پروژه پیاده‌سازی واحد محاسبات و منطق
درس معماری کامپیوتر
تاریخ تحویل: ۳۰ اردیبهشت

پیاده سازی واحد محاسبات و منطق[۱] با استفاده از زبان توصیف سخت‌افزار

هدف از این پروژه پیاده‌سازی یک واحد محاسبات و منطق است که به صورت همگام[۲] کار می‌کند، تمامی اعداد اعشاری ممیز ثابت بوده و سه بیت سمت راست آن ها قسمت اعشاری و ۵ بیت سمت چپ قسمت صحیح عدد را تشکیل می‌دهند. توجه شود در این پروژه برای راحتی کار، فرض شده همه‌ی اعداد مثبت هستند.

ورودی‌های شما هشت بیت برای عدد اول (a)، هشت بیت برای عدد دوم (b) و سه بیت برای انتخاب عملیات مورد نظری است که واحد محاسبات و منطق باید انجام دهد. خروجی این ماژول نیز باید هشت بیت نتیجه‌ی عملیات و ۳ بیت برای پرچم‌های خروجی باشد.

پرچم‌های خروجی عبارتند از:

i. پرچم سرریز: این پرچم از خروجی CarryOut آخرین جمع کننده یا BorrowOut آخرین تفریق کننده یا سرریز تقسیم کننده و ضرب کننده بسته به نوع عملیات تامین می‌شود.
ii. پرچم صفر: این بیت، در صورتی که نتیجه محاسبات خروجی واحد محاسبات و منطق، صفر بود، باید یک شود.
iii. پرچم علامت: این پرچم نمایانگر علامت خروجی است، در صورت یک بودن یعنی علامت حاصل منفی و در صورت صفر بودن مبین علامت مثبت حاصل است.

واحد محاسبات و منطق به صورت کلی از سه بخش زیر تشکیل شده:

- واحد کنترل[۳]
- واحد محاسبات[۴]
- واحد منطق[۵]

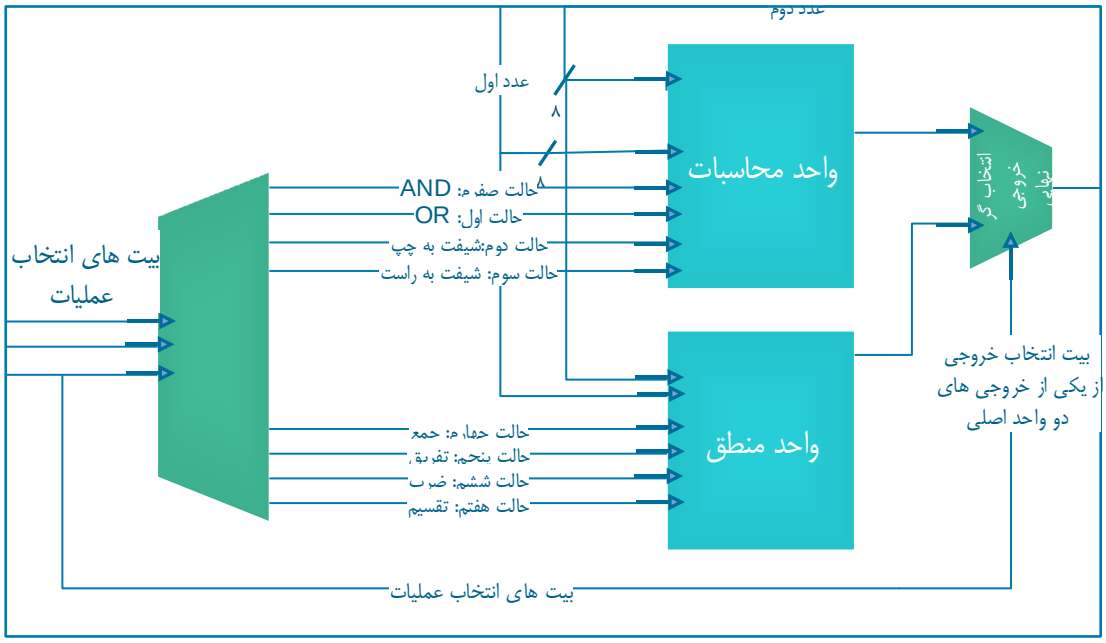
۱. واحد کنترل

در این واحد دو وظیفه انجام می شود: ۱- رساندن تمام بیت‌های ورودی (ورودی a، ورودی b و پرچم ها و غیره) به دو ماژول اصلی محاسبات و منطق. در واقع در اینجا شما باید در متن توصیفی، از واحد محاسبات و از واحد منطق نمونه‌سازی کنید[۶] و نحوه‌ی تعامل این واحدها با خروجی‌ها و ورودی‌ها را تعیین کنید (ورودی‌ها را به هر دو ماژول یعنی ماژول محاسبات و ماژول منطق بدهید و خروجی‌های مناسب را طبق سه بیت کنترل معین کنید).

پیاده‌سازی این واحد به صورت کلی علاوه بر موارد بالا شامل یک رمزگشا[۷] ۳ به ۸، که ورودی‌های آن از ورودی‌های کنترل ماژول اصلی تامین می‌شوند و چهار بیت کم ارزش خروجی آن به عنوان بیت‌های کنترل به ماژول محاسبات و چهار بیت پر ارزش آن به عنوان بیت‌های کنترل مطابق شکل به ماژول منطق داده می‌شوند.

---

۱ Arithmetic and logic unit
۲ Asynchronous
۳ Control unit
۴ Arithmetic unit
۵ Logical unit
۶ Instantiation
۷ Decoder



در شکل بالا دو ورودی a و b به صورت ۸ بیت موازی با هم وارد ماژول واحد کنترل می‌شوند و به هر دو واحد محاسبات و واحد منطق وارد می‌شوند (توجه کنید که در این شکل از آوردن پرچم‌ها خودداری شده اما شما ملزم به پیاده‌سازی آن هستید)، همینطور خروجی‌های هر یک از واحد ها به صورت دو گذرگاه ۸ بیتی هستند که توسط یک رمزکننده[۸] که از بین دو ورودی ۸ بیتی توسط پرارزش‌ترین بیت کنترلی انتخاب، مقادیر یکی از این دو گذرگاه ۸ بیتی به خروجی ماژول کنترل منعکس می‌شود.

– بدیهی است پرچم‌های خروجی هم مانند ۸ بیت حاصل انتخاب می‌شوند.

– توجه کنید معمولا در پروژه‌های بزرگ، به دلیل بزرگ بودن واحدهای کنترل از پرداختن به آن در دل سایر ماژول‌ها خودداری می‌شود و به جای آن از یک ماژول کنترل جداگانه استفاده می‌شود. لذا نسبت به تفکیک این قسمت از بقیه قسمت‌های مدار توجه لازم را مبذول دارید.

– برای محاسبه‌ی تاخیر و مساحت رمزگشا می‌توانید پس از اطلاع یافتن از نحوه‌ی پیاده‌سازی آن از طریق این صفحه و تسهیم‌کننده[۹] نیز از این صفحه اقدام کنید.

---

[۸] Encoder

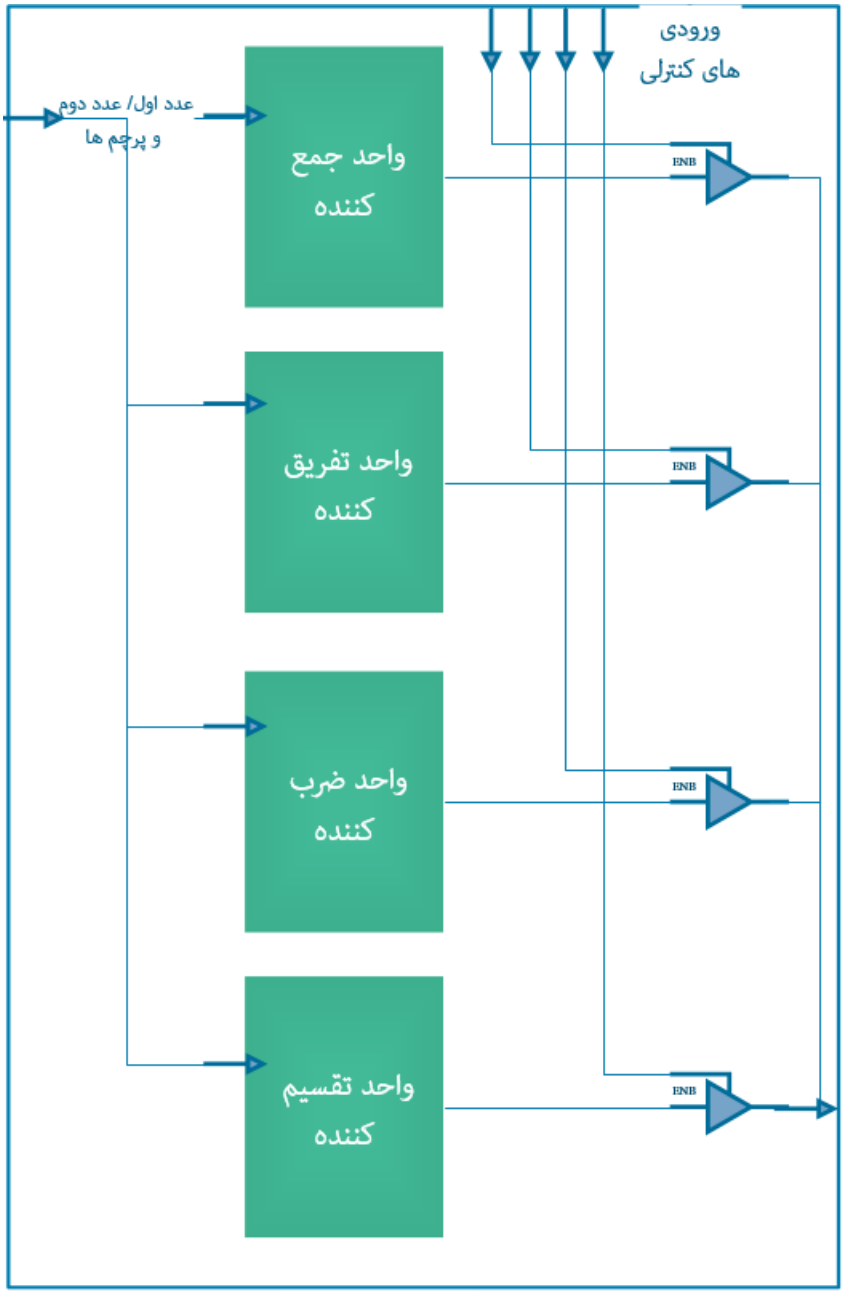
[۹] Multiplexer

۲. واحد محاسبات

این واحد خود از چهار واحد مجزا تشکیل شده است:

- واحد جمع
- واحد تفریق
- واحد ضرب
- واحد تقسیم

شما در واحد محاسبات باید از هر یک از واحد های بالا نمونه بسازید و ورودی‌های واحد محاسبات که خود خروجی‌هایی از واحد کنترل هستند را به همه‌ی واحد ها بدهید، خروجی محاسبات باید با توجه به چهار بیت کنترلی وارد شده از ماژول کنترل از بین خروجی های ماژول‌های جمع، تفریق، ضرب ویا تقسیم انتخاب شوند.



واحد محاسبات

واحد جمع

در این واحد شما باید دو جمع کننده‌ی چهار بیتی با منطق پیش بینی‌کننده ی رقم نقلی[۱۰] طراحی کنید و با متصل کردن آبشاری این دو جمع کننده یک جمع کننده‌ی هشت بیتی ممیز ثابت بسازید. در این روش شما چهار بیت کم ارزش را به ورودی یکی از این جمع کننده‌ها خواهید داد و مقدار ورودی بیت نقلی این جمع‌کننده چهار بیتی را صفر می گذارید و رقم نقلی خروجی آن را به رقم ورودی نقلی جمع‌کننده چهار بیتی بعدی بدهید و در آخر خروجی نقلی جمع‌کننده پیش‌بینی کننده‌ی دوم را به عنوان پرچم سرریز به خروجی ماژول اصلی اعمال کنید.

واحد تفریق

برای پیاده‌سازی واحد تفریق باید از دو تفریق کننده‌ی پیش‌بینی کننده‌ی رقم قرضی [۱۱] چهار بیتی استفاده کرده و با وصل کردن آبشاری دو عدد از این ماژول‌ها به هم یک تفریق‌کننده‌ی هشت بیتی بسازید. برای به دست آوردن مدار مربوط به یک تفریق کننده‌ی پیش‌بینی‌کننده رقم قرضی ۴ بیتی مسیر زیر را طی کنید تا به مدار مربوطه برسید:

i. ابتدا تصور کنید قرار است یک تفریق‌کننده‌ی یک بیتی بسازید که دو بیت را از هم کم می‌کند $(a_n - b_n)$ و یک بیت برای ورودی قرض دارد که همان مقداری است که رقم کم ارزش‌تر (رقمی که توسط تفریق‌کننده سمت راست محاسبه

[۱۰] Carry look ahead adder
[۱۱] Borrow look ahead subtractor

می‌شود) اگر بخواهد از این رقم چیزی قرض بگیرد این بیت را یک می‌کند و در غیر این صورت این بیت صفر است(bin_n)، از طرفی اگر این واحد برای تفریق دو مقدار اصلی دچار کمبود ارزش شد، باید بتواند از رقم پرارزش‌تر که سمت چپ این جمع‌کننده قرار می‌گیرد، مقداری قرض بگیرد ($bout_n$). پس یک بیت هم به عنوان خروجی دارد که اگر یک باشد، یعنی این واحد دچار کمبود ارزش شده و از رقم پرارزش‌تر قرض گرفته است و اگر صفر بود یعنی دچار کمبود ارزش نشده. همینطور یک بیت دیگر برای حاصل تفریق وجود خواهد داشت (sub_n).

ii. قطعا برای به دست آوردن حاصل (sub_n) مشکلی نخواهد بود (به فرض از یک Full subtractor استفاده شود) و قسمت مشکل پیش‌بینی رقم قرضی است! برای این کار تمام حالاتی که برای دو بیتی که قصد کم‌کردن آن ها را داریم (a_n و b_n) در نظر می گیریم (a=0, b=1/a=0, b=0/a=1, b=0/a=1, b=1) که در حالاتی که a و b هر دو صفر یا هر دو یک باشند هر چه مقدار bin_n باشد به $bout_n$ منتقل می‌شود و در صورتی که a=0 و b=1 است مستقل از مقدار bin_n مقدار $bout_n$ برابر با یک شده و در حالتی که a=1 و b=0 باشد چه مقدار bin_n یک باشد یا صفر باشد مقدار $bout_n$ همیشه صفر خواهد بود.

iii. حال با کمک دو گام قبل و ذهنیت کلی از جمع‌کننده‌ی پیش‌بینی کننده‌ی نقلی وجود دارد، به توصیف تفریق‌کننده‌ی نقلی چهار بیتی بپردازید.

واحد ضرب

در این واحد نیز شما باید قادر به ضرب دو عدد اعشاری ۸ بیتی ممیز شناور باشید و در پایان ۸ بیت دارای ارزش بیشتر را به عنوان خروجی اعلام کنید. با توجه به اینکه کل پیاده سازی به صورت آسنکرون انجام می شود الگوریتم پیشنهادی ضرب کننده ی آرایه ای است، اما در کل روش پیاده سازی این قسمت اختیاری است اما به خاطر داشته باشید که حتما توضیحات کافی راجع به روش خود را در گزارش خود قید کنید.

در صورت پیاده‌سازی ناهمگام ضرب‌کننده به شما نمرهی امتیازی تعلق خواهد گرفت (بدیهی است در صورت پیاده سازی سنکرون به ورودی های اصلی سیستم باید یک ورودی برای کلاک هم اضافه کنید).

واحد تقسیم

در این واحد شما باید ۸ بیت ورودی اول را به عنوان مقسوم و چهار بیت کم ارزش ورودی دوم را به عنوان مقسوم علیه در نظر بگیرید و ۴ بیت خارج قسمت را در ۴ بیت پر ارزش خروجی و ۳بیت باقیمانده را در ۳ بیت کم ارزش خروجی منعکس کنید.

این واحد نیز همانند واحد ضرب الگوریتمی اختیاری دارد. اما در صورت استفاده از الگوریتم زیر نمره‌ی امتیازی به شما تعلق خواهد گرفت:

هدف این بخش تولید ۸ بیت عدد اعشاری حاصل از تقسیم دو عدد هشت بیتی ورودی است، روش انجام این کار تقسیم با بازیابی است.

در ادامه دو الگوریتم برای تقسیم معرفی می‌گردد که الگوریتم دوم (تقسیم بدون بازیابی صرفا برای مقایسه‌ی آن با الگوریتم تقسیم با بازیابی آمده است):

تقسیم با بازیابی[۱۲]:

الگوریتم تقسیم با بازیابی به این صورت است که در ابتدا باقیمانده پاره‌ای با مقدار مقسوم، مقداردهی اولیه می‌شود. سپس در هر مرحله مقسوم علیه را از n–بیت پرارزش باقیمانده پاره‌ای[۱۳] کم می‌شود (مکمل ۲ مقسوم را با n–بیت پرارزش باقیمانده پارهای جمع می‌شود)، اگر حاصل مثبت باشد، بیت متناظر از خارج قسمت یک و اگر حاصل منفی باشد، بیت متناظر از خارج قسمت صفر می‌شود و مقسوم را با باقیمانده پاره‌ای جمع می‌گردد تا مقدار قبلی آن بازیابی شود. در حقیقت در این روش باقیمانده پاره‌ای منفی برای مراحل میانی نیز پذیرفته نیست و بلافاصله به تصحیح آن اقدام می‌شود. الگوریتم این تقسیم در زیر آمده است:

[۱۲] Restoring division
[۱۳] Partial reminder

1. $w_0 = x$
2. for $i = 0$ to $n-1$ do
   2.1. $w_i = 2w_i - d$
   2.2. if $(w_i \geq 0)$
   $w_{i+1} = w_i; q_{n-1-i} = 1;$
   else
   $w_{i+1} = w_i + d; q_{n-1-i} = 0;$

تقسیم بدون بازیابی[۱۴]:

الگوریتم تقسیم بدون بازیابی به این صورت است که باقیمانده پاره‌ای منفی که در یک مرحله به دست می‌آید پذیرفته شده (یعنی بلافاصله با انجام عملیات جمع آن را تصحیح نمی‌گردد) و تصحیح آن را در مرحله بعد با انجام عملیات جمع انجام صورت می‌گیرد. الگوریتم تقسیم بدون بازیابی در زیر آمده است. از آنجا که باقیمانده نهایی باید حتما مثبت باشد (توجه کنید که در اینجا تقسیم اعداد مثبت را توضیح داده می‌شود، در نتیجه باقیمانده نهایی باید مثبت باشد)، بنابراین در مرحله آخر اگر باقیمانده پاره‌ای منفی باشد، یک مرحله تصحیح نهایی وجود خواهد داشت (مرحله‌ی چهارم) .

الگوریتم این روش نیز به صورت زیر است:

1. $w_0 = x$
2. $w_1 = 2w_0 - d$
3. for $i = 1$ to $n-1$ do
   3.1. if $(w_i \geq 0)$
   $w_{i+1} = 2w_i - d; q_{n-1-i} = 1;$
   else
   $w_{i+1} = 2w_i + d; q_{n-1-i} = 0;$
4. if $(w_n < 0)$
   $w_n = w_n + d; q_0 = 0;$
   else
   $q_0 = 1;$

۳. واحد منطق

این واحد نیز همانند واحد محاسبات خود از ۴ زیر واحد زیر تشکیل شده است:

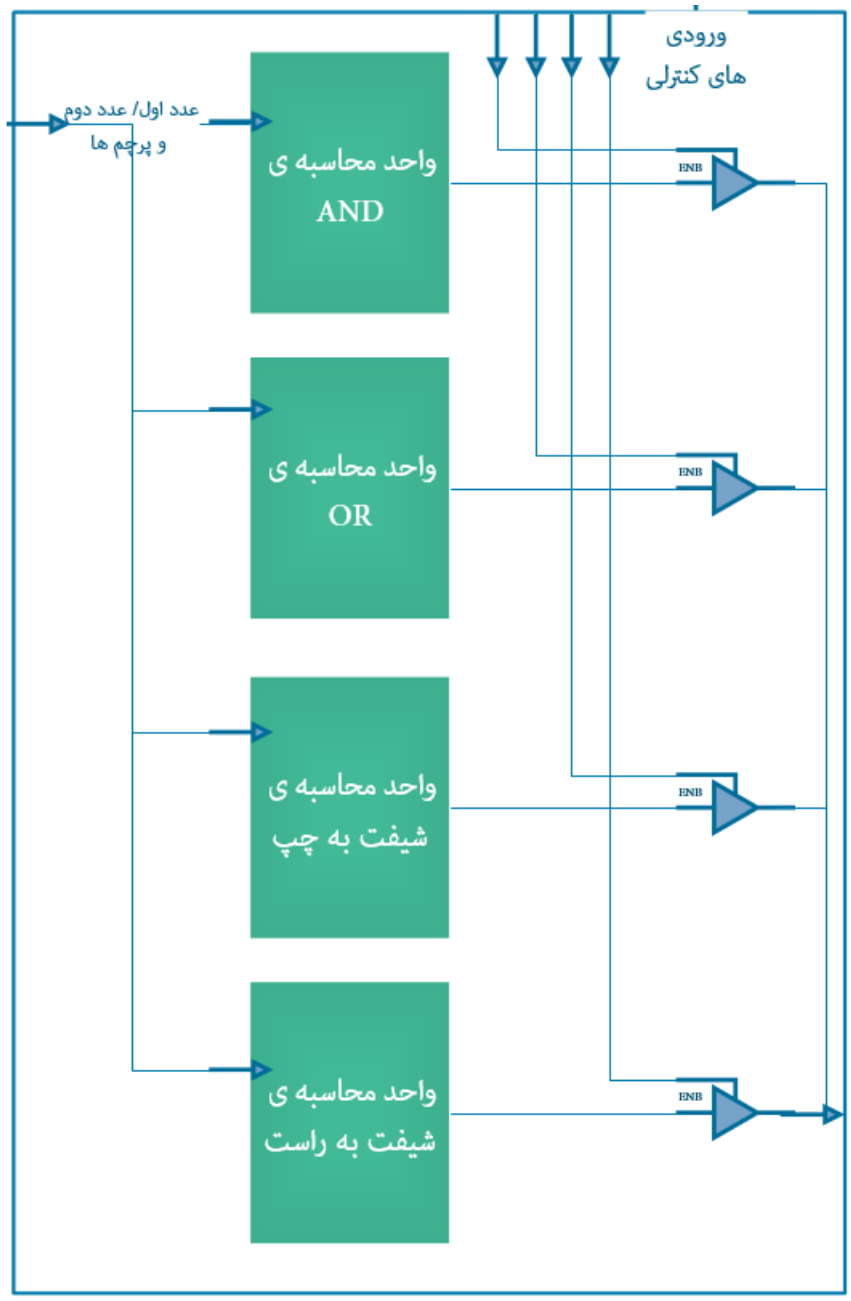
- واحد محاسبه‌ی AND
- واحد محاسبه‌ی OR
- واحد محاسبه‌ی شیفت حسابی به چپ
- واحد محاسبه‌ی شیفت حسابی به راست

و همانند واحد محاسبات همه‌ی ورودی ها را به هر چهار واحد محاسبه‌ی AND ، OR، شیفت به چپ و شیفت به راست می‌دهد و توسط چهار خط کنترل ورودی (که چهار بیت پر ارزش از ۸ بیت کنترل کلی حاصل از رمزگشا[۱۵] هستند) به انتخاب خروجی ماژول از یکی از خروجی های چهار واحد منطقی می پردازد.

---

[۱۴] Non restoring division

[۱۵] decoder



واحد منطق

واحد محاسبه‌ی AND

این واحد خروجی‌ای برابر AND تک تک بیت‌های متناظر دو ورودی a و b تولید می‌کند.

واحد محاسبه‌ی OR

این واحد خروجی‌ای برابر OR تک تک بیت‌های متناظر دو ورودی a و b تولید می‌کند.

واحد محاسبه‌ی شیفت حسابی به چپ

این واحد باید با گرفتن یک عدد هشت بیتی (بدون ملاحضه‌ی ممیز) آنرا یک بیت به سمت چپ شیفت دهد و بیتی که از شیفت به چپ در منتها الیه سمت راست عدد خالی می‌ماند را با صفر پر کند.

توجه: هر چند پیاده سازی این قسمت به کمک روش سنکرون ممکن است، اما پیاده سازی آسنکرون ساده تر است.

واحد محاسبه‌ی شیفت حسابی به راست

این واحد باید با گرفتن یک عدد هشت بیتی (بدون ملاحضه‌ی ممیز) آنرا یک بیت به سمت راست شیفت دهد و بیتی که از شیفت به راست در منتها الیه سمت چپ عدد خالی می ماند را با بیتی که قبلا در این مکان حضور داشته است پر کند.

توجه: هر چند پیاده سازی این قسمت به کمک روش سنکرون ممکن است، اما پیاده سازی آسنکرون ساده تر است.

بخش امتیازی ممیز شناور

این پروژه علاوه بر بخش های امتیازی ضرب و تقسیم، که با این رنگ در صورت پروژه مشخص شده اند، شامل دو بخش امتیازی دیگر زیر است:

۱- تمام به صورت کلی مبتنی بر اعداد ممیز ثابت است. در این بخش شما باید همه ی مراحل بالا را به جای استفاده از اعداد ممیز ثابت با استفاده از اعداد ممیز شناور توصیف کنید و برای این کار باید یک بیت علامت ۴ بیت fraction و سه بیت برای exponent در نظر بگیرید و پیاده سازی را مطابق جزوه‌ای که در مودل قرار داده شده، دنبال کنید.

۲- بعد از تست پروژه در محیطی مثل Modelsim طراحی مورد نظر خود را روی بورد FPGA پیاده‌سازی کرده و آن را به صورت کامل تست کنید.

توجه

I. در واقع سه پروژه ی نوروز، واحد محاسبات و منطق و پروژه ی پایانی در یک راستا و برای الحاق به هم طراحی شده‌اند بنابراین در جهت پیاده‌سازی بهینه و منطقی این پروژه بکوشید. (و اگر پروژه ی نوروز شما کامل نیست نگران نباشید چون در نهایت کلیدی برای استفاده از این پروژه در اختیار شما قرار خواهد گرفت)

II. زمان تحویل حضوری پروژه متعاقبا اعلام خواهد شد.

III. هر چند شکل پیاده‌سازی در ماژول‌هایی که الگوریتم آن صراحتا ذکر نشده مطابق خواست شماست، اما شیوه‌ی ماژول-بندی کلی باید مطابق طرحی که در صورت پروژه توضیح داده شده است باشد.

IV. در کد شما به هیچ عنوان نباید از امکانات سطح بالا مثل جمع و تفریق عادی (C<= A-B) استفاده شده باشد و پیاده سازی سطح گیت تمامی مراحل اجباری است (طبیعتا در این شرایط کد شما سنتز پذیر هم خواهد بود).

V. برای پروژه باید گزارشی تدارک ببینید (نه کوتاه‌تر از نیمی از تعریف پروژه) و در آن نحوه‌ی کلی پیاده‌سازی خود و نام‌هایی که برای ورودی‌ها و خروجی‌های ماژول‌های خود در نظر گرفتید را ذکر کنید. توجه کنید که گزارش شما حتماً باید تایپ شده باشد و در آن به ذکر تاخیر و مساحت هر یک از ماژول ها مطابق منطق پیاده سازی خود و ماژول کلی بپردازید (توجه کنید که در این قسمت تاخیر و مساحت شما باید مطابق طرح خودتان باشد)

VI. برای تست پروژه نیز باید در نهایت یک فضای آزمون[۱۶] ساده طراحی کنید و یکبار هر یک از ۸ عملیات گفته شده را تست کنید (توجه کنید که در تحویل حضوری به این قسمت نیز نمره تعلق خواهد گرفت).

VII. تمام مراحل پروژه باید به صورت انفرادی باشد و در صورت مشاهده‌ی هر گونه مشابهت غیر عادی، بعد از تحویل حضوری و بررسی کدها نمره‌ی همه‌ی طرفین درگیر بدون هیچ اغماضی **صفر** در نظر گرفته خواهد شد!

VIII. برای شبیه‌سازی می‌توانید از نرم افزارهایی مثل Modelsim, Quartus, Xelinx ISE, ... استفاده نمایید ولی ترجیح بر Modelsim است.

IX. از قرار دادن هر گونه فایل دست نوشته در گزارش خودداری کنید.

X. در پایان همه‌ی فایل های مربوطه را در یک فایل zip با قالب زیر ارسال کنید:
Farzan_dehbashi_9231038.zip

موفق باشید

[۱۶] Test bench