دانشگاه صنعتی شریف

دانشکده مهندسی برق

طراحی سیستم های مبتنی بر FPGA/ASIC

گزارش نهایی پروژه

استاد درس: دکتر مهدی شعبانی

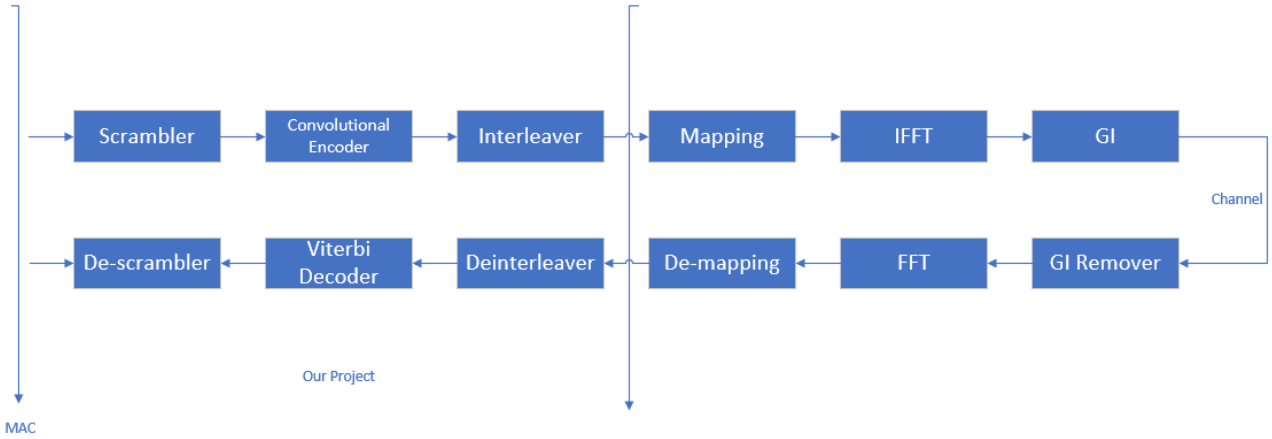
نام و نام خانوادگی: امیرحسین اسدیان

شماره دانشجویی: ۹۶۱۰۱۱۸۷

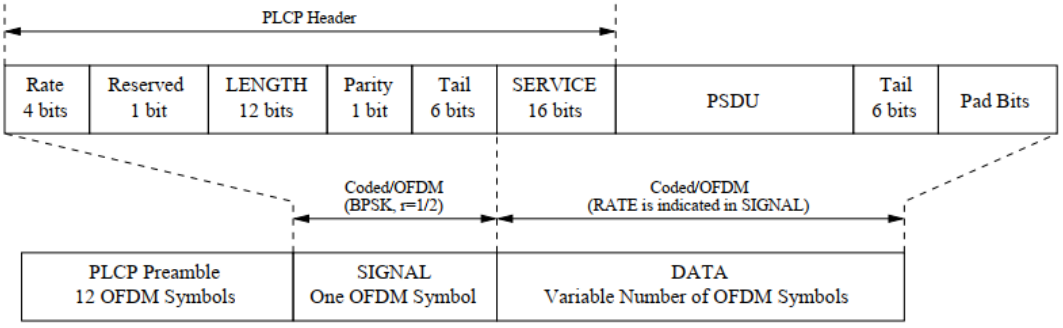
بهمن ماه ۱۳۹۹

مقدمه

در این گزارش طراحی و پیاده سازی لایه فیزیکی استاندارد ۸۰۲.۱۱a در سمت گیرنده و فرستنده مورد بحث قرار گرفته است. بلوک های استفاده شده در سمت فرستنده در شکل زیر آورده شده است



لایه فیزیکی فرستنده داده ها را از لایه MAC می گیرد و به ترتیب عملیات مختلف بلوک های بالا بر روی آن انجام می شود. سپس داده ها ارسال می شوند و در سمت گیرنده پس از دریافت و گذر از بلوک های نمایش داده شده به لایه MAC داده می شوند. داده ها بر طبق فرمت زیر از لایه MAC گرفته می شوند:



در ادامه به توضیح هر یک از بلوک های طراحی شده در این پروژه می پردازیم.

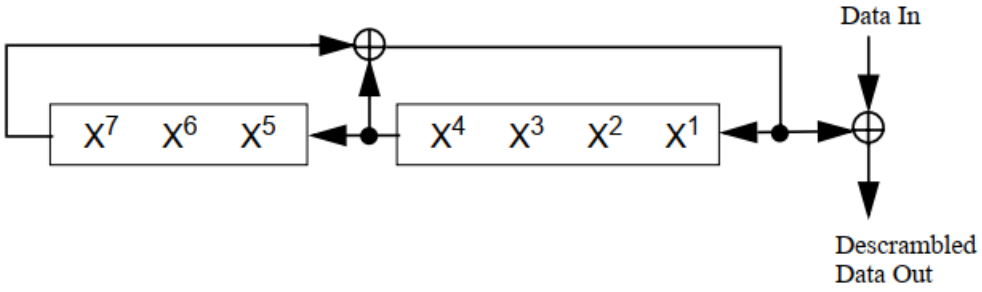
بورد انتخاب شده: Virtex۶، xc۶vcx۷۵t-۱۱ff۴۸۴

داده های مربوط به استفاده از منابع مربوط به این سخت افزار است.

کدهای HDL در نرم افزار ISE Design Suite ورژن ۱۴.۷ نوشته و سنتز شده است و همچنین کدهای تست بنچ در نرم افزار Modelsim ورژن ۱۰.۶d شبیه سازی شده اند.

بلوک Scrambler و Descrambler:

در استاندارد ۸۰۲.۱۱a از بلوک Scrambler زیر استفاده می شود. این بلوک ۷ رجیستر و دو XOR دارد. ۱۲۷ حالت مختلف را ایجاد می کند و در هر پالس ساعت یک خروجی می دهد. ۷ بیت اول از بیت های SERVICE است و حالت اولیه یک مقدار غیر صفر است تا بتوانیم با دادن ۷ بیت صفر ورودی در ابتدا و ذخیره مقدار داده شده، Descramble را در گیرنده انجام دهیم.

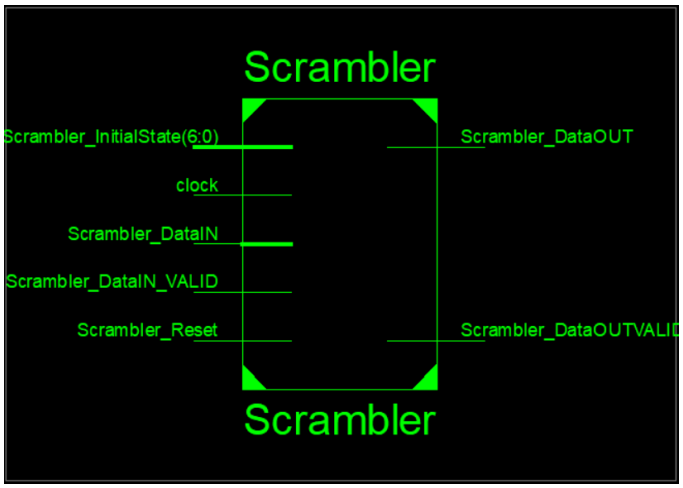


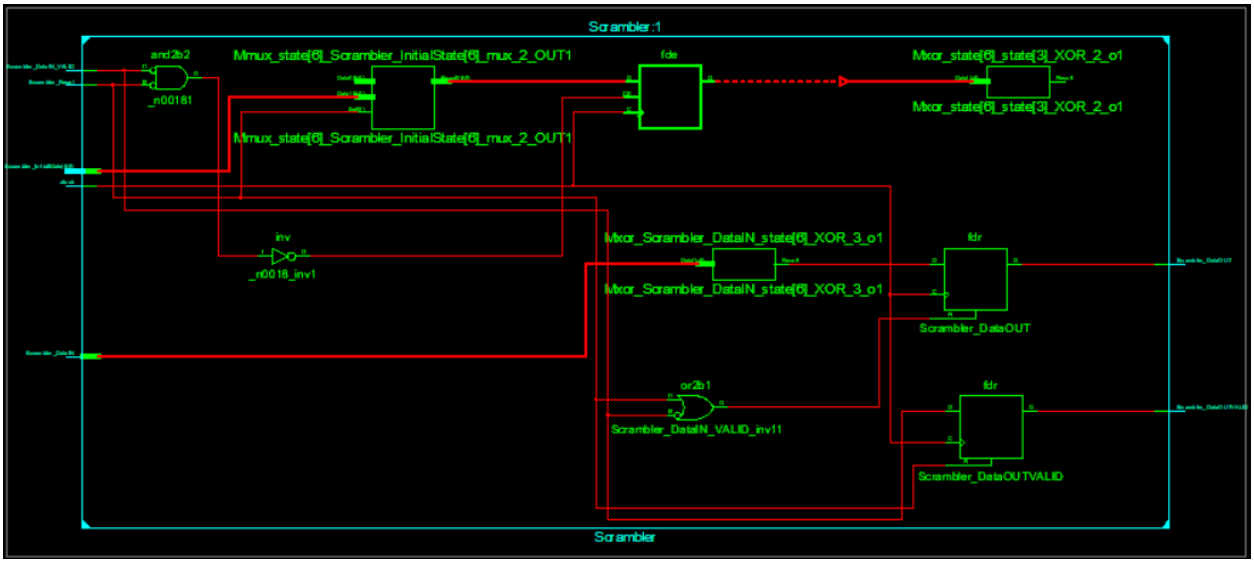
داده های خروجی این بلوک به convolutional Encoder در سمت گیرنده و در سمت فرستنده به لایه MAC ارسال می شوند. بلوک های Scrambler و Descrambler مشابه یکدیگر هستند.

سیگنال های ورودی و خروجی:

- کلاک و ریست
- حالت اولیه (ورودی ۷ بیتی برای تنظیم حالت اولیه در Scrambler، به عنوان ورودی تعریف شده است تا اگر نیاز بود بسته به شرایط تغییر کند، ممکن باشد.)
- دیتای ورودی (تک بیت ورودی) به همراه سیگنال اعتبار سنجی (مشخص می کند که آیا داده های ورودی اعتبار دارند یا خیر)
- دیتای خروجی (تک بیت خروجی) به همراه سیگنال اعتبار سنجی

تصویر بلوک طراحی شده: (دو بلوک Scrambler و Descrambler مشابه یکدیگرند.)



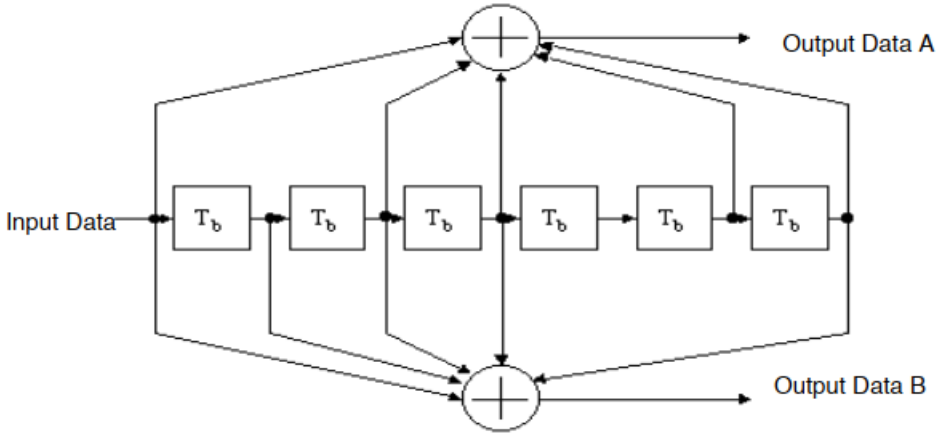


استفاده از منابع:

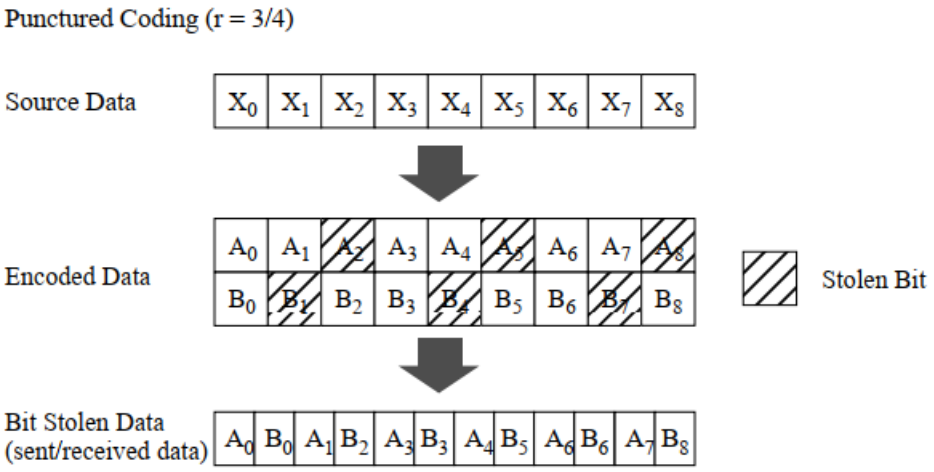
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	8	93120	0%
Number of Slice LUTs	10	46560	0%
Number of fully used LUT-FF pairs	8	10	80%
Number of bonded IOBs	13	240	5%
Number of BUFG/BUFGCTRLs	1	32	3%

بلوک Convolutional Encoder:

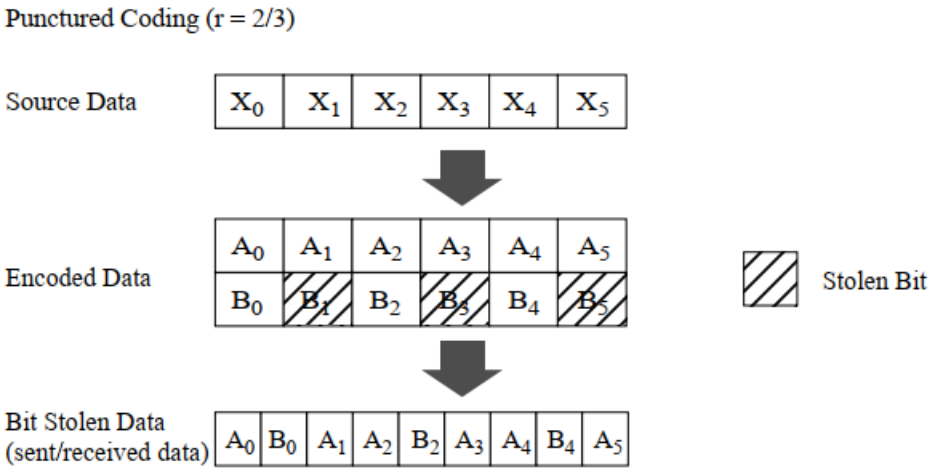
در این بلوک مشابه دیاگرام زیر، داده ها کد می شوند. این بلوک در سمت فرستنده بعد از Scrambler قرار می گیرد. هر داده ورودی دو داده خروجی خواهد داشت. در هر پالس ساعت دو خروجی داده می شود.



سپس داده ها بر اساس Rate تعریف شده Puncture می شوند. در این عمل به ضرورت تعدادی از بیت ها حذف شده و داده ها به صورت تک بیت خروجی داده می شوند. برای پیاده سازی Puncture از آی پی FIFO Generator v9.3 استفاده شده است. داده های خروجی A و B در FIFO نوشته می شوند. سپس در هنگام خواندن با توجه به Rate، Puncture می شوند و داده ای که مورد نظر است به خروجی داده می شود. برای مثال در حالت ½ که به معنی آن است هر داده ورودی دو داده خروجی تحویل می دهد، هر دو مقدار A و B پس از خواندن از FIFO به ترتیب (اول A) در خروجی ظاهر می شوند. در حالت ¾ که به معنی آن است هر ۳ داده ورودی، ۴ داده خروجی خواهد داشت، طبق شکل زیر (از استاندارد) خروجی تولید می شود:



در حالت 2/3 نیز خروجی طبق شکل زیر از FIFO خوانده می شود:



Coding Rate های مختلف برای Rate های گوناگون در جدول زیر آورده شده است:

Data rate (Mbits/s)	Modulation	Coding rate (R)
6	BPSK	1/2
9	BPSK	3/4
12	QPSK	1/2
18	QPSK	3/4
24	16-QAM	1/2
36	16-QAM	3/4
48	64-QAM	2/3
54	64-QAM	3/4

مشخصات آی پی FIFO:

از حالت Block RAM استفاده شده است. همچنین بر روی مود First-Word Fall-Through تنظیم شده است که باعث صفر بودن Latency خواندن داده بعد از فعال کردن بیت کنترلی RD_EN می شود.

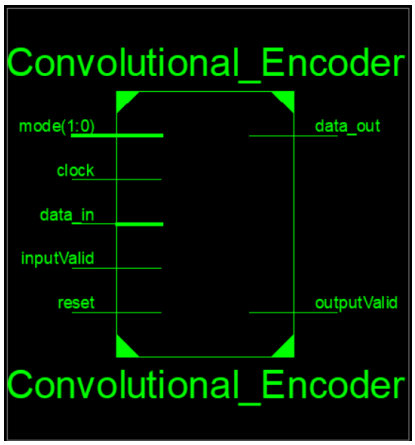
دو بیت ورودی برای نوشتن و دو بیت خروجی برای خواندن قرار داده شده است.

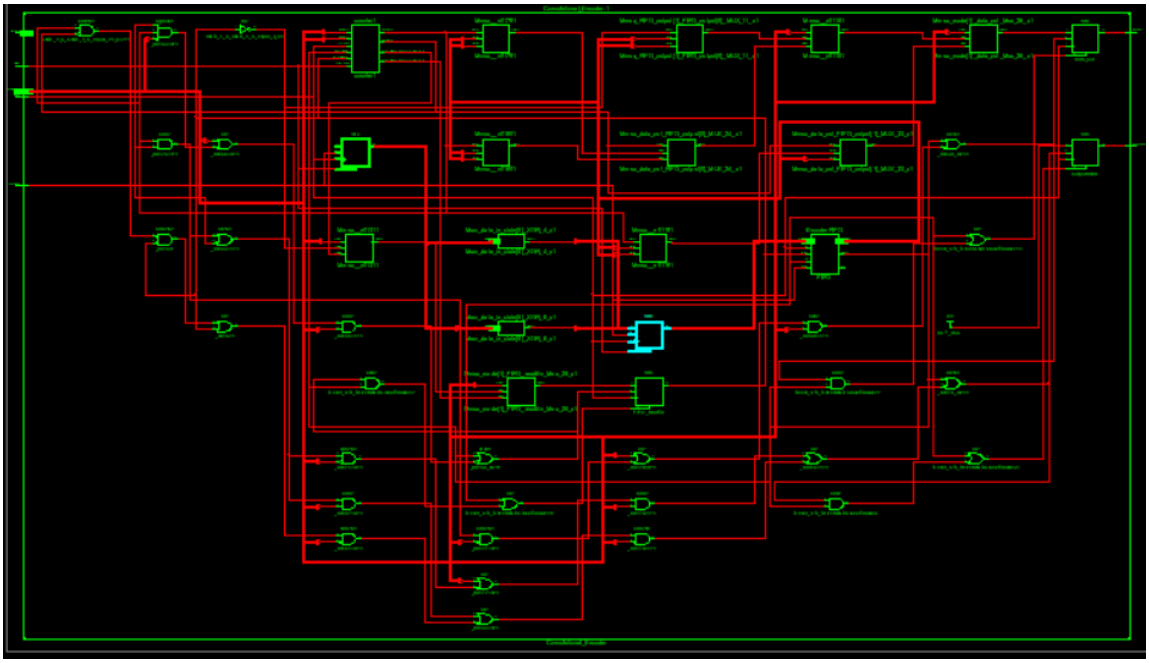
مقدار Write Depth، 32768 (برابر با نصف تعداد داده های مجاز – چون حداکثر این مقدار کلاک خواندن از نوشتن عقب خواهد بود.) تنظیم شده است.

سیگنال های ورودی و خروجی:

- کلاک و ریست
- Mode (مربوط به عمل Puncture – با توجه به Rate در ماژول فرستنده تعیین می شود.)
- دیتای ورودی (تک بیت ورودی) به همراه سیگنال اعتبار سنجی (مشخص می کند که آیا داده های ورودی اعتبار دارند یا خیر)
- دیتای خروجی (تک بیت خروجی) به همراه سیگنال اعتبار سنجی

تصویر بلوک طراحی شده:





استفاده از منابع:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1061	93120	1%
Number of Slice LUTs	1995	46560	4%
Number of fully used LUT-FF pairs	932	2124	43%
Number of bonded IOBs	42	240	17%
Number of Block RAM/FIFO	2	156	1%
Number of BUFG/BUFGCTRLs	1	32	3%

بلوک Interleaver و Deinterleaver:

در این بلوک داده های ورودی جایگشت داده می شوند. این دو جایگشت در رابطه های ۱۵ و ۱۶ (برای Interleaver) و ۱۸ و ۱۹ (برای Deinterleaver) در استاندارد آورده شده است: (رابطه های ۱۸ و ۱۹ برعکس روابط ۱۵ و ۱۶ هستند.)

$$i = (N_{CBPS}/16)\,(k \bmod 16) + \text{floor}(k/16) \qquad k = 0,1,\ldots,N_{CBPS} - 1 \tag{15}$$

$$j = s \times \text{floor}(i/s) + (i + N_{CBPS} - \text{floor}(16 \times i/N_{CBPS})) \bmod s \qquad i = 0,1,\ldots N_{CBPS} - 1 \tag{16}$$

بلوک Interleaver در فرستنده و بلوک Deinterleaver در گیرنده قرار دارند. برای Rate های ۶، ۹، ۱۲ و ۱۸ مگابیت بر ثانیه رابطه دوم تاثیری ندارد زیرا s برابر ۱ می شود. فرمول s:

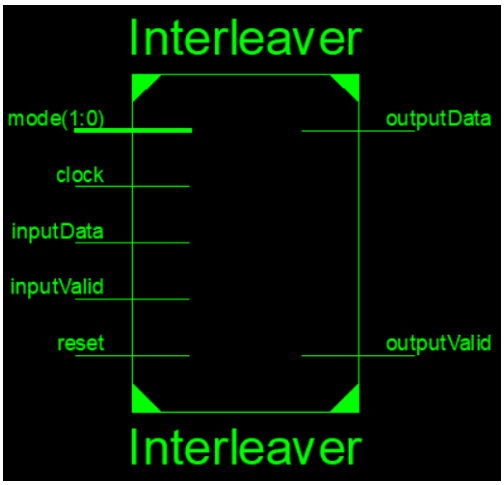
$$s = \max(N_{BPSC}/2,1)$$

این دو بلوک مشابه یکدیگر هستند با این تفاوت که رابطه های جایگشت متفاوت است.

سیگنال های ورودی و خروجی:

- کلاک و ریست
- Mode (مربوط به عمل Puncture – با توجه به Rate در ماژول فرستنده تعیین می شود.)
- دیتای ورودی (تک بیت ورودی) به همراه سیگنال اعتبار سنجی
- دیتای خروجی (تک بیت خروجی) به همراه سیگنال اعتبار سنجی

تصویر بلوک طراحی شده:



استفاده از منابع:

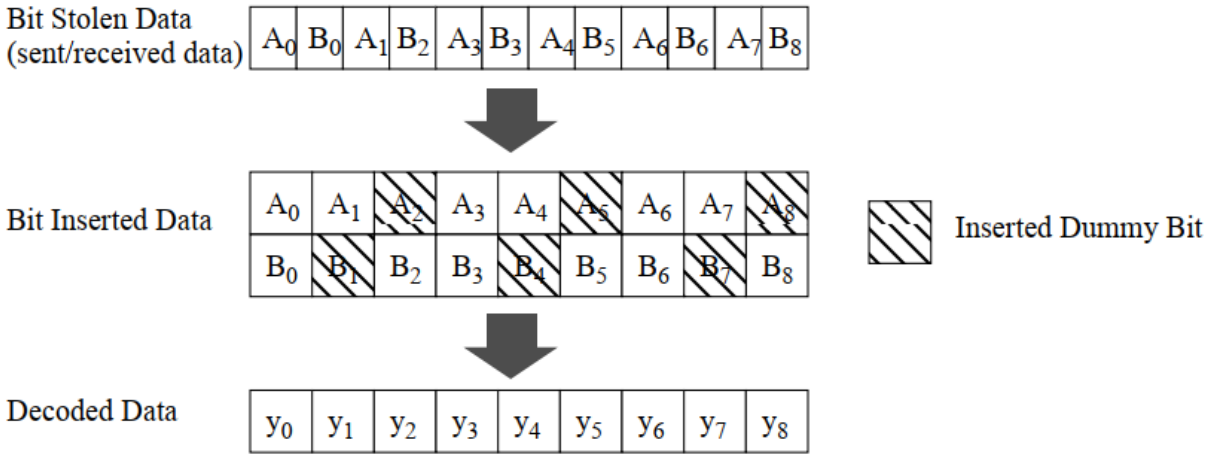
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	637	93120	0%
Number of Slice LUTs	1475	46560	3%
Number of fully used LUT-FF pairs	629	1483	42%
Number of bonded IOBs	8	240	3%
Number of BUFG/BUFGCTRLs	1	32	3%

بلوک Serial to Parallel:

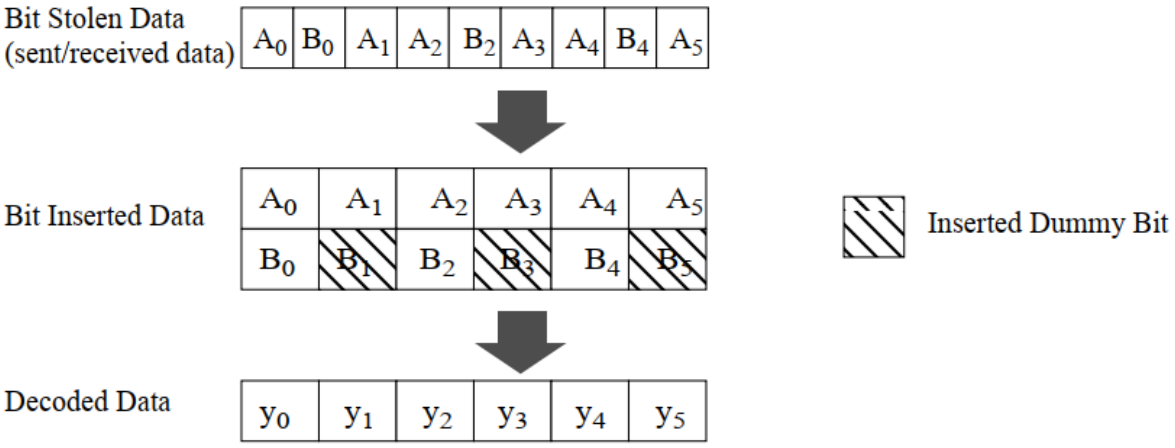
در این بلاک عمل عکس Puncture انجام می شود و بین داده ها بر اساس coding rate در صورت نیاز صفر قرار داده می شود. این بلوک تک بیت ورودی و دو بیت خروجی دارد که در سمت گیرنده بین بلوک های Deinterleaver و Viterbi decoder قرار می گیرد.

عمل عکس puncture طبق استاندارد به صورت زیر انجام می شود:

برای Coding Rate برابر ¾ :



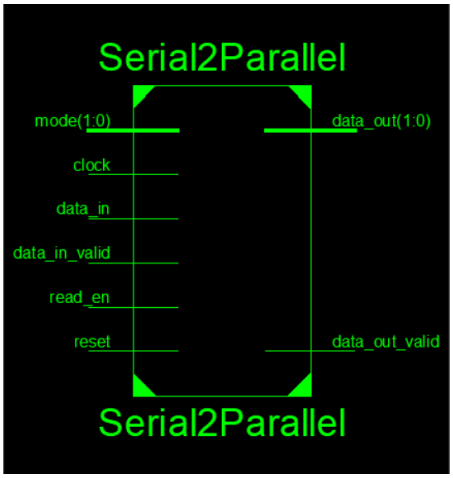
برای Coding Rate برابر 2/3:



سیگنال های ورودی و خروجی:

- کلاک و ریست
- دیتای ورودی (تک بیت ورودی) به همراه سیگنال اعتبارسنجی
- دیتای خروجی (تک بیت خروجی) به همراه سیگنال اعتبار سنجی
- Mode (مربوط به عمل Puncture – با توجه به Rate در ماژول فرستنده تعیین می شود.)
- Read_en : این سیگنال برای خواندن از این بلوک قرار گرفته است. ماژول گیرنده، پس از ورود نصف سیگنال های دریافت شده از Deinterleaver، از این بلوک داده ها رو می خواند. (به خاطر آنکه خواندن سریع تر از نوشتن نباشد.)

تصویر بلوک طراحی شده:

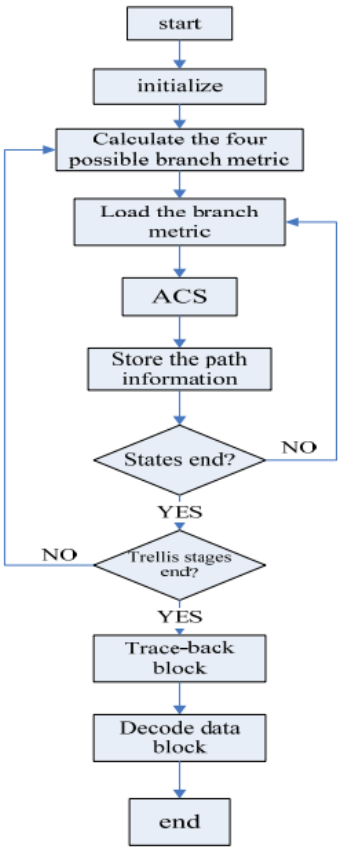


استفاده از منابع:

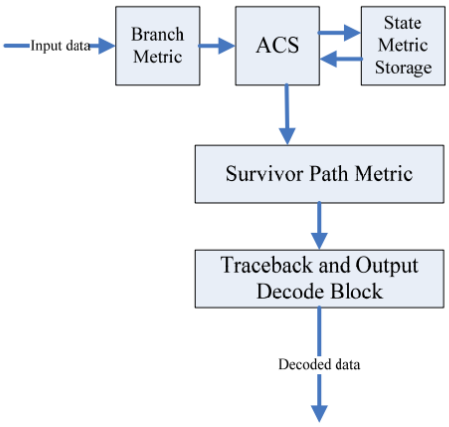
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	631	93120	0%
Number of Slice LUTs	1021	46560	2%
Number of fully used LUT-FF pairs	589	1063	55%
Number of bonded IOBs	8	240	3%
Number of BUFG/BUFGCTRLs	1	32	3%

بلوک Viterbi decoder:

در این پیاده سازی برای دیکود کردن داده های گیرنده، از الگوریتم Viterbi استفاده می کنیم. این الگوریتم با استفاده از فلوچارت زیر طراحی شده است:



نمودار بلوک های مختلف در طراحی صورت گرفته:



این الگوریتم با استفاده از نمودار Trellis داده ها رو دیکود می کند. در ۶ کلاک اول حالت های مختلف با شروع از صفر بدست می آیند، در کلاک های بعدی برای هر دو ورودی یک فاصله Hamming با ورودی های ممکن محاسبه می شود و سپس طبق فاصله های گره ها تا هر مرحله بهترین مسیر انتخاب می شود. (ACS) در انتها نیز با برگشت بر روی گره های بهترین مسیر خروجی تعیین می شود.

توضیحات بیشتر نحوه کار این الگوریتم در لینک زیر قابل دریافت است:

https://www.youtube.com/watch?v=dKIf6mQUfnY

ایده های استفاده شده:

با توجه به آنکه بیت ورودی به انکودر در هر گره از نمودار ترلیس با توجه به زوج بودن و یا فرد بودن شماره گره به صورت یکتا تعیین می شود لذا دیکود کردن همزمان با عملیات Trace-back انجام گرفته است.

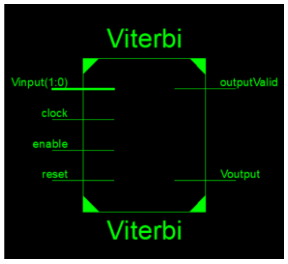
مقادیر خروجی برای هر استیت به صورت یکتا در گذر به استیت بعدی برای هر گره تعیین می شود. این مقادیر نیز در ابتدا ذخیره شده و هر بار استفاده می شوند.

برای ذخیره سازی مسیر بهتر تنها یک بیت مورد استفاده قرار گرفته است که نشان می دهد استیت قبلی در MSB یک داشته است یا صفر که پس از شیفت دیگر موجود نیست.

سیگنال های ورودی و خروجی:

- کلاک و ریست
- enable (سیگنال مربوط به فعال سازی ماژول)
- دیتای ورودی (دو بیت ورودی)
- دیتای خروجی (تک بیت خروجی) به همراه سیگنال اعتبار سنجی
- پارامتر N: طول دیاگرام trellis

تصویر بلوک طراحی شده:



استفاده از منابع:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	631	93120	0%
Number of Slice LUTs	1021	46560	2%
Number of fully used LUT-FF pairs	589	1063	55%
Number of bonded IOBs	8	240	3%
Number of BUFG/BUFGCTRLs	1	32	3%

فرستنده:

پس از طراحی و تست ماژول های ذکر شده، در ماژول فرستنده، بلوک ها در کنار یکدیگر قرار گرفتند.

در ماژول فرستنده، ابتدا لایه مک درخواست شروع فرآیند (PHY_TXSTART_req) را صادر می کند و TXVECTOR را به لایه فیزیکی می دهد. TXVECTOR شامل پارامتر های زیر است:

Parameter	Associate primitive	Value
LENGTH	PHY-TXSTART.request (TXVECTOR)	1–4095
DATATRATE	PHY-TXSTART.request (TXVECTOR)	6, 9, 12, 18, 24, 36, 48, and 54 (Support of 6, 12, and 24 data rates is mandatory.)
SERVICE	PHY-TXSTART.request (TXVECTOR)	Scrambler initialization; 7 null bits + 9 reserved null bits
TXPWR_LEVEL	PHY-TXSTART.request (TXVECTOR)	1–8

با استفاده از این اطلاعات می توان قسمت مربوط به SIGNAL را بدست آورد:

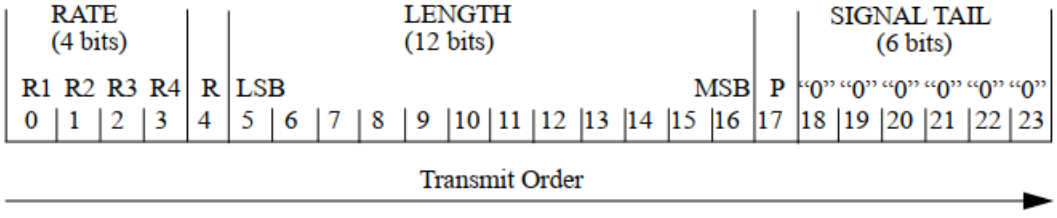
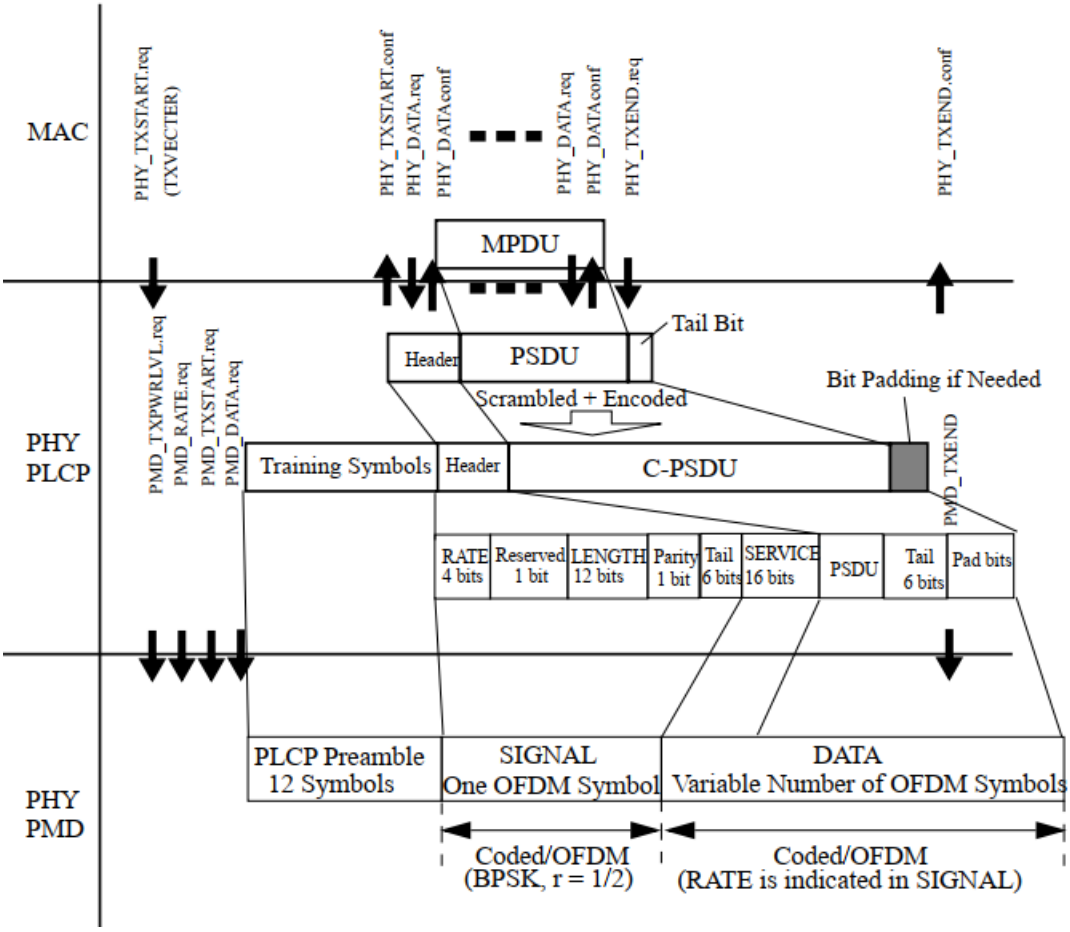


Figure 111—SIGNAL field bit assignment

سپس لایه فیزیکی دستور PHY_TXSTART_conf را به معنای تایید دریافت دستور مک صادر می کند. پس از این تایید لایه مک، دستور PHY_DATA_req را صادر می کند. این دستور برای شروع فرآیند ارسال داده قرار گرفته است. سپس داده ها در هر پالس ساعت به لایه فیزیکی داده می شوند.

در شکل زیر عملیات مربوط به ارتباط لایه فیزیکی و MAC به تصویر کشیده شده است:



داده های PSDU پس از دریافت ذخیره می شوند. حال می بایست عملیات Scramble انجام می شود. برای این منظور ابتدا قسمت SERVICE به ورودی ماژول Scrambler داده می شود. سپس داده های PSDU به ورودی آن داده می شوند و سپس ۶ بیت Tail و در انتها Pad bits به ورودی ماژول Scrambler داده می شود.

Pad bits ها برای آن قرار می گیرند که قسمت DATA مضربی از تعداد بیت های هر سمبل OFDM شود.

پس از Scramble، بلوک Convolutional encoder و پس از آن بلوک Interleaver قرار دارد.

این بلوک ها با اتصال سیگنال خروجی ماژول قبل به بعد و همچنین اتصال سیگنال اعتبار سنجی خروجی ماژول قبل به سیگنال اعتبار سنجی ورودی ماژول بعد به طور مناسب به هم متصل شده اند.

سیگنال های ورودی و خروجی:

- کلاک و ریست
- enable (سیگنال مربوط به فعال سازی ماژول)
- دیتای ورودی (تک بیت ورودی)
- دیتای خروجی (تک بیت خروجی) به همراه سیگنال اعتبار سنجی
- TXVECTOR (توضیح داده شده در بالا)
- PHY_TXSTART_req و PHY_TXSTART_conf و PHY_DATA_req و PHY_DATA_conf.

استفاده از منابع:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1061	93120	1%
Number of Slice LUTs	1995	46560	4%
Number of fully used LUT-FF pairs	932	2124	43%
Number of bonded IOBs	42	240	17%
Number of Block RAM/FIFO	2	156	1%
Number of BUFG/BUFGCTRLs	1	32	3%

گزارش زمانی:

Minimum period: 18.238ns (Maximum Frequency: 54.832MHz)

Minimum input arrival time before clock: 3.395ns

Maximum output required time after clock: 0.783ns

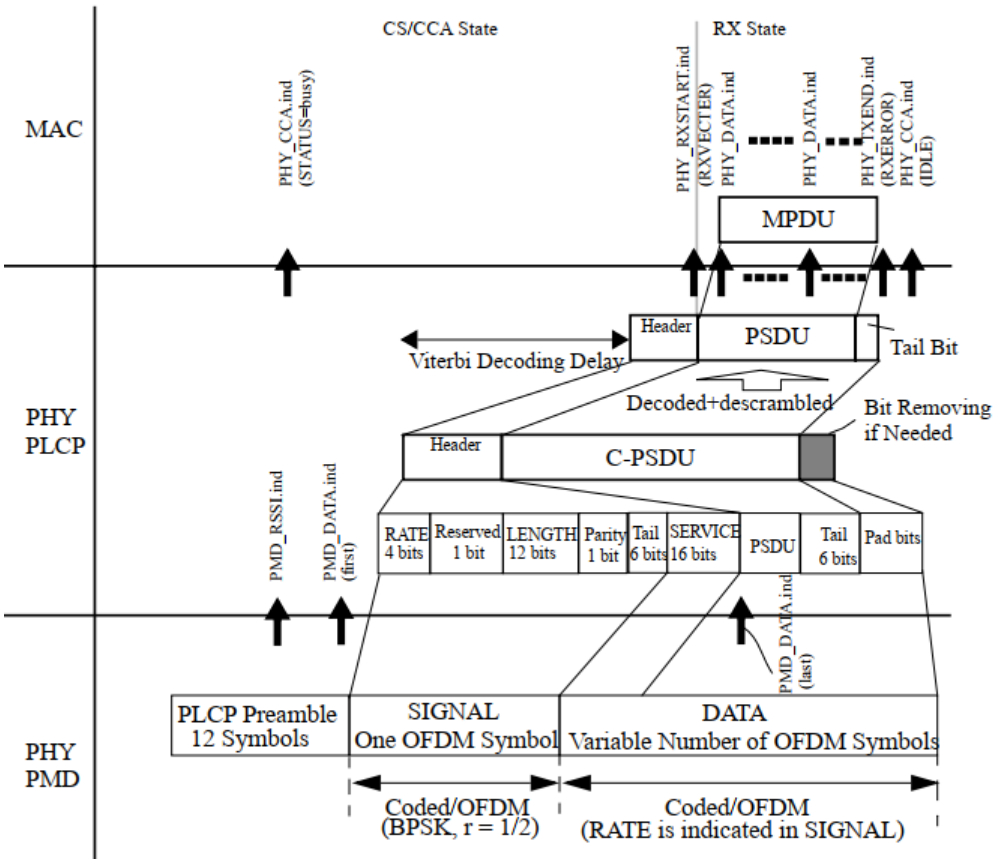
Maximum combinational path delay: No path found

گیرنده:

پس از طراحی و تست ماژول های ذکر شده برای این بخش، در ماژول فرستنده، بلوک ها در کنار یکدیگر قرار گرفتند.

در این ماژول با فرض آنکه داده ها از قسمت های FFT و Remove GI گذشته اند، به Deinterleaver وارد می شوند. سپس داده ها به بلوک Serial2Parallel وارد می شوند و داده های دو تایی وارد ویتربی دیکودر می شوند. در انتها نیز وارد Descrambler شده و به عنوان خروجی از ماژول خارج می شوند.

فرآیند گیرنده در استاندارد:



استفاده از منابع:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	7918	93120	8%
Number of Slice LUTs	33365	46560	71%
Number of fully used LUT-FF pairs	1461	39822	3%
Number of bonded IOBs	42	240	17%
Number of BUFG/BUFGCTRLs	1	32	3%

گزارش زمانی:

Minimum period: 296.941ns (Maximum Frequency: 3.368MHz)

Minimum input arrival time before clock: 4.333ns

Maximum output required time after clock: 0.777ns

Maximum combinational path delay: No path found

تست و اعتبارسنجی صحت پیاده سازی:

فرآیند تست:

ابتدا داده های مثال زده شده در استاندارد بر روی هر یک از ماژول ها به طور مجزا تست شده است.

سپس با استفاده از نرم افزار Matlab داده های رندوم تولید شده، خروجی مورد نظر تولید شده و در فایل های جداگانه ای ذخیره شده است.

تست بنچ نوشته شده به زبان وریلاگ فایل های ورودی را می خواند و خروجی ماژول را در فایل دیگری ذخیره می کند.

سپس خروجی HDL و خروجی مورد انتظار تولید شده توسط نرم افزار Matlab با هم در محیط متلب مقایسه می شوند.

در زیر اطلاعات مربوط به نحوه تست هر ماژول ذکر شده است:

ماژول های Scrambler و Descrambler:

برای قسمت های مربوط به Scrambler و Descrambler تست بنچ های مربوطه با نام های Scrambler_tb و Descrambler_tb قابل دسترس است.

ابتدا کد متلب (فایل ScramblerDescrambler.m)، قسمت های ۱ و ۲ می بایست اجرا شود.

در قسمت ۱ این کد متلب، یک ورودی تصادفی ۱۰۰ تایی تولید خواهد شد و در فایلی به نام Scrambler_DataIn_Matlab ذخیره می شود. همان مقادیر به عنوان مقادیر مورد انتظار خروجی از Descrambler نیز در فایلی به نام Descrambler_DataOut_Matlab ذخیره می شود.

در قسمت ۲ این کد، خروجی اسکرمبل شده توسط متلب تولید می شود. اساس این کد، مدار شکل ۱۱۳ در استاندارد است. خروجی این فایل به نام Scrambler_DataOut_Matlab ذخیره می شود. این خروجی همان ورودی داده شده به دی اسکرمبلر است. لذا در فایلی به نام Descrambler_DataIn_Matlab داده های بدست آمده ذخیره می شوند.

همچنین سری تولید شده در این مدار با حالت اولیه مورد نظر نیز در یک فایل به نام Scrambler_Sequence_Matlab ذخیره می شود تا در صورت نیاز با کد HDL مقایسه شود.

برای اطمینان بیشتر، مقادیر عددی بدست آمده با خروجی تابع wlanScramble که تابع آماده نرم افزار متلب است مقایسه می شود و در صورت صحت کد، فایل های ذکر شده نوشته می شوند.

پس از اجرای کد متلب، تست بنچ نوشته شده در هر بخش اجرا می شود. تست بنچ ها، خروجی و سری تولیدی را در فایل هایی به نام Scrambler_DataOut_HDL و Scrambler_Sequence_HDL ذخیره می شوند. (برای دی اسکرمبل نیز به همین ترتیب)

پس از اجرای تست بنچ برای مقایسه نتایج قسمت های ۳ و ۴ کد متلب نتایج خروجی متلب و HDL را مقایسه می کند.

قسمت های ۵ و ۶ نیز سری تولیدی توسط متلب و HDL را مقایسه می کند.

لازم به ذکر است برای اجرای صحیح کد متلب، باید پوشه پیش فرض متلب بر پوشه اصلی پروژه تنظیم شده باشد.

تمامی فایل های ذکر شده در پوشه اصلی متلب قابل دسترس هستند.

زمان لازم برای اجرای تست بنچ: اگر برای ۲.۲ میکرو ثانیه تست بنچ ها اجرا شوند، فایل ها به درستی تولید می شوند.

پس از اجرای صحیح تست بنچ ها خروجی زیر در متلب نمایش داده می شود:

File Scrambler_DataIn_Matlab.txt generated successfully!

File Descrambler_DataOut_Matlab.txt generated successfully!

My generated scrambler output data & Matlab function output are equal!

File Scrambler_DataOut_Matlab.txt generated successfully!

File Descrambler_DataIn_Matlab.txt generated successfully!

File Scrambler_Sequence_Matlab.txt generated successfully!

File Descrambler_Sequence_Matlab.txt generated successfully!

Scrambler output data for Matlab and HDL are eqaul!

Descrambler output data for Matlab and HDL are eqaul!

Scrambler Sequences for Matlab and HDL are equal!

Descrambler Sequences for Matlab and HDL are equal!

در تست بنچ نیز، سری تولیدی توسط مدار توصیف شده با استفاده از دستور monitor نمایش داده می شود.

ماژول Convolutional Encoder:

برای تست ماژول دو نوع ورودی قابل دسترس است:

۱) ورودی مربوط به مثال استاندارد: ۹۴ داده اول اسکرمبل شده از جدول G۱۶ که با ۶ بیت صفر در انتها در فایل ConvEncoder_DataIn.txt قابل دسترس است.
۲) ورودی رندوم تولید شده توسط نرم افزار متلب با اجرای بخش دو فایل ConvEncoder.m که در فایلی به نام ConvEncoder_DataIn_Matlab.txt ذخیره می شود.

در تست بنچ به طور پیشفرض داده ها از فایل اول خوانده می شوند. برای تست اعداد رندوم می بایست نام فایل در خط ۶۳، ۶۴ و ۶۵ تغییر پیدا کند.

قسمت سوم فایل متلب ConvEncoder.m مربوط به تولید خروجی انکودر توسط متلب، قسمت چهارم مربوط به نوشتن خروجی داده ها در فایلی به نام ConvEncoder_DataOut_Matlab.txt است که می بایست به ترتیب اجرا شود.

قسمت پنجم مربوط به مقایسه داده های متلب و وریلاگ است که پس از اجرای تست بنچ قابل اجرا می باشد. اگر داده های خروجی یکی باشد پیغام زیر صادر می شود:

Convolutional encoder output data for Matlab and HDL are equal!

ماژول Viterbi Decoder:

کد متلب با نام VitebiDecoder.m در پوشه پروژه موجود است. در این کد با استفاده از توابع متلب مقادیر کد شده و دیکود شده بدست می آیند و سپس در یک فایل تکست به نام Viterbi_DataOut_Matlab.txt ذخیره می شوند.

فایل تست بنچ به نام Viterbi_tb.v در پوشه پروژه موجود است. داده های ورودی از فایلی به نام Viterbi_DataIn.txt خوانده می شوند. این داده ها به صورت دو بیت دو بیت در آن فایل ذخیره شده اند. داده ها مقادیر کد شده توسط انکودر پیاده شده هستند و در هر کلاک یک سطر (دو داده) به عنوان ورودی به ماژول داده می شود.

مقادیر حاصل در فایلی به نام Viterbi_DataOut_HDL.txt ذخیره می شود.

در بخش آخر کد متلب دو فایل با هم مقایسه می شوند و در صورت برابری دو مقدار دیکود شده عبارت زیر به نمایش در می آید:

Convolutional encoder output data for Matlab and HDL are equal!

برای مقادیر مثال استاندارد ماژول تست مقادیر خروجی با مقادیر متلب برابر بودند.

مدت زمان لازم برای اجرای تست بنچ: ۴.۲ میکروثانیه

برای قسمت Interleaver تست بنچ مربوطه با نام Interleaver_tb.v قابل دسترس است.

برای تست ماژول از یک ورودی رندوم تولید شده توسط نرم افزار متلب استفاده می کنیم. با اجرای بخش اول فایل Interleaver.m رشته ورودی به طول ۹۶ (برابر با دو سمبول ۴۸ بیتی) در فایلی به نام Interleaver_DataIn.txt ذخیره می شود.

ماژول های Interleaver و Deinterleaver:

قسمت سوم فایل متلب Interleaver.m مربوط به تولید خروجی اینترلیور توسط متلب، قسمت چهارم مربوط به نوشتن خروجی داده ها در فایلی به نام Interleaver_DataOut_Matlab.txt است که می بایست به ترتیب اجرا شود.

قسمت پنجم مربوط به مقایسه داده های متلب و وریلاگ است که پس از اجرای تست بنچ قابل اجرا می باشد. اگر داده های خروجی یکی باشد پیغام زیر صادر می شود:

interleaver output data for Matlab and HDL are equal!

برای بخش Deinterleaver، چون حاصل خروجی بازای خروجی Interleaver برابر ورودی Interleaver است در فایل Deinterleaver.m ابتدا فایل ورودی Deinterleaver ساخته می شود.

سپس تست بنچ (فایل Deinterleaver_tb.v) اجرا شده و خروجی آن در فایلی به نام
Deinterleaver_DataOut_HDL.txt ذخیره می شود. قسمت دوم فایل متلب نیز این خروجی و ورودی اولیه به interleaver را مقایسه می کند و در صورت تساوی پیغام زیر را صادر می کند:

Deinterleaver output data for Matlab and HDL are equal!

هر دو قسمت انجام شده است و تایید یکسان بودن خروجی HDL و خروجی متلب با استفاده از روش بیان شده گرفته شده است.

ماژول فرستنده:

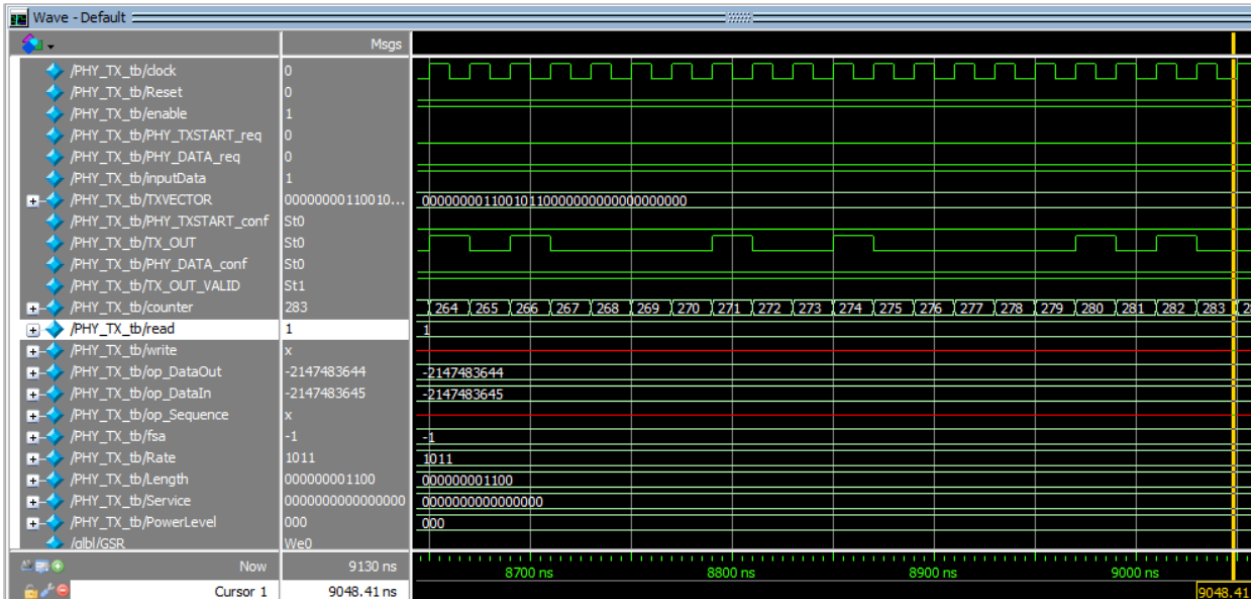
برای تست این ماژول، ابتدا ماژول Scrambler به صورت مجزا تست شد. سپس ماژول Convolutional Encoder به همراه Scrambler تست شد و در نهایت ماژول فرستنده با در کنار هم قرار دادن هر سه ماژول تست شد.

برای تست این ماژول می توان قسمت مربوط به output را که با کامنت مشخص شده است به هر یک از ماژول های ذکر شده اختصاص داد و خروجی را مقایسه کرد.

فایل های متلب TX_Scrambler، ConvEncoder_TX و Interleaver_TX مربوط به این قسمت هستند.

در تست بنچ نیز برای هر خروجی لازم است عدد counter مورد نظر را تا یکی قبل از تعداد مورد انتظار قرار دهیم. (از صفر شروع می شود.)

شکل شبیه سازی شده توسط نرم افزار Modelsim:



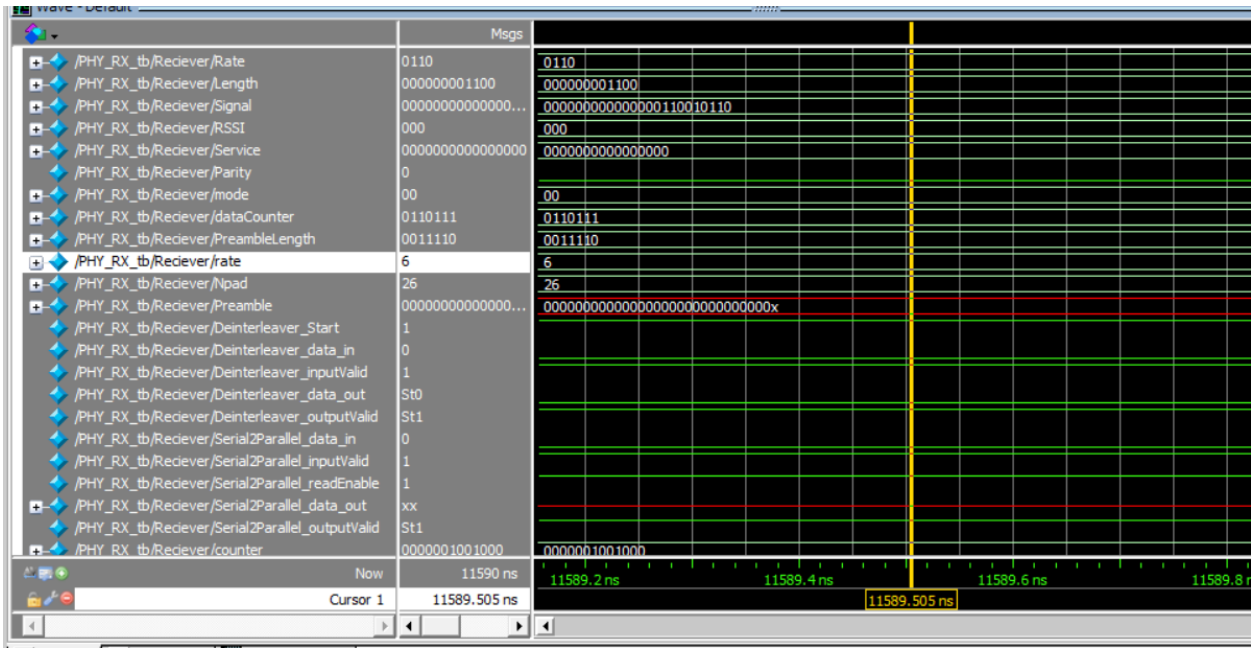
ماژول گیرنده:

برای تست این ماژول، ابتدا ماژول Deinterleaver به صورت مجزا تست شد. سپس ماژول Serial2Parallel و Viterbi اضافه شدند و در انتها نیز ماژول Descrambler به فرآیند تست اضافه شد.

فایل Deinterleaver_RX.m، و فایل های مربوط به ویتربی و Descrambler تست های گذشته قابل تایید است.

در تست بنچ نیز برای هر خروجی لازم است عدد counter مورد نظر را تا یکی قبل از تعداد مورد انتظار قرار دهیم. (از صفر شروع می شود.)

شکل شبیه سازی شده توسط نرم افزار Modelsim:



تمامی فایل های ذکر شده در فولدر اصلی قرار دارند.

شبیه سازی دیگر ماژول ها در گزارش های فازهای قبلی آورده شده است.

تغییرات نسبت به فازهای گذشته:

ماژول های Scrambler و Descrambler تغییری نداشتند.

در ماژول Encoder با اضافه کردن آی پی FIFO عمل Puncture برای Rate های امتیازی انجام شد.

ماژول Viterbi تغییری نداشته است.

ماژول Serial2Parallel برای ارتباط بین Deinterleaver و Viterbi اضافه شده است.

ماژول interleaver برای Rate های امتیازی ارتقا یافته است و تمامی rate ها را پوشش می دهد.

ماژول های فرستنده و گیرنده به طور کامل نوشته و تست شده اند.