برای رسیدن به این حد ظرفیت ایجاد کردند. ایده اصلی که موجب عملکرد حیرت انگیز این کدهای زنجیره ای گشت، استفاده از روش کدبرداری تکراری بود که ضمن کارایی مناسب از پیچیدگی کمی نیز برخودار بود. ابداع این کدها دریچه جدیدی بر روی محققان علم کدینگ باز نموده و توجه آنان را به کدبرداری تکراری معطوف نمود[۳].

یکی از عوامل قابل تامل در به کارگیری این کد ها در سیستم های زمان حقیقی، تاخیر طولانی در عملیات کدبرداری آنها می باشد. این مسئله انگیزه انجام تحقیقات بسیاری به منظور برطرف کردن محدودیت مذکور گشته است. یکی از بررسی های انجام شده در این زمینه، بررسی امکان عملکرد همزمان دو کدبردار SISO بوده است که نتیجه آن کاهش تاخیر زمان کدبرداری تا حدود نصف می باشد[۴].

در این مقاله نشان می دهیم که زمان تاخیر کدبرداری کدهای توربو با طول فریم داده نسبت مستقیم دارد و از آنجا که در اکثر کاربردهای عملی با فریم های طولانی مواجه می باشیم، بنابراین یافتن تکنیک های جدیدی که بتواند باعث کاهش زمان عملیات کدگذاری و به خصوص کدبرداری کدهای توربو شود، بسیار حائز اهمیت می باشد. در این مقاله به بررسی و شبیه سازی ایده تقسیم شاخه های اصلی کدگذار و کدبردار توربو به زیرشاخه های موازی پرداخته و نشان می دهیم که کدهای توربو چند شاخه، علاوه بر اینکه دارای تاخیر زمانی عملیات کدگذاری و کدبرداری بسیار کمتری نسبت به کدهای توربو عادی می باشند، عملکردی نزدیک و قابل مقایسه با این کدها دارند.

در ادامه مقاله، پس از معرفی مشخصات کدگذار و کدبردار کد توربو ای که مبنای شبیه سازی قرار گرفته است، به محاسبه زمان کدبرداری کدهای توربو پرداخته و پس از آن ساختار و نحوه عملکرد کدهای توربو چندشاخه را معرفی خواهیم نمود.

۲- کدگذار توربو

کدگذار توربو شامل ترکیب موازی دو کدگذار کانولوشنال سیستماتیک بازگشتی(RSC) است که توسط یک بلوک لایه گذار[1] از هم جدا شده اند. در این روش کدکردن، دنباله اطلاعات به طور مستقیم به یکی از کدگذارهای کانولوشنال وارد شده و نیز همین دنباله پس از گذشتن از یک بلوک لایه گذار، وارد کدگذار کانولوشنال دیگر می شود. دنباله کدشده حاصل شامل خروجی کدگذار اول (بیت های سیستماتیک و بیت های پریتی) و قسمت پریتی کدگذار دوم می باشد(الگوریتم غربالگری یک در میان).

۳- کدبردار تکراری توربو

طراحی کدبردار توربو با الهام از ساختار کدگذار آن، به صورت ترکیب دو عنصر کدبردار ورودی-خروجی نرم (SISO) می باشد که توسط یک لایه گذار به یکدیگر متصل شده اند. عملکرد این کدبردار بر اساس استفاده از تخمین رشته ارسالی در خروجی هر یک از کدبردارها و استفاده از همین تخمین در ورودی کدبردار دیگر و تکرار این عملیات(کدبرداری تکراری) می باشد. این روش موجب می گردد که کدبرداری با عملکرد مناسب، نرخ خطای کم و در عین حال با حجم محاسبات پایین قابل پیاده سازی باشد. روش های متفاوتی برای کدبرداری توربو وجود دارد که از جمله آنها، الگوریتم های Log-Map، Max-Log-Map و SOVA می باشند. در این مقاله روش Log-Map مبنای شبیه سازی قرار گرفته است[۵]. در این روش کدبرداری به منظور تخمین داده ارسالی، پارامتری به نام لگاریتم شرطی نسبت احتمالات مورد استفاده واقع می شود که طبق رابطه(۱)، محاسبه می شود و علامت آن نشانگر غالب بودن احتمال صفر بودن و یا یک بودن بیت اطلاعات ارسالی u_k به شرط بیت دریافتی در گیرنده (y) و قدرمطلق آن بیانگر میزان اطمینان این احتمال می باشد.

$$L(u_k) \approx \ln\left(\frac{P(u_k=+1/y)}{P(u_k=-1/y)}\right) \qquad (۱)$$

طبق مرجع[۵] ، این پارامتر از رابطه (۲) محاسبه می گردد.

$$L(u_k/y) = \ln\left(\frac{\sum_{u_k=+1}^{(\hat{s},s)\Rightarrow} \exp(\mathrm{A}_{k-1}(\hat{s})+\Gamma_k(\hat{s},s)+\mathrm{B}_k(s))}{\sum_{u_k=-1}^{(\hat{s},s)\Rightarrow} \exp(\mathrm{A}_{k-1}(\hat{s})+\Gamma_k(\hat{s},s)+\mathrm{B}_k(s))}\right) \qquad (۲)$$

قبل از محاسبه رابطه فوق، بایستی پارامترهای $\mathrm{A}_k(s)$ (احتمال اینکه ترلیس در لحظه k در حالت s باشد و رشته دریافتی از ابتدا تا این لحظه برابر $y_{j<k}$ باشد) و $\mathrm{B}_k(s)$ (احتمال اینکه ترلیس در لحظه k در حالت s باشد و رشته دریافتی از این لحظه تا آخر برابر $y_{j>k}$ باشد) محاسبه گردند که از روابط بازگشتی (۳) و (۴) بدست می آیند.

$$\mathrm{A}_k(s) \cong \ln(\alpha_k(s)) = \ln\left(\sum_{all\ \hat{s}} \exp[\mathrm{A}_{k-1}(\hat{s})+\Gamma_k(\hat{s},s)]\right) \qquad (۳)$$

$$\mathrm{B}_{k-1}(\hat{s}) \cong \ln(\beta_k(s)) = \ln\left(\sum_{all\ s} \exp[\mathrm{B}_k(s)+\Gamma_k(\hat{s},s)]\right) \qquad (۴)$$

همچنین $\Gamma_k(\hat{s},s)$ (احتمال اینکه رشته دریافتی در لحظه k برابر y_k بوده و وضعیت ترلیس از حالت $\hat{s}$ در لحظه $k-1$ به حالت s در لحظه k تغییر حالت بدهد) ، مطابق رابطه (۵) تعریف می گردد.

$$\Gamma_k(\hat{s},s) = \sum_{i=1}^{n} \ln p(y_k/x_{ki}) + \ln C + u_k L(u_k)/2 \qquad (۵)$$

که در آن ثابت C از رابطه (۶) بدست می آید.

[1] - Interleaver

$$C = \frac{e^{-L(u_u)/2}}{1+e^{-L(u_u)}} \qquad (۶)$$

این روش کدبرداری به دلیل استفاده از خروجی هر کدبردار در کدبردار دیگر، دارای ماهیت تکراری می باشد.

۴- محاسبه زمان تاخیر محاسبات عملیات کدبرداری

در این بخش به محاسبه زمان تاخیر محاسبات مورد نیاز در روش کدبرداری Log-Map، با فرض طول فریم n، تعداد حالات 2^m در دیاگرام ترلیس و نیز معلوم فرض کردن جدول(۱) برای زمان مورد نیاز محاسبات ریاضی با توجه به نوع و توان پردازشگر مورد استفاده، می پردازیم.

جدول(۱): زمان مورد نیاز محاسبات ریاضی اصلی

جمع	ضرب	تقسیم	توان	لگاریتم
T_{sum}	T_{mul}	T_{div}	$T_{\exp}$	$T_{\log}$

حجم محاسبات مربوط به هر بار تکرار به صورت زیر بدست می آید.

الف) زمان مورد نیاز برای محاسبه ضرایب $\Gamma_k(\hat{s},s)$

برای بدست آوردن زمان مذکور، رابطه (۵) را به صورت رابطه (۷) بازنویسی می کنیم.

$$\Gamma_k(\hat{s},s) = \sum_{i=1}^{n} \ln p(y_k / x_{ki}) - \ln(1+\exp(-L(u_k))) + (\frac{u_k - 1}{2})L(u_k) \qquad (۷)$$

برای محاسبه کل زمان تاخیر محاسبات رابطه فوق در طول هر تکرار، ابتدا باید زمان لازم برای عملیات محاسبه ضریب $\Gamma_k(\hat{s},s)$ هر شاخه تخمین زده شود. در این محاسبه باید این نکته را مورد توجه قرار داد که پارامترهای $\ln(1+\exp(-L(u_k)))$ و $(\frac{u_k-1}{2})L(u_k)$، برای هر مرحله kام فقط یکبار محاسبه می گردند، بنابراین تاخیر محاسبه هر کدام از ضرایب فوق، با توجه به اینکه u_{ki} دارای دو حالت ممکن ۱ و ۱- می باشد، طبق رابطه (۸) برابر است با:

$$T_\Gamma = 3T_{sum} + 2T_{\log} \qquad (۸)$$

با توجه به اینکه تعداد کل Γهای غیر صفر برابر $n\times 2\times 2^m$ می باشد و نیز با در نظرگرفتن زمان مورد نیاز محاسبه پارامترهای ثابت هر مرحله یعنی $\{\ln(1+\exp(-L(u_k)))\}$ و $\{(\frac{u_k-1}{2})L(u_k)\}$، کل زمان لازم برای محاسبه ضرایب $\Gamma_k(\hat{s},s)$ به صورت رابطه (۹) محاسبه می گردد.

$$T_{\Gamma total} = k\times 2\times 2^m \times (2T_{\log} + 3T_{sum}) + k\times(2T_{sum} + T_{div} + T_{\log} + T_{\exp}) \qquad (۹)$$

ب) زمان مورد نیاز برای محاسبه ضرایب $\mathrm{A}_k(s)$

با توجه به رابطه(۳) و دقت به این نکته که $\Gamma_k(\hat{s},s)$، تنها برای دو مقدار $\hat{s}$ غیر صفر است، زمان لازم برای محاسبه هر پارامتر A_k از رابطه (۱۰) بدست می آید:

$$T_{\mathrm{A}} = 3T_{sum} + 2T_{\exp} + T_{\log} \qquad (۱۰)$$

حال با توجه به اینکه تعداد A_kها برابر $2^m\times k$ می باشد، زمان کل مورد نیاز برای محاسبه A_kها مطابق رابطه (۱۱) می باشد.

$$T_{\mathrm{A}total} = k\times 2^m \times (3T_{sum} + 2T_{\exp} + T_{\log}) \qquad (۱۱)$$

ج) زمان مورد نیاز برای محاسبه ضرایب $\mathrm{B}_{k-1}(\hat{s})$

با توجه به رابطه(۴)، به طور مشابه قسمت(ب)، زمان مورد نیاز برای محاسبه B_kها نیز از رابطه (۱۱) حاصل می گردد.($T_{Btotal} = T_{Atotal}$)

د) زمان مورد نیاز برای محاسبه خروجی $L(u_k / y)$

با توجه به رابطه(۲) و اینکه تعداد $\Gamma_k(\hat{s},s)$ معتبر برای هر k برابر $2^m\times 2$ می باشد، تاخیر محاسبه این پارامتر مطابق رابطه (۱۲) می باشد:

$$T_L = k\times[T_{\log} + T_{div} + 2\times 2^m(T_{\exp} + 3T_{sum})] \qquad (۱۲)$$

بنابراین زمان لازم برای انجام کدبرداری در هر یک از کدبردارها و برای هر مرتبه تکرار در روش Log-Map برابر رابطه (۱۳) است:

$$T = T_{Atotal} + T_{Btotal} + T_{\Gamma total} + T_L$$
$$= k \times [2^{m+1}(3T_{\log} + 9T_{sum} + 3T_{\exp}) + 2T_{sum} + 2T_{div} + 2T_{\log} + T_{\exp}] \quad (۱۳)$$

ملاحظه می گردد که این زمان تاخیر انجام عملیات با طول فریم یعنی k نسبت مستقیم دارد و این نکته می تواند باعث ایجاد محدودیت هایی در کاربرد این کدها گردد.

۵- کاهش زمان کدبرداری

همان طور که قبلا اشاره کردیم، عملکرد توربو کدها به ظرفیت شانون بسیار نزدیک می باشد و این خصوصیت موجب شده است تا این کدها در سیستم هایی که عملکردی مطلوب و نزدیک به ایده آل نیاز دارند و همچنین با محدودیت پهنای باند مواجه می باشند، مورد استفاده واقع گردند. البته همان طور که در بخش قبل محاسبه کردیم، روش کدبرداری تکراری، تاخیر قابل توجهی ایجاد نموده و در نتیجه باعث فاصله گرفتن سیستم های حاوی کدهای توربو از حالت زمان حقیقی می گردد. به همین دلیل یافتن روش هایی که بتواند باعث کاهش زمان تاخیر گردد، از جمله موضوعاتی است که همواره مورد توجه محققان بوده است. در این مقاله به پیاده سازی ایده ای می پردازیم که به میزان قابل توجهی قادر به کاهش زمان کدبرداری و کدگذاری توربو می باشد.

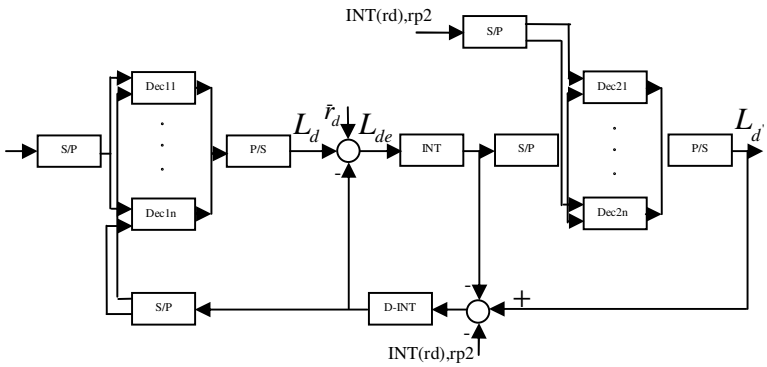
چنانچه پیش از این ذکر کردیم، بخش عمده تاخیر کدهای توربو به دلیل استفاده از روش کدبرداری تکراری می باشد. از آنجا که در هر یک از دورهای تکرار این روش، هر کدام از کدبردارها از اطلاعات اکتسابی بدست آمده از کدبردار دیگر به عنوان ورودی، در تخمین داده استفاده می نمایند، در نتیجه این کدبردارها قادر به عملکرد موازی و همزمان با یکدیگر نمی باشند. این موضوع یکی از عوامل کند شدن فرایند کدبرداری به حساب می آید، خصوصا اینکه معمولا فریم های اطلاعات بسیار طولانی بوده و در نتیجه در هر یک از دورهای تکرار، زمان قابل توجهی به کدبرداری یک فریم اختصاص می یابد، که در ضمن این مدت کدبردار دیگر ناچار به منتظر ماندن برای دریافت نتیجه بوده و بنابراین قادر به انجام عملیات نمی باشد.

در راستای کاهش زمان کدگذاری و خصوصا کدبرداری فریم های طولانی، در این مقاله، هر کدام از کدگذارهای RSC و کدبردارهای SISO، به چند شاخه موازی مشابه تقسیم شده اند که هر شاخه برای انجام عملیات بر روی کسری از یک فریم در نظر گرفته شده است. ایده پیاده سازی کدگذارها و کدبردارهای RSC به وسیله n شاخه فرعی موازی، موجب می گردد که در صورتی که عملیات هرکدام از این شاخه ها توسط یک پردازشگر مستقل و در نتیجه به صورت موازی و همزمان با هم صورت پذیرد، زمان لازم برای انجام عملیات کدگذاری و کدبرداری تقریبا به میزان $\frac{1}{n}$ کاهش یابد و به این ترتیب یکی از مشکلات استفاده از کدهای توربو یعنی تاخیر زمانی طولانی عملیات کدبرداری به میزان چشمگیری کاهش می یابد. البته استفاده از این روش باعث اندکی افت عملکرد سیستم می شود که این افت عملکرد نسبت به افزایش طول فریم روند کاهشی دارد، به طوری که در فریم های با طول بزرگ عملکرد این روش به کدهای توربو عادی بسیار نزدیک می باشد. در این روش می توان متناسب با مشخصات مورد نیاز سیستم و در نظر گرفتن میزان محدودیت زمانی، تعداد بهینه برای شاخه های موازی را تعیین نمود و بدین ترتیب بین کاهش زمان تاخیر و عملکرد سیستم یک توازن برقرار نمود.

۶- پیاده سازی کدبردار و کدگذار توربو توسط شاخه های موازی

در این بخش به بیان مشخصات و بررسی نحوه پیاده سازی روش مذکور می پردازیم. بدین منظور در شکل های ۱ و ۲ بلوک دیاگرام کدگذار و کدبردار کدهای توربو چند شاخه به ترتیب با n_c و n_d شاخه موازی آورده شده است.

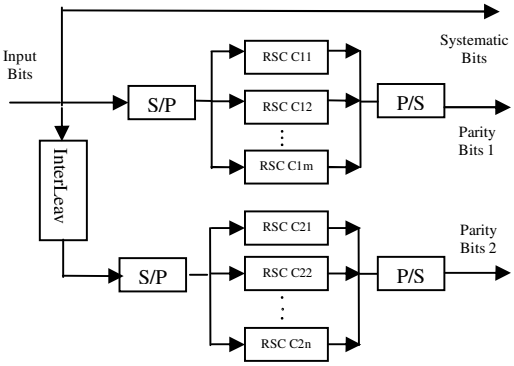
در این مقاله کد توربو با چند جمله ای مشخصه [7;5] و نرخ کد $\frac{1}{2}$، توسط الگوی غربالگری یک در میان



شکل۲: کدگذار توربو توسط شاخه های موازی

پریتی ها و لایه گذار تصادفی مبنای شبیه سازی قرار گرفته است. در این روش پیاده سازی، در هر یک از دو شاخه اصلی کدگذار، به جای قرار دادن یک عنصر کدگذار RSC، از یک مجموعه کدگذار با همان مشخصات و به صورت موازی استفاده نموده ایم. در این شیوه، اگر تعداد کدگذارهای موازی n_c باشد، فریم ارسالی را قبل از اعمال به هر شاخه کدگذار، توسط مبدل سریال به موازی، به n_c فریم کوچکتر با طول $\frac{n}{n_c}$ تبدیل نموده و سپس هر کدام از این فریم های کوچکتر را کدکرده و از مبدل موازی به سریال عبور می دهیم تا خروجی نهایی بدست آید.

بصورت مشابه در کدبردار نیز در هر یک از دو شاخه اصلی ، به جای قرار دادن یک عنصر کدبردار SISO، از یک مجموعه کدبردار با همان مشخصات، به صورت موازی استفاده می نماییم. اگر تعداد کدبردارهای موازی n_r باشد، ابتدا فریم دریافتی را توسط مبدل سریال به موازی، به n_r فریم کوچکتر با طول $\frac{n}{n_r}$ تبدیل نموده، سپس هر کدام از آنها به یکی از کدگذارهای موازی اعمال می شود و با داشتن اطلاعات پیشین کدبردار دیگر کدبرداری شده و خروجی های حاصل از مبدل موازی به سریال عبور داده می شود تا خروجی نهایی به طول n بدست آید.

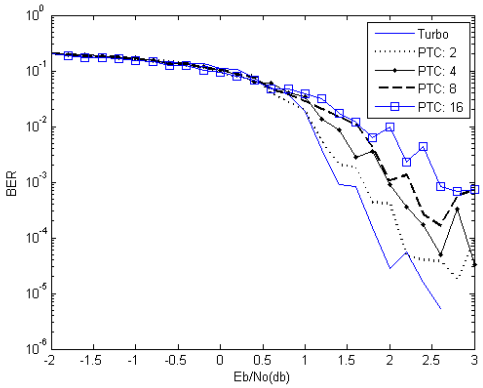


شکل۱: کدگذار توربو توسط شاخه های موازی

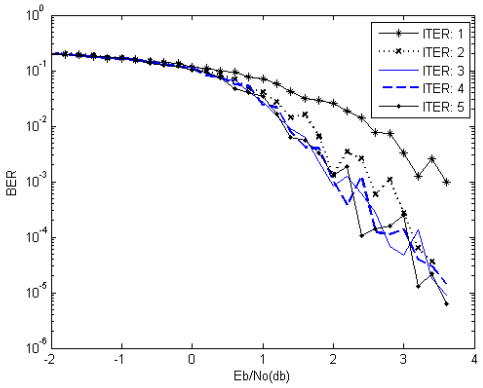
۷- نتایج شبیه سازی

در شکل ۳، عملکرد کد توربو عادی دارای طول فریم ۱۶۰۰ و تعداد ۱۰ فریم ارسالی با کد توربو طراحی شده با تعداد شاخه ۲، ۴، ۸ و ۱۶مقایسه شده است. با توجه به این شکل می توان نتیجه گرفت که کدهای توربو مذکور، ضمن داشتن زمان کدبرداری بسیار کمتر، افت عملکرد چندانی نسبت به کد توربو معمولی ندارند.

در شکل ۴، عملکرد کد توربو دارای چهار شاخه موازی برای تعداد تکرارهای ۱، ۲، ۳، ۴ و ۵ رسم شده است و بیانگر این است که همانند کدهای توربو عادی با افزایش تعداد تکرارها عملکرد این کدها بهبود می یابد.

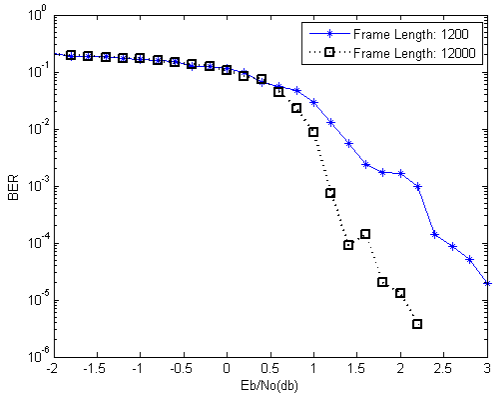


شکل۳: مقایسه عملکرد توربو ۲، ۴،۸ و۱۶ شاخه با توربو عادی



شکل۴: عملکرد توربو با ۴ شاخه موازی به ازای تعداد تکرارهای مختلف

در شکل ۵، عملکرد کد توربو با چهار شاخه موازی برای طول فریم های ۱۲۰۰ و ۱۲۰۰۰ با یکدیگر مقایسه شده است. با توجه به شکل در می یابیم که افزایش طول فریم باعث بهبود عملکرد کدهای توربو مذکور میگردد.



شکل ۵: مقایسه عمکلرد توربو با ۴ شاخه موازی در طول فریم های **۱۲۰۰** و **۱۲۰۰۰**

۸- نتیجه گیری

در این مقاله، شاخه های اصلی کدگذار و کدبردار توربو را به چند زیرشاخه موازی تقسیم نموده و بدین ترتیب با فراهم کردن امکان پردازش همزمان چندین پردازشگر موازی، زمان عملیات کدگذاری و کدبرداری را به میزان قابل توجهی کاهش داده ایم، همچنین مقایسه نتایج این روش با کدهای توربو معمولی، حاکی از عملکرد مناسب و قابل قبول این شیوه به خصوص برای فریم های بزرگ می باشد.

سپاسگذاری

از مرکز تحقیقات مخابرات که این پروژه را تحت حمایت مالی خود قرار داده تقدیر و تشکر می گردد.

مراجع

[1] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal 27*, pp. 379–427, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon Limit error- correcting coding and decoding: turbo-codes,” in *Proc. IEEE Int. Comm. Conf*., Geneva, pp. 1064-1070, May 1993.

[3] A.Burr, “Turbo-codes: the ultimate error control codes?,” *Electronics & Communication Engineering Journa*l, pp. 155-165, August 2001.

[4]Y. Wang, J. Zhang, M. Fossorier, J. S. Yedidia, “Reduced latency turbo decoding,” in *Proc. IEEE Sixth SPAWC Workshop*, pp. 930-934.

[5] P. Robertson, E. Villebrun and P. Hoeher, “A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain,” in *Proceedings of the International Conference on Communications, (Seattle, United States)*, pp.1009-1013, June 1995.