In the name of God

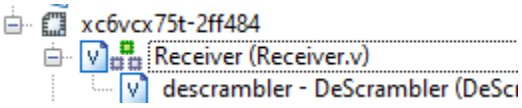
Arsalan Firoozi - 97102225

Project - Phase 1

کد های وریلاگ و متلب را به تمامی کامنت گذاری کردم و عملکرد آن در کد مشخص شده است.

Specification:

ساختار گیرنده به صورت زیر است:



ورودی و خروجی ها Receiver:

```
// Instantiate the module
Receiver instance_name (
    .Clk(Clk),
    .Reset(Reset),
    .x(x),
    .y(y),
    .num_pads(num_pads)
    );
```

این ماژول در صورتی که بیت های Preamble را در ورودی دریافت کند، داده ورودی را پردازش می کند.
در ورودی به صورت سریال تمام بیت های فریم را می گیرد و در خروجی به صورت سریال داده Descrambled را تحویل می دهد.

ورودی و خروجی های DeScrambler:

```
// Instantiate the module
DeScrambler instance_name (
    .Clk(Clk),
    .Reset(Reset),
    .x(x),
    .y(y),
    .Start(Start)
    );
```

این ماژول اگر بیت Start یک شود شروع به فعالیت می کند و در ورودی به صورت سریال داده های بخش Data فریم را دریافت می کند.
در خروجی به صورت سریال Descrambled شده این بیت ها را تحویل می دهد.
با پیاده سازی بخش Receiver به مشخصات زیر از Design رسیدم:

Device	
Family	Virtex6
Part	xc6vcx75t
Package	ff484
Temp Grade	Commercial
Process	Typical
Speed Grade	-2

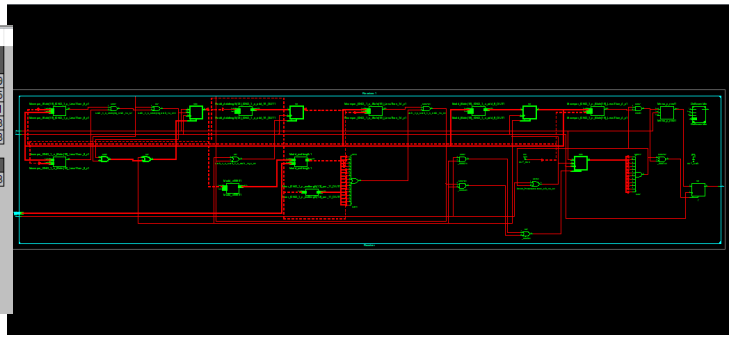
On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	--	--
Logic	0.000	111	46560	0
Signals	0.000	139	--	--
IOs	0.000	7	240	3
Leakage	1.293			
Total	1.293			

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.619	0.000	0.619
Vccaux	2.500	0.045	0.000	0.045
Vcco25	2.500	0.001	0.000	0.001
MGTAVcc	1.000	0.303	0.000	0.303
MGTAVtt	1.200	0.213	0.000	0.213

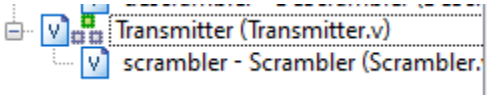
Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.7	81.5	53.5

Supply Power (W)	Total	Dynamic	Quiescent
	1.293	0.000	1.293



ساختار فرستنده به صورت زیر است:



ورودی و خروجی ها Transmitter:

```
// Instantiate the module
Transmitter instance_name (
    .Clk(Clk),
    .Reset(Reset),
    .x(x),
    .y(y),
    .Enable(Enable),
    .num_pads(num_pads)
    );
```

این ماژول در صورتی که بیت Enable یک باشد شروع به ارسال یک فریم می کند.
در ورودی این ماژول به صورت سریال تمام فریم بدون اسکرمبل و بدون Preamble دریافت می شود.
در خروجی فریم اسکرمبل شده که قبل از آن Preamble ارسال شده است به صورت سریال داده می شود.

ورودی و خروجی های Scrambler:

```
// Instantiate the module
Scrambler instance_name (
    .Clk(Clk),
    .Reset(Reset),
    .Start(Start),
    .Seed(Seed),
    .x(x),
    .y(y)
    );
```

این ماژول اگر بیت Start یک شود شروع به فعالیت می کند و در ورودی به صورت سریال داده های بخش Data فریم را دریافت می کند.
در خروجی به صورت سریال Scrambled شده این بیت ها را تحویل می دهد.
با پیاده سازی بخش Transmitter به مشخصات زیر از Design رسیدم:

Device	
Family	Virtex6
Part	xc6vcx75t
Package	ff484
Temp Grade	Commercial
Process	Typical
Speed Grade	-2

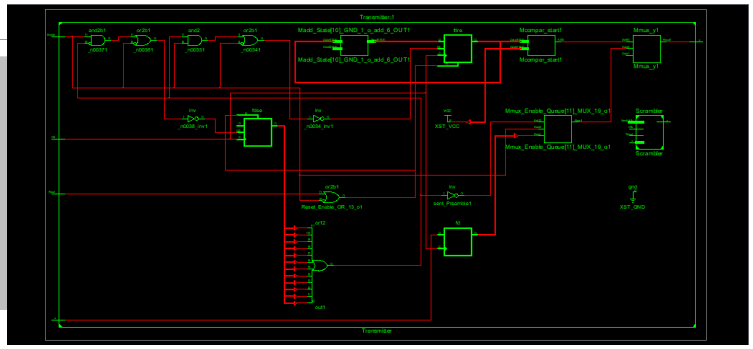
On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	--	--
Logic	0.000	160	46560	0
Signals	0.000	176	--	--
IOs	0.000	8	240	3
Leakage	1.293			
Total	1.293			

Supply Summary				
Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.619	0.000	0.619
Vccaux	2.500	0.045	0.000	0.045
Vcco25	2.500	0.001	0.000	0.001
MGTAVcc	1.000	0.303	0.000	0.303
MGTAVtt	1.200	0.213	0.000	0.213

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat_Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.7	81.5	53.5

Supply Power (W)	Total	Dynamic	Quiescent
	1.293	0.000	1.293



Design:

من در این فاز از 2 ماژول زیر برای گیرنده استفاده کردم:

1. **Receiver**: این ماژول در اصل کنترلر دی اسکرمبلر است و تصمیم میگیرید که دیتا ورودی باید دی اسکرمبلر شود یا نه. در این ماژول دیتا ورودی بررسی می شود و در صورتی که Preamble دیده شد همانند یک فریم بیت های بعدی را در نظر میگیرد. با توجه به فیلد Length در ورودی، این ماژول تا پایان فریم اطلاعات ورودی را پردازش میکند. و پس از آن باز برای دریافت شروع فریم جدید منتظر می ماند.
   جهت کاهش T_Critical Path در خروجی این ماژول یک فلیپ فلاپ قرار دادم.
   این ماژول علاوه بر کلاک ، سیگنال ریست و یک مقدار 3 بیتی نشان دهنده طول pad bits، یک بیت ورودی میگیرد و یک بیت خروجی می دهد.
   برای پیاده سازی این ماژول از یک رجیستر Length برای نگه داری فیلد طول داده استفاده می کنم که در سایکل هایی که داده های مربوط به طول داده وارد مدار می شود آپدیت می شود.
   همچنین از یک رجیستر Preamble برای برای تشخیص 12 بیت اولیه فریم استفاده شده که همیشه در حال آپدیت شدن است.
   Premablecheck نیز یک flag است که نشان دهنده خواندن اطلاعات فریم است که در صورت تطابق پریمبل فعال می شود و تا پایان دریافت فریم فعال می ماند.
   از endlength استفاده می شود برای تشخیص آخرین بیت دریافت شده.

2. **DeScrambler**: در این ماژول با توجه به اینکه هفت بیت اولیه فیلد سرویس قبل از اسکرمبلر صفر بوده است، ابتدا از طریق اولین هفت بیت پس از یک شدن سیگنال Start، با توجه به منطق نشان داده شده در زیر Seed را بدست می آورد.

Clk	State	Out
1	g f e d c b a	g⊕d
2	F e d c b a g⊕d	f⊕c
3	e d c b a g⊕d f⊕c	e⊕b
4	d c b a g⊕d f⊕c e⊕b	d⊕a
5	c b a g⊕d f⊕c e⊕b d⊕a	c⊕g⊕d
6	b a g⊕d f⊕c e⊕b d⊕a c⊕g⊕d	b⊕f⊕c
7	a g⊕d f⊕c e⊕b d⊕a c⊕g⊕d b⊕f⊕c	a⊕e⊕b
8	g⊕d f⊕c e⊕b d⊕a c⊕g⊕d b⊕f⊕c a⊕e⊕b	

با استفاده از بیت های هفت بیت اول می توان State در کلاک هشتم را بدست آورد. پس میتوان بدون مشکل از این State هشتم به بعد ادامه داد و تنها کافی است به ازای هفت بیت اول دریافتی، در خروجی صفر داشته باشیم. از این پس از همان منطق اسکرمبلر برای دی اسکرمبل کردن استفاده می شود. چون اگر یک بیت 2 بار در یک متغیر XOR شود همان متغیر بدست می آید.

در این ماژول علاوه بر ورودی های Clk Reset یک ورودی Start نیز دارد که در صورتی که فعال باشد اسکرمبل می کند. یک بیت ورودی که برای دریافت ورودی به صورت سریال است، و یک بیت خروجی که برای داده اسکرمبل شده به صورت سریالی است.

من در این فاز از 2 ماژول زیر برای فرستنده استفاده کردم:

1. **Transmitter:** این ماژول کنترلر اسکرمبلر است و تعیین می کند بیت های ورودی اسکرمبل شوند یا نه. همچنین تعداد بیت های ورودی سنجیده می شود و پس از ارسال کامل فریم دوباره Preamble فرستاده می شود و فرض می شود یک فریم جدید آمده است.
   به دلیل اینکه به صورت داخلی باید بیت های Preamble فرستاده شود و به عنوان فریم های ورودی این ماژول(طبق دیاگرام شکل زیر) نیست، بایستی 12 کلاک ابتدایی یک فرستاد. که برای اینکه این 12 کلاک از سرعت ارسال کم نکند از یک شیفت رجیستر 12 بیتی استفاده شده است.

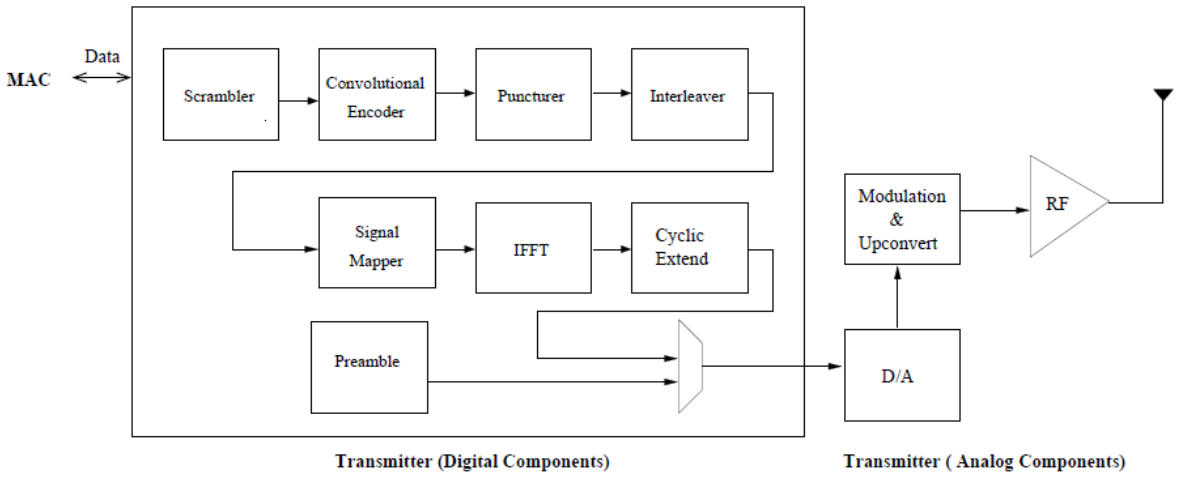


Figure 1: Top Level Diagram of an 802.11a Transmitter.

این ماژول علاوه بر کلاک ، سیگنال ریست و یک مقدار 3 بیتی نشان دهنده طول pad bits، یک بیت ورودی میگیرد و یک بیت خروجی می دهد.

برای پیاده سازی این ماژول از یک رجیستر Length برای نگه داری فیلد طول داده استفاده می کنم که در سایکل هایی که داده های مربوط به طول داده وارد مدار می شود آپدیت می شود.

همچنین از یک رجیستر Preamble برای برای تشخیص 12 بیت اولیه فریم استفاده شده که همیشه در حال آپدیت شدن است.

Premablecheck نیز یک flag است که نشان دهنده خواندن اطلاعات فریم است که در صورت تطابق پریمبل فعال می شود و تا پایان دریافت فریم فعال می ماند.

از endlength استفاده می شود برای تشخیص آخرین بیت ارسال شده.

2. **Scrambler :** با توجه به دیاگرام شکل زیر از یک شیفت رجیستر برای پیاده سازی آن استفاده کردم.

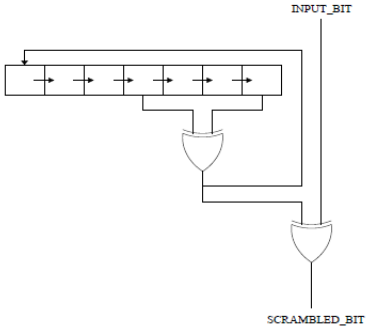


Figure 5: Simple Implementation of Scrambler.

در این ماژول علاوه بر ورودی های Clk Reset یک ورودی Start نیز دارد که در صورتی که فعال باشد اسکرمبل می کند. یک بیت ورودی که برای دریافت ورودی به صورت سریال است، و یک بیت خروجی که برای داده اسکرمبل شده به صورت سریالی است. همچنین به عنوان یک ورودی دیگر 7 بیتی به عنوان seed اولیه برای شروع گرفته می شود که در مدار من آن را برابر 111 دسیمال قرار دادم.

User Document:

1. **Transmitter :** ریست این مدار را به ریست کلی وصل کنید. از لحاظ زمانی باید ابتدا Enable را یک کرد و در همان سیکل از بیت MSB فیلد سرویس شروع شود تا پایان بیت LSB فیلد DATA. در خروجی از موقعی که Enable یک می شود داده خواهد داشت تا 12 بیت پس از دریافت آخرین بیت. همچنین قبل از بیت 18 در ورودی بایستی به عنوان ورودی یک سیگنال سه بیتی num_pads که نشان دهنده تعداد بیت های pad-bits است و از طریق محاسبه زیر باید حساب شود به آن داده شود:

   Ceil(Number of pad bits / byte)

2. **Receiver :** ریست این مدار را به ریست کلی وصل کنید. از لحاظ زمانی تا زمانی که شروع فریم تشخیص داده نشده است در خروجی صفر وجود دارد ودر صورت دریافت Preamble در ورودی بلافاصله دریافت بقیه فیلد ها تا پایان بیت های آن فریم خوانده می شود.
   در خروجی از موقعی که 12 بیت Preamble دریافت می شود داده موثق خواهد داشت. همچنین قبل از بیت 30 در ورودی بایستی به عنوان ورودی یک سیگنال سه بیتی num_pads که نشان دهنده تعداد بیت های pad-bits است و از طریق محاسبه زیر باید حساب شود به آن داده شود:

   Ceil(Number of pad bits / byte)

Testing:

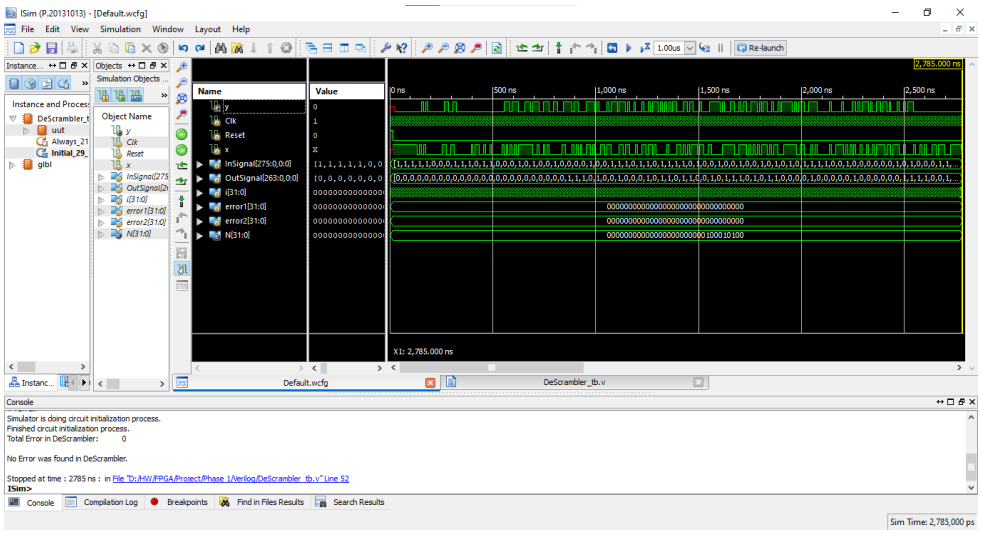
تست بنچ DeScrambler_tb ==> Receiver:

در این تست بنچ من از دو فایل که در کد متلب ساخته شده میخوانم:

1. In.txt: شامل کل بیت های یک فریم رندوم است.
2. Out_Scramble.txt: شامل خروجی دی اسکرمبل است که کل فریم است بدون بخش Preamble

در تست بنچ محتوان Out_Scramble را به عنوان ورودی به Reveiver می دهم و خروجی را با In.txt چک میکنم.

با توجه به عکس زیر به تطابق 100% رسیدم:



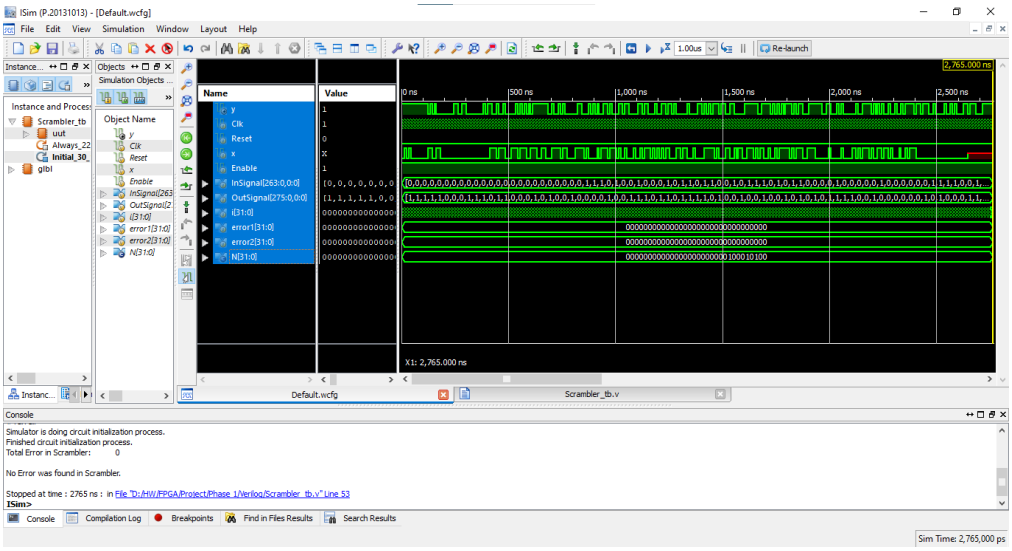
تست بنچ Scrambler_tb ==> Transmitter:

در این تست بنچ من از دو فایل که در کد متلب ساخته شده میخوانم:

3. In.txt: شامل کل بیت های یک فریم رندوم است.
4. Out_Scramble.txt: شامل خروجی دی اسکرمبل است که کل فریم است بدون بخش Preamble

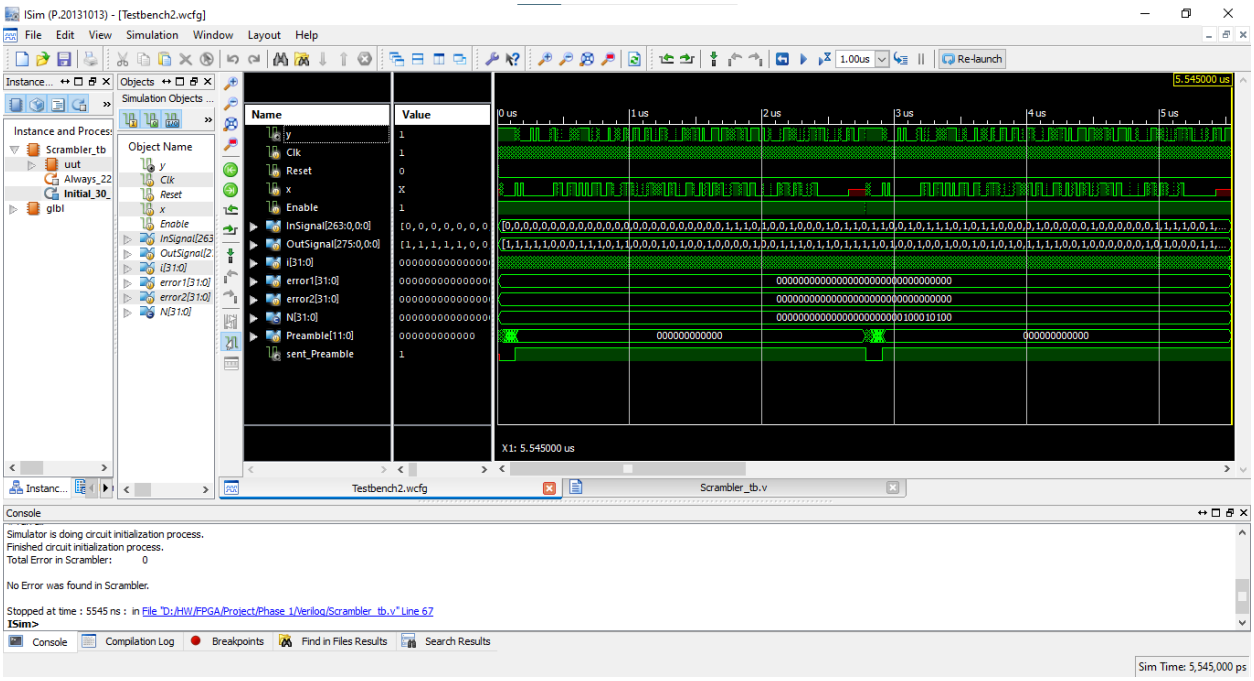
در تست بنچ محتوان In را به عنوان ورودی به Reveiver می دهم و خروجی را با Out_Scramble.txt چک میکنم.

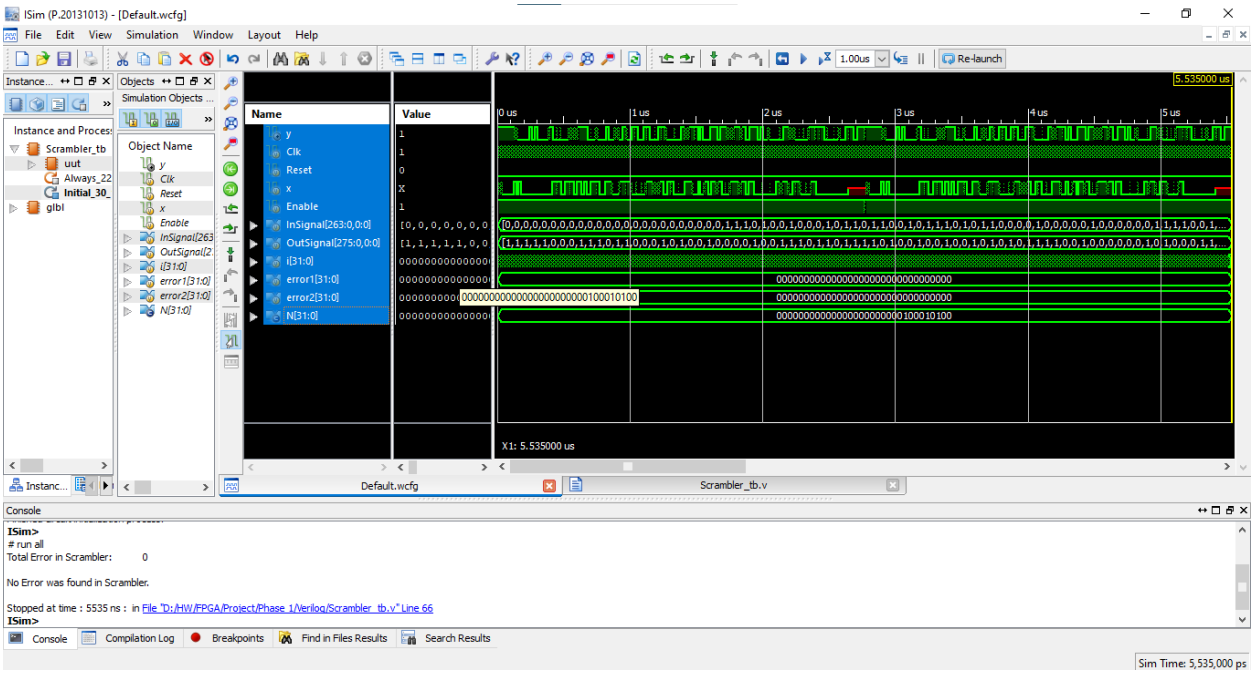
با توجه به عکس زیر به تطابق 100% رسیدم:



برای تولید بیت های تست بنچ از کد متلب استفاده کردم. در همان کد متلب الگوریتم پیاده شده در وریلاگ را نیز پیاده سازی کردم و از صحت آن مطمئن شدم.

همچنین من 2 فریم دیتا را هم به فرستنده دادم و هم به گیرنده. در هر دو ماژول با توجه به نتایج زیر نتایج مطلوب دریافت شد.





نتایج بالا تایید می کند که ورودی و خروجی بلوک زیر یکسان است: