

به نام خدا

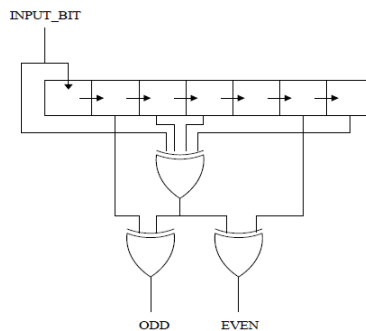
ارسلان فیروزی - ۹۷۱۰۲۲۲۵

پروژه فاز ۲ - FPGA

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

Specification:

۱. ساختار Encoder: به صورت شکل زیر است:



با استفاده از ۳ XOR خروجی های Odd و Even به ازای هر ورودی تحویل داده می شود. چون ورودی و خروجی به صورت سریال است، در خروجی برای اینکه به صورت سریال داده ها تحویل داده شود، بایستی از یک صف استفاده می کردم. که با استفاده از یک شیفت رجیستر و یک مدار ترتیبی برای تعیین مکان پر شدن اطلاعات در این شیفت رجیستر این کار را انجام دادم.

ورودی و خروجی های انکودر:

```
// Instantiate the module
Encoder instance_name (
    .Clk(Clk),
    .Reset(Reset),
    .x(x),
    .Out(Out),
    .Start(Start)
);
```

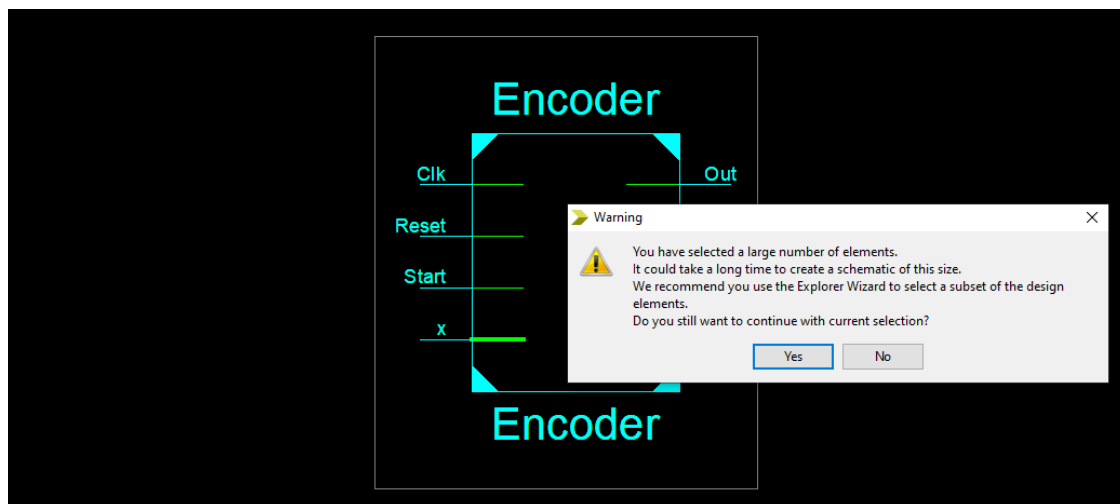
در این مازول برای انکود کردن ورودی، بایستی به صورت سریال از همان سیکلی که سیگنال Start یک می شود، ورودی ها داده شود. در خروجی دقیقا از یک سیکل بعد سیگنال معتبر وجود دارد.

با پیاده سازی این بخش به مشخصات زیر از لحاظ توان و RTL و میزان استفاده از منابع FPGA رسیدم:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1049	93120	1%
Number of Slice LUTs	2311	46560	4%
Number of fully used LUT-FF pairs	1034	2326	44%
Number of bonded IOBs	5	240	2%
Number of BUFG/BUFGCTRLs	1	32	3%

Device		On-Chip	Power (W)	Used	Available	Utilization (%)	Supply Summary		Total	Dynamic	Quiescent	
Family	Virtex6	Clocks	0.000	1	---	---	Source	Voltage	Current (A)	Current (A)	Current (A)	
Part	xc6vcx75t	Logic	0.000	2298	46560	5	Vccint	1.000	0.619	0.000	0.619	
Package	ff484	Signals	0.000	2338	---	---	Vccaux	2.500	0.045	0.000	0.045	
Temp Grade	Commercial	IOs	0.000	5	240	2	Vcco25	2.500	0.001	0.000	0.001	
Process	Typical	Leakage	1.293				MGTA Vcc	1.000	0.303	0.000	0.303	
Speed Grade	-2	Total	1.293				MGTA Vtt	1.200	0.213	0.000	0.213	
Environment		Thermal Properties										
Ambient Temp (C)	50.0	Effective TJA (C/W)	2.7	Max Ambient (C)	81.5	Junction Temp (C)	53.5	Supply Power (W)		Total	Dynamic	Quiescent
Use custom TJA?	No									1.293	0.000	1.293
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											

به دلیل این خطا و بنظر نداشتن سخت افزار کافی در لپ تاپ من پس از این خطا قادر به نمایش RTL نبود و نتوانستم در اینجا این مورد را اضافه کنم.




۲. ساختار دیکودر: (Decoder_Viterbi)

ورودی و خروجی ها:

```
// Instantiate the module
Decoder_Viterbi instance_name (
    .Clk(Clk),
    .Reset(Reset),
    .x(x),
    .Out(Out),
    .Start(Start),
    .Valid(Valid),
    .Length(Length)
);
```

در این ماژول برای دیکود کردن ورودی، بایستی به صورت سریال از همان سیکلی که سیگنال Start یک می شود، ورودی ها داده شود. سپس پس از اینکه به تعداد $Length * 2$ (یکی از ورودی های ماژول) بیت در ورودی دریافت کرد، در $(1 + Length)$ سایکل بعد به تعداد Length سیکل خروجی معتبر خواهید داشت و خروجی معتبر با سیگنال Valid در خروجی مشخص می شود.

با پیاده سازی این بخش به مشخصات زیر از لحاظ توان و RTL و میزان استفاده از منابع FPGA رسیدم:

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	1,174	93,120	1%	
Number used as Flip Flops	1,174			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	4,320	46,560	9%	
Number used as logic	3,664	46,560	7%	
Number using O6 output only	1,736			
Number using O5 output only	896			
Number using O5 and O6	1,032			
Number used as ROM	0			
Number used as Memory	528	16,720	3%	
Number used as Dual Port RAM	0			
Number used as Single Port RAM	528			
Number using O6 output only	528			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Shift Register	0			
Number used exclusively as route-thrus	128			
Number with same-slice register load	0			
Number of occupied Slices	1,200	11,640	10%	
Number of LUT Flip Flop pairs used	4,322			
Number with an unused Flip Flop	3,467	4,322	80%	
Number with an unused LUT	2	4,322	1%	
Number of fully used LUT-FF pairs	853	4,322	19%	
Number of unique control sets	7			
Number of slice register sites lost to control set restrictions	10	93,120	1%	
Number of bonded IOBs	15	240	6%	
Number of RAMB36E1/FIFO36E1s	0	156	0%	
Number of RAMB18E1/FIFO18E1s	0	312	0%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of ILOGICE1/ISERDESE1s	0	360	0%	
Number of OLOGICE1/OSERDESE1s	0	360	0%	
Number of BSCANs	0	4	0%	
Number of BUFHCEs	0	72	0%	
Number of BUFIODQSs	0	36	0%	
Number of BUFRs	0	18	0%	
Number of CAPTUREs	0	1	0%	
Number of DSP48E1s	0	288	0%	
Number of EFUSE_USRs	0	1	0%	

Device		On-Chip	Power (W)	Used	Available	Utilization (%)	Supply Summary		Total	Dynamic	Quiescent
Family	Virtex6	Clocks	0.000	1	---	---	Source	Voltage	Current (A)	Current (A)	Current (A)
Part	xc6vcx75t	Logic	0.000	4320	46560	9	Vccint	1.000	0.619	0.000	0.619
Package	ff484	Signals	0.000	4264	---	---	Vccaux	2.500	0.045	0.000	0.045
Temp Grade	Commercial	IOs	0.000	15	240	6	Vcco25	2.500	0.001	0.000	0.001
Process	Typical	Leakage	1.293	---	---	---	MGTAVcc	1.000	0.303	0.000	0.303
Speed Grade	-2	Total	1.293	---	---	---	MGTAVt	1.200	0.213	0.000	0.213

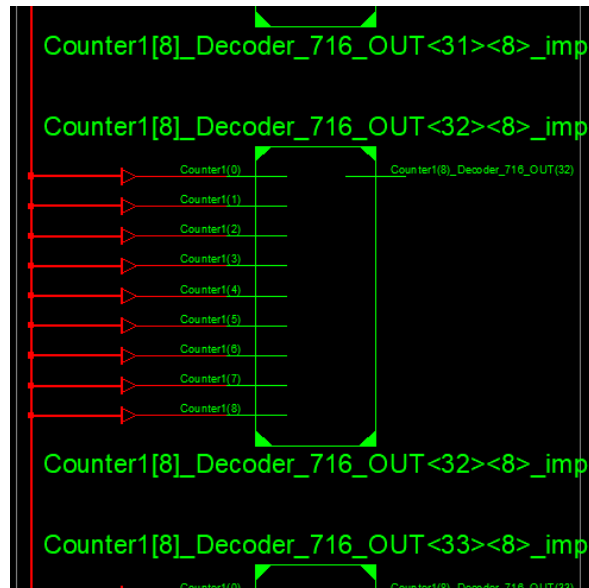
Environment		Thermal Properties		Effective TJA	Max Ambient	Junction Temp
Ambient Temp (C)	50.0			(C/W)	(C)	(C)
Use custom TJA?	No			2.7	81.5	53.5
Custom TJA (C/W)	NA					
Airflow (LFM)	250					
Heat Sink	Medium Profile					
Custom TSA (C/W)	NA					
Board Selection	Medium (10"x10")					
# of Board Layers	8 to 11					
Custom TJB (C/W)	NA					
Board Temperature (C)	NA					

Supply		Total	Dynamic	Quiescent
Supply	Power (W)	1.293	0.000	1.293

بخش دیگری از RTL:



بخشی از RTL:



Design:

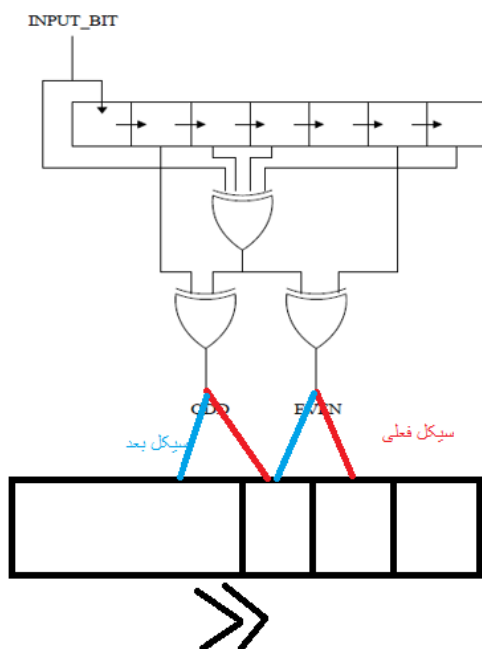
۱. Encoder: برای انکودر من از یک شیفت رجیستر که در هر سیکل یک بیت به سمت راست شیفت پیدا می کند و یک سیگنال که از یک مدار ترتیبی حاصل می شود و تعیین می کند که برای ذخیره بیت های زوج و فرد در این شیفت رجیستر در چه خانه ای از آن استفاده شود، استفاده کردم.

چون در هر سیکل یک بیت به راست شیفت وجود دارد و ما در هر سیکل نیاز داریم یک جفت بیت ذخیره کنیم، جایگاه ذخیره کافی است یک شمارنده با شروع از صفر که در هر سیکل یک واحد افزایش می یابد باشد.

این ماژول حداکثر فریم با تعداد بیت ۲۰۴۸ را پشتیبانی می کند.

در این ماژول علاوه بر ریست از یک سیگنال Start استفاده می کنم که کنترلر هرگاه دیتا موثق در ورودی این ماژول قرار داد، باید این سیگنال را یک کند. در صورت صفر بودن عملکرد ریست را خواهد داشت.

ورودی ها به صورت سریال دریافت و خروجی نیز به صورت سریال خواهد بود.



این ماژول به طور کامل تست شد و با متلب تطابق کامل داشت.

۲. Decoder-Viterbi :

در این ماژول باید الگوریتم داینامیک پروگرامینگ ویتربی را پیاده سازی کنم. برای اینکار مطابق توضیحات زیر کد وریداگ زده شده و به تمامی کامنت گذاری شده است. این ماژول با چند State کار می کند: (در پرانتز تعداد سایکلی که هرکدام نیاز دارند نوشته شده است). \leq توسط TotalControl این state ها کنترل می شوند.

- صفر: پر کردن و محاسبه مقادیر ارور همه مسیر ها در حالت های بهینه و نگه داری نحوه تغییر state ها در هر یک از مسیر های بهینه ($2 * \text{Length}$ سایکل)
 - یک: پیدا کردن بهترین مسیر از مسیر های بهینه محاسبه شده و بازگشت به عقب و به دست آوردن تک تک بیت های خروجی و ذخیره در یک صف برای ارسال خروجی در state دوم. ($\text{Length} + 1$ سایکل)
 - دو: خروجی دادن تمام بیت های ذخیره شده در صف (Length سایکل)
- تمام بیت های خروجی همان بیت MSB در state های بهینه ترین مسیر است.

در این ماژول از دو بخش حافظه، یکی برای نگه داری هزینه های مسیر های بهینه به ابعاد 2^6 Cost_Length Bits x و یکی برای نگه داری مسیر های بهینه به ابعاد $2^6 \times \text{Frame_Length} \times 6$ Bits استفاده شده است.

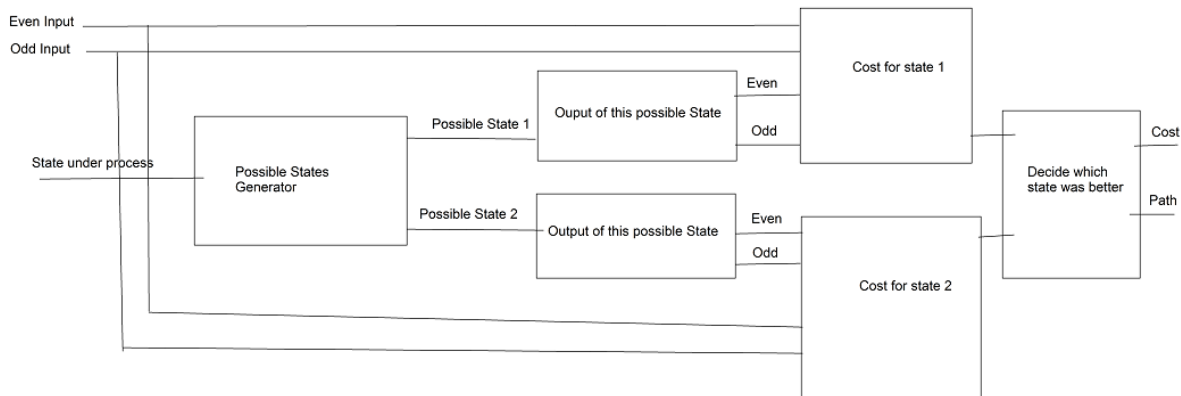
از شمارنده Counter1 برای کنترل تعداد سایکل های هر state استفاده کردم که در پرانتز توضیحات state ها نوشته شده است.

به دلیل اینکه در state صفر باید برای هر عملیات از ۲ بیت آخر در ورودی استفاده شود و یکی در میان پردازش انجام شود، از سیگنال Valid_i که فرکانس نصف کلاک را دارد استفاده کردم، از ۲ بیت رجیستر In که نماینده دو بیت ورودی آخر است، و سیگنال flag استفاده می شود.

علت استفاده از flag این است که در ریست سیگنال Valid_i را صفر میکنیم و مواقعی که صفر شده است پردازش را انجام میدهیم. درست پس از ریست یا سیگنال Start این سیگنال

صفر است درحالیکه دو بیت ورودی صحیح گرفته نشده است. برای همین از سیگنال **flag** استفاده کردم تا اولین باری که **Valid_i** صفر است را نادیده بگیرم.

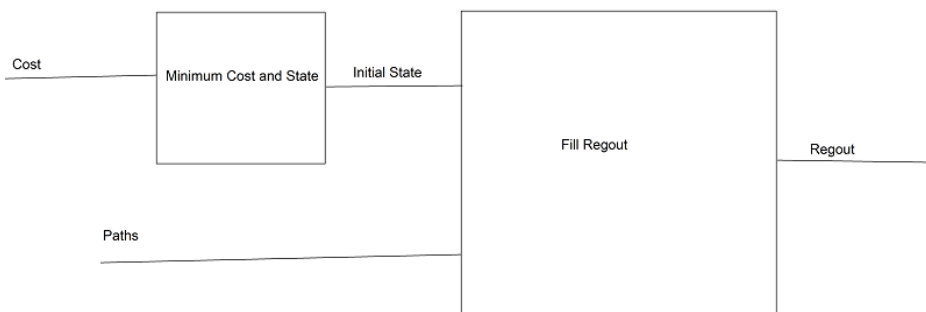
در **state** صفر میتوانم از شماتیک زیر برای محاسبه **Cost** و مسیر مناسب استفاده کنم: (در کد **always** بلوک اول می شود).



state های احتمالی همان **state** مورد بررسی است که یک بیت به چپ شیفت داده شده و **LSB** میتواند صفر و یک باشد. در نتیجه ۲ **state** احتمالی وجود دارد.

محاسبه خروجی این **state** های احتمالی از طریق **xor** برخی بیت های آن

در **state** یک از یک بلوک برای محاسبه کمینه کاست بین تمام **state** ها و بلوکی برای پرکردن **RegOut** استفاده کردم:



در **state** دوم نیز که تنها یک رجیستر شیفت داده شده است.

در این ماژول علاوه بر ریست از یک سیگنال **Start** استفاده می کنم که کنترلر هرگاه دیتا موثق در ورودی این ماژول قرار داد، باید این سیگنال را یک کند. در صورت صفر بودن عملکرد ریست را خواهد داشت.

ورودی ها به صورت سریال دریافت و خروجی نیز به صورت سریال خواهد بود.

خروجی معتبر با سیگنال **Valid** در خروجی مشخص می شود.

User Documents:

۱. **Encoder**: برای شروع انکود کردن بایستی در یک سیکل سیگنال **Start** یک شود و در همان

سیکل ورودی به صورت سریال به پورت **X** داده شود. در دومین سیکل پس از یک کردن

سیگنال **Start**، خروجی معتبر خواهیم داشت. در خروجی ابتدا بیت **Even** و سپس بیت **Odd**

انتقال داده می شود. در یک بار یک کردن **Start** می توان حداکثر فریم با تعداد بیت ۲۰۴۸ را انکود کرد.

۲. **Decoder_Viterbi**: برای استفاده از این ماژول باید در ورودی تعداد بیت هایی که می خواهیم

دیکود کنیم **Length** را ذخیره کرد. (این سیگنال می تواند دیر تر از سیگنال **Start** تعیین شود

و در این صورت باید مقداری برابر با همه بیت ها ۱ داشته باشد.) سپس باید سیگنال **Start** را

یک کرد و در همان سیکل به تعداد $Length * 2$ سیکل داده انکود شده به صورت سریال به

مدار داده شود و سپس پس از $Length + 1$ سیکل از آخرین بیت، در خروجی دیتا موثق

خواهیم داشت. سیگنال **Valid** در صورت خروجی معتبر یک خواهد بود.

Testing:

تست بنچ **Encoder** ➔ **Encoder_tb** :

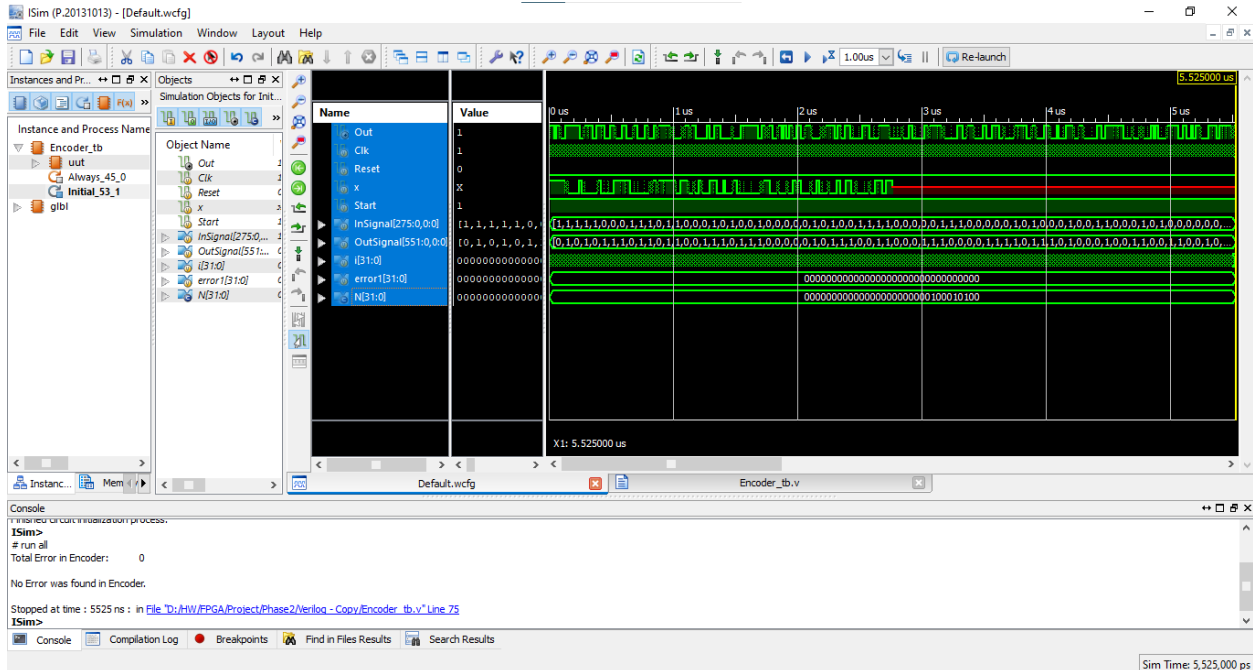
در این تست بنچ من از دو فایل که در کد متلب ساخته شده میخوانم:

۱. شامل کل بیت های یک فریم رندوم است. **Out_Scramble.txt**:

۲. شامل خروجی دیکودر در متلب است. **Out_Encoder.txt**.

خروجی ماژول را با Out_Encoder مقایسه میکنم.

با توجه به عکس زیر به تطابق ۱۰۰٪ رسیدم:



تست بنچ Decoder_Viterbi → Decoder_Viterbi_tb :

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

۱. شامل کل بیت های یک فریم رندوم است. : Out_Encoder.txt

۲. شامل خروجی دیکودر در متلب است. Out_DeCoder2.txt

خروجی ماژول را با Out_DeCoder2 مقایسه میکنم.

با توجه به عکس زیر به تطابق ۱۰۰٪ رسیدم:

