سیستم‌های مبتنی بر ASIC و FPGA

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر مهدی شعبانی

دانشکده مهندسی برق

پروژه فاز چهارم

نام و نام‌خانوادگی: عرفان نصرتی

شماره دانشجویی: ۹۷۱۰۲۵۵۸

فهرست مطالب

۱ مقدمه

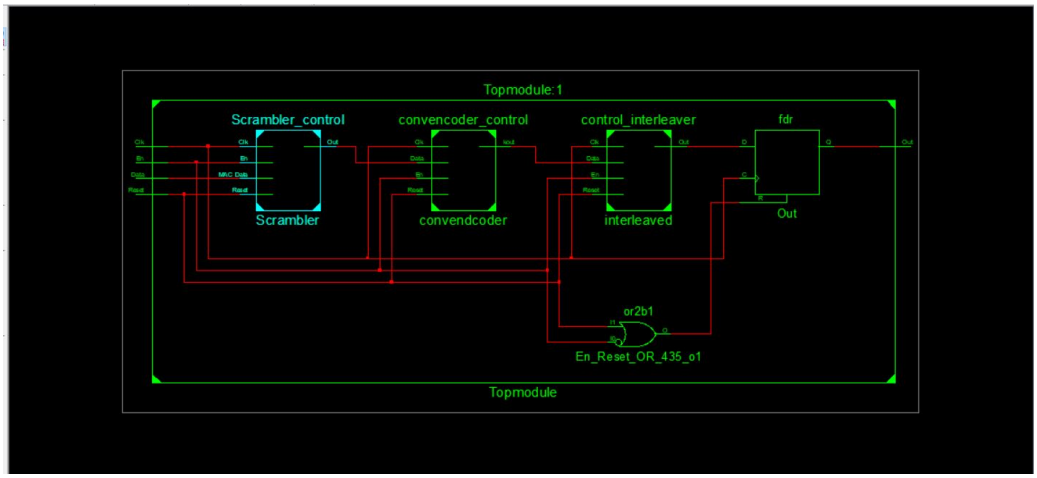
در این فاز که فاز نهایی پروژه است به طراحی ساختار فرستنده و گیرنده می‌پردازیم و باید ماژول هایی که در فاز های قبل مورد استفاده قرار گرفته است را با هم ارتباط دهیم تا در گیرنده و فرستنده آن را استفاده کنیم.

۲ فایل بندی

نحوه قرار گیری فایل های پروژه به این صورت است که پوشه اصلی ۲ فولدر به نام های Verilog و متلب قرار دارد و در هر یک از پوشه ها کد های فاز مربوطه در پوشه فاز مربوطه قرار گرفته اند. کد متلب هر سه تست کیس های فرستنده و گیرنده و فرستنده گیرنده را در یک کد درست می‌کند. تنها کافیست برای صحت سنجی تست کیس ها را توسط متلب درست کنید و در پوشه های پروژه دلخواه کپی کنید و تست بنچ را اجرا کنید.
در این قسمت در پوشه وریلاگ ۳ پوشه دیده می‌شود که یکی فرستنده و دیگری گیرنده و فولدر سوم فرستنده و گیرنده است.

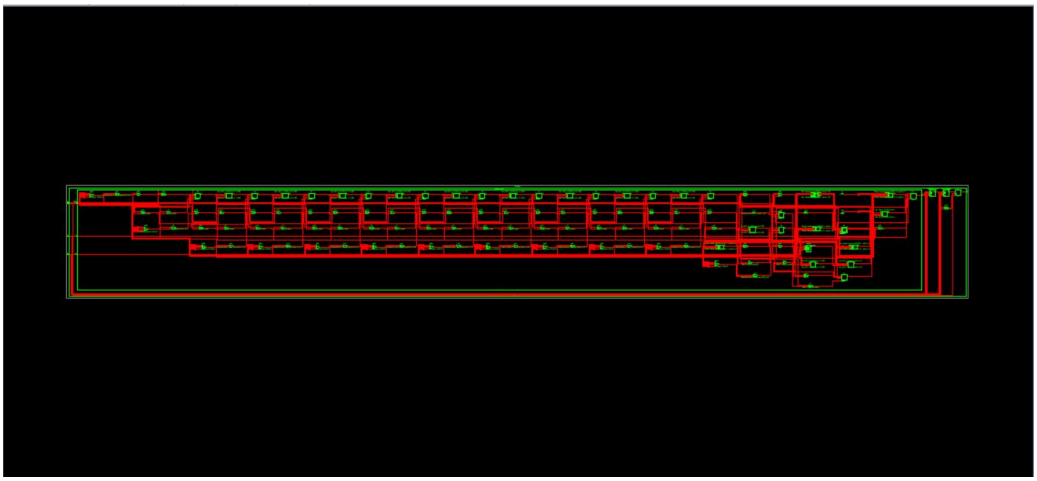
۳ فرستنده

ساختار کلی فرستنده به این شکل است که دیتای آمده ابتدا باید اسمبل شود سپس دیتای اسمبل شده به کانولوشنال انکودر داده می‌شود و سپس دیتای انکود شده تحویل اینترلیور می‌شود.
ساختار استفاده شده در این فرستنده به صورت پسیو است یعنی ماژول های ما سیگنال En به دیگری نمی‌فرستند و هر ماژول در فاز قبل یک ماژول بالا دستی کنترلی دارد که این ماژول بالا دستی اینگونه عمل می‌کند که قسمت preamble و signal را جدا کرده و اگر ماژول ما دارای دیلی باشد آن را در یک شیفت رجیستر ذخیره می‌کند و در موقع مناسب preamble و signal را به خروجی میدهد و خروجی ماژول را نیز بعد از آن به خروجی می‌دهد.
به این صورت هر ماژول کل فریم را دریافت کرده و فریم را به بعدی تحویل میدهد. در ساختار این فرستنده باید قسمت سیگنال داده وارد انکودر شود اما چون در فاز دو این پیاده سازی انجام نشده بود ساختار فرستنده و گیرنده من این اشکال را دارند که تنها قسمت دیتای دیتا ورودی را از همه ماژول ها عبور می‌دهند در صورتی که قسمت سیگنال باید از انکودر و ویتربی عبور داده شود.
حال همانطور که گفتیم ماژول ها پسیو هستند و یک دیگر را صدا نمی‌کنند. و روش فعال شدن آنها روشی است که گیرنده می‌فهمد سیگنال فعال شده است.یعنی از قسمت پری‌امبل برای اینکه ماژول ها بفهمند دیتایی آمده است استفاده شده است.
از آنجایی که پری‌امبل ما از ۱۲ بیت یک تشکیل شده است در هر ماژول فرستنده و یا گیرنده یک ماشین حالت وجود دارد که بررسی می‌کند زمانی که ۱۲ یک پشت سر هم دریافت کرد به ماژول دستور روشن شدن می‌دهد و هم زمان با این استیت ماشین یک رجیستر مقدار ورودی را ذخیره می‌کند تا پری‌امبل و قسمت سیگنال که بدون تغییر به ماژول بعد منتقل می‌شود را در خود ذخیره کند و در موقع مناسب به خروجی انتقال دهد.



شمای کلی فرستنده است که به این صورت عمل می‌کند که هر ماژول فاز قبل یک ماژول کنترلی دارد که با آمدن پری‌امبل به ماژول دست پایین خود دستور روشن شدن و انجام عملیات را می‌دهد سپس در موقع مناسب دیتای پری‌امبل و سیگنال

را به خروجی اضافه کرده و به خروجی بعدی می‌دهد.



در اینجا RTL کنترلی اسمبلر را می‌بینیم همانطور که دیده می‌شود در قسمت اول و ابتدایی این ماژول تعداد رجیستر و کانتر هستند که نقش ماشین حالت را برای تشخیص پری‌امبل انجام می‌دهند.

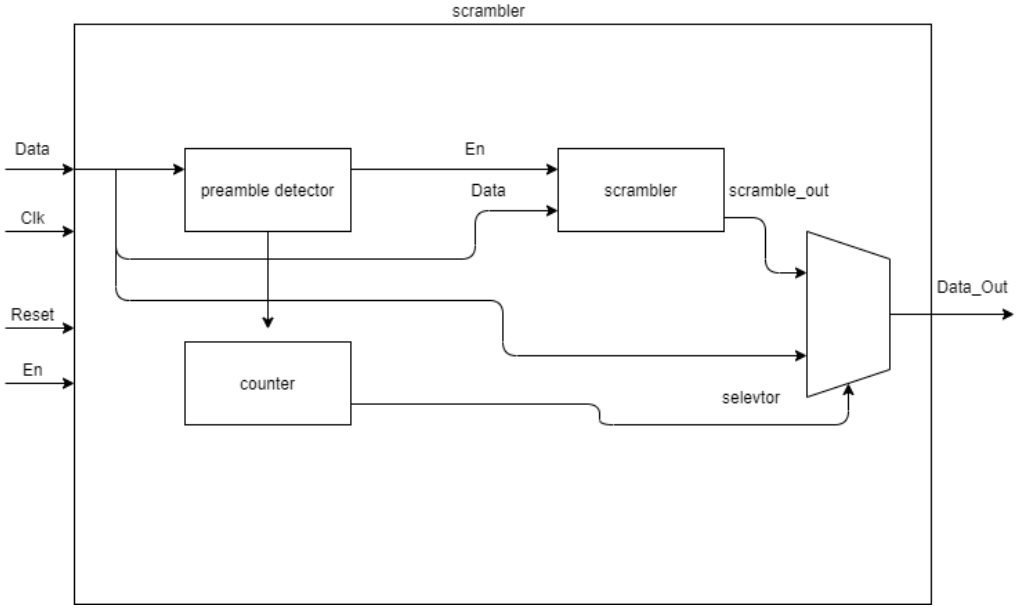
```
44
45      always@(posedge Clk) begin
46         if (~En || Reset ) begin
47           Out <= 1'b0;   // the Scrambler is disable and sends 0
48           Preamble_signal_length <= 7'b0;
49           start_scramble <= 1'b0;
50           FIFO <= 12'b0;
51         end
52         else if ( Preamble_signal_length < 7'd12 )begin
53           Out <= 1'b1;
54           FIFO[Preamble_signal_length] <= MAC_Data;
55           Preamble_signal_length <= Preamble_signal_length + 1'b1;
56         end
57         else if ( Preamble_signal_length < 7'd36 ) begin
58           Out <= FIFO[0];
59           FIFO <= {MAC_Data,FIFO[11:1]};
60           Preamble_signal_length <= Preamble_signal_length + 1'b1;
61         end
62         else if ( Preamble_signal_length < 7'd84 ) begin
63           start_scramble <= 1'b1;
64           Out <= Scrambled_data;
65           FIFO <={MAC_Data,FIFO[11:1]};
66           Preamble_signal_length <= Preamble_signal_length + 1'b1;
67         end
68         else begin
69           Out <= 1'b0;
70           FIFO <= 12'b0;
71         end
72      end
```

در اینجا برای نمونه کد کنترلر اسمبلر آورده شده است که همانطور که گفته شد در صورت تشخیص پری‌امبل ابتدا قسمت سیگنال را دریافت سپس ماژول دست پایینی یعنی اسمبلر را فعال می‌کند و دیتای آنرا به خروجی‌میدهد. لازم به ذکر است در ماژولی مانند اسمبلر که بدون تاخیر دیتای ورودی را به خروجی می دهد و به ازای هر ورودی یه خروجی داریم لازم نیست از شیفت رجیستر استفاده شود و فقط کافیست مشخص شود کدام قسمت از دیتا باید وارد ماژول شود.

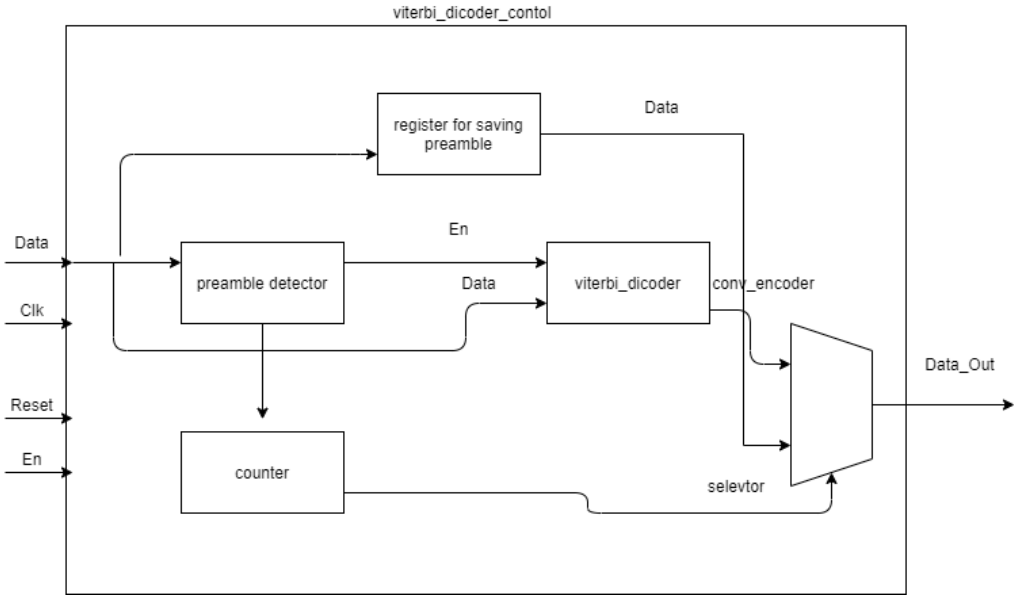
اما در ماژولی مانند ویتربی یا اینترلیور که تاخیر در انتقال ورودی به خروجی دارند ابتدا باید پری امبل و سیگنال در یک رجیستر ذخیره شود سپس به قسمت اول خروجی اضافه شود.

در زیر شکلی از شمای کلی طراحی کنترل کننده ها آمده است در ماژول هایی که با دریافت خروجی بدون تاخیر دیتا به بیرون می‌دهند دیده می‌شود از ساختار رجیستر برای رجیستر کردن دیتا استفاده نشده است.

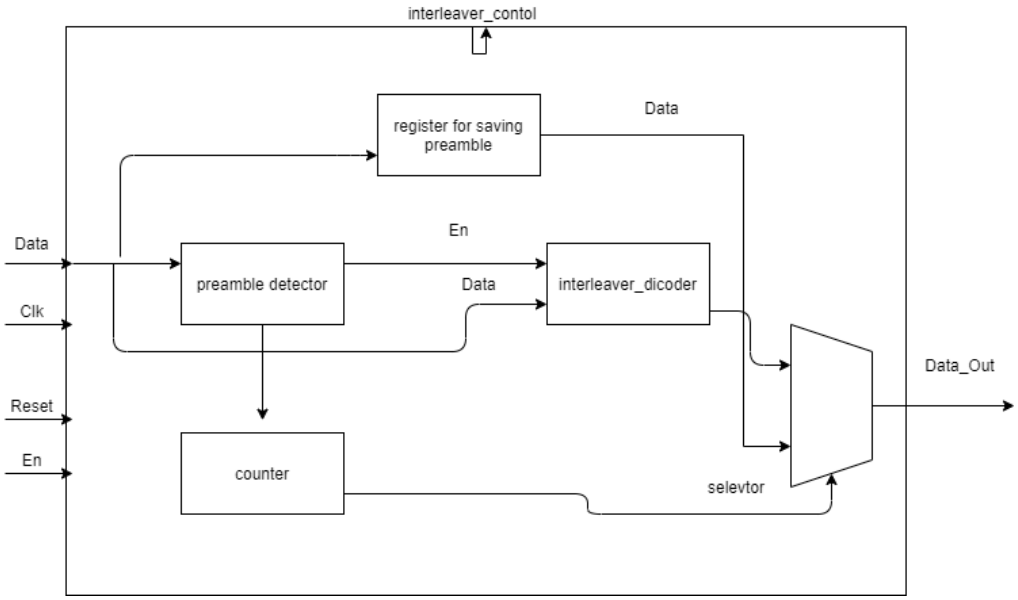
۱.۳ scrambler



۲.۳ viterbi

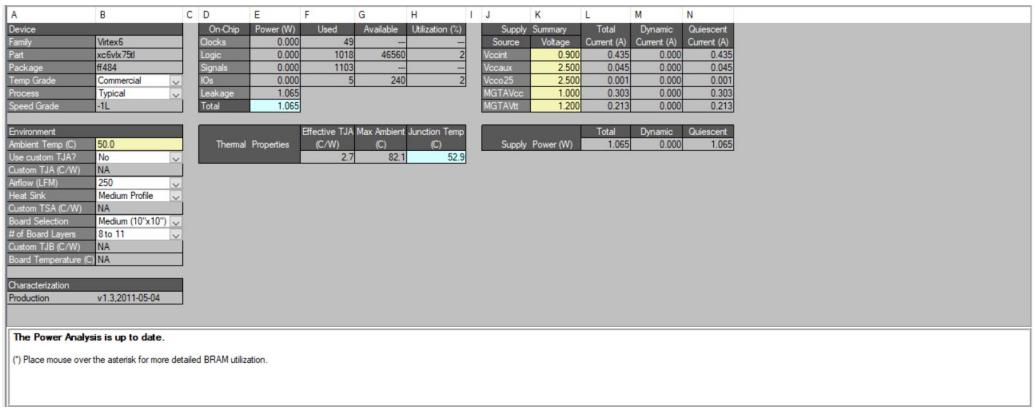


interleaver ۳.۳



power ۴.۳

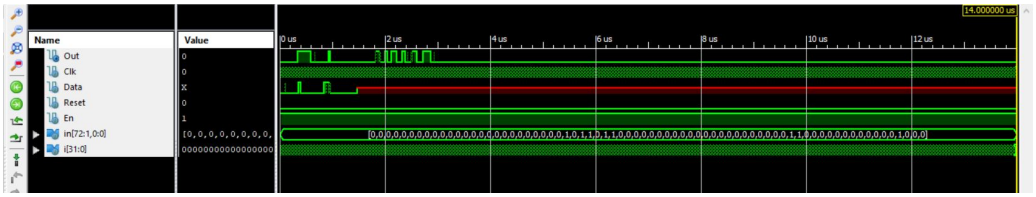
ماژول های ما به علت کوچکی و استفاده کم از ریسورس در ماژول های استفاده شده توان و سطح زیادی از چیپ را اشغال نمی کنند و به همین علت این قسمت زیاد اعتبار ندارد زیرا همانطور که در عکس زیر آمده است ما تنها از ۱۰۰۰ رجیستر و ۱۰۰۰ LUT از میان ریسورس های درون FPGA استفاده کردیم (FPGA استفاده شده همانطور که در دستورکار گفته شده است Vertex ۶ است). و به همین دلیل اکثر توان تلف شده در چیپ توان leakage است و ما توانی قابل مقایسه با این ماژول استفاده نمی‌کنیم.
همچنین در ورودی و خروجی ها ما ۴ ورودی کلاک و ریست و دیتا و فعال کننده را داریم و یک خروجی دیتا. پس در مجموع ۵ پین ورودی و خروجی استفاده شده است.



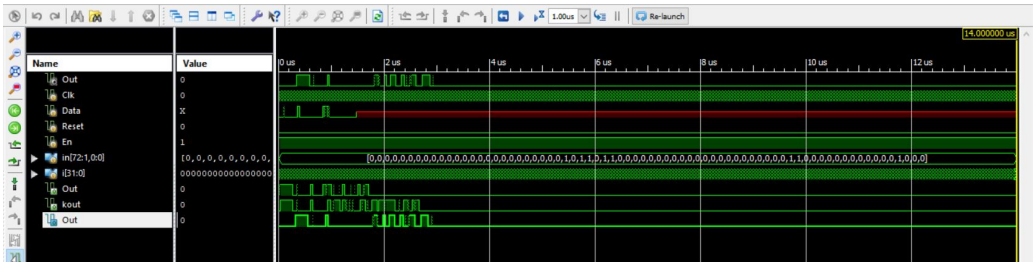
user ۵.۳

عمل کرد دستگاه به این صورت است که شما قسمت دیتا فریم به علاوه سیگنال را در ورودی به ماژول میدهد و با فعال کردن En و ریست کردن مدار. مدار آماده گرفتن دیتا می‌شود و از زمانی که سیگنال EN یک شود دیتا را در ورودی می‌گیرید در این ساختار پیاده سازی شده ما دو بیت را در ورودی گرفته و در خروجی تحویل می‌دهیم.
قسمت پری‌امبل ماژول scramble به دیتا افزوده می‌شود (به طور مشابه در descramble از دیتا جدا می‌شود). و سپس توسط کنترل های توضیح داده شده در بالا دیتا وارد ماژول کنترل می‌شود و قسمت سیگنال و پری‌امبل به ماژول بعدی انتقال پیدا می کنند.

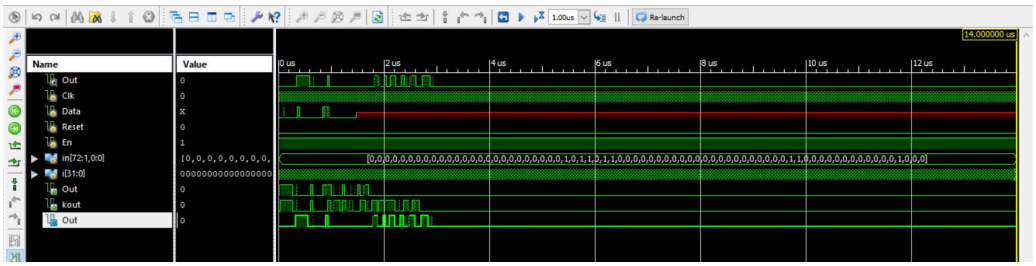
test ۶.۳



شکل ۱: فقط ورودی و خروجی



شکل ۲: ماژول های میانی به ترتیب

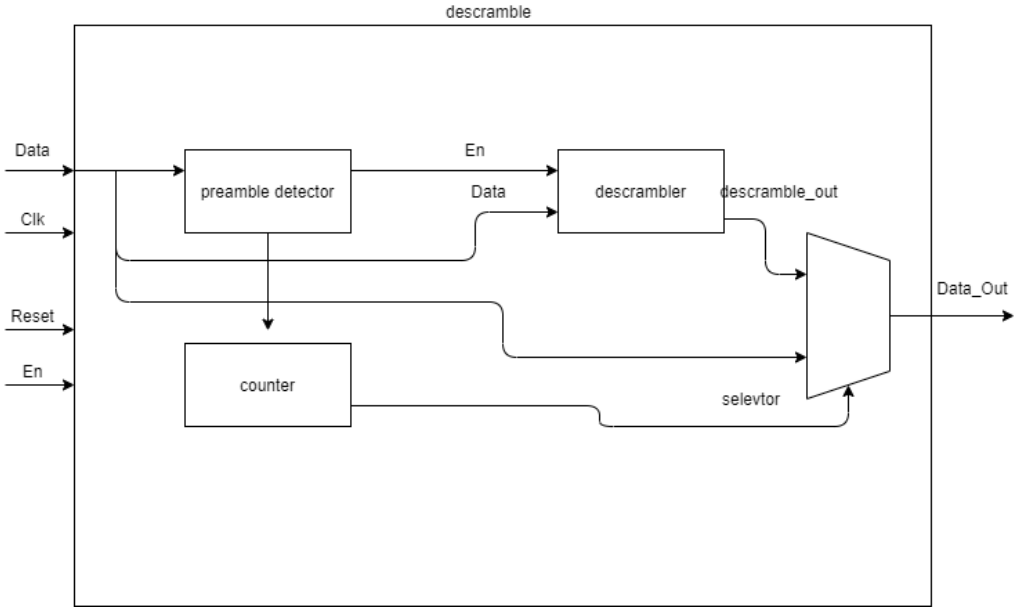


شکل ۳: همه خروجی ها و ورودی

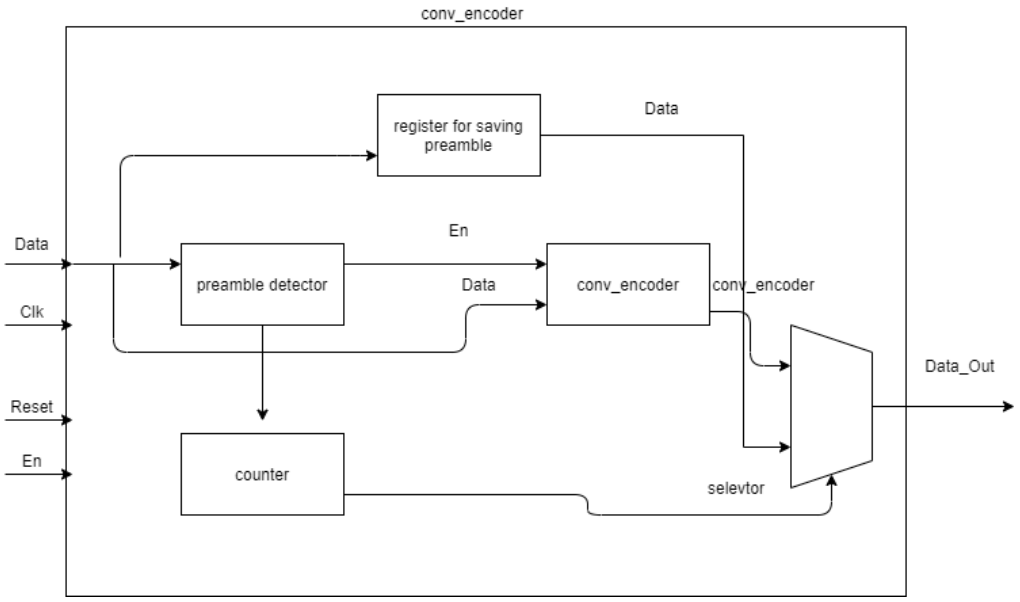
۴ reciever

درگیرنده نیز ساختاری مشابه فرستنده استفاده شده است و از همان روش پسیو برای فهمدین آمدن دیتا استفاده شده است. یعنی ماژول هایی که در رسیور هستند با دریافت پری‌امبل شروع به کار کرده و هر ماژول پری‌امبل و قسمت سیگنال داده را گرفته و بعد از انجام عملیات به بیرون می‌دهد. برای جلوگیری از حشو از گفتن دوباره مطالب بالا پرهیز می‌کنیم و به آوزدن مشخصات و نتایج در این قسمت بسنده می‌کنیم.

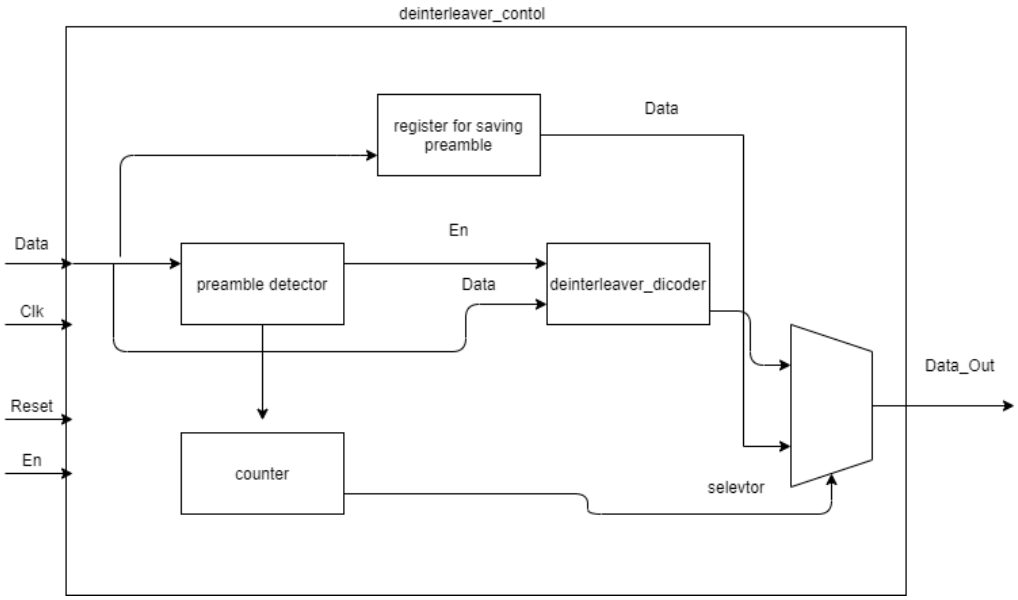
۱.۴ descramble



۲.۴ encoder

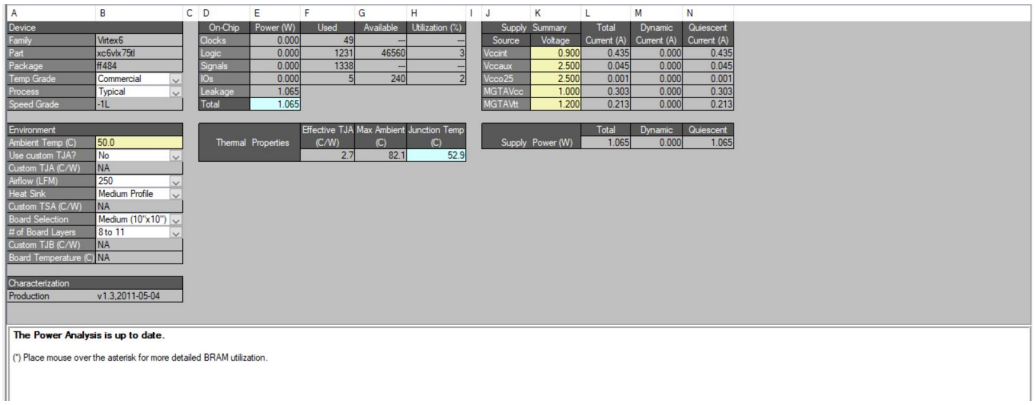


deinterleaver ۳.۴



power ۴.۴

مانند ماژول فرستنده زیر ماژول هایی که در این ماژول استفاده شده است باز هم ماژول هایی هستند که زیاد ریسورس مصرف نمی‌کنند اما با این تفاوت که در این قسمت ماژول ویتربی به خاطر ساختار DP بودن خود یک ماژول نسبت بزرگ حساب می‌شود و تفاوت چشم گیری در تعداد رجیسترها و LUT ها دیده می‌شود بعنی تعداد آنها تقریبا ۷.۱ برابر شده است و با توجه با اینکه میدانیم ریسورس های استفاده شده در قسمت deinterleaver و descramble تفاوت چشم گیری با هم ندارند. همانطور که در گزارش فاز دوم نیز آوزده شده است ماژول ویتربی بیشتر ریسورس های این قسمت را دارد.

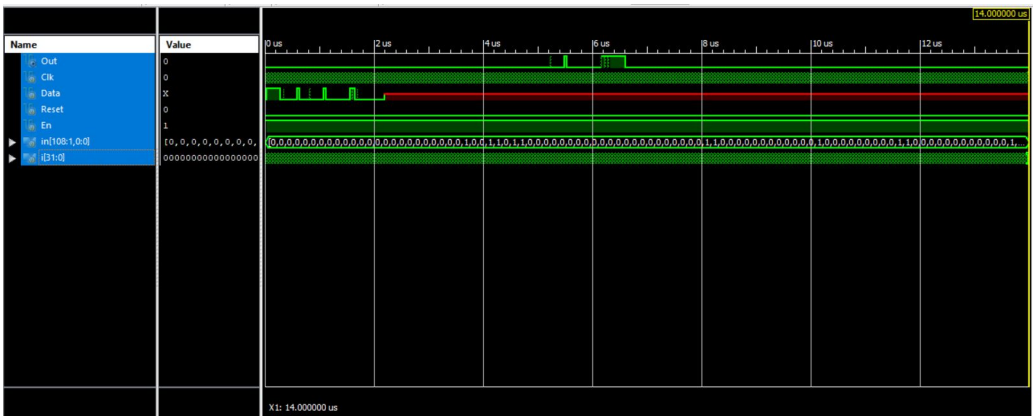


همچنین در ورودی و خروجی ها ما ۴ ورودی کلاک و ریست و دیتا و فعال کننده را داریم و یک خروجی دیتا. پس در مجموع ۵ پین ورودی و خروجی استفاده شده است.

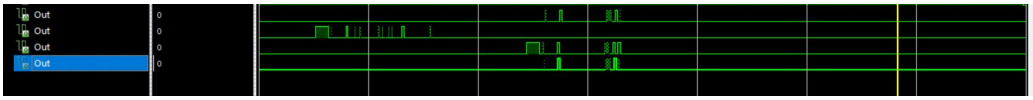
user ۵.۴

در این قسمت چون گیرنده یک عنصر پسیو است باید زمانی که می‌خواهیم یک دیتا را دریافت کنیم سیگنال EN را فعال کنیم و یک کلاک هم فرکانس با کلاک فرستنده تولید کنیم تا گیرنده و فرستنده با هم سنکرون باشند. و سپس گیرنده با دریافت پری‌امبل خودش به صورت خودکار شروع به پراسس کردن دیتا ها و باز تولید دیتای فرستاده شده می کند.

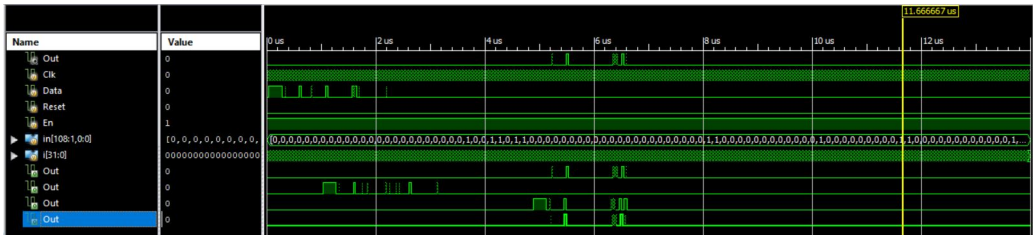
test ۶.۴



شکل ۴: فقط ورودی و خروجی



شکل ۵: ماژول های میانی یه ترتیب

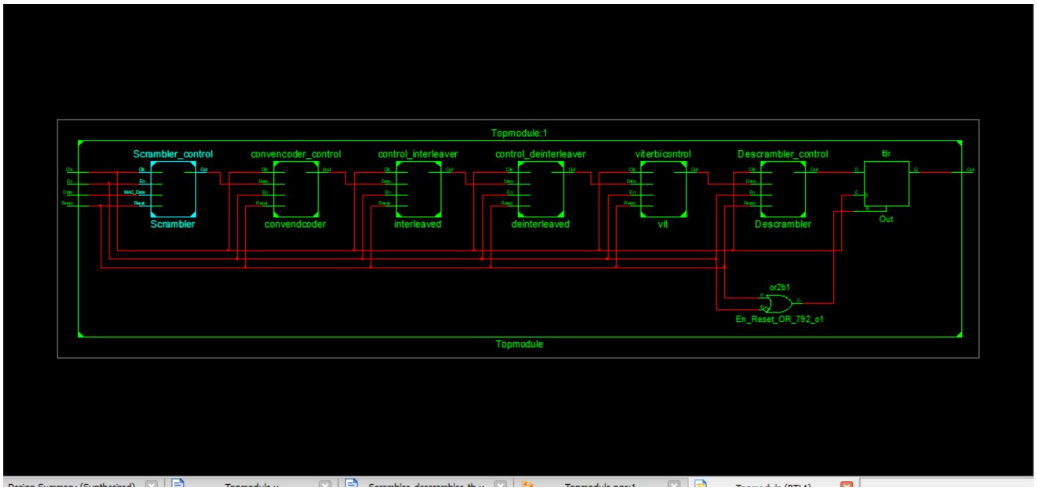


شکل ۶: همه خروجی ها و ورودی

همانطور که دیده می شود در خروجی همه ماژول ها پری‌امبل دیده می‌شود (مفدار پری‌امبل ۱۲ تا یک است). به جز دی‌اسکرمبلر زیرا این ماژول دیتا را به خارج تحویل می‌دهد.

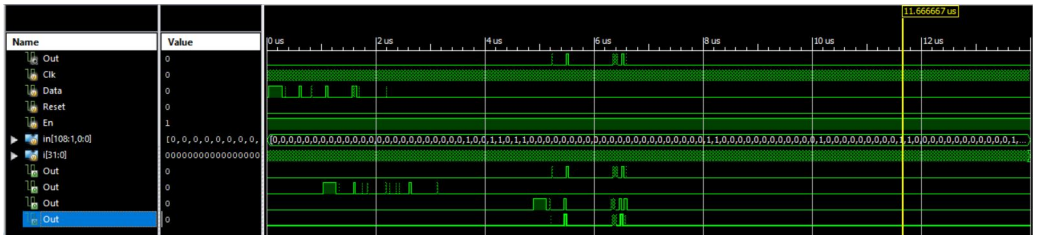
۵ گیرنده و فرستنده

این ماژول تنها پشت هم چیدن ماژول های فرستنده و گیرنده است زیرا همانطور که گفته شد این المان ها پسیو هستند. در اینجا باید همان ورودی که در وردی می‌دهیم را در خروجی دریافت کنیم.

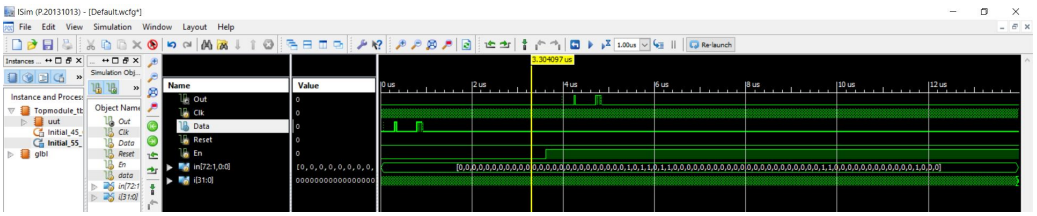


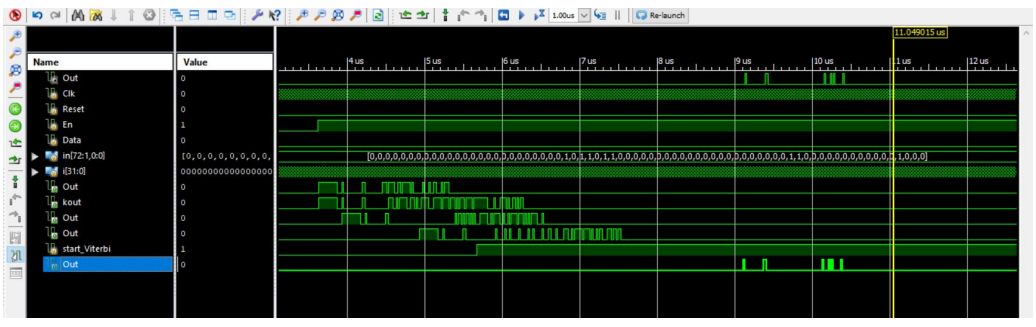
شکل ۷: فقط ورودی و خروجی

۱.۵ test



شکل ۸: فقط ورودی و خروجی





پایان