|  |  |  |
| --- | --- | --- |
|  | به نام خدا |  |
| **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر**  **ESL**  **گزارش** **پروژه‌ دوم** | | |

|  |  |
| --- | --- |
| امیرحسام جعفری راد | نام و نام خانوادگی |
| 810100247 | شماره‌ دانشجویی |
|  | تاریخ ارسال گزارش |

­

فهرست

[بخش اول) استفاده از ابزار Fixed Point 1](#_Toc166532889)

[بخش دوم) استفاده از ابزار HDL coder 7](#_Toc166532890)

# بخش اول) استفاده از ابزار Fixed Point

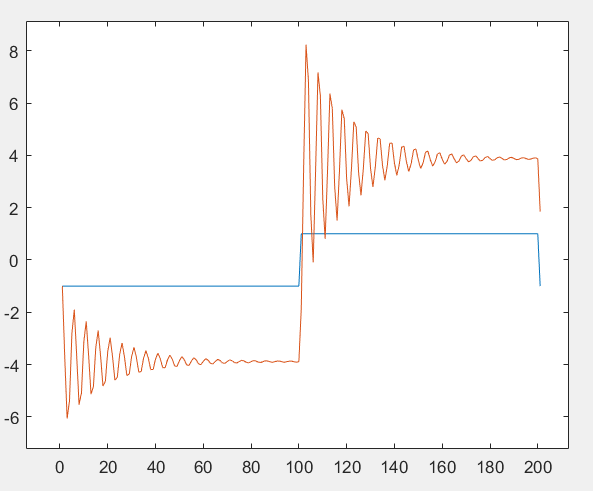
1. دو مورد از مزایا و معایب تبدیل floating point به fixed point را بیان کنید:

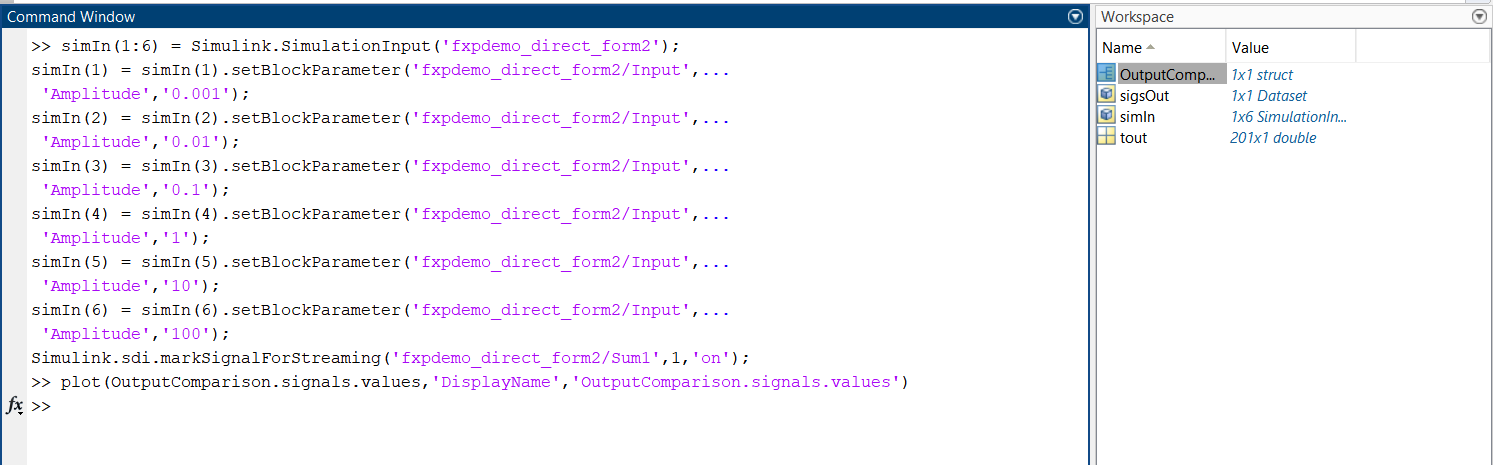
در حالت fixed point سیستم دقت و کنترل پذیری بیشتری نسبت به حالت ممیز شناور دارد. همچنین در پیاده سازی سخت افزاری این سیستم به المان های کمتر و ساده تری نیاز است که باعث کاهش هزینه ساخت و افزایش کارایی و سرعت نیز می‌شود.

در سیستم ممیز ثابت امکان رخ دادن underflow یا overflow خیلی بیشتر از سیستم ممیز شناور است چرا که مدیریت این دو موضوع باید هنگام طراحی به درستی انجام شود. همچنین در تعداد بیت های یکسان برای هردو سیستم، سیستم ممیز ثابت دامنه و رنج محدود تری نسبت به سیستم ممیز شناور دارد.

* 1. **بررسی فیلتر دیجیتال**

1. نتیجه شبیه‌سازی و نمودار output comparison:





نمودار خروجی برحسب ورودی پس از اعمال کد متلب مربوطه به شرح بالا می‌باشد.

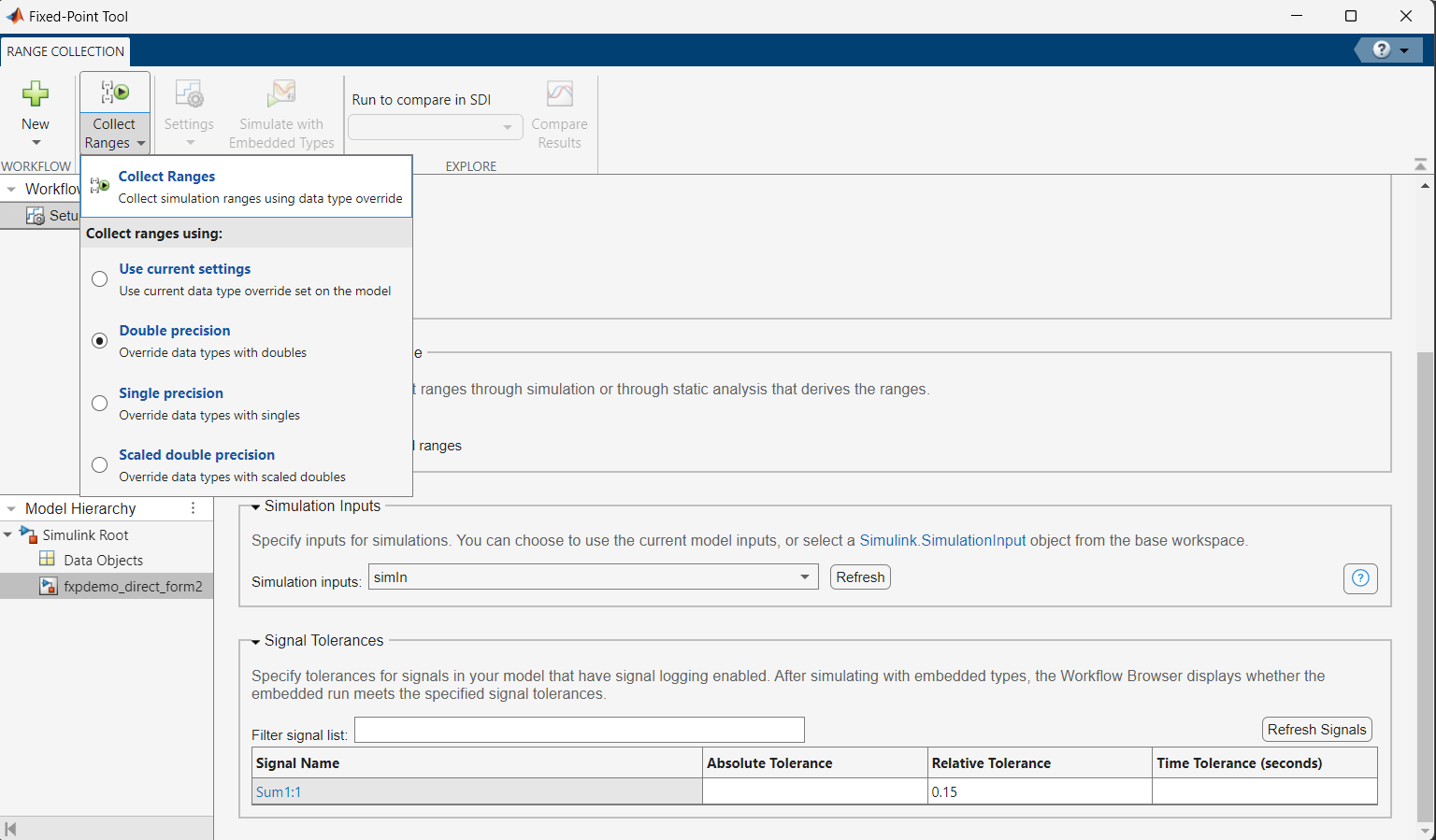
در این نمودار رابطه ورودی ها به فرم

و رابطه خروجی به فرم می‌باشد.

خروجی این معادله تقریبا برابر با بخش سبز رنگ در تصویر می‌باشد که با نمودار اصلی برابر و بر آن منطبق است.

* 1. **تغییر مبنای فیلتر دیجیتال به Fixed Point و مقایسه با مبنای Floating Point**

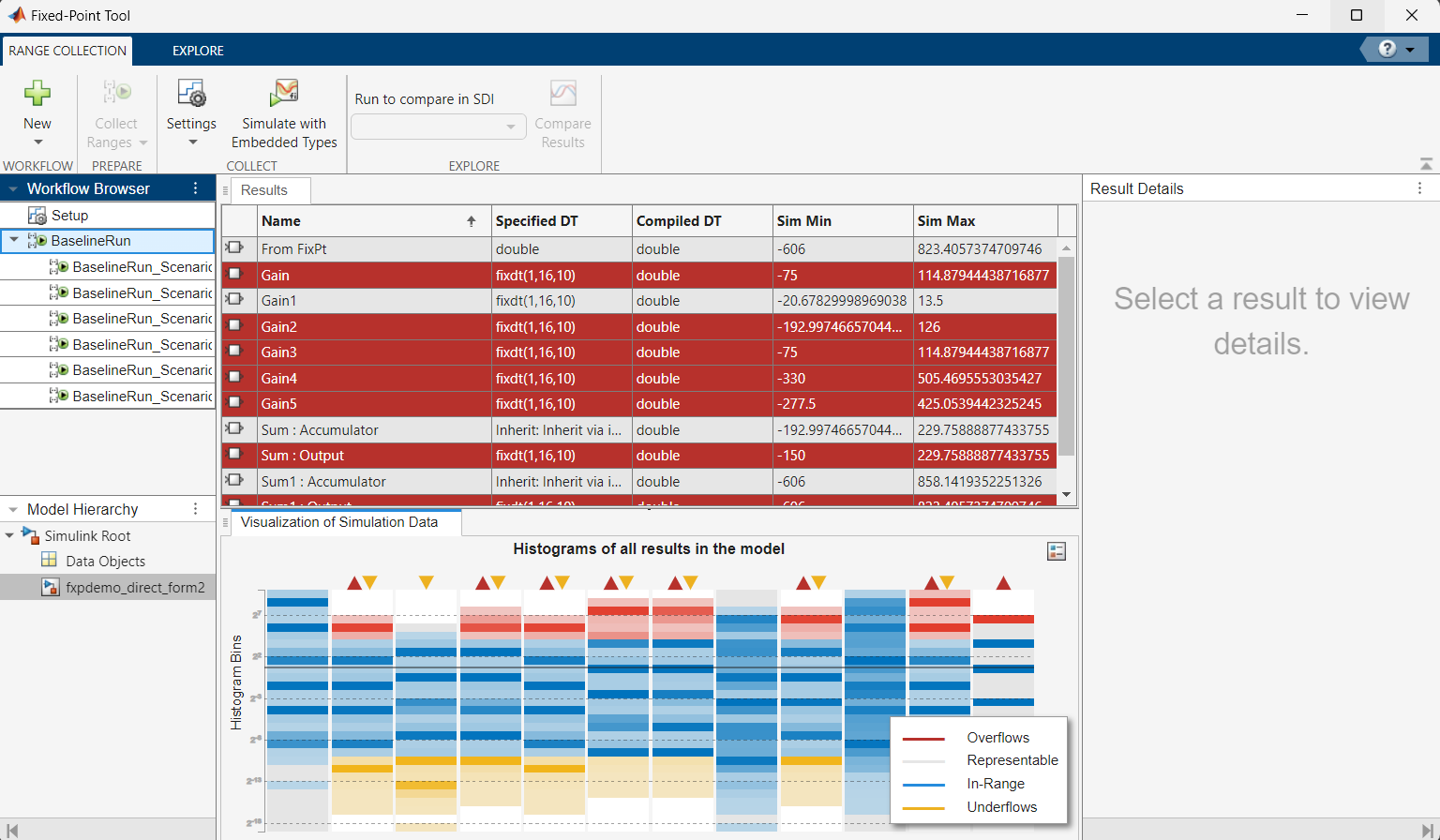
تنظیمات گفته شده مطابق تصویر اعمال شده است.

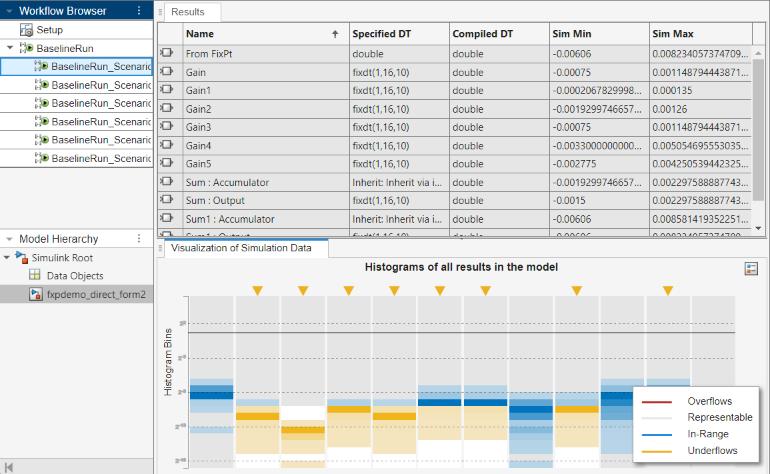
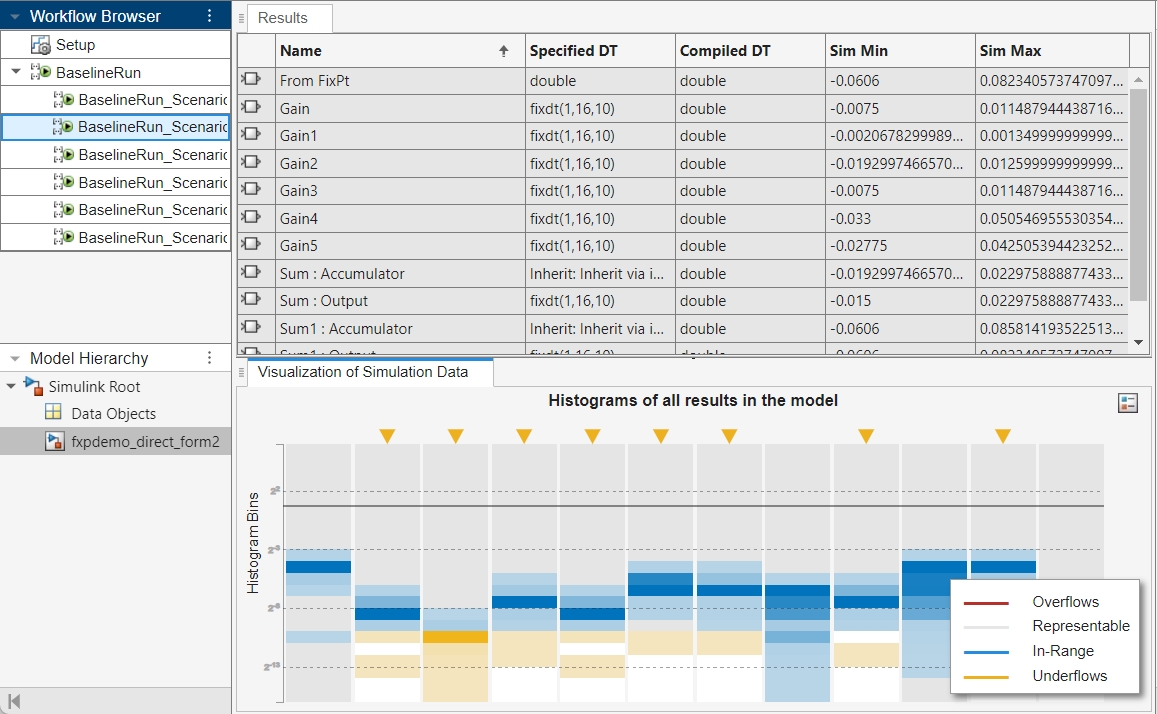


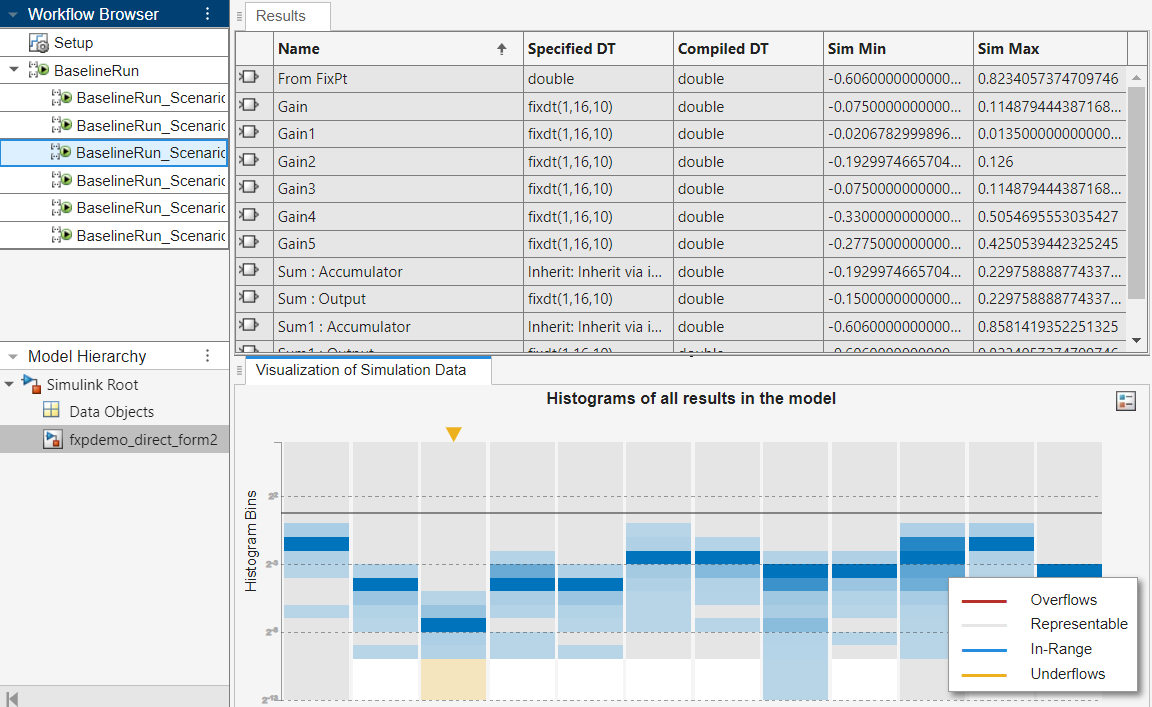
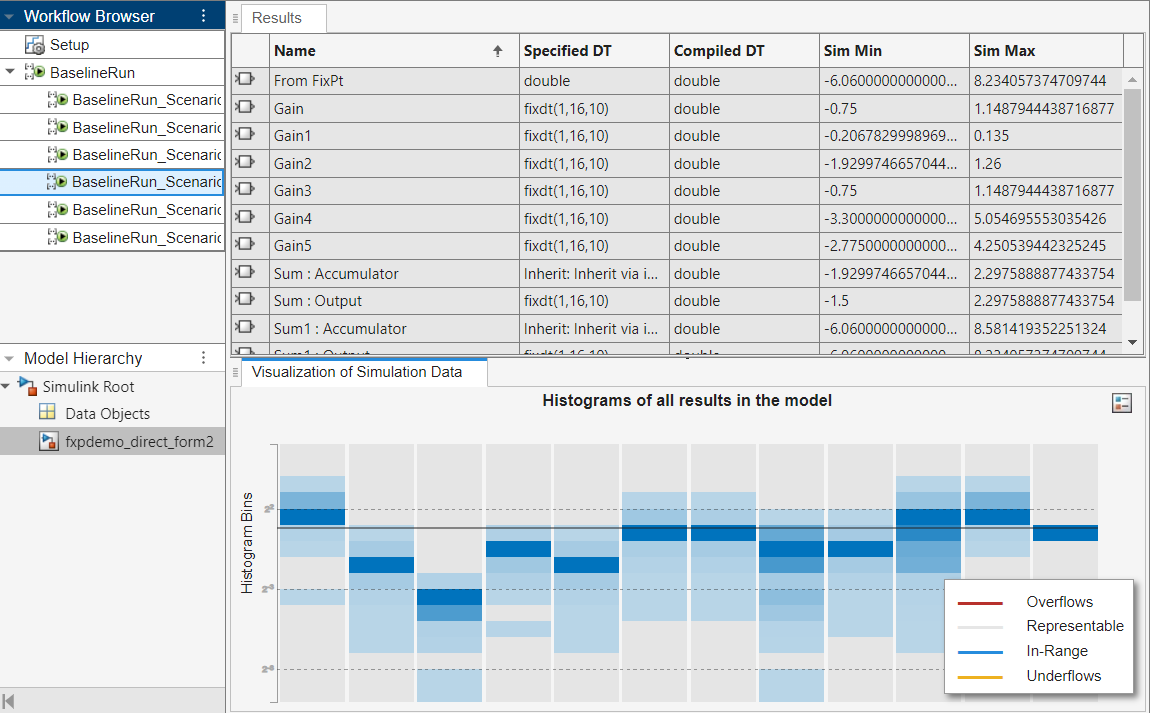
1. وضعیت underflow و overflow به ازای هر 6 ورودی و رابطه آن‌ها با سیگنال ورودی:

مشاهده می‌شود که در ابتدا میزان زیادی underflow رخ داده است که مطابق انتظار است. چرا که در ابتدا ورودی ها کوچک بوده و به ازای این ورودی ها سیستم دچار underflow شده است. انتظار میرود با افزایش دامنه ورودی میزان underflow کاهش پیدا کند که در نمودار نیز قابل مشاهده است و در مرحله ای که ورودی ها به میزان متوسطی رسیده اند دیگر اثری از underflow دیده نمیشود و مقدار آن برابر با صفر خواهد شد. همچنین در صورت افزایش بیش از حد ورودی انتظار میرود سیستم دچار overflow شود که در تصاویر نیز مشاهده می‌شود.

نمودار مربوطه:



نمودار های سناریو های 1 الی 6 به تفکیک نیز به شرح ذیل می‌باشد:



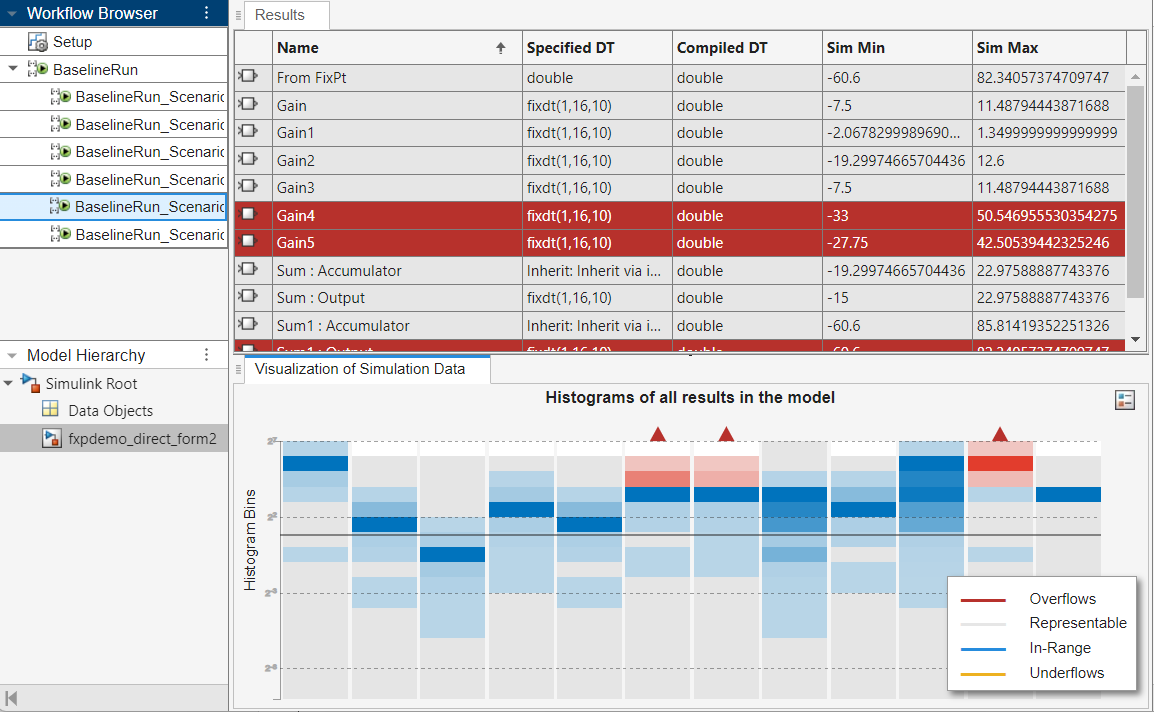
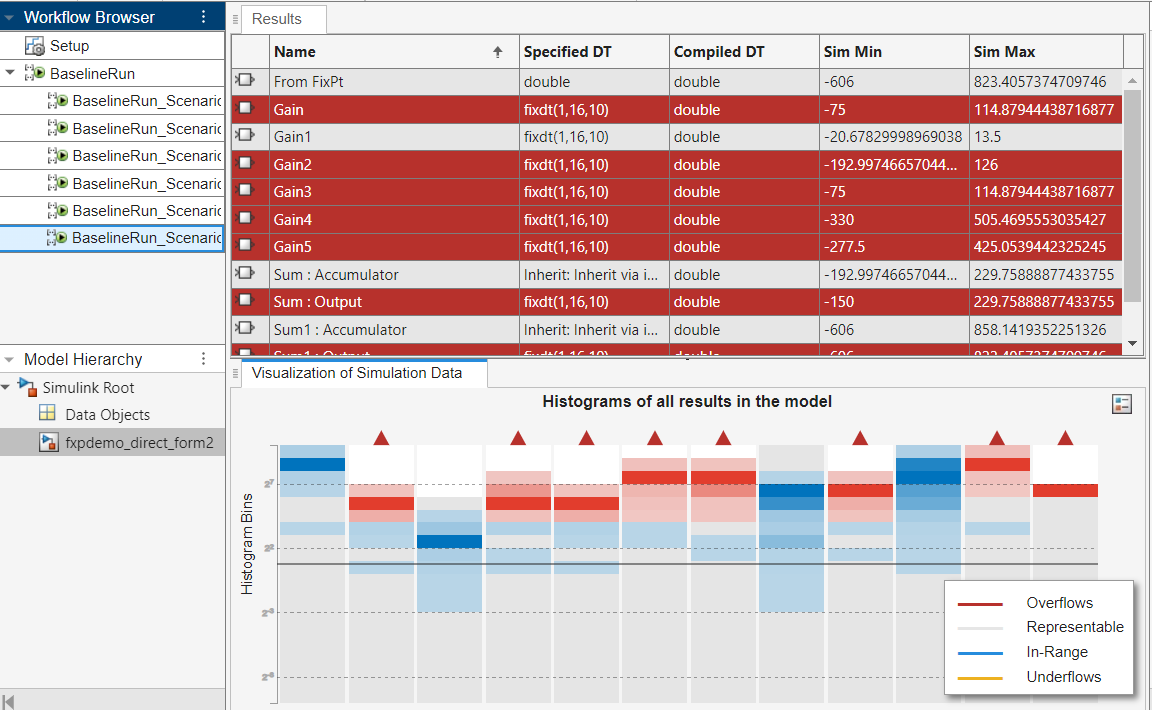
حالت دوم

حالت اول

حالت اول

حالت چهارم

حالت سوم

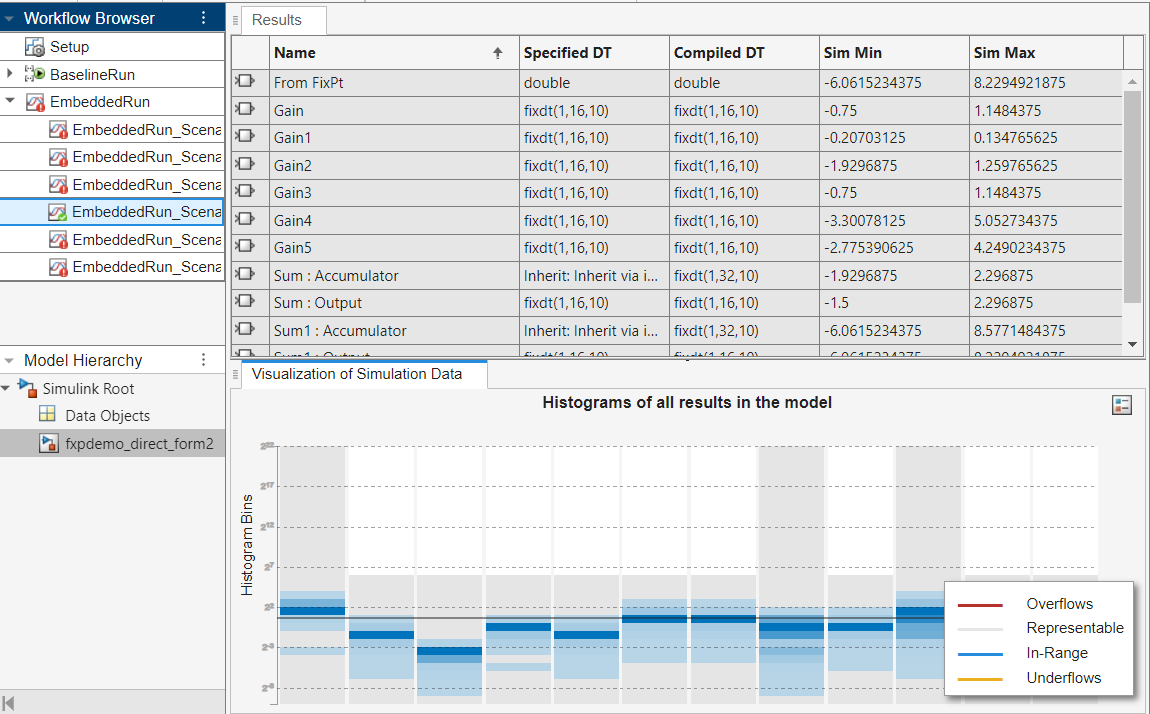
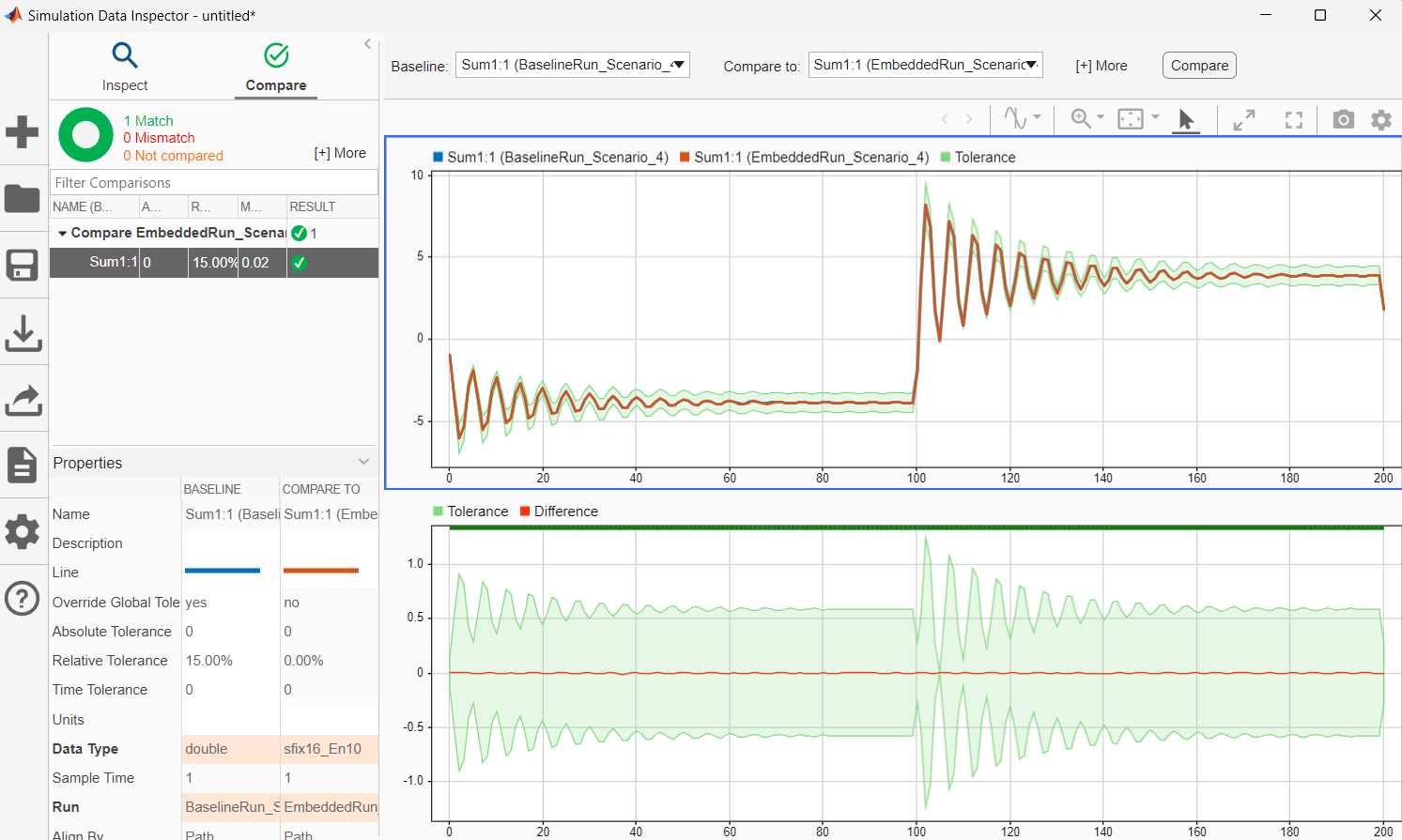


حالت ششم

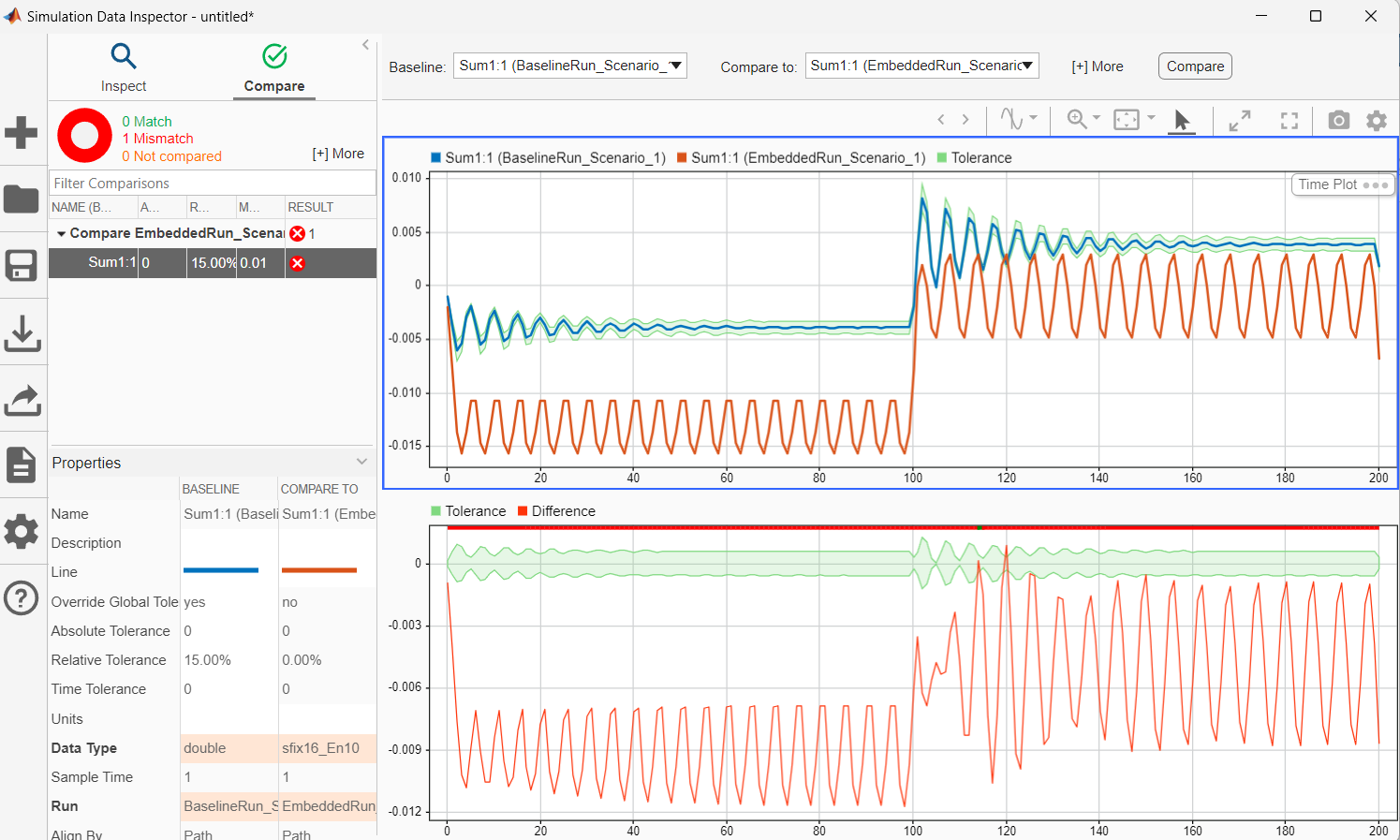
حالت پنجم

1. تصویر صفحه نمایش داده شده:

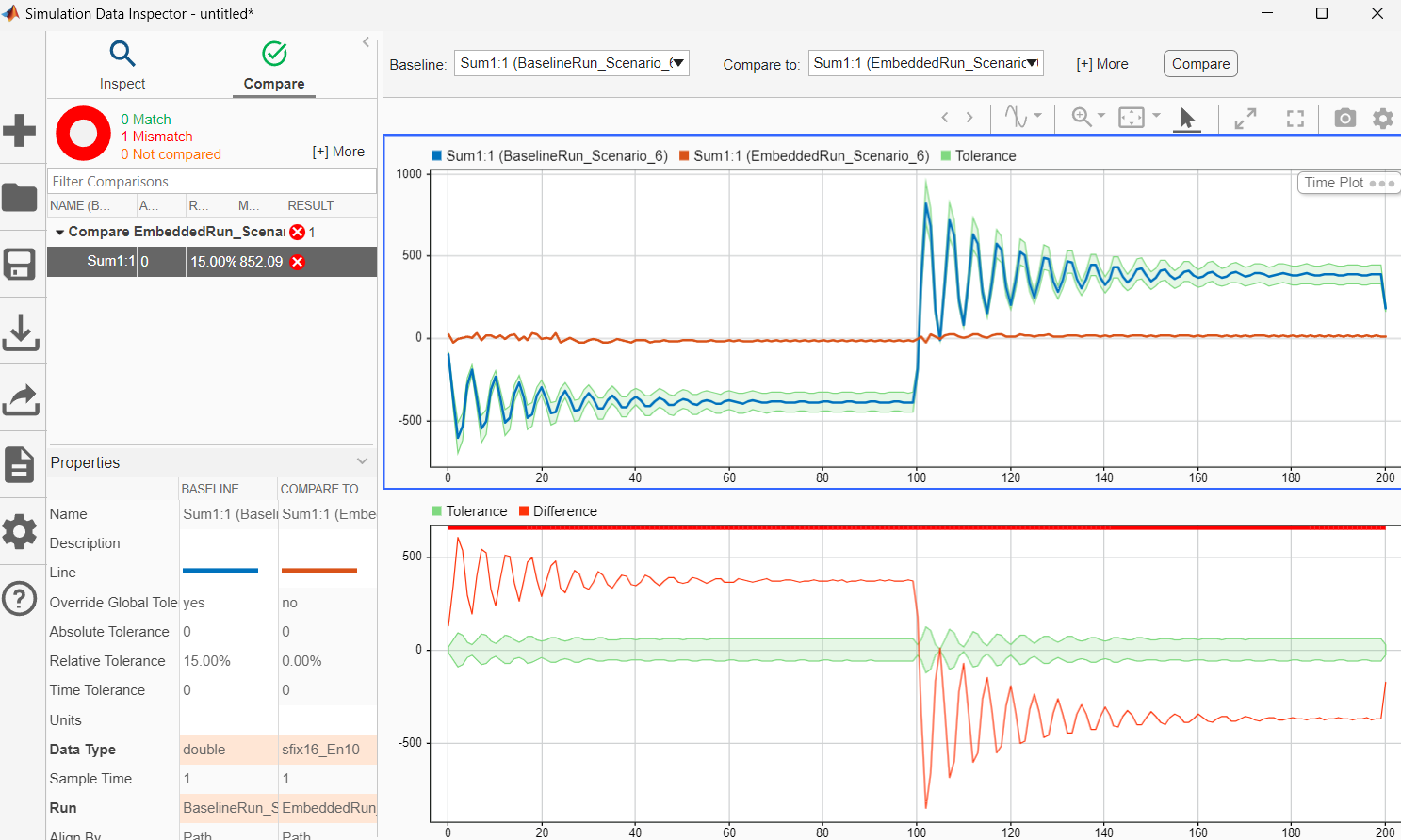
همانطور که مشاهده می‌شود تنها سناریو شماره 4 پاسخ مطلوب را تولید کرده که به شرح ذیل می‌باشد:



1. مرحله 4 برای سناریو اول و ششم:



سناریو اول: در این حالت خروجی دچار underflow شده است و به همین دلیل مطابق تصویر خروجی در قسمت مطلوب و قابل نمایش قرار نگرفته است.



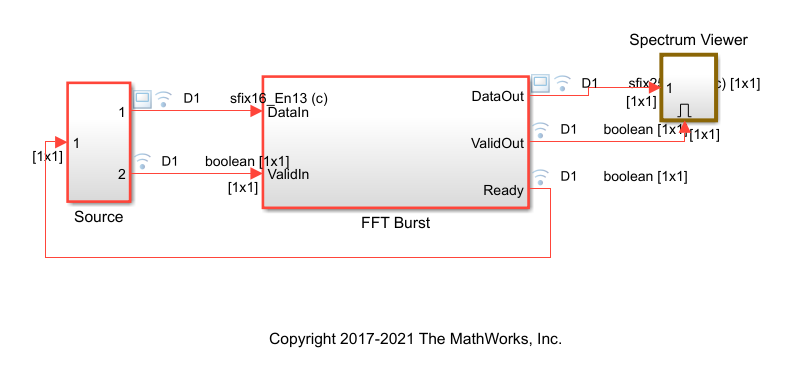
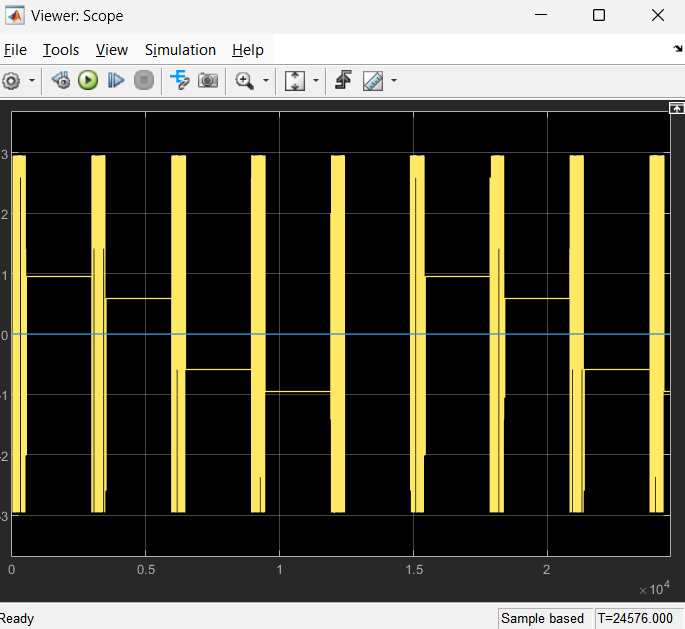
سناریو ششم: در این حالت سیستم دچار overflow شده است و به همین علت مطابق تصویر خروجی خروجی در بخش مطلوب و

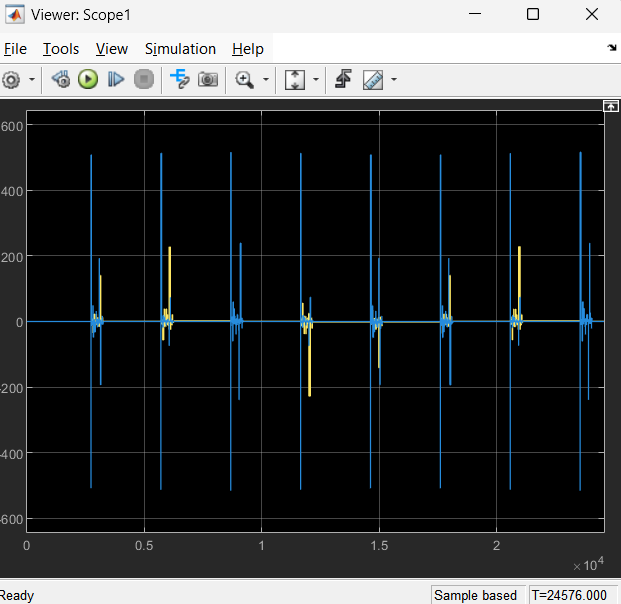
قابل نمایش قرار نگرفته است.

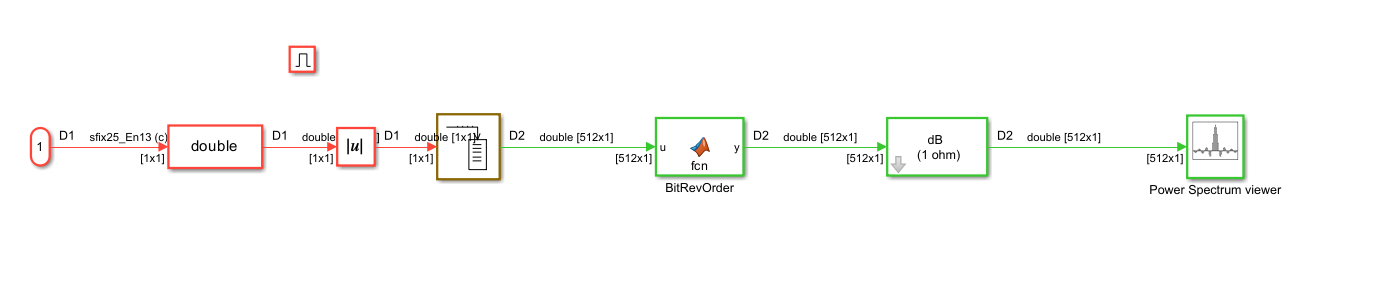
# بخش دوم) استفاده از ابزار HDL coder

* 1. **بررسی مدل FFT**

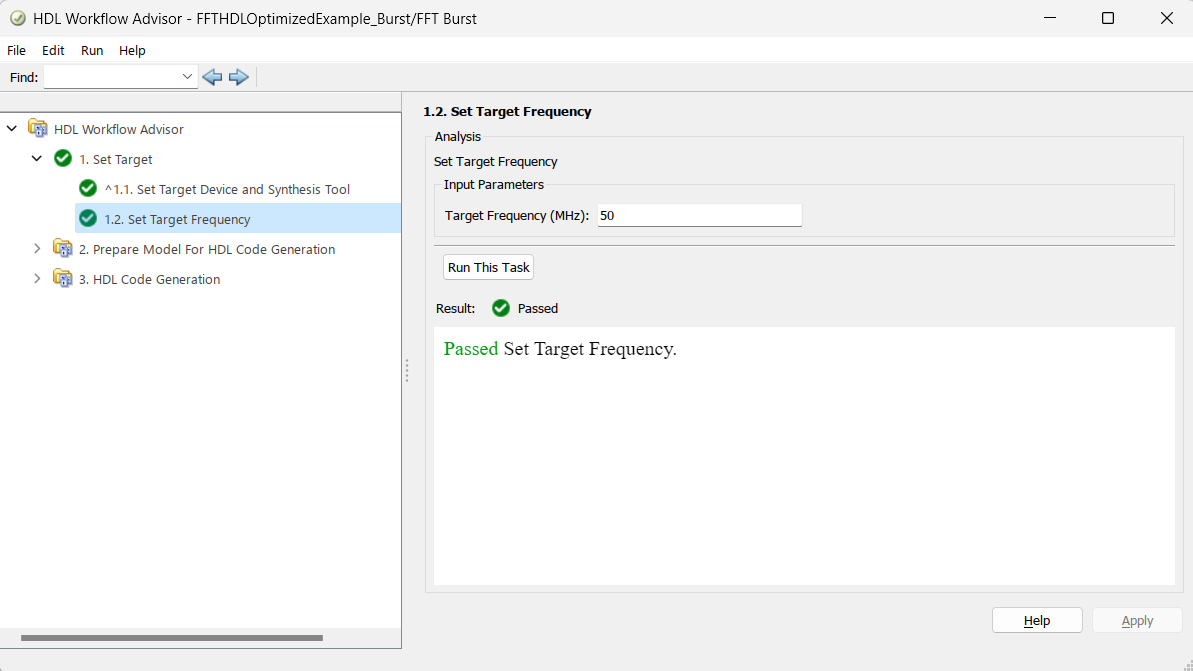
**1-**تصویر scope خروجی و ورودی و همچنین spectrum viewer

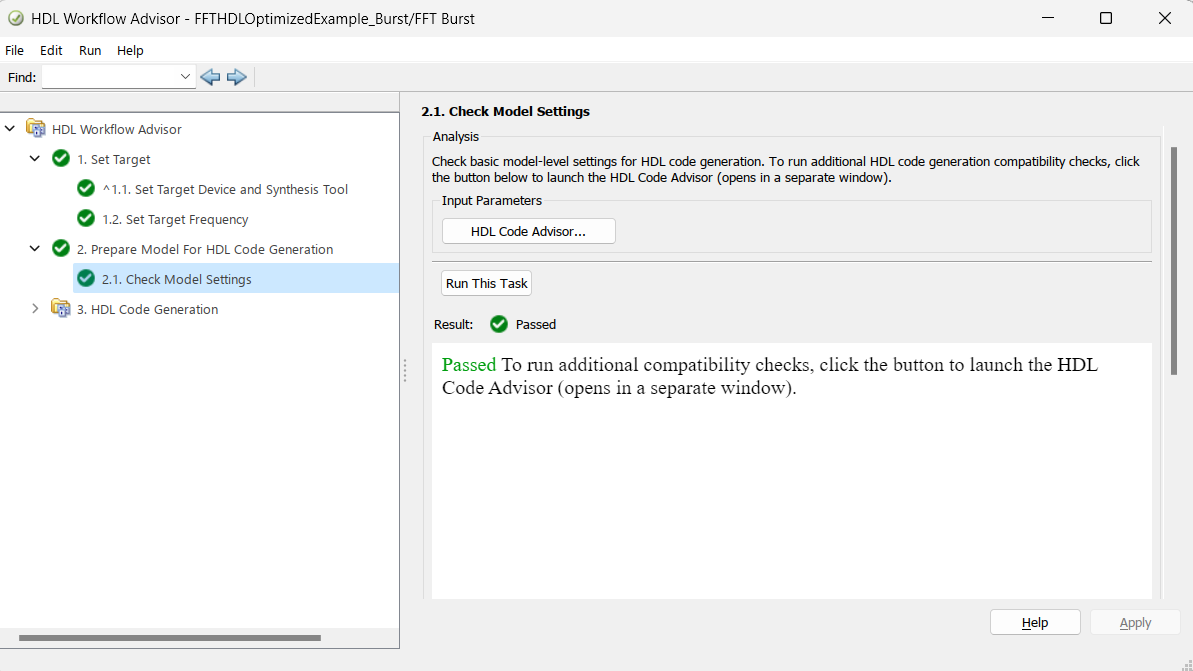
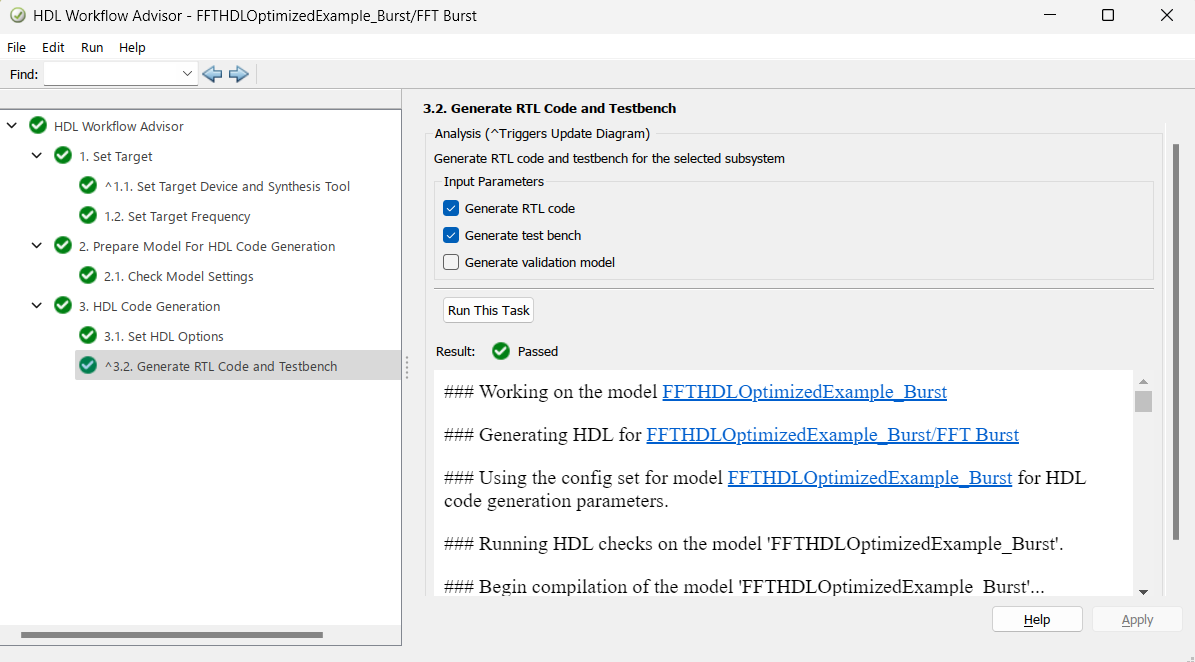






**2-2) سنتز مدل و تولید کد Verilog مدل FFT**

1. نتیجه Run و موفقیت آمیز بودن مرحله:

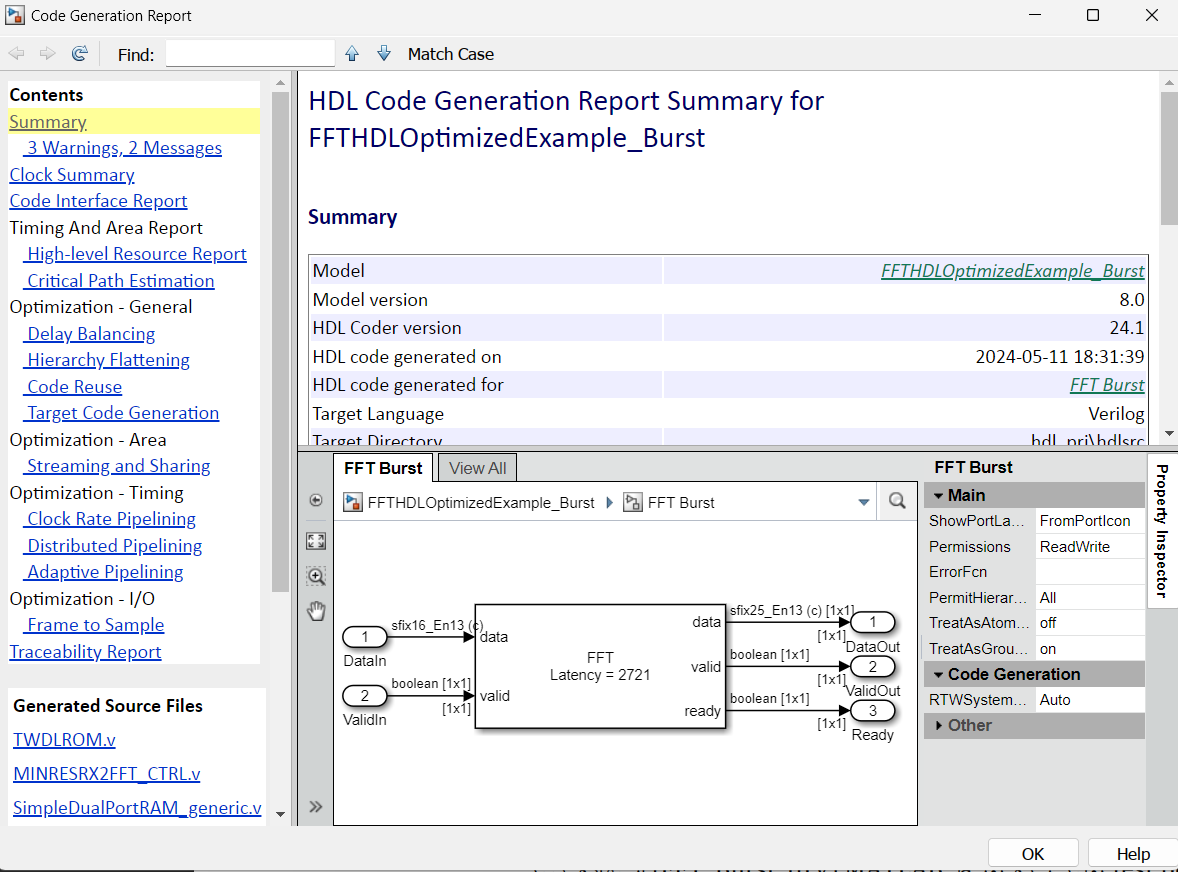


سوم:

دوم:

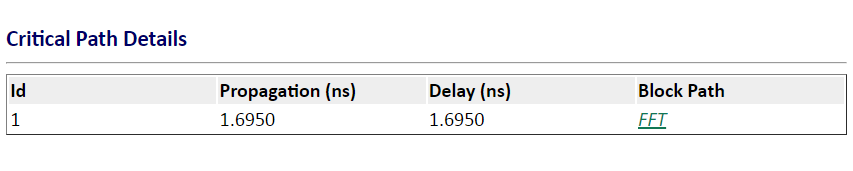
اول:

1. تصویر Latency ماژول:



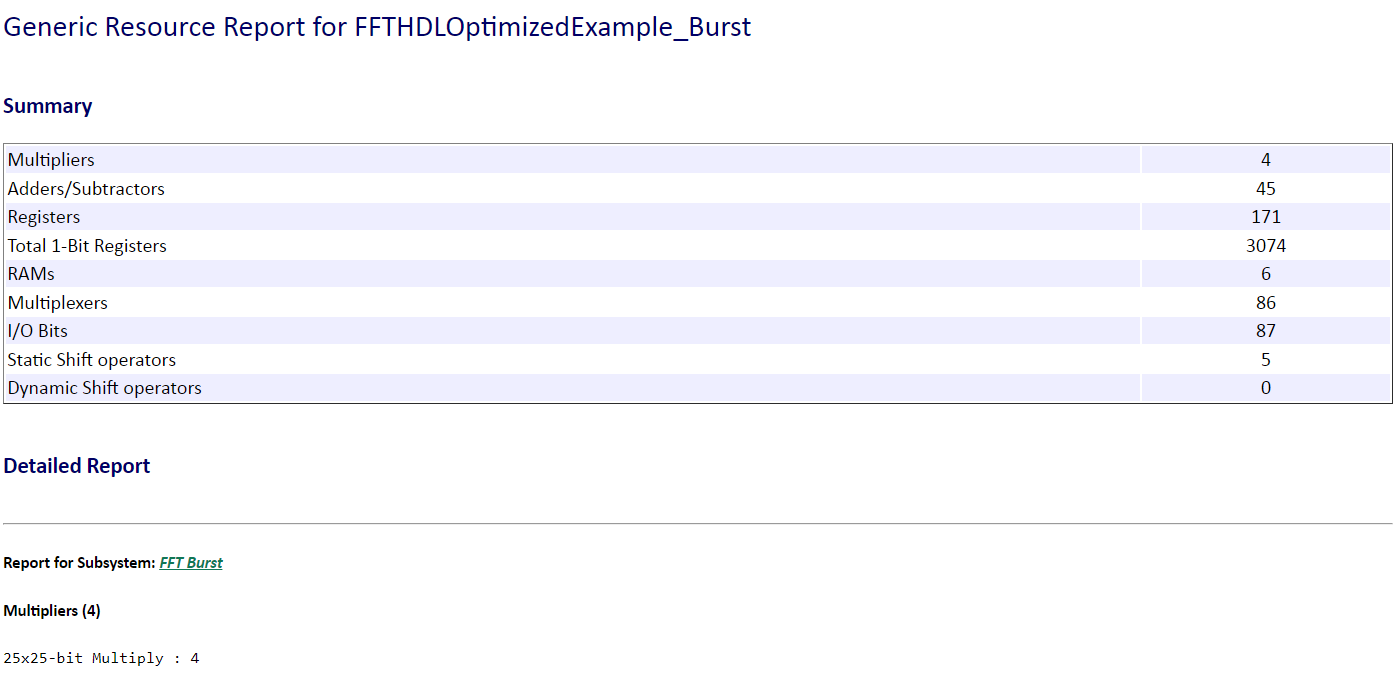
Latency به معنای مدت زمانی که این ماژول شروع به کار میکند تا لحظه ای که خروجی میدهد و کارش تمام می‌شود می‌باشد.

1. تصویر critical path estimation و مفهوم آن:

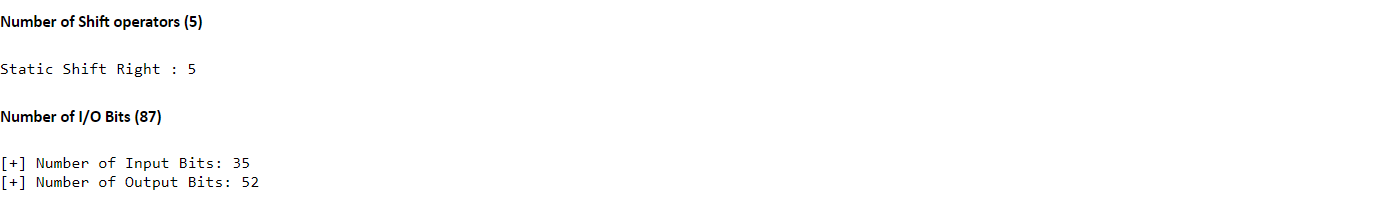


critical path estimation به معنای طولانی ترین و بیشتری تاخیری که ماژول در مدار خود دارد تا ورودی به خروجی برسد می‌باشد.

1. تصویر تمام گزارش‎‌های مربوط به high-level resource report

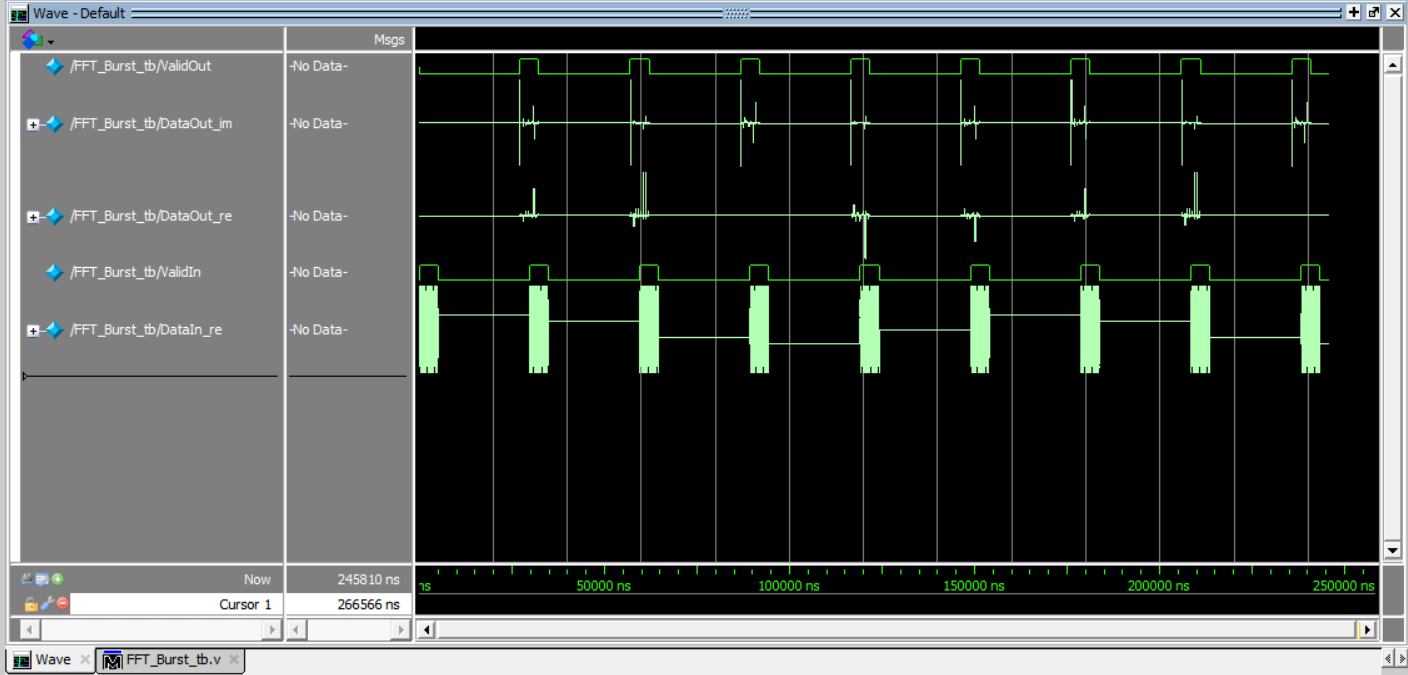






**3-2) شبیه‌سازی کدهای تولید شده توسط HDL coder**

1. تصویر سیگنال‌های validOut, dataOut\_im, dataOut\_re, validIn, dataIn\_re در شبیه‌سازی:

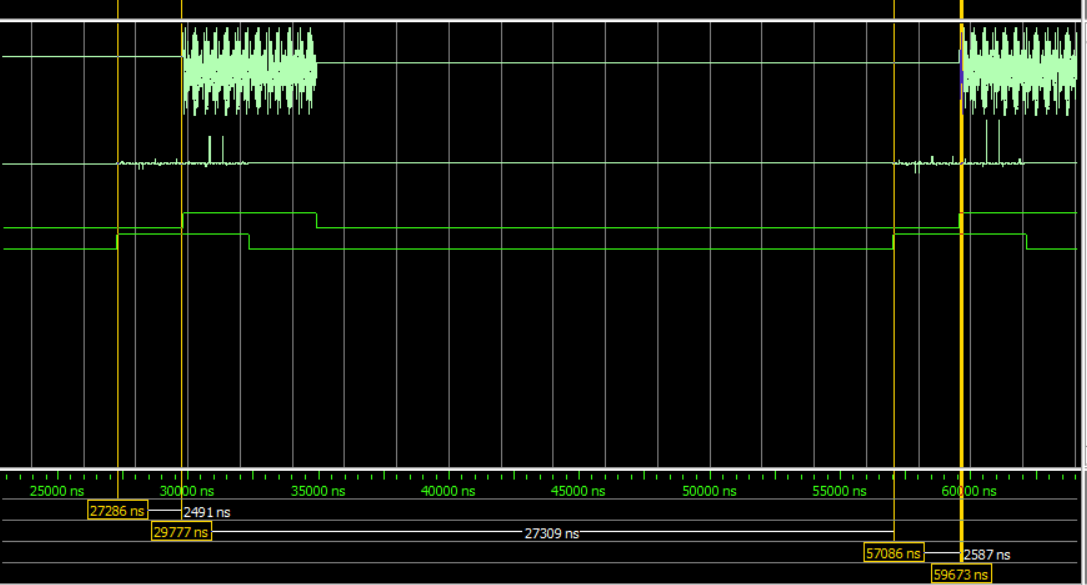
****

سیگنال DataIn\_re بخش حقیقی سیگنال ورودی، سیگنال DataOut\_re بخش حقیقی سیگنال خروجی و سیگنال DataOut\_im بخش موهومی سیگنال خروجی را نمایش می‌دهد.

1. مقایسه نتیجه شبیه‌سازی modelsim با سیگنال‌های خروجی Simulink:

همانطور که در تصویر بالا نشان داده شده است دقیقا همان سیگنال ورودی که در سیمولینک بود به رنگ آبی و همان سیگنال خروجی با رنگ بنفش مشخص شده اند. در واقع نتیجه سیمولینک و شبیه سازی مطابق انتظار یکسان است.

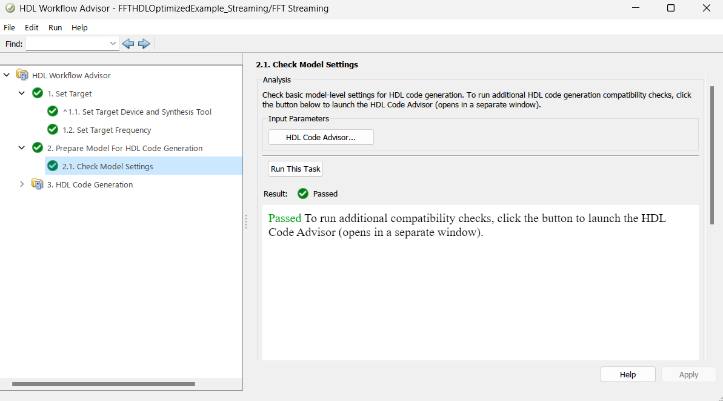
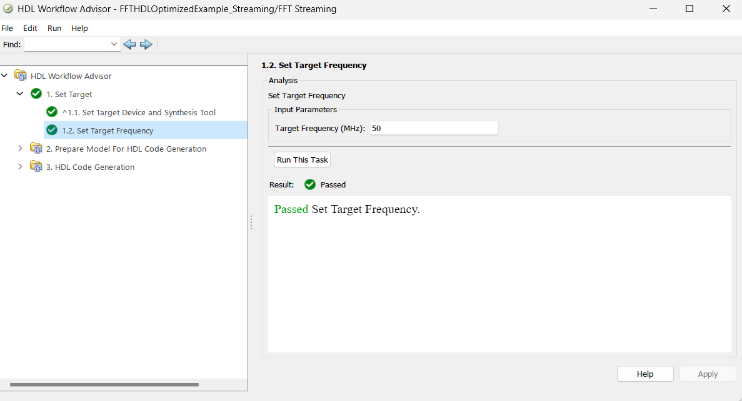
1. استخراج مقدار Latency:



همانطور که در تصویر مشخص است میزان تاخیر یعنی فاصله زمانی میان فعال شدن سیگنال های validIn و validOut که در دوحالت این مقدار استخراج شده است. علت تفاوت جزئی موجود در این حالت نیز مربوط به تست بنچ مورد استفاده در شبیه سازی است.

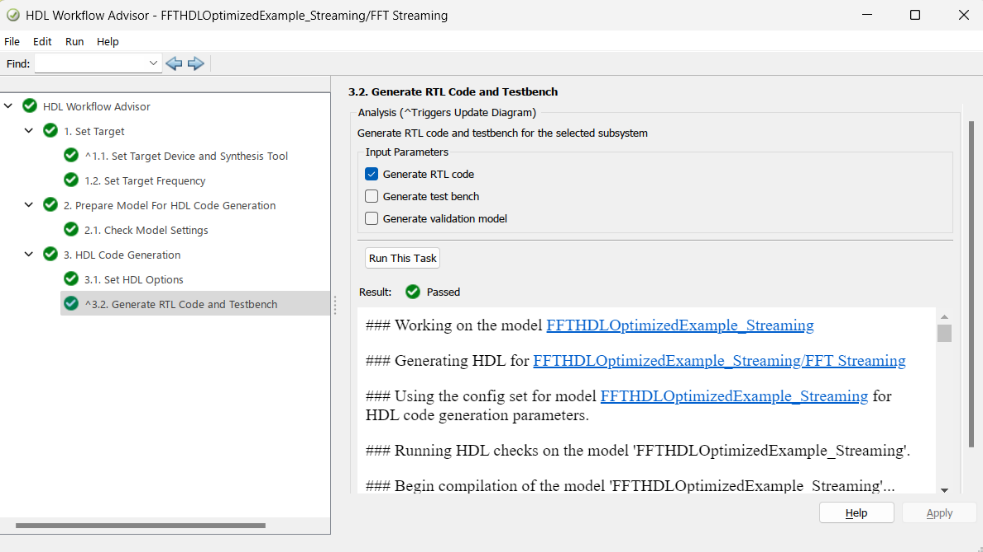
1. نحوه کار واحد Butterfly Unit:

این واحد یکی از موثرترین واحد ها در پیاده سازی ماژول FFT میباشد و به این صورت کار میکند که ابتدا دو ورودی با تایپ اعداد گنگ دریافت کرده و بر اساس اپلیکیشن و نیاز عملیاتی را بر روی آن ها انجام می‌دهد. درواقع تفاوت اصلی این واحد با بقیه در نوع ورودی های قابل قبول آن است که اعداد گنگ هستند.

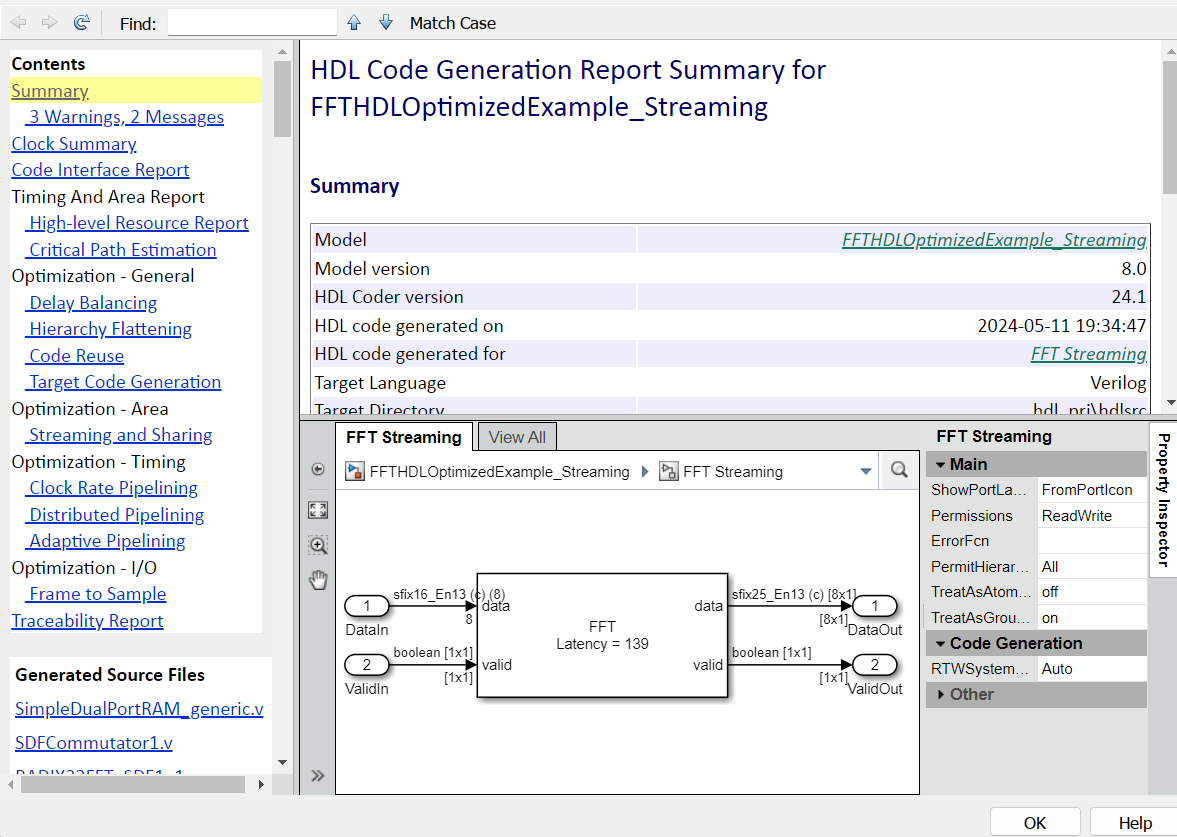
1. مراحل 2 تا 5 برای فایل FFTHDLOptimizedExample\_Streaming.slx:

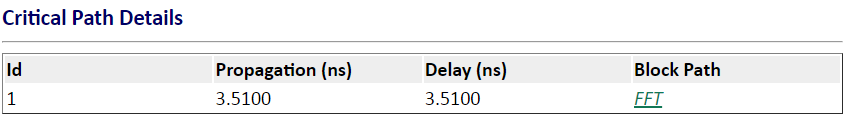
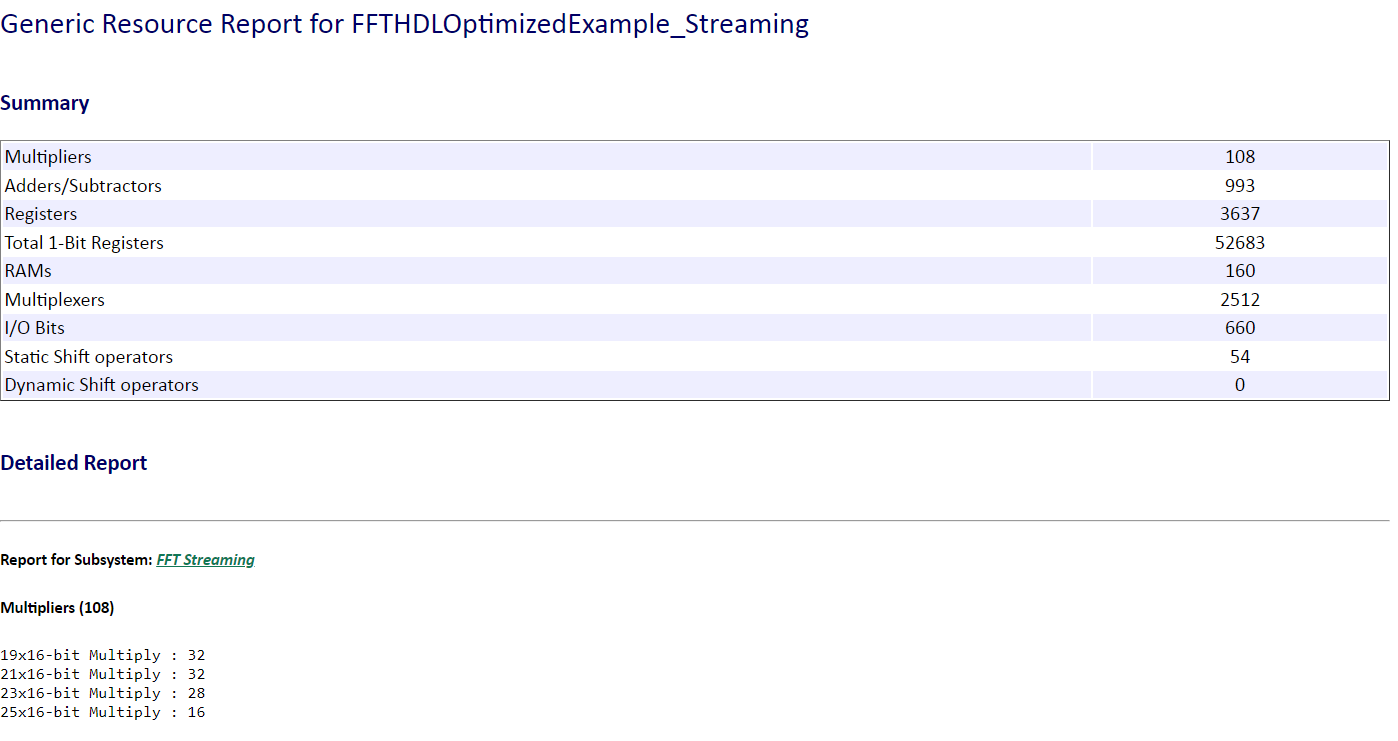
مرحله دوم

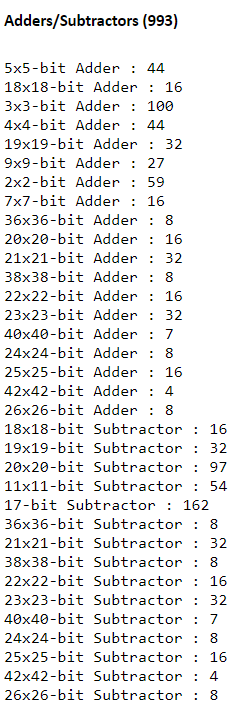
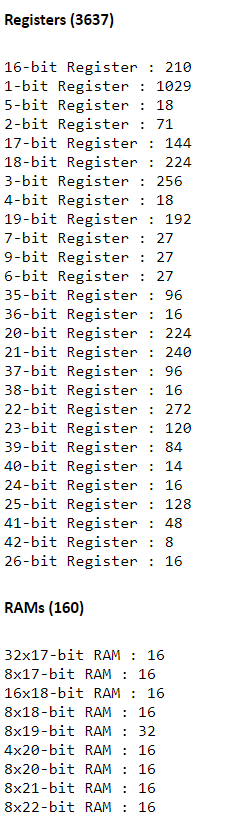
مرحله اول

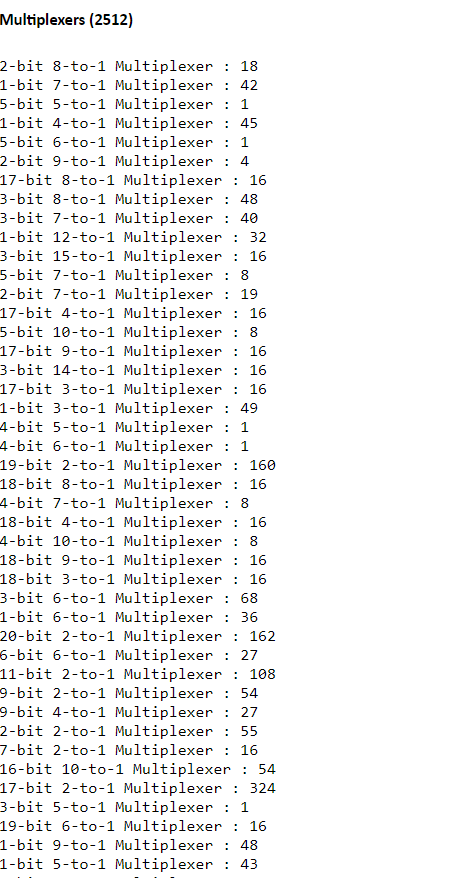
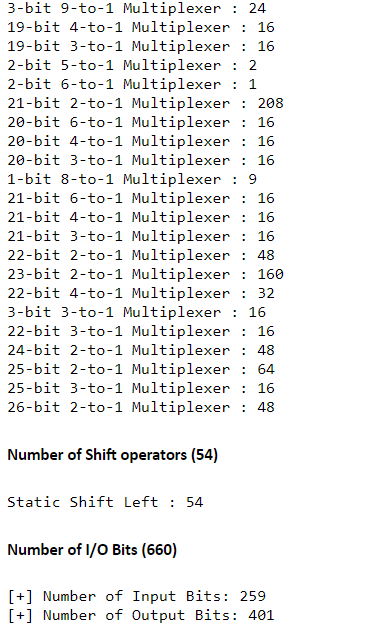


مرحله سوم









1. مقایسه و تحلیل حالت Burst و Streaming:

Latency: مطابق نتایج بدست آمده حالت Streaming دارای Latency بسیار کمتری نسبت به حالت Burst دارد و سریع تر انجام می‌شود و لذا نیازمند منابع بیشتری می‌باشد. درنتیجه برای زمان هایی که تاخیر latency برای ما اهمیت دارد و محدودیتی از نظر هزینه و منابع نداریم گزینه مناسب تری است.

Critical Path: مطابق نتایج بدست آمده حالت Streaming دارای مسیر بحرانی و تاخیر بیشتری نسبت به حالت Burst دارد.

1. تفاوت میان ارتباط Burst و Streaming:

در حالت Streaming سیستم در هر لحظه متناسب با داده های ورودی عملیات مرتبط را انجام میدهد و به محض تغییری در ورودی، در همان حین عملیات جدید بر اساس ورودی جدید انجام میشود درحالیکه در حالت Burst ابتدا داده ها بصورت پکیج های کاملی بسته بندی شده و پس از آماده شدن تمامی ورودی ها سیستم بصورت همزمان شروع به کار می‌کند.