This technical report is available in **English** as a PDF file named **Convolution_Report_English** along with the project files.

نام پروژه : پیاده‌سازی کانولوشن مبتنی بر FPGA (صفحه پروژه در گیت‌هاب)

نام طراح : میلاد زادرفیع

تاریخ ایجاد و نسخه پروژه : آبان ۱۴۰۴ – ورژن ۱.۰

نرم‌افزار و زبان طراحی : Xilinx ISE 14.7 – VHDL

راه‌های ارتباطی با طراح :

https://www.linkedin.com/in/zadrafee
https://www.github.com/zadrafee

این مستند بخشی از مجموعه پروژه‌های FPGA است که در وب‌سایت LogLab منتشر می‌شود. برای مشاهده سایر پروژه‌های مرتبط و مقالات فنی، به وب‌سایت رسمی من مراجعه کنید:

http://www.LogLab.ir

شرح فایل‌ها و فولدرهای اصلی پروژه

Convolution_MATLAB	Compare.m	ساخت ورودی و مقایسه خروجی ISE و MATLAB
	Convolution.slx	شبیه‌سازی کانولوشن در Simulink
Convolution_VHDL	Convolution_VHDL.xise	پروژه کانولوشن در ISE
	Convolution_Top.vhd	تاپ ماژول کانولوشن در ISE
	Convolution_VHDL_tb.vhd	تست بنچ کانولوشن در ISE
	Multiply_Conjugate.vhd	ماژول ضرب کننده مختلط و مزدوج کننده
Convolution_Report_Persian.pdf		گزارش فنی کانولوشن به زبان فارسی
Convolution_Report_English.pdf		گزارش فنی کانولوشن به زبان انگلیسی
Convolution_RTL.pdf		شماتیک RTL طرح کانولوشن پیاده‌سازی شده
xfft_ds260.pdf		دیتاشیت IP FFT نسخه ۷.۱ Native

چکیده

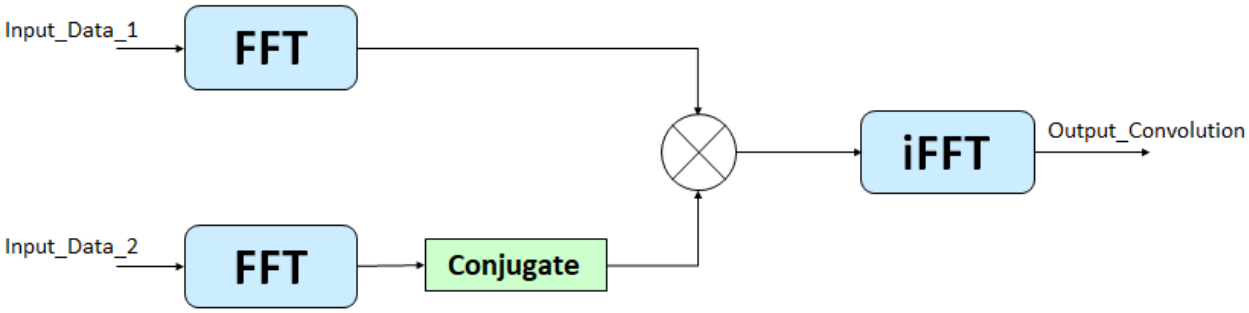
در این پروژه، عمل کانولوشن گسسته با استفاده از زبان VHDL و در محیط Xilinx ISE پیاده‌سازی شده است. این طراحی با هدف پردازش سریع داده‌ها بر روی FPGA انجام شده و می‌تواند به عنوان بخشی از سیستم‌های پردازش تصویر و سیگنال مورد استفاده قرار گیرد.

مقدمه

کانولوشن یک عملگر ریاضی است که برای ترکیب دو سیگنال یا تابع استفاده می‌شود تا نشان دهد چگونه شکل یکی از آن‌ها روی دیگری تأثیر می‌گذارد.

به زبان ساده‌تر، کانولوشن یعنی یک سیگنال را روی سیگنال دیگر حرکت دهیم (شیفت بدهیم) و در هر موقعیت، میزان هم‌پوشانی آن‌ها را محاسبه کنیم. نتیجه این فرایند، سیگنال جدیدی است که بیانگر میزان شباهت یا تأثیر متقابل دو سیگنال در طول زمان است. این تعریفی ساده از کانولوشن، به دور از محاسبات و اثبات‌های پیچیده ریاضی است. در حوزه پردازش سیگنال و تصویر، کانولوشن به ما کمک می‌کند تا عملیات‌هایی مثل فیلتر کردن، استخراج ویژگی، حذف نویز، تشخیص لبه‌ها، یا هموارسازی داده‌ها را انجام دهیم.

توصیف کلی سیستم و IP FFT

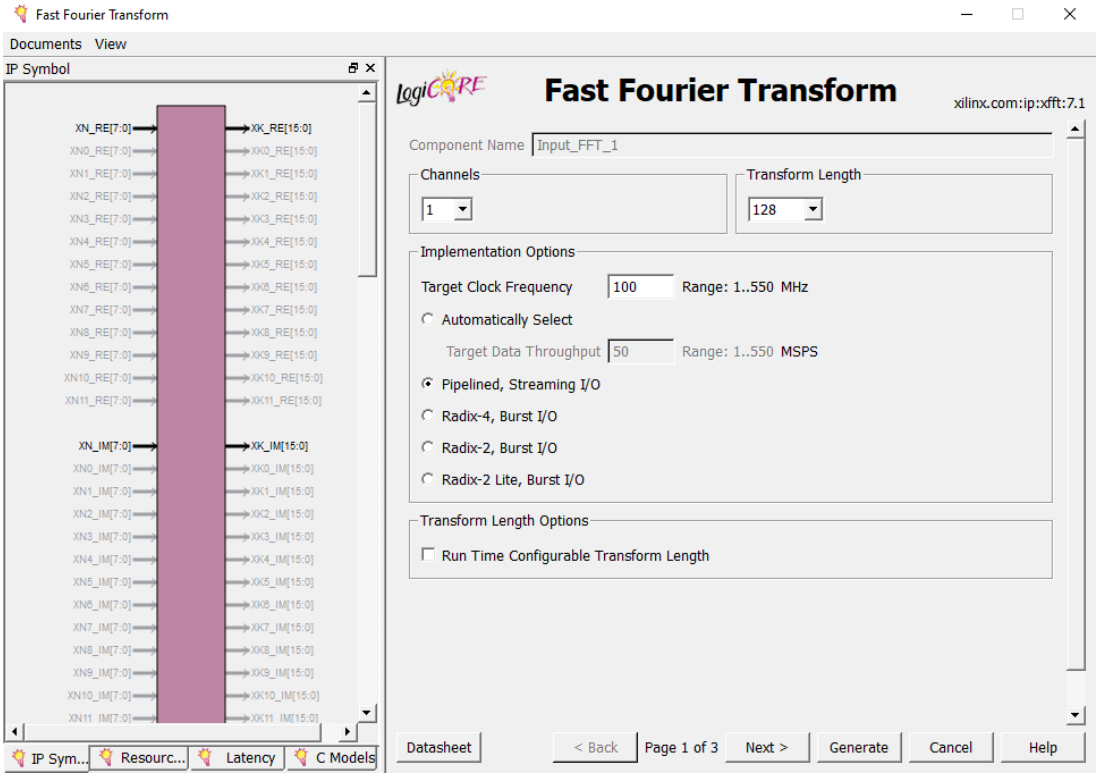


در تصویر بالا شمای کلی سیستم کانولوشن پیاده‌سازی شده در این پروژه را مشاهده می‌کنید. نمونه اعمال شده به ورودی Input_Data_1 توسط نرم‌افزار MATLAB به کمک تابع rand تولید شده است که یک Vector ۱۲۸ عددی است و همین طور نمونه اعمال شده به ورودی Input_Data_2 همان ۱۲۸ نمونه قبلی بوده که به تعداد ۱۰ عدد شیفت پیدا کرده است. کد تولید این ورودی‌ها در فایل Compare در پوشه Convolution_MATLAB موجود است.

روش کار دقیقا همان اصولی است که در کتاب‌های سیگنال و سیستم آورده شده است. دو ورودی که هر کدام دارای ۱۲۸ نمونه هستند را به دو FFT با مشخصات یکسان می‌دهیم، البته بیان این نکته در اینجا حائز اهمیت است که استفاده از دو FFT جهت فهم راحت‌تر بوده در حالی که روش اصولی‌تر استفاده از یک FFT دو کاناله

است که باعث می‌شود منابع سخت افزاری کمتری نیز استفاده شود البته سرعت تولید خروجی هم به نسبت کمتر می‌شود. ورودی‌های اعمالی ما اعداد صحیح سه رقمی کوچکتر از ۲۰۰ هستند به همین خاطر می‌بایست ورودی FFT ها ۸ بیت باینری باشند لذا در صفحه تنظیمات GUI مربوط به FFT باید Transform Length را بر روی ۱۲۸ تنظیم کنیم. در این پروژه فرض شده کلاک مدار ما 100Mhz است. پس Target Clock Frequency را بر روی 100Mhz تنظیم می‌کنیم.

در این پروژه از FFT نوع Native نسخه ۷.۱ استفاده شده است (دیتاشیت این FFT ضمیمه فایل‌های پروژه شده است) ، لذا بدیهی است از این کد نمی‌توان به طور مستقیم برای FFT با پروتکل ارتباطی AXI4-Stream استفاده کرد و نیاز به برخی اصلاحات و تغییرات دارد البته اصول کار یکسان است فقط نیاز است با نحوه کارکرد پروتکل ارتباطی AXI4-Stream آشنا باشید تا تغییرات لازم را برای عملکرد صحیح بر روی کد اعمال کنید.

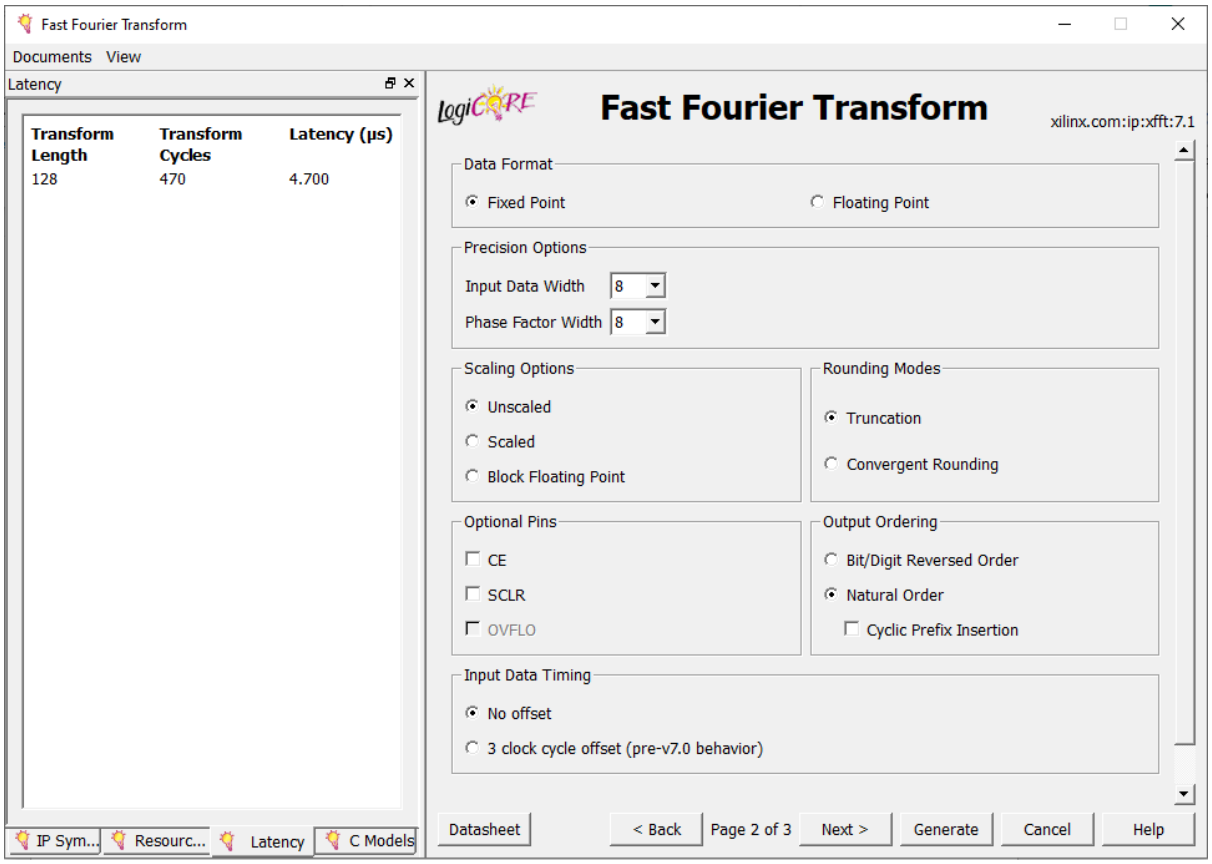


در بخش Implementation Options نوع معماری که FFT قرار است پیاده‌سازی کند را مشخص می‌کنیم. در جدول زیر تفاوت‌های هر کدام از انواع پیاده‌سازی را مشاهده می‌کنید.

نوع معماری	سرعت	تاخیر	مصرف منابع	کاربرد
Pipelined, Streaming	بسیار بالا	کم	زیاد	پردازش زنده (Real-time)
Radix-4, Burst	بالا	متوسط	متوسط	سیستم‌های نیمه‌زنده
Radix-2, Burst	متوسط	زیاد	کم	سیستم‌های کوچک
Radix-2 Lite, Burst	کم	زیاد	خیلی کم	کاربردهای کم‌هزینه

ما در این پروژه نوع پیاده‌سازی را بر روی Pipelined, Streaming تنظیم کردیم تا Throughput بالا و Latency پایین داشته باشیم. Latency ، FFT ما در این پروژه $4.7\mu s$ است.

در صورتی که پروژه شما به گونه‌ای باشد که نیاز داشته باشید طول تبدیل (تعداد نقاط FFT) را به صورت زنده تغییر دهید باید از بخش Transform Length Options کمک بگیرید و تیک گزینه Run Time Configurable Transform Length را بزنید.



در صفحه بعدی تنظیمات IP FFT می‌توانید نوع کوانتیزاسیون را مشخص کنید. در صورت انتخاب Fixed Point حداقل تعداد بیت با کمترین میزان خطا برای کلیه ورودی‌های اصلی و میانی در نظر گرفته می‌شود. با انتخاب گزینه Floating Point همه محاسبات با بالاترین میزان دقت انجام می‌گیرد و البته که منابع بسیاری را استفاده می‌کند که مطلوب ما نیست. همان طور که پیشتر دلیل‌اش را گفتیم ورودی‌ها را ۸ بیتی انتخاب می‌کنیم. ضریب Phase Factor همان ضریب Twiddle Factor در Butterfly هاست. انتخاب دقیق این ضریب نیاز به محاسبات دارد ولی به طور معمول آن را عددی برابر با تعداد ورودی‌ها انتخاب می‌کنند.

به تغییر محل نقطه باینری عمل Scaling گفته می‌شود ما با توجه به نیازمندی‌هایمان در این پروژه آن را بر روی Unscaled تنظیم می‌کنیم.

بخش Rounding Modes نوع کوتاه کردن عرض بیت سیگنال‌ها را مشخص می‌کند در صورتی که بر روی Truncation تنظیم شده باشد بخش کم ارزش بیت‌ها را بدون گرد کردن برش می‌دهد ولی حالت دیگر یعنی

Convergent Rounding علاوه بر برش دادن و کم کردن بیت‌ها برای ایجاد دقت بهتر آن را به صورت مناسب گرد هم می‌کند که پرواضح است باعث استفاده منابع سخت افزاری بیشتری می‌شود. انتخاب این گزینه بستگی به نوع پروژه شما و دقت مورد نیاز و همین طور منابع سخت افزاری موجود دارد. ما در این پروژه آن را بر روی Truncation تنظیم کردیم.

گزینه CE مختصر شده Clock Enable بوده و به منظور کنترل سیگنال کلاک مورد استفاده قرار می‌گیرد.

SCLR مختصر شده عبارت Synchronous Clear است و سیگنالی برای پاک کردن یا ریست همزمان (Synchronous Reset) کل هسته‌ی FFT در هنگام کار است، وقتی پین SCLR فعال (1) شود، تمام رجیسترهای داخلی FFT پاک می‌شوند. شمارنده‌ها، بافرها و وضعیت درونی به حالت اولیه برمی‌گردند. این کار در لبه‌ی فعال کلاک (clk) انجام می‌شود به همین دلیل به آن Synchronous می‌گویند. و زمانی که سیگنال دوباره ’0‘ شود، FFT از ابتدا آماده‌ی دریافت داده می‌شود.

تفاوت SCLR با ARESET یا ACLK

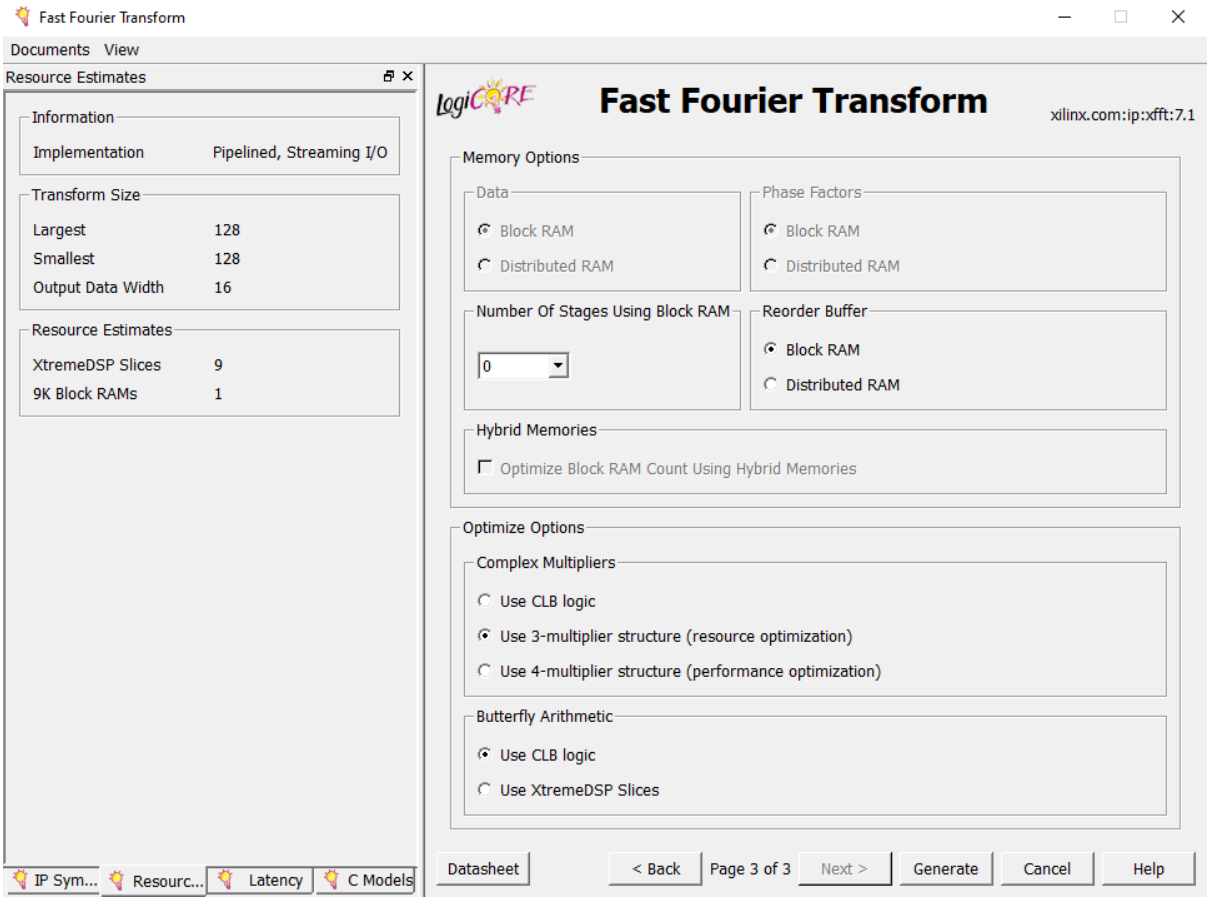
همان طور که گفته شد SCLR نوع ریستاش Synchronous Reset است و فقط در لبه‌ی کلاک عمل می‌کند که معمولاً سطح فعال ’1‘ دارد. ولی در IP های جدیدتر ARESETN یا ACLK نوع ریست‌شان Asynchronous Reset بوده و بلافاصله بدون وابستگی به کلاک عمل می‌کند، و معمولا سطح فعال ’0‘ دارد.

در بخش Output Ordering نوع نمایش خروجی مشخص می شود که Bit/Digit Reversed Order باشد یا Natural Order ، در روش Natural Order خروجی‌ها به همان ترتیب طبیعی Index هایشان از FFT خارج می‌شوند. و اما در روش Bit/Digit Reversed Order هسته FFT برای سریع بودن از یه ساختار خاص به اسم Butterfly استفاده می‌کند. این ساختار باعث می‌شود ترتیب داده‌ها در ورودی یا خروجی، دیگر به ترتیب معمول (0,1,2,3,…) نباشند. به همین خاطر، در انتهای محاسبه‌ی FFT، داده‌ها به‌صورت «درهم‌ریخته» درمی‌آیند یعنی indexهایشان به ترتیب خاصی جابجا می‌شوندکه به آن Bit-Reversed Order یا Digit-Reversed Order می‌گویند.

Index طبیعی	باینری	Bit-Reversed	عدد جدید
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3

در این طرح چون ما نیاز داریم خروجی‌ها به صورت مرتب شده بر اساس Index هایشان باشد از روش Natural Order استفاده می‌کنیم.

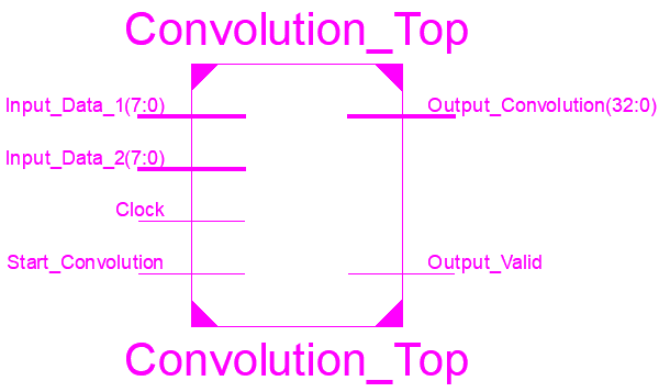
در بخش انتهایی این قسمت بخشی به نام Input Data Timing وجود دارد که مربوط به IP های FFT قدیمی با نسخه قبل از ۷ می‌باشد. در صورتی که از نسخه ۷ به بعد استفاده می‌کنید این گزینه را بر روی No Offset قرار دهید.



در صفحه انتهایی تنظیمات مربوط به حافظه وجود دارد که می‌توان نوع حافظه Data و Phase Factors را مشخص کرد که از نوع Block RAM باشد یا Distibuted RAM ، معمولا به دلیل اینکه حافظه‌های Distributed RAM فضای زیادی از FPGA ما را اشغال می‌کنند و باعث کند شدن تراشه FPGA می‌شوند به صورت Default بر روی Block RAM تنظیم شده‌اند.

بخش انتهایی تنظیمات یعنی Optimize Options جهت بهینه‌سازی ضرب‌کننده‌های مختلط بوده و طبق توضیحات نوشته شده توسط شرکت Xilinx در زیر بخش Complex Multiplier گزینه دوم یعنی Use 3-multiplier structure به لحاظ منابع مصرفی گزینه بهینه‌ای است و گزینه سوم یعنی Use 4-multiplier structure به لحاظ عملکرد (منظور Throughput و Latency) نتایج بهتری را رقم می‌زند ولی منابع سخت‌افزاری بیشتری را مصرف می‌کند.

توضیح ماژول‌ها

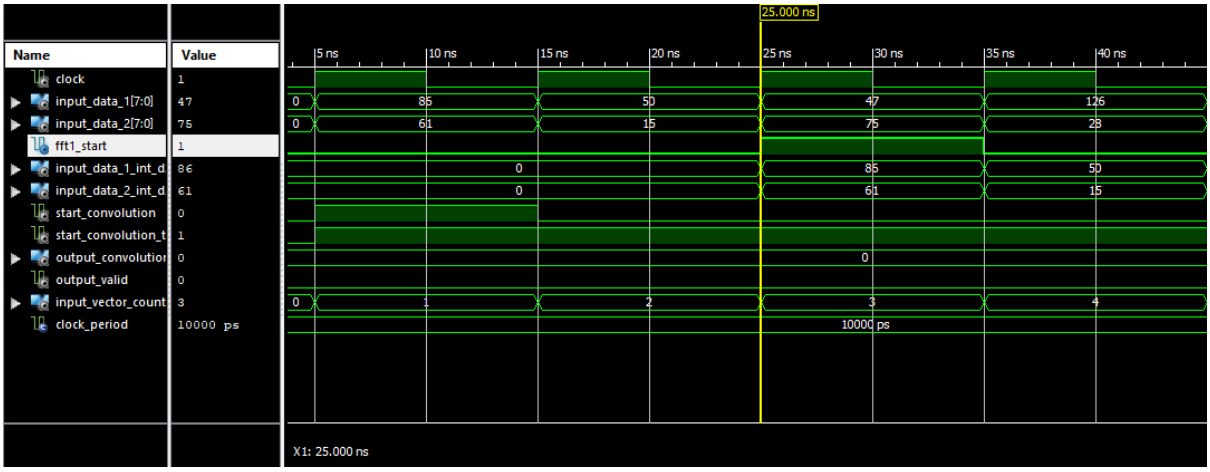


عنوان ماژول اصلی پروژه ما Convolution_Top است، این ماژول دارای ورودی‌های Clock (تک بیتی)، Input_Data_1 و Input_Data_2 (۸ بیتی)، Start_Convolution (تک بیتی) و خروجی‌های Output_Convolution (۳۳ بیتی) و Output_Valid (تک بیتی) است. قلب تپنده پروژه ما IP FFT است پس لازم است با Port های آن آشنا شویم. در جدول زیر با پایه‌های FFT به همراه توضیحاتشان آمده است.

نام Port	مختصر شده	توضیحات
clk	Clock	سیگنال کلاک
start	Start	شروع اجرای عملیات FFT
xn_re	$x_{(n)}$ Real	بخش حقیقی داده ورودی
xn_im	$x_{(n)}$ Imaginary	بخش موهومی داده ورودی
fwd_inv	Forward/Inverse	تعیین نوع تبدیل (FFT=1 و IFFT=0)
fwd_inv_we	Forward/Inverse Write Enable	فعال‌سازی ثبت مقدار fwd_inv
rfd	Ready For Data	وقتی ۱ شود یعنی ماژول آماده دریافت داده جدید است.
xn_index	$x_{(n)}$ Index	شماره (ایندکس) داده‌ای که باید ارسال شود.
busy	Busy	نشان می‌دهد FFT در حال پردازش است.
edone	Early Done	پایان زودتر از موعد برای هم‌زمان‌سازی یا کنترل.
done	Done	عملیات FFT به‌طور کامل تمام شده.
dv	Data Valid	داده خروجی معتبر و آماده خواندن است.
xk_index	$x_{(k)}$ Index	شماره (ایندکس) نمونه خروجی FFT
xk_re	$x_{(k)}$ Real	بخش حقیقی خروجی FFT
xk_im	$x_{(k)}$ Imaginary	بخش موهومی خروجی FFT

ورودی‌ها از طریق سیگنال‌های میانی تعریف شده به پورت‌های xn_re و xn_im متصل می‌شوند، پس از به پایان رسیدن کار FFTها می‌بایست طبق شماتیکی که در ابتدای این مقاله آورده شد، خروجی FFTها در یکدیگر ضرب مختلط شوند، قبل از اینکه خروجی‌ها در هم ضرب شوند می‌بایست خروجی یکی از FFTها مزدوج شود یعنی بخش موهومی آن در یک منفی ضرب شود بعد عمل ضرب انجام شود. برای این منظور یک ماژول به نام Multiply_Conjugate طراحی گردیده است تا هم زمان هم عمل مزدوج کردن و هم ضرب مختلط را انجام دهد. سپس بعد از محاسبه حاصل ضرب که سیگنالی ۲۵ بیتی است (۲۵ بیت برای بخش حقیقی و ۲۵ بیت برای بخش موهومی) باید این مقدار را به FFT نهایی بدهیم و آن را در Mode معکوس تنظیم کنیم. سایر توضیحات راجب iFFT همانند توضیحات مراحل قبل است با این تفاوت که عرض بیت ورودی iFFT برابر ۲۵ بیت است و ضریب Phase Factor یا همان Twiddle Factor را بر روی ۱۶ بیت تنظیم می‌کنیم و همین طور به سیگنال fwd_inv مقدار ’0‘ می‌دهیم تا به صورت معکوس کار کند.

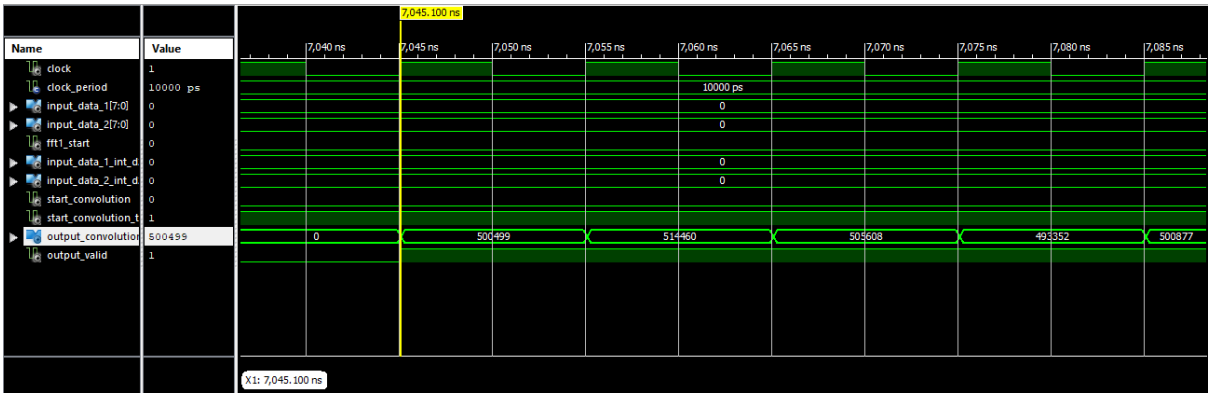
نکته : در کد VHDL ، ما ورودی‌ها را به اندازه ۲ کلاک تاخیر داده‌ایم یا به عبارتی یک Delay Line دو کلاک ایجاد کرده‌ایم دلیل این امر چیست ؟ به شکل زیر نگاه کنید :



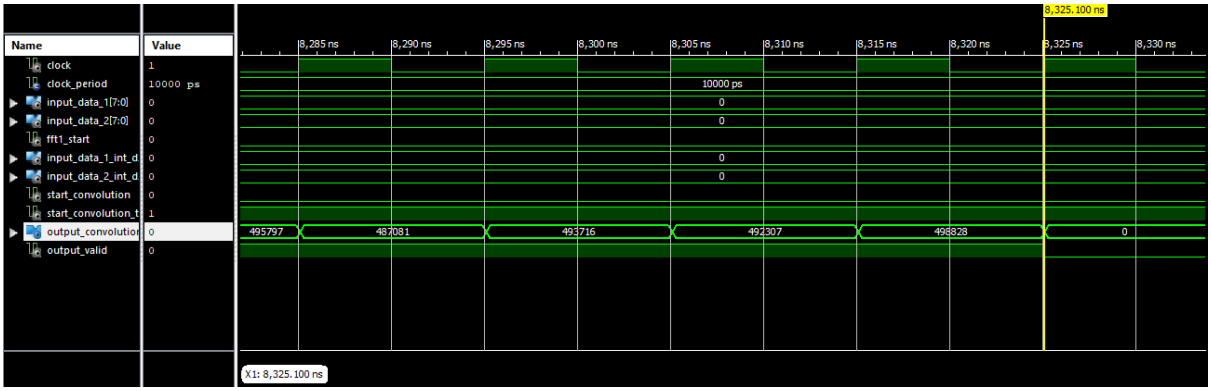
با توجه به ساختار کد و نوع کدنویسی سیگنال Start ، FFT ما که آن را با FFT1_Start نشان می‌دهیم دو کلاک تاخیر دارد. به منظور جبران این دو کلاک تاخیر به ناچار مجبوریم سیگنال ورودی را نیز با دو کلاک تاخیر به FFT اعمال کنیم تا تمام نمونه‌ها از ابتدا به طور کامل منتقل شوند.

با توجه به اینکه پریود زمانی کلاک مدار ما 10ns است و ۱۲۸ نمونه را باید به FFT اعمال کنیم و هر نمونه در یک کلاک اعمال می‌شود بدون در نظر گرفتن Propagation Delay ها 1280ns طول می‌کشد تا تمام ۱۲۸ نمونه به FFT اعمال شود. شروع تولید خروجی iFFT از زمان 7045ns بوده و در 8325ns تبدیل ۱۲۸ نمونه ما پایان یافته و به عبارتی خروجی iFFT ما کامل می‌شود، البته پر واضح است تست سخت افزاری این پروژه بر روی FPGA زمان‌های متفاوتی (بیشتر) از مقادیر ذکر شده در این مقاله نتیجه می‌دهد.

شکل موج شروع پاسخ iFFT در نرم‌افزار ISim



شکل موج بخش پایانی پاسخ iFFT در نرم‌افزار ISim



نکات تکمیلی

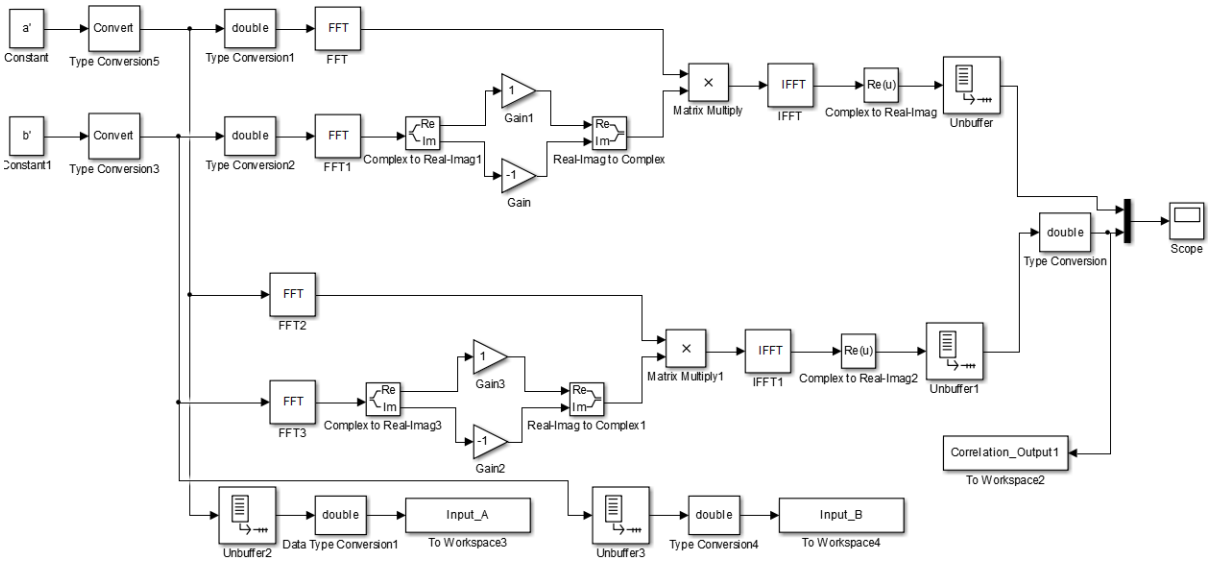
دقت داشته باشید برای عملکرد صحیح Test bench می‌بایست مسیر (آدرس) فایل‌های متنی txt را مطابق با مسیر موجود در سیستم عامل خود تغییر دهید در غیر اینصورت با پیغام خطا مواجه خواهید شد.

در فایل‌های این پروژه فولدری به نام Convolution_MATLAB وجود دارد که شامل فایل‌های تولید ورودی، مقایسه نتایج حاصل از ISE و MATLAB و فایل شبیه‌سازی در نرم‌افزار MATLAB آمده است. فایل شبیه‌سازی (Simulink) با نام Convolution دقیقا همین پروژه را در Simulink شبیه‌سازی می‌کند. هدف از شبیه‌سازی این طرح در نرم‌افزار MATLAB کوانتیزاسیون ورودی‌ها و عملگرها و بدست آوردن مدل Fixed point است.

براساس دیتاشیت IP FFT، این پروژه بر روی تراشه‌های سری Virtex-7 ,Kintex-7, Virtex-6, Virtex-5, Virtex-4, Spartan-6, Spartan-3/XA, Spartan-3E/XA, Spartan-3A/3AN/3A DSP/XA قابل پیاده‌سازی است. برای سری‌های جدیدتر تراشه‌های Xilinx می‌بایست از FFT با نسخه‌های جدیدتر استفاده کرد که پروتکل ارتباطی آنها از نوع AXI4-Stream است.

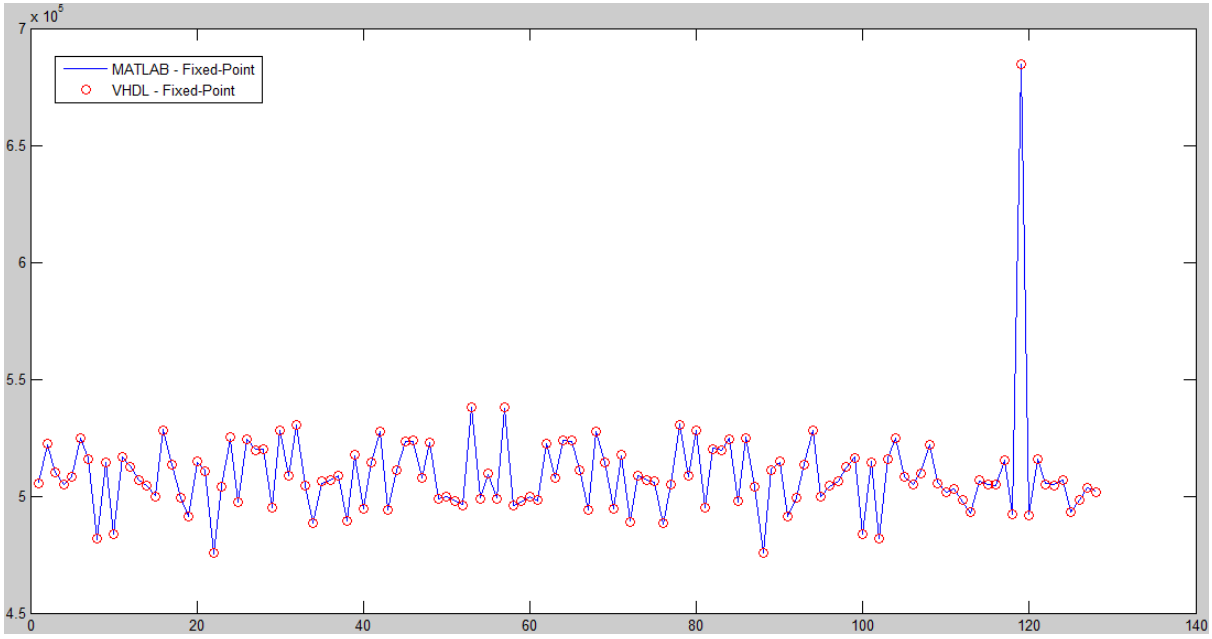
نتایج و تحلیل عملکرد

در شکل زیر نمایی از محیط Simulink در نرم‌افزار MATLAB را مشاهده می‌کنید.



بخش بالایی تصویر فوق مدل Floating Point طرح کانولوشن را پیاده‌سازی کرده و بخش پایینی قسمت Fixed Point کانولوشن را مدلسازی کرده است.

خروجی حاصل از پیاده‌سازی کانولوشن به زبان VHDL در نرم‌افزار ISE را با خروجی حاصل از نرم‌افزار MATLAB مقایسه کرده‌ایم، نتیجه این مقایسه در شکل زیر آمده است:



خطوط آبی رنگ نتایج حاصل از شبیه‌سازی در نرم‌افزار MATLAB بوده و دایره‌های توخالی قرمز رنگ نتایج حاصل از شبیه‌سازی در نرم‌افزار ISE که به زبان VHDL پیاده‌سازی شده است، همان طور که مشاهده می‌کنید. نتایج با دقت بالایی بر هم منطبق هستند.