a

پیاده سازی فیلتر FIR مرتبه ۲۰ میانگذر با تقارن و پایپلاین

گزارش پروژه درس مدار منطقی پیشرفته

استاد میرزا کوچکی

انجام دهندگان : (گروه ۳)

محمد حسین حسنی سرزه

۹۹۴۱۱۳۰۸

امین فیضی شهری

۹۹۴۱۲۴۱۱

Contents

طراحی فیلتر و مشخصات آن 4

مشخصات فیلتر 6

Quantization و Scaling 6

مقایسه فیلتر ها در سیمولینک 8

بررسی مدل بدون PipeLine 8

مقایسه با متلب 11

بررسی مدل همراه با PipeLine 12

مقایسه با متلب 13

طراحی فیلتر و مشخصات آن

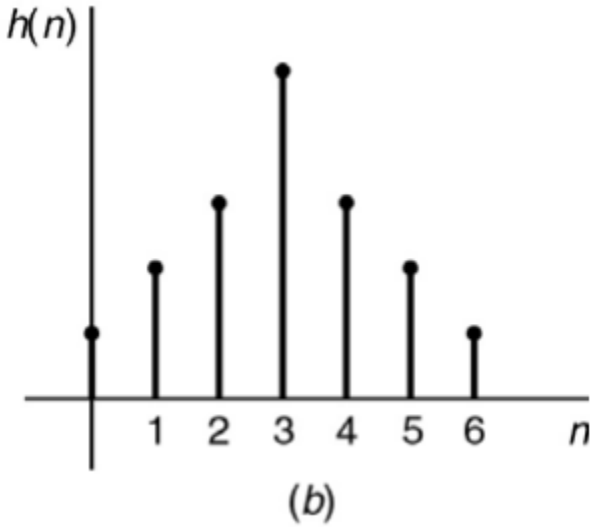
به طور کلی فیلتر های FIR به ۴ نوع تقسیم بندی میشوند

رابطه مرتبه(درجه) فیلتر با تعداد ضرایب :

$$order = number(Coeffs) - 1$$

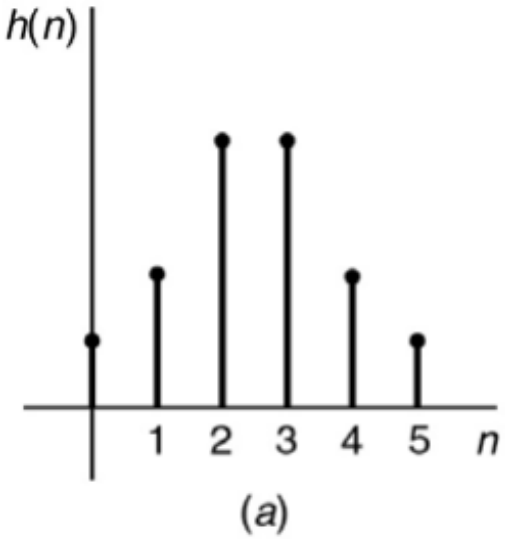
نوع ۱) فیلتر متقارن با تعداد ضرایب فرد یا مرتبه زوج

در این پروژه از این نوع بخصوص استفاده شده است.



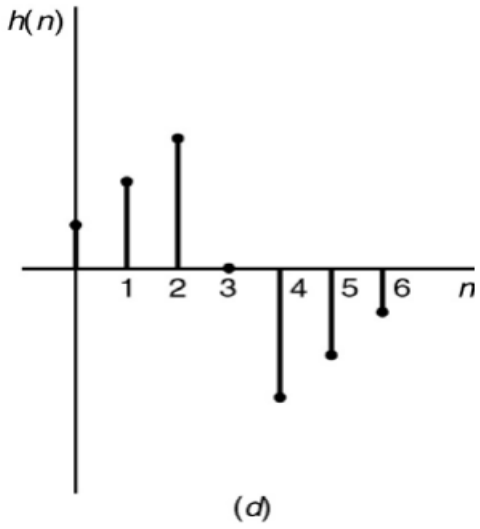
این نوع فیلتر ، برخلاف سایر محدودیتی در پیاده سازی ندارد .

نوع ۲) فیلتر متقارن با تعداد ضرایب زوج یا مرتبه فرد



بدلیل وجود صفر در $z = -1$ این فیلتر برای موارد بالاگذر مناسب نیست.

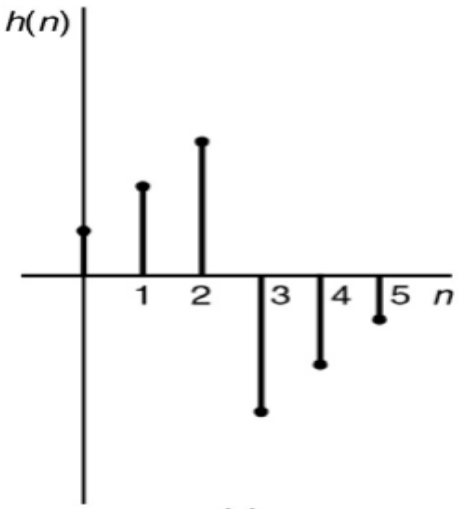
نوع ۳) فیلتر نا متقارن با تعداد ضرایب فرد یا مرتبه زوج



(d)

بدلیل وجود صفر در $z = 1,-1$ برای استفاده بالاگذر ، پایین گذر و یا میان نگذر مناسب نیست.

نوع ۴) فیلتر نا متقارن با تعداد ضرایب زوج یا مرتبه فرد

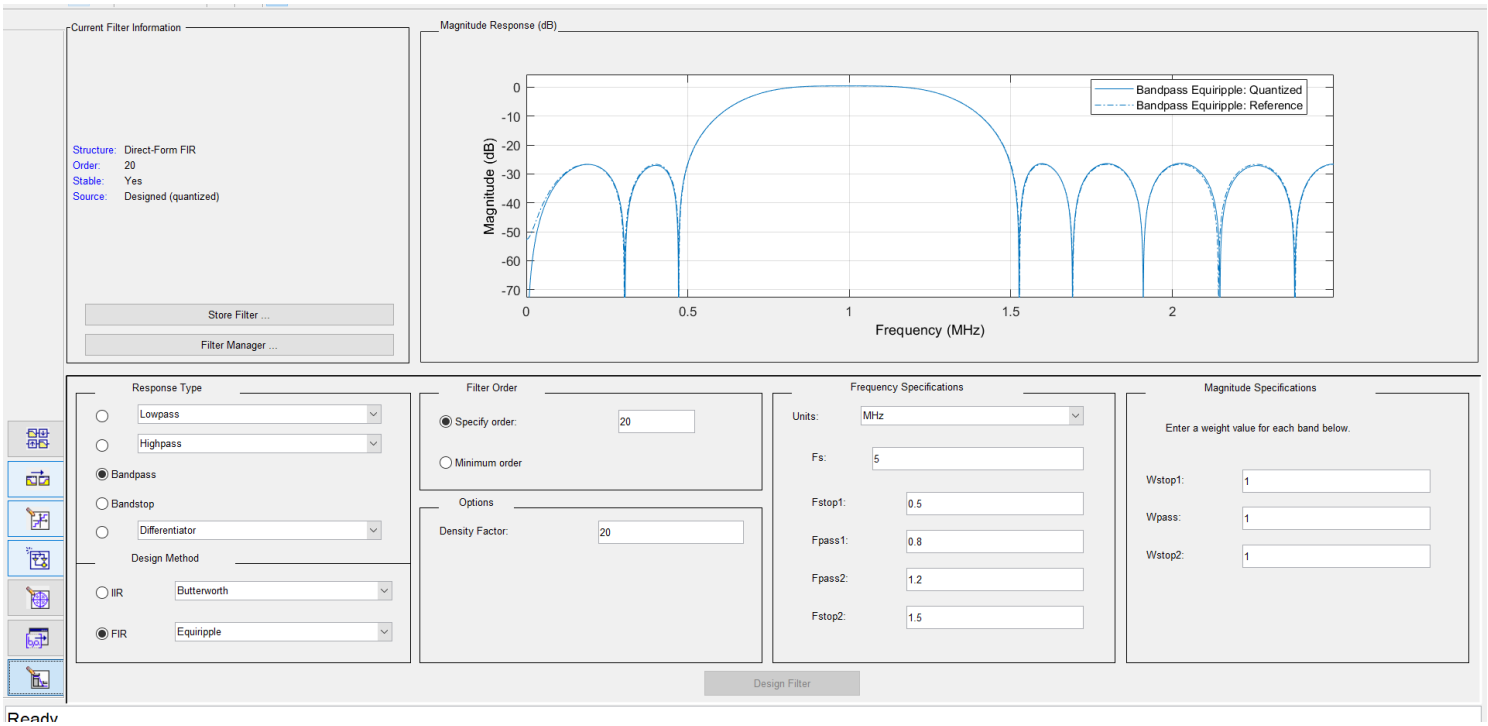


(c)

این نوع نیز بدلیل وجود صفر در $z = 1$ برای موارد پایین گذر مناسب نیست.

مشخصات فیلتر

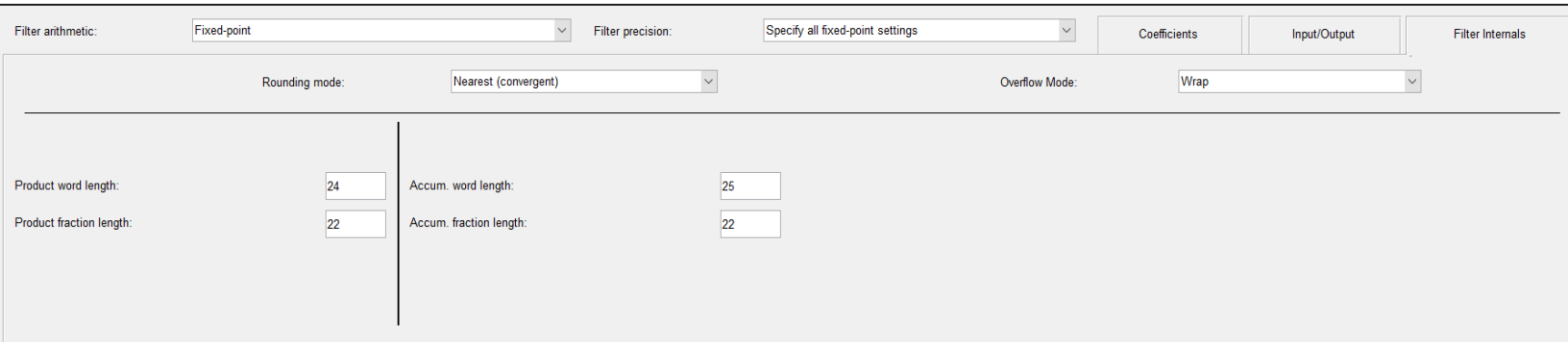
این فیلتر تماما با افزونه fdatool نرم افزار matlab طراحی شده و در ادامه به فرمت fixed point تبدیل شد.

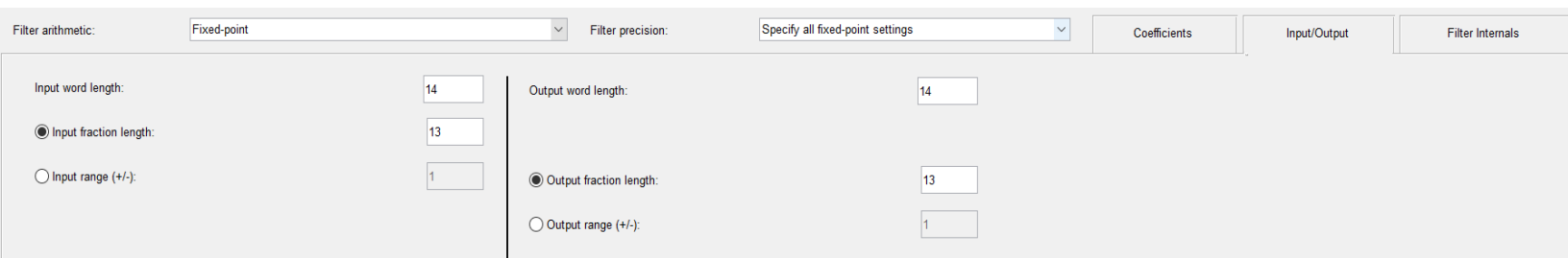


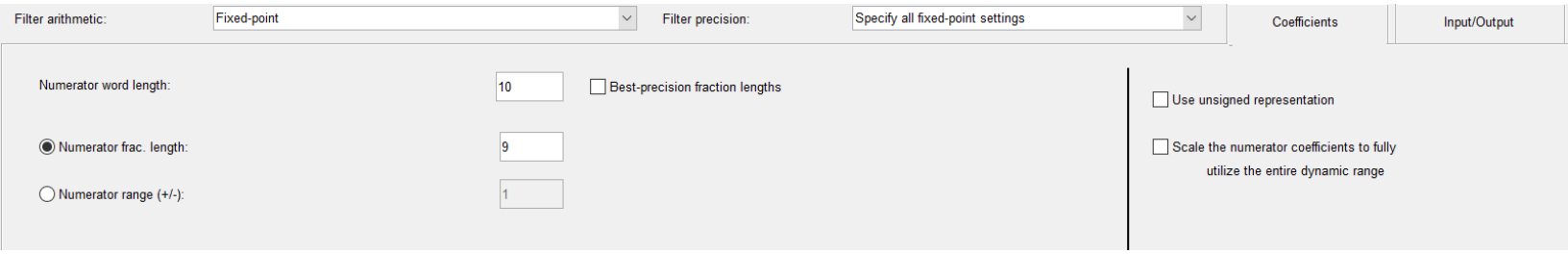
شکل ۱) مشخصات فیلتر

Scaling و Quantization

در بخش بعدی به سراغ تبدیل ضرایب به Fixed Point و مقایسه فیلتر حاصل با Floating Point میرویم :

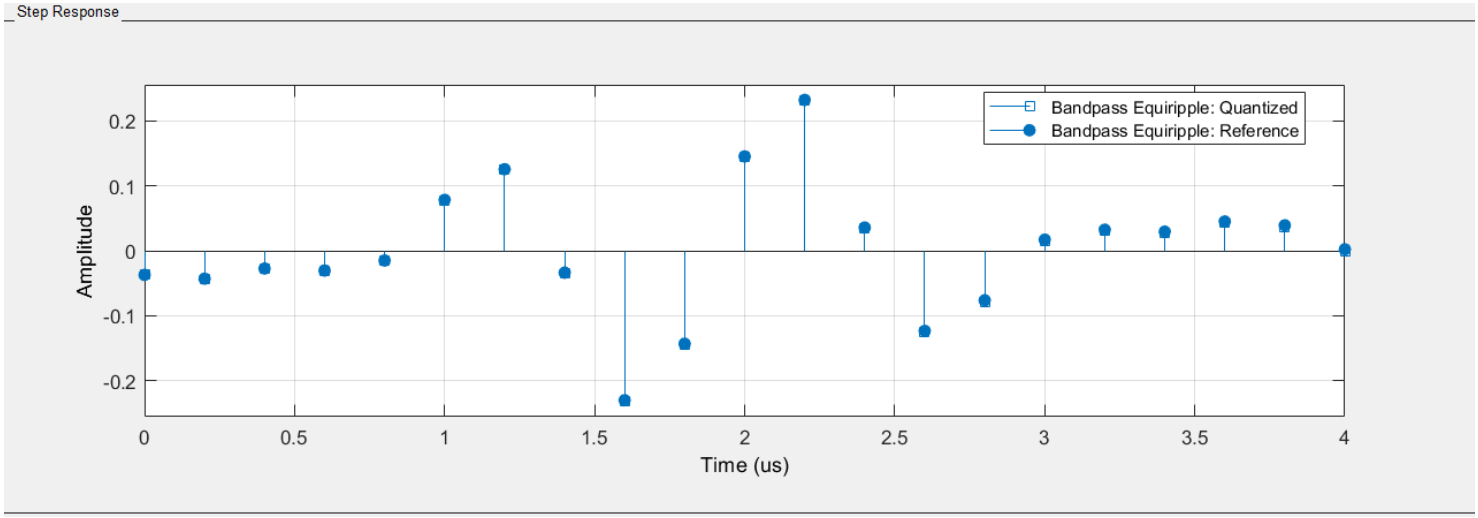






در این مرحله عرض بیت ورودی و خروجی که به طور تقریبی محاسبه میشود را ۱۴ بیت با ۱ بیت قسمت صحیح همچنین برای ضرایب، عرض بیت را ۱۰ که ۹ بیت آن قسمت کسری و یک بیت مقدار صحیح است ،در نظر گرفته میشود.

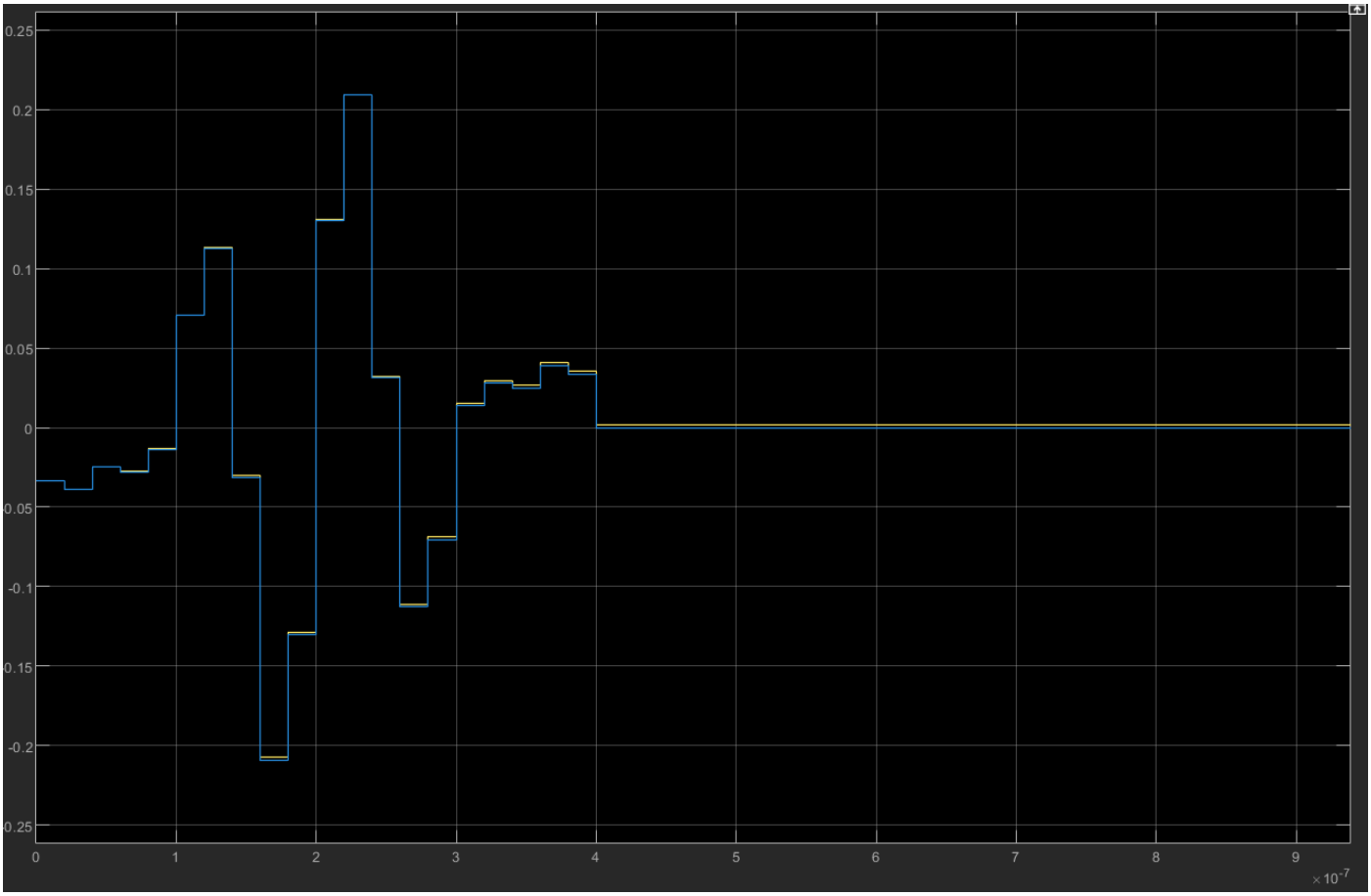
در شکل ۱) حاصل نشان داده شده که با مقدار Floating Point (نمودار نقطه چین) مقایسه شده.



شکل ۲) مقایسه پاسخ پله ها

در شکل ۲) نیز مقایسه پاسخ پله ها را میبینیم که به طور واضح مقدار کوانتیزه شده ضرایب ، تقریب خوبی از مقدار ممیز شناور آن هاست.

مقایسه فیلتر ها در سیمولینک

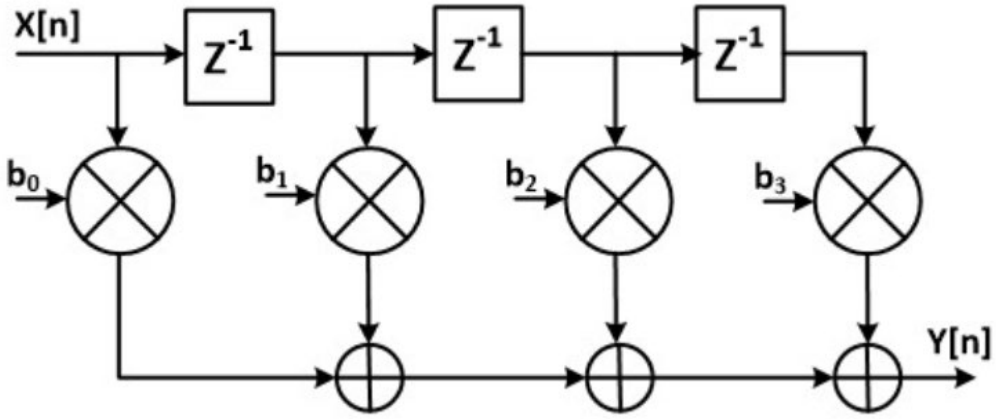


شکل ۳) مقایسه پاسخ پله (آبی مدل فیکس پوینت زرد مدل ممیز شناور)

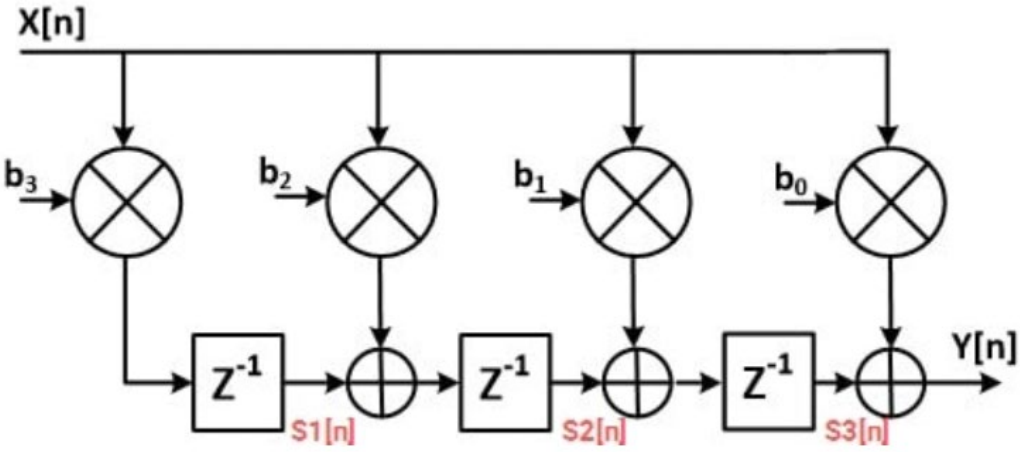
مطابق شکل ۳) پاسخ پله مدل کوانتزه شده که با تقریب خوبی مطابق مدل اصلی یا ممیز شناور است.

بررسی مدل بدون PipeLine

پیاده سازی فیلتر ها FIR به دو روش Direct و Transposed انجام میشود که ما دراینجا به مدل Direct پرداختیم :



ساختار Direct Form فیلتر FIR



ساختار Transposed Structure فیلتر FIR

شکل ۴) تفاوت مدل Direct و Transposed

در مدلTransposed تاخیر زمانی محاسبه جمع ها کمتر خواهد بود پس نسبت به مدل Direct بهینه تر است اما پیاده سازی آن پیچیده تر خواهد بود.

در پیاده سازی فیلتر این پروژه در مدل Direct بدون PipeLine کافیست در یک پروسه خطوط تاخیر(Delay Line) ها را بسازیم و سپس ضرب در ضرایب فیلتر کنیم و نتیجه را با هم به یکباره جمع کنیم و به خروجی بدهیم.

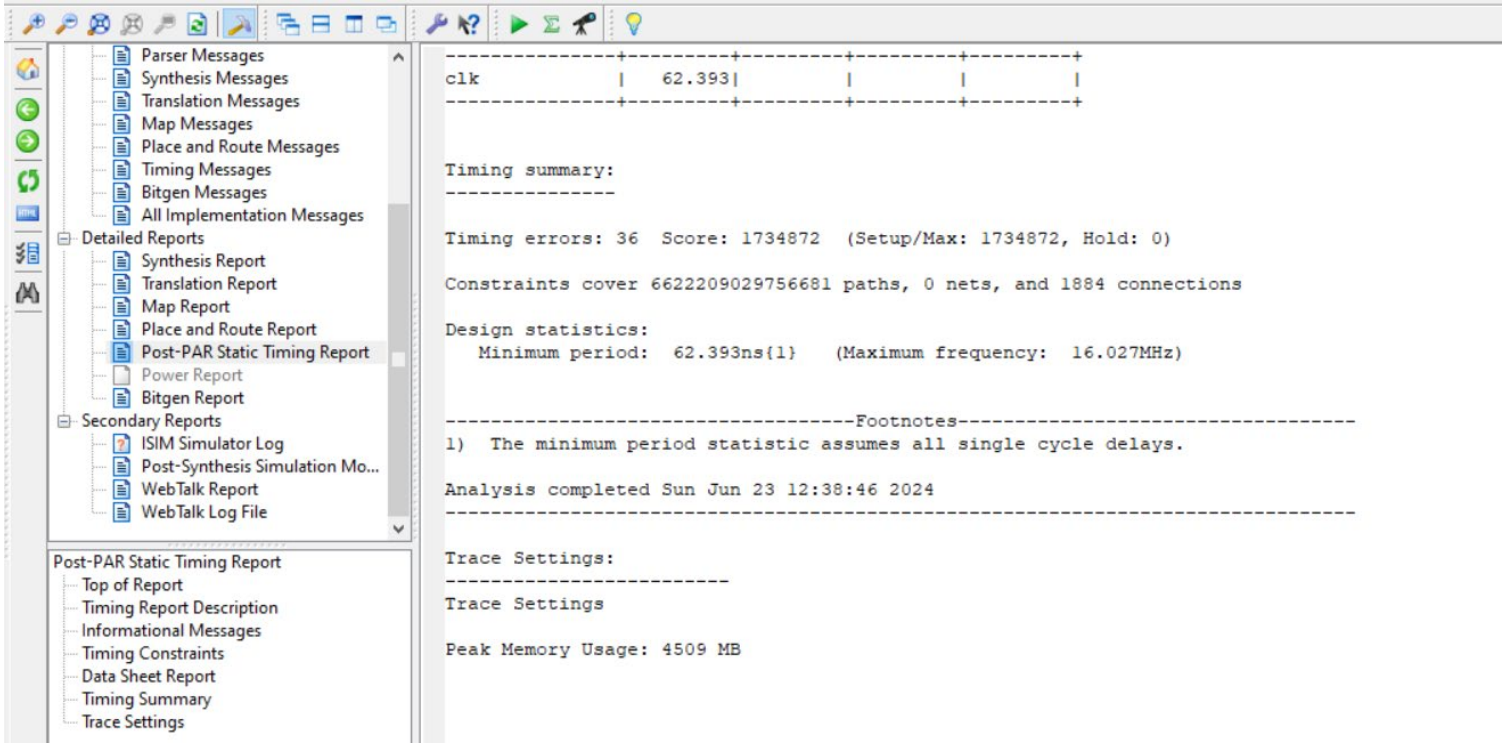
پس به تعداد ضرایب فیلتر یعنی ۲۱، عمل ضرب خواهیم داشت که نتیجه این ضرب ها را باید با هم جمع کنیم تا به خروجی بدهیم.

- برای صحت سنجی پیاده سازی پس از سنتز ماژول باید برای آن تست بنچ نوشته شود که خروجی را در فایلی ذخیره کند سپس نتایج این فایل را در نرم افزار متلب خوانده و با فیلتر پیاده شده اصلی با ضرایب ممیز شناور مقایسه میشود .
- نوشتن روی فایل با دستور ساده $fwrite انجام میشود.
- برای بررسی تاخیر های زمانی این ماژول و طراحی باید ابتدا فایلی به فرمت UCF به نام Constrain به ماژول اضافه کنیم تا بتوانیم Post-PAR Static Timing Report را مشاهده کنیم

در این فایل به تعریف فرکانس کلاک بسنده کردیم :

```
NET "clk" PERIOD = 10 ns;
```

با این کار میتوان گزارش کاملی از تاخیر ها زمانی مانند Critical Path و یا Maximum Clock Frequency را بدست آورد.

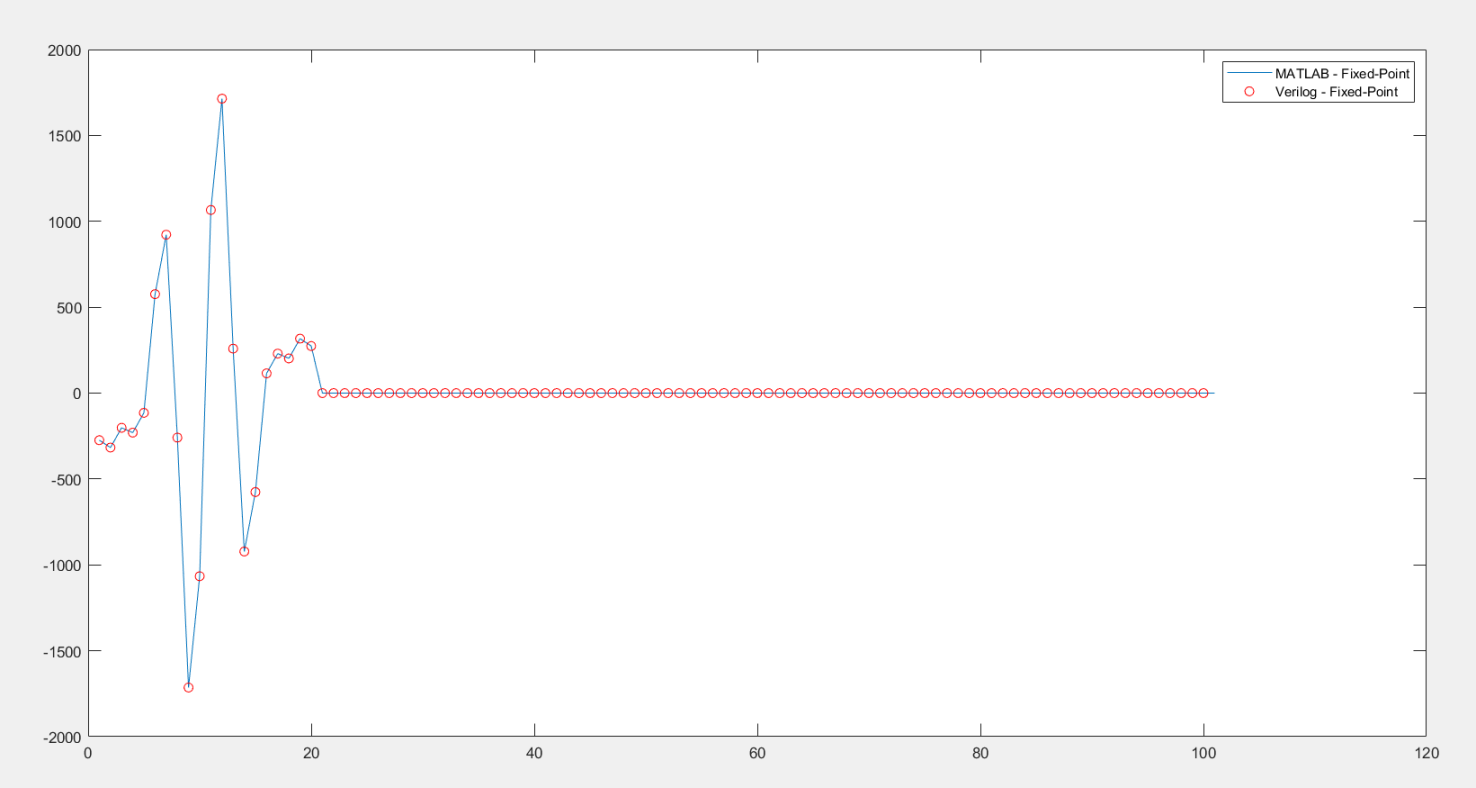


شکل ۵) گزارش سنتز

همانطور که مشخص است بیشترین کلاکی که میتوان به سیستم اعمال کرد برابر 16.027 MHz خواهد بود که مقدار بسیار کمی است.

مقایسه با متلب

کافیست خروجی تست بنچ را به متلب دهیم و با فیلتر طراحی شده اصلی مقایسه کنیم



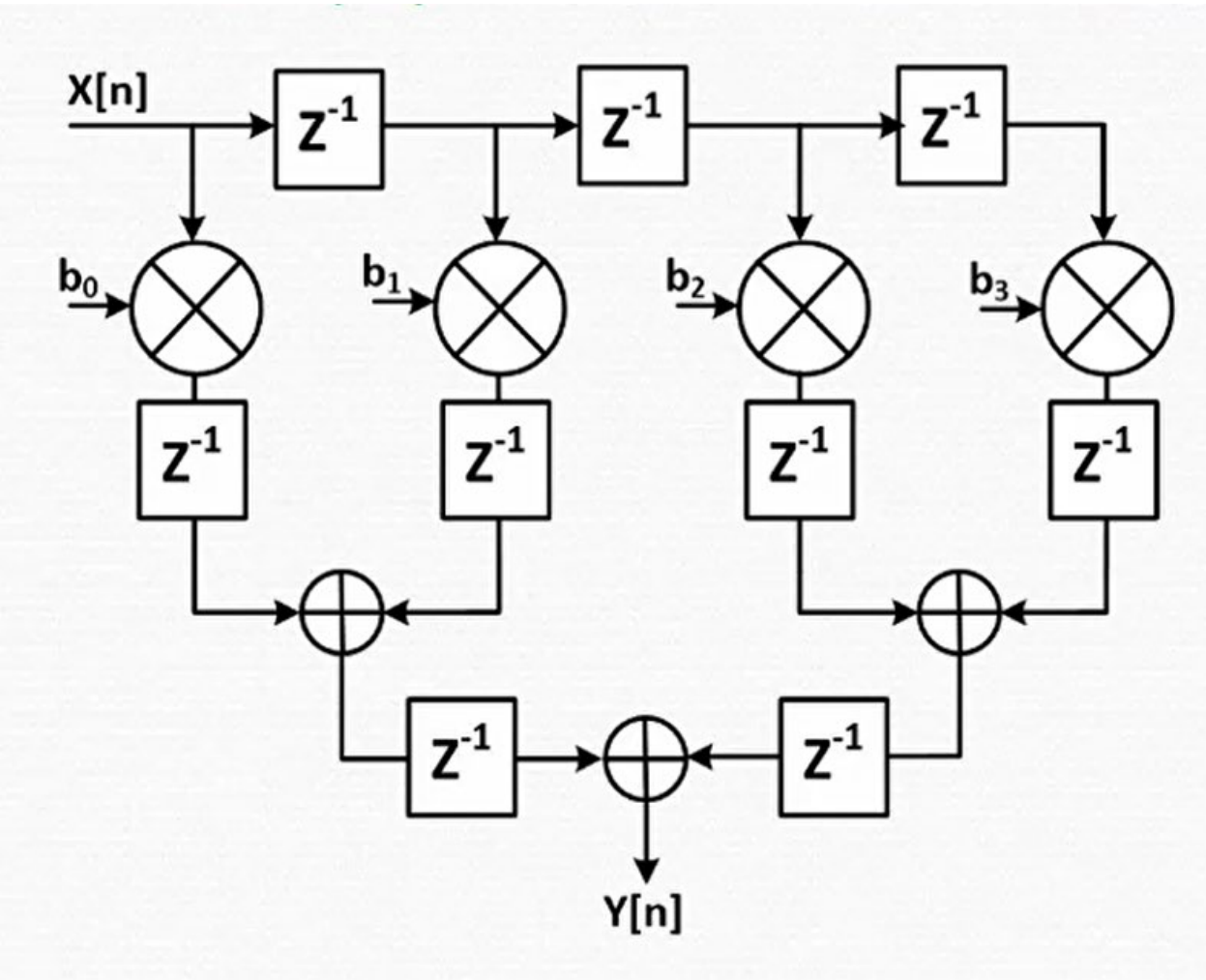
شکل ۶) مقایسه خروجی وریلاگ با فیلتر ممیز شناور اصلی

همانطور که مشخص است خروجی تست بنچ دقیقا همان خروجی فیلتر طراحی شده با فرمت Float است.

تنها عیبی که میتوان به این طراحی گرفت بحث زمانی یعنی ماکزیمم کلاک قابل اعمال است که برای رفع این مشکل به سراغ Pipe Line میرویم.

بررسی مدل همراه با PipeLine

برای پیاده سازی مدل با پایپ لاین تنها کاری که باید بکنیم این است که عملیات های طولانی جمع را به جمع های دو بخشی کوچک تر تقسیم کنیم و حاصل را رجیستر کنیم با این کار برای رسیدن به ۱ داده خروجی نیاز نیست منتظر کل عملیات بمانیم چون بخشی از محاسبات از قبل و در مراحل قبلی انجام شده.

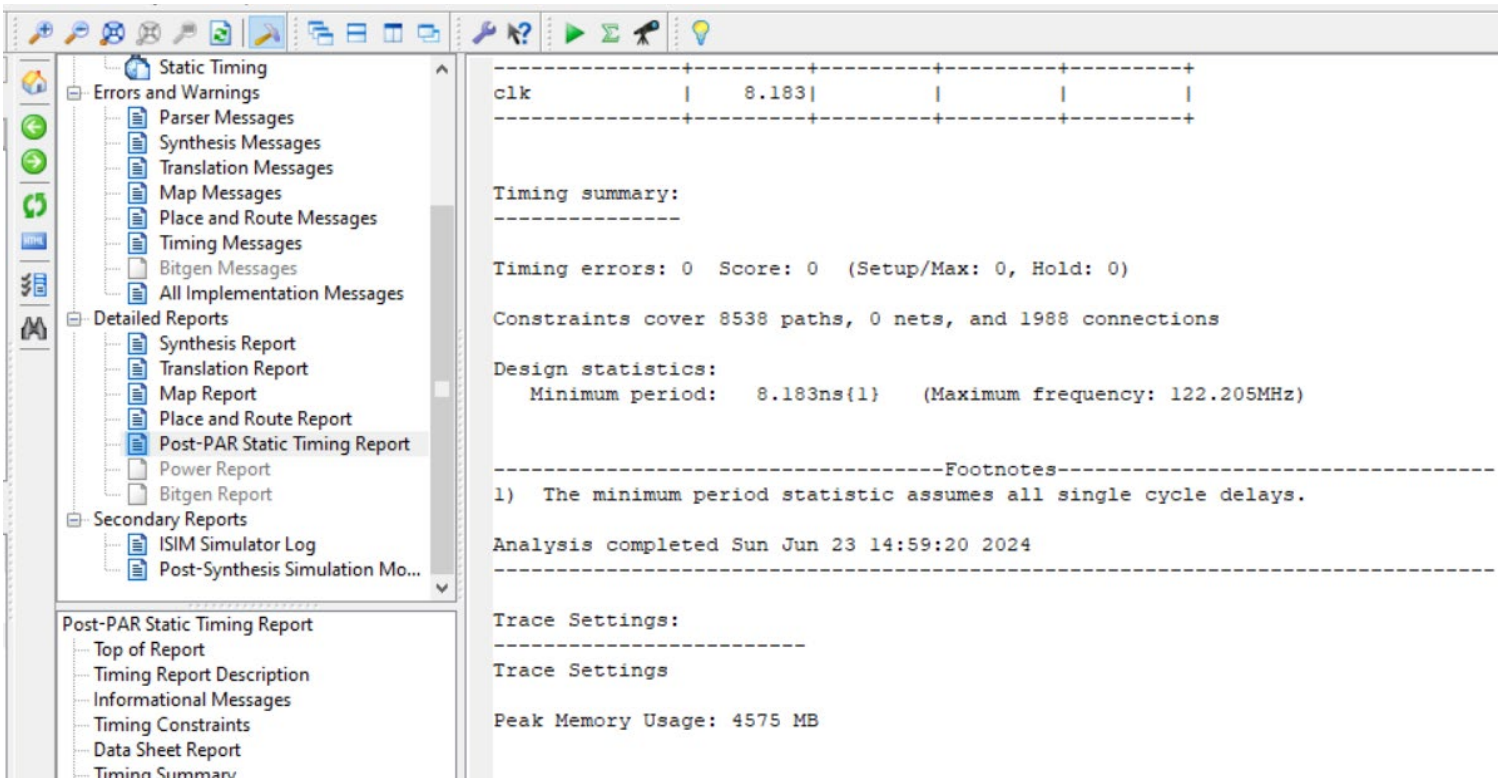


شکل ۷) مدل ساده پایپ لاین

در کد این مدل کافیست خروجی تمام ضرب ها را دو به دو به صورت non Blocking جمع کنیم و دوباره خروجی این جمع ها را با هم جمع کنیم تا در نهایت به ۲ یا ۳ جمع با دو عامل برسیم .

مقایسه با متلب

خروجی این ماژول دقیقا همان اعداد مدل بدون پایپ لاین میشود پس نیازی به مقایسه نیست اما در بخش گزارش زمانی (Timing Report) خواهیم داشت :



شکل ۸) گزارش زمانی ماژول با پایپ لاین

همانطور که مشخص است بیشترین کلاک قابل اعمال سیستم به شکل چشم گیری افزایش یافته و به مقدار 122.205MHz رسیده که نشان از کم شدن مسیر بحرانی و تاخیر سیستم میدهد.