تمرین ۵ درس طراحی کامپیوتری سیستم های دیجیتال

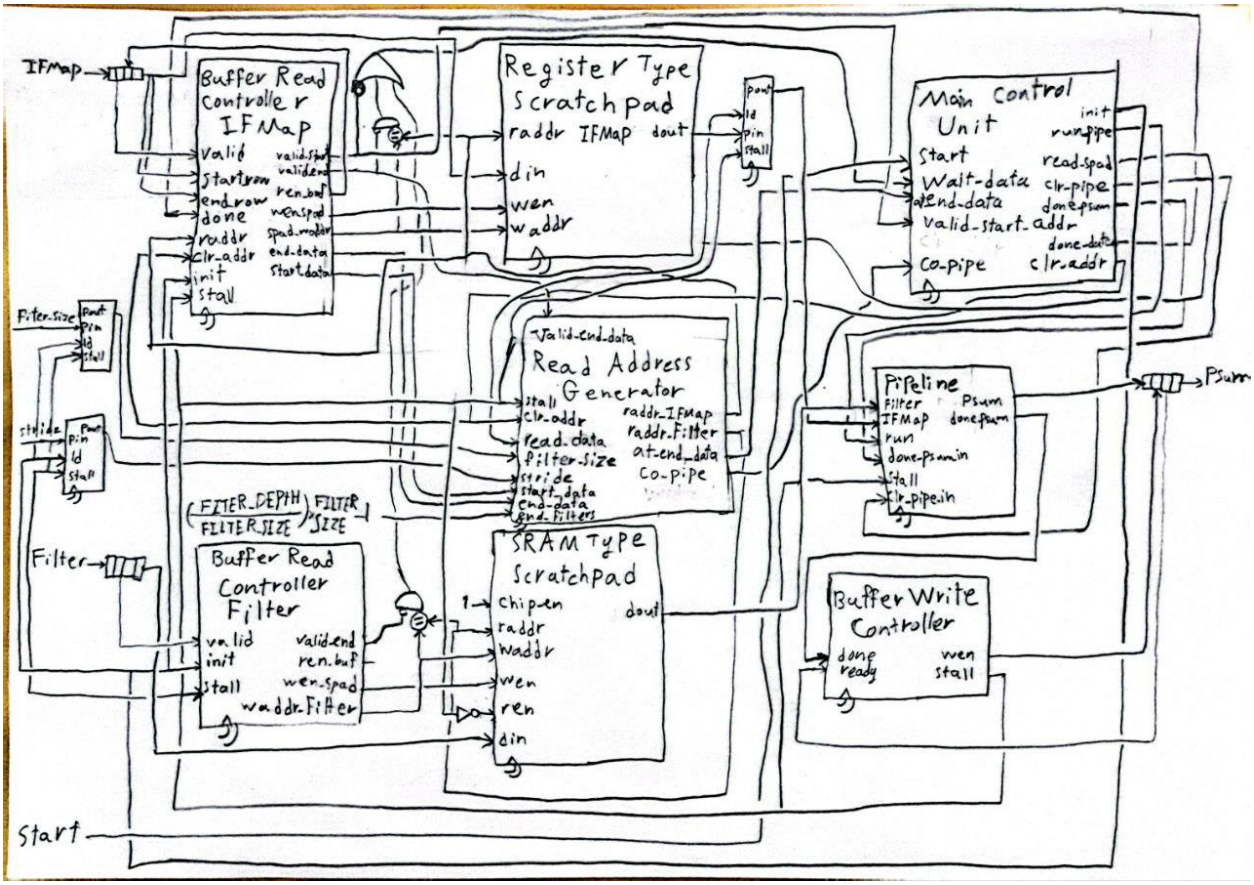
بابک حسینی محتشم ۸۱۰۱۰۱۴۰۸

کیارش خراسانی ۸۱۰۱۰۱۴۱۳

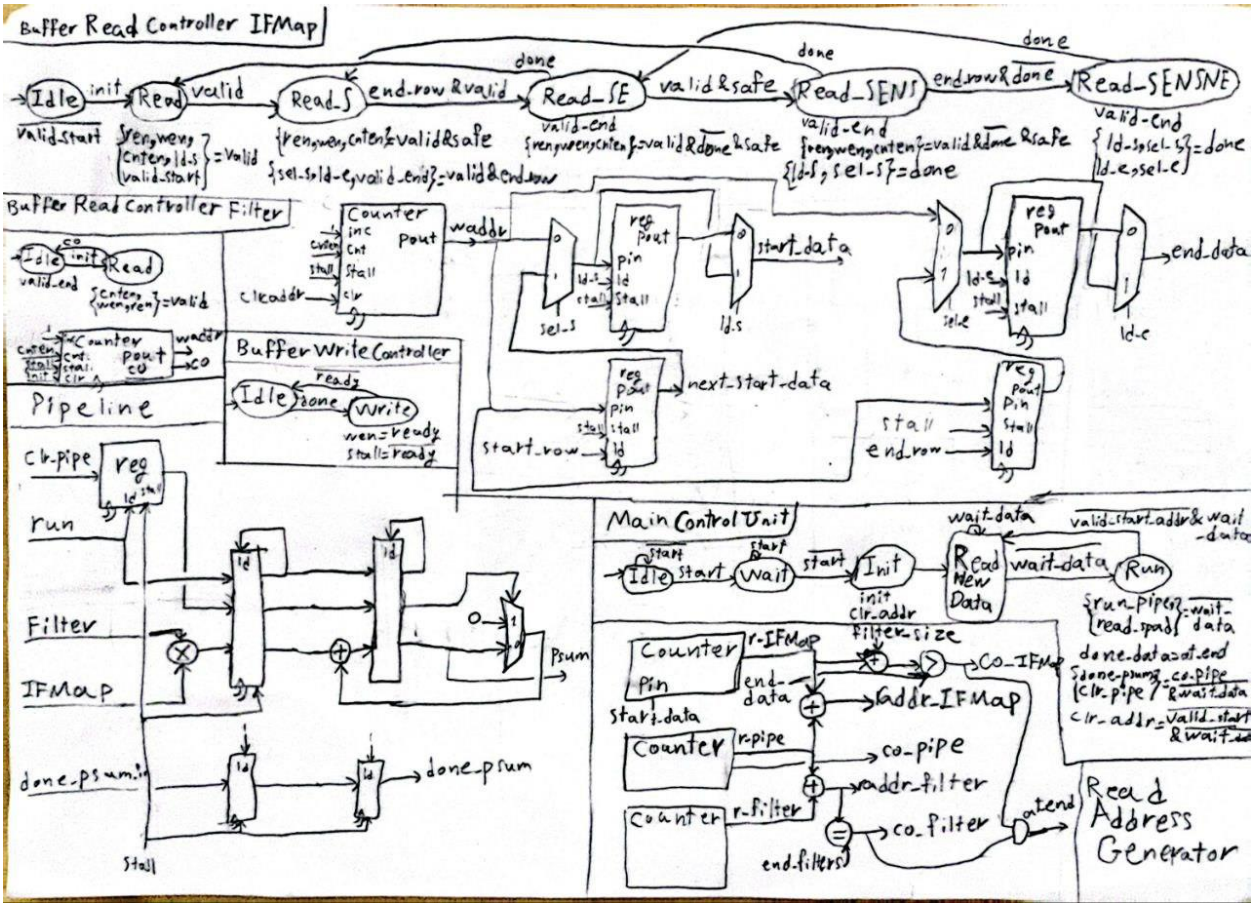
در این تمرین هدف، پیاده سازی کامل کردن قسمت محاسباتی ماژول eyeris بود. در این تمرین، به ماژول PE تمرین قبل، سه حالت مختلف اضافه کردیم و کارکرد آن را تا حدودی تغییر دادیم به طوری که، با هر بار گرفتن کانولوشن رو یک پنجره، نتیجه در scratchpad جدیدی ذخیره می‌شود و در نهایت پس از اتمام تمام محاسبات، نتایج ذخیره شده با input_psum که به عنوان ورودی داده می‌شود جمع شده و به خروجی می‌رود.

ابتدا طرح کلی را که طراحی کردیم مشاهده میکنیم.

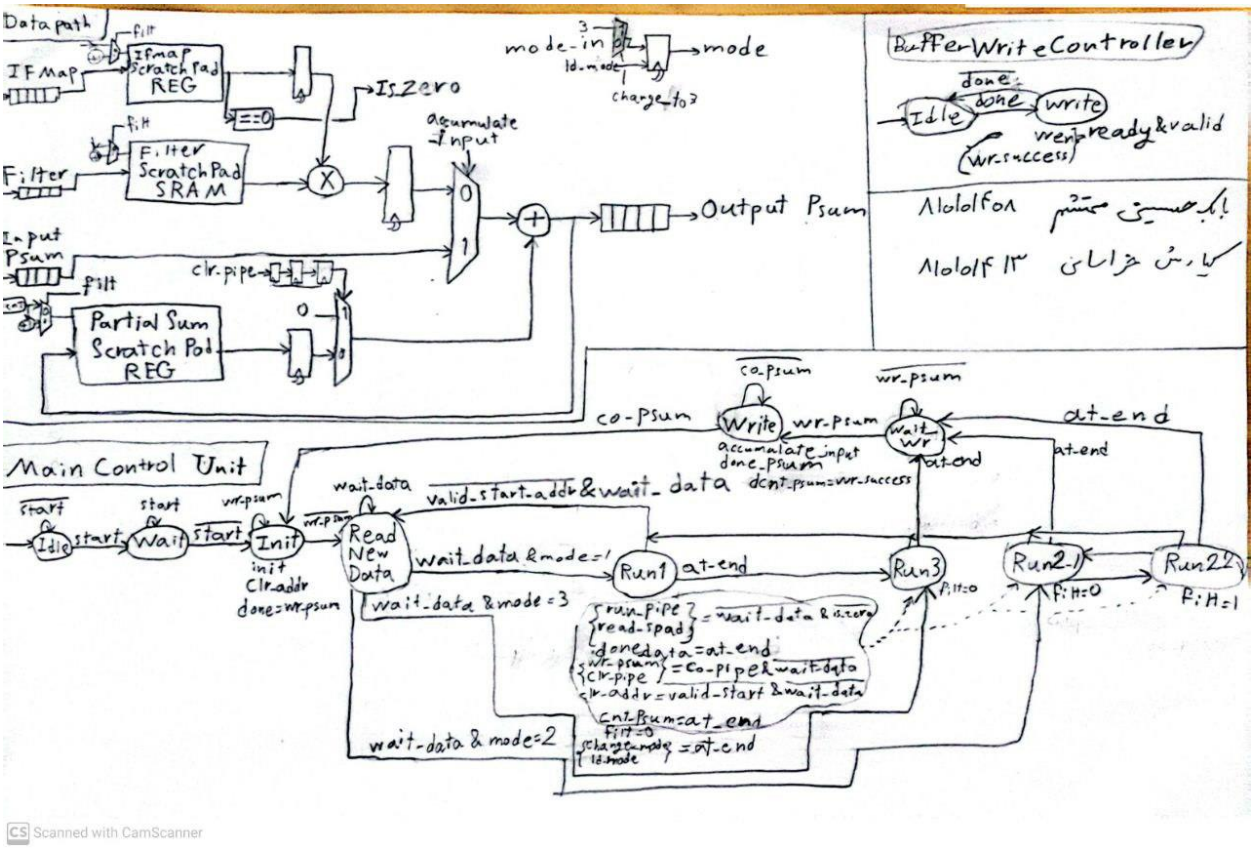
طرح نهایی مسیر داده تمرین قبلی:



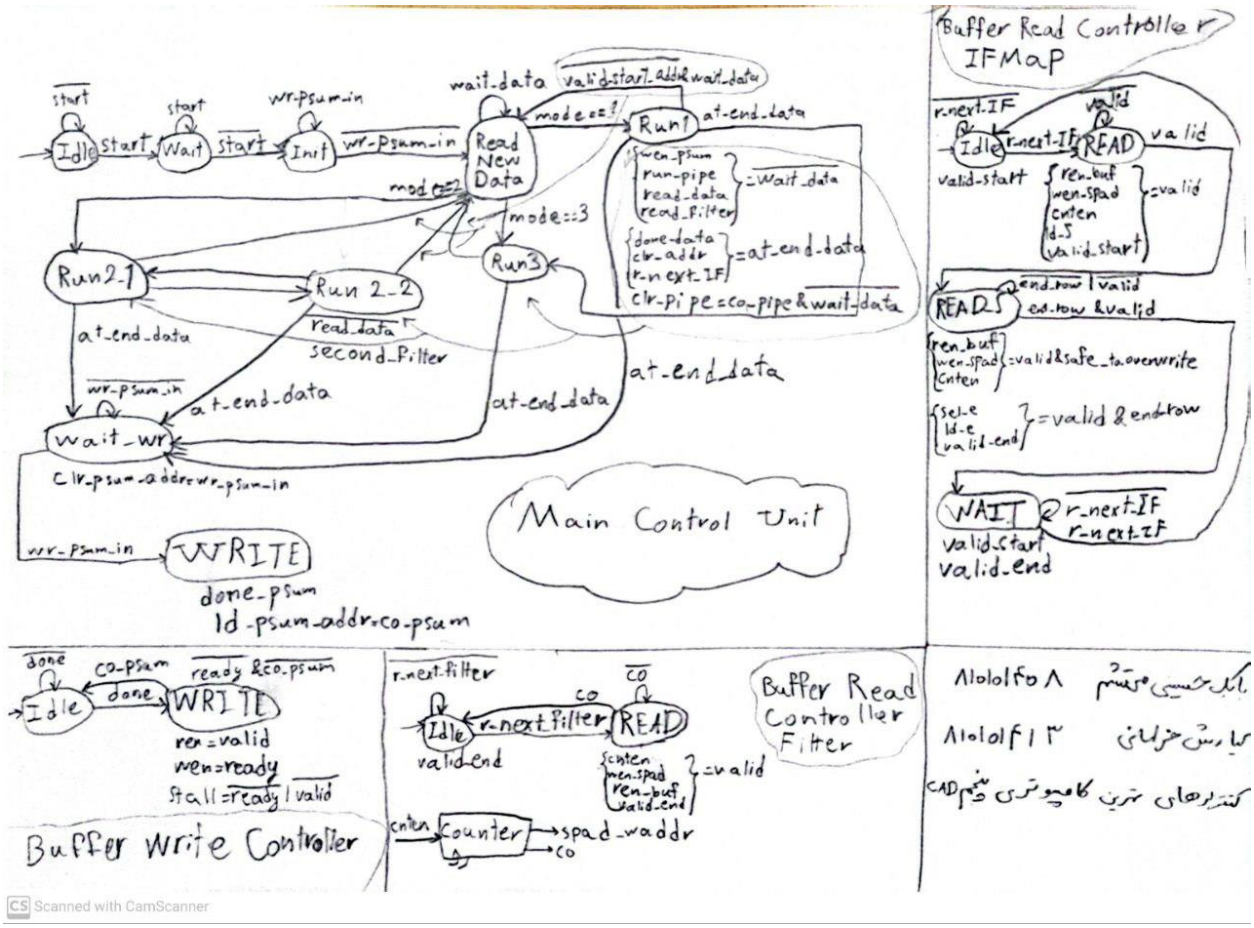
طرح نهایی هر یک از ماژول ها در تمرین قبلی:



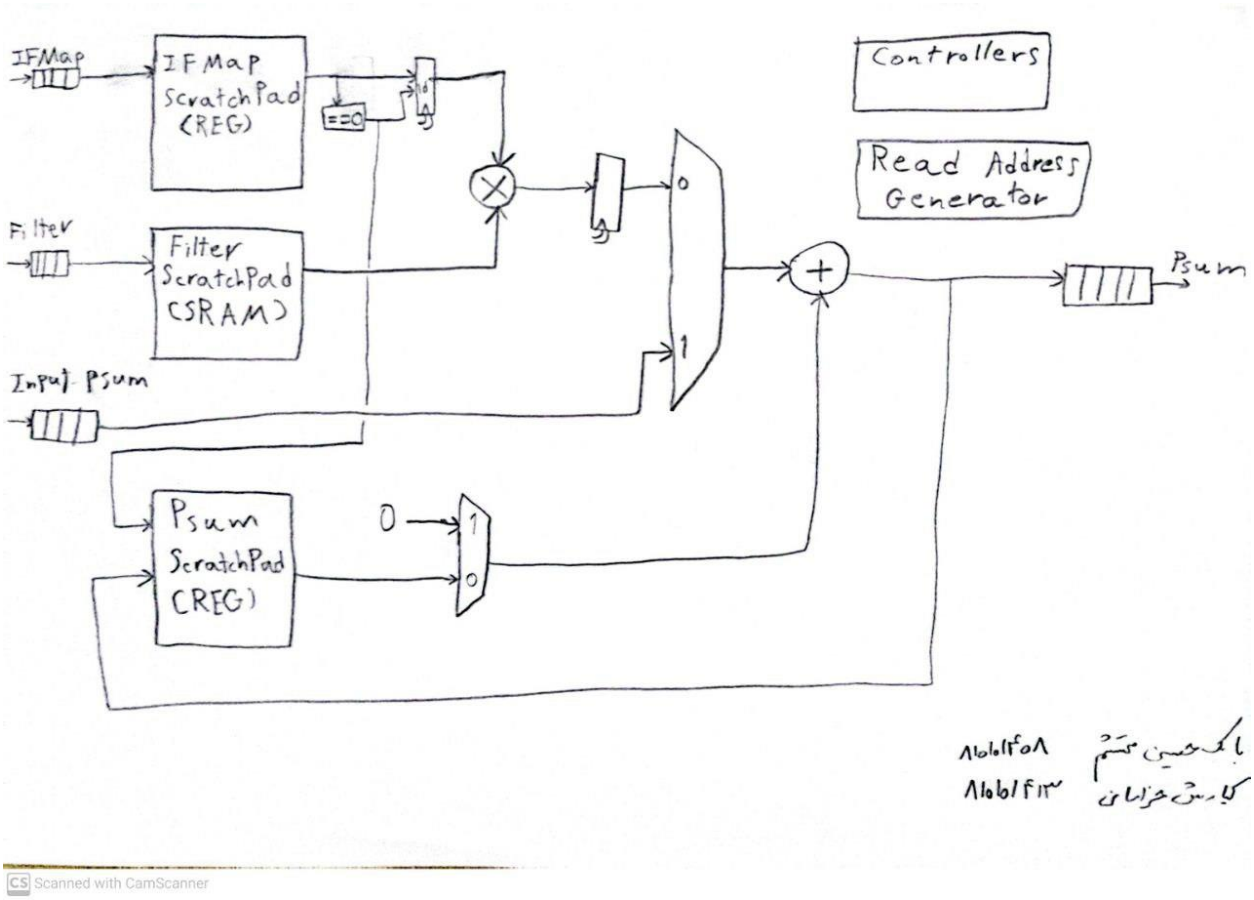
طرح اولیه این تمرین:



طرح نهایی کنترلرهای این تمرین:



طرح نهایی مسیرداده این تمرین:



تغییرات ایجاد شده در این تمرین نسبت به تمرین قبلی:

ماژول کنترل کننده اصلی:
در تمرین پیش، این کنترلر استیتی به نام RUN داشت که مدیریت کل ماژول در این استیت رخ می‌داد. در این تمرین ما به جای یک استیت RUN چهار استیت RUN1,RUN2_1,RUN2_2,RUN3 داریم. از استیت READ_NEW_DATA که برای خواندن داده جدید استفاده می‌شود، با توجه به مقدار mode ورودی، به یکی از این استیت‌های اجرا می‌رویم. کارکرد کلی این استیت‌ها مشابه است.

RUN3: در این استیت مانند تمرین قبل، سیگنال‌های کنترلی به ماژول‌های مختلف فرستاده می‌شوند. برخی از این سیگنال‌ها عبارتند از: run_pipe برای اجرای پایپ‌لاین، done_data به معنای اتمام محاسبات و نوشتن در خروجی، read_data/read_filter برای فرستادن داده بعدی به پایپ‌لاین و....

در این استیت مود سوم اجرا میشود و اگر نیاز به خواندن داده باشد به استیت read_new_state میرویم و اگر محاسبات سطر فعلی تمام شود به WAIT_WR میرویم.

RUN1: سیگنال‌های کنترلی تقریبا یکسانی با RUN3 دارد ولی وقتی که یک سطر حساب شد، به RUN3 می‌رود و سیگنال خواندن سطر بعدی را میفرستد.

RUN2_1/RUN2_2: کنترلر در صورتی که مود ۲ باشد، مدام بین این دو استیت جابه‌جا میشود و در استیتی RUN2_2 نیازی به خواندن داده نیست چون با داده قبلی کار داریم پس سیگنال read_data غیرفعال است و همچنین second_filter فعال است که مشخص کنیم در حال انجام محاسبه برای فیلتر دوم هستیم.

دو استیت WAIT_WR و WRITE در این تمرین به کنترلر اصلی اضافه شده اند که در تمرین قبلی وجود نداشتند:

WAIT_WR: تا وقتی دستور نوشتن در خروجی از بیرون نیاید در این استیت می‌مانیم.

WRITE: در این استیت نتایج ذخیره شده را با ورودی input_psum جمع، و به بافر Psum میفرستیم.

تغییرات Buffer Write Controller:

تنها نیاز بود، که این سیگنال با input_psum نیز handshake کند تا داده های ورودی را دریافت کند. پس یک سیگنا خروجی ren به این کنترلر اضافه کردیم و همچنین سیگنال ready بافر را نیز به این کنترلر میدهیم.

تغییرات Buffer Read Controller Filter:

چون برخلاف پروژه قبل پس از هر بار اجرا باید فیلتر جدید خوانده شود پس تغییراتی اندک در این کنترلر لازم بود. به طوری که با دریافت سیگنال r_next_Filter این کنترلر از استیت IDLE به READ میرود و فیلتر جدید را میخواند.

تغییرات Buffer Read Controller IFMap:

برخلاف تمرین قبل، در این تمرین قصد نداریم تا حد امکان داده بخوانیم و باید هر داده را در زمان مناسب بخوانیم پس برخی استیت‌های این کنترلر را حذف و تغییر دادیم، حال استیت‌های این کنترلر بدین صورت است:

IDLE: منتظر سیگنال r_next_IF میمانیم تا سطر بعدی را ورودی بگیریم و با فعال شدن این سیگنال به READ میرویم.

READ: در این استیت ابتدا یک داده دریافت و به استیت READ_S میرویم. همچنین آدرس داده دریافت شده را نیز ذخیره میکنیم که نشان دهنده شروع داده هاست.

READ_S: تا رسیدن به داده با end_row، داده دریافت میکنیم و درنهایت به استیت WAIT میرویم.

WAIT: در این استیت هم مانند IDLE کار خاصی انجام نمیدهیم و تنها سیگنال های معتبر بودن داده ها را برخلاف استیت IDLE فعال میکنیم. با آمدن سیگنال r_next_IF دوباره به READ میرویم.

تغییرات Read Address Generator:

در این تمرین برای مود دوم، ما این گونه پیاده سازی کردیم، که داده IFMap فعلی را با دو خانه فعلی دو فیلتر حساب کرده و سپس به داده بعدی برویم پس شمارنده ای که در تمرین قبلی نشان دهنده فیلتر و داده nام بود را به دو شمارنده جدا برای filter و IF تبدیل کردیم که اگر مود دو بود، پس از هر دو محاسبه، شمارنده IF یکی افزایش میابد در صورتی که Filter با هر محاسبه افزایش میابد.

تغییرات پایپ لاین:

در این تمرین برخلاف تمرین قبل نیاز بود که خروجی ذخیره شده از بیرون بیاید. همچنین ممکن است به جای جمع مقدار psum ذخیره شده با حاصل ضرب جدید، نیاز باشد که آن را با input_psum جمع کنیم. هر دوی این موارد با استفاده از مولتی پلکسر، درست شدند. همچنین برخی سیگنال های جدید اضافه شده در این پروژه نیز از پایپ لاین عبور میکنند مانند سیگنال is_zero.

تغییرات مسیرداده:

به مسیرداده Psum Scratchpad را برای ذخیره Psumهای محاسبه شده اضافه کردیم. همچنین از دو شمارنده برای شمارش آدرس خواندن و نوشتن برای این حافظه استفاده کردیم. برای مود دوم، نیاز بود که یکی در میان از دو خانه مجاور حافظه بخوانیم که با گذاشتن یک مولتی پلکسر، در صورتی که مود دو باشد، مقدار آدرس فعلی یا آدرس پیشین را انتخاب میکنیم. همچنین سیگنال، is_zero را نیز برای بخش امتیازی اضافه کردیم که با سیگنال read_data که از کنترلر میاید and شده و به ورودی load رجیستر جلوی IFMap Scratchpad میرود تا در صورتی که داده ورودی صفر بود، به رجیستر وارد نشود. همچنین این سیگنال پس از گذشتن از پایپ لاین، با wen_Psum نیز and میشود و به wen حافظه psum میرود تا این داده داخل حافظه نوشته نشود.

در نهایت ماژول را با اجرای تست کیس‌های داده شده و تست کیس خودمان تست میکنیم.

برای مثال مقادیر psum scratchpad برای تست کیس آخر مود سوم: