**گزارش دستورکار یازدهم آزمایشگاه معماری کامپیوتر**

نگار موقتیان، 9831062

**ماژول floating\_point\_adder**

در این آزمایش می­خواهیم یک مدار برای جمع اعداد اعشاری با نمایش single precision طراحی کنیم. قالب این اعداد به صورت زیر می­باشد:



به عبارتی علامت عدد با چپ­ترین بیت مشخص می­شود، پس از آن نمای عدد اعشاری در قالب مکمل دو افزونه بایاس 1 و در 8 بیت مشخص می­شود. در نهایت نیز قسمت اعشاری عدد مشخص می­شود. طبق این قالب برای هر عدد هنجار شده مانند A داریم:

تنها عددی که از این قالب پیروی نمی­کند عدد صفر است زیرا نمی­توان آن را به شکل هنجار شده نوشت. برای حل این مشکل کوچک­ترین عدد قابل نمایش (اپسیلون) را برابر با صفر در نظر می­گیریم. این عدد نمایشی تمام صفر دارد و در برنامه با سیگنالی به نام zero مشخص شده­است.

حال با توجه به این نکات الگوریتمی که این جمع کننده باید طبق آن عمل کند را بررسی می­کنیم.

1. در ابتدا سه حالت خاص را که عدد صفر در آن دخیل است در نظر می­گیریم:
2. عدد A برابر با صفر باشد: در این صورت حاصل جمع خروجی برابر با B خواهد بود.
3. عدد B برابر با صفر باشد: در این صورت حاصل جمع خروجی برابر با A خواهد بود.
4. اعداد A و B قرینۀ یکدیگر باشند (جز در بیت اول، تمامی بیت­های آن­ها برابر باشد): در این صورت حاصل جمع خروجی برابر با صفر خواهد بود.

و اگر هیچ یک از این حالات پیش نیامد جمع دو عدد را طبق ادامۀ الگوریتم انجام می­دهیم.

1. ابتدا نمای اعداد را یکی می­کنیم. برای این کار با استفاده از جمع کنندۀ مکمل گیر نماها را از یکدیگر کم می­کنیم تا از این طریق نمای بزرگ­تر را پیدا کرده و اختلاف نماها را نیز بیابیم. اگر حاصل دارای رقم نقلی بود یعنی نمای عدد اول از عدد دوم بزرگ­تر است. بنابراین نمای عدد کوچک­تر را به عدد بزرگ­تر می­رسانیم و قسمت اعشار آن را به اندازۀ اختلاف نماها به سمت راست شیفت می­دهیم. اگر حاصل دارای رقم نقلی نبود نمای عدد دوم از عدد اول بزرگ­تر است و اینبار نمای عدد اول را به عدد دوم می­رسانیم.
2. حال بخش­های اعشاری قابل جمع کردن با یکدیگر هستند. از آن­جایی که اعداد علامت­دار هستند طبق الگوریتم جمع اندازه-علامت دو عدد را جمع می­کنیم (اگر علامت­ها یکسان بود اعداد را جمع کرده و علامت آن­ها را به عنوان علامت می­گذاریم، در غیر این صورت عدد کوچک­تر را از عدد بزرگ­تر کم کرده و علامت عدد بزرگ­تر را به عنوان علامت می­گذاریم).
3. در نهایت با چک کردن حالات مختلف حاصل مرحلۀ 4، عدد بدست آمده را هنجار می­کنیم.

با توجه به این توضیحات جمع­کنندۀ مورد نظر را به صورت Behavioral پیاده سازی می­کنیم. سپس در  
test bench مربوطه، با دادن مقادیر مختلف به ورودی­های A و B رفتار خروجی را بررسی می­کنیم.

در test bench نوشته شده به ترتیب مقادیر زیر به ورودی­های A و B داده شده­اند و انتظار خروجی­های زیر را داریم:

1. A = 0 00000000 00000000000000000000000

B = 0 10000011 01001000000000000000000

S = 0 10000011 01001000000000000000000 (طبق حالت خاص اول)

1. A = 1 10000001 01100000000000000000000

B = 0 10000001 01100000000000000000000

S = 0 00000000 00000000000000000000000 (طبق حالت خاص سوم)

1. A = 0 10000010 11100000000000000000000

Sign = +1, exponent = 2, fraction = 0.111 🡪 A = + (22 × 1.111) = + (111.1) = +7.5

B = 0 10000011 01001000000000000000000

Sign = +1, exponent = 3, fraction = 0.01001 🡪 A = + (23 × 1.01001) = + (1010. 01) = +10.25

S = 0 10000100 00011100000000000000000

A + B = +17.75 = 10001.11 = 1.000111 × 24

1. A = 1 10000001 01100000000000000000000

Sign = -1, exponent = 1, fraction = 0.011 🡪 A = -(21 × 1.011) = -(10.11) = -2.75

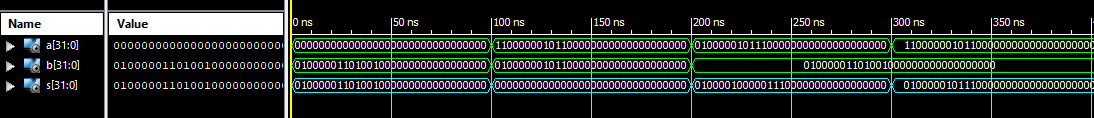
B = 0 10000011 01001000000000000000000

Sign = +1, exponent = 3, fraction = 0.01001 🡪 A = + (23 × 1.01001) = + (1010. 01) = +10.25

S = 0 10000010 11100000000000000000000

A + B = +7.5 = 111.1 = 1.111 × 22

و شکل سیگنال­های ورودی و خروجی شبیه سازی مدار با توجه به این مقادیر مطابق شکل زیر است.



که با داده­های بالا مطابقت کامل دارد.