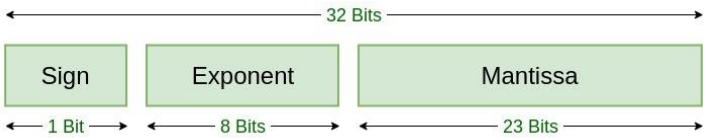
گزارش دستورکار یازدهم آزمایشگاه معماری کامپیوتر

نگار موقتیان، ۹۸۳۱۰۶۲

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 ۱ و در ۸ بیت مشخص می‌شود. در نهایت نیز قسمت اعشاری عدد مشخص می‌شود. طبق این قالب برای هر عدد هنجار شده مانند A داریم:

$$A = (-1)^{Sign} \times 2^{Exponent} \times (1.Mantissa)$$

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.

۱- در ابتدا سه حالت خاص را که عدد صفر در آن دخیل است در نظر می‌گیریم:
   ۱. عدد A برابر با صفر باشد: در این صورت حاصل جمع خروجی برابر با B خواهد بود.
   ۲. عدد B برابر با صفر باشد: در این صورت حاصل جمع خروجی برابر با A خواهد بود.
   ۳. اعداد A و B قرینهٔ 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 ادامهٔ الگوریتم انجام می‌دهیم.

۲- ابتدا نمای اعداد را یکی می‌کنیم. برای این کار با استفاده از جمع کنندهٔ 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 اندازهٔ 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.

۳- 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).

۴- در نهایت با چک کردن حالات مختلف حاصل مرحلهٔ ۴، عدد بدست آمده را هنجار می‌کنیم.

با توجه به این توضیحات جمع‌کنندهٔ 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)

2. A = 1 10000001 01100000000000000000000
   B = 0 10000001 01100000000000000000000
   S = 0 00000000 00000000000000000000000 (طبق حالت خاص سوم)

3. A = 0 10000010 11100000000000000000000
   Sign = +1, exponent = 2, fraction = 0.111 → A = + ($2^2 \times 1.111$) = + (111.1) = +7.5
   B = 0 10000011 01001000000000000000000
   Sign = +1, exponent = 3, fraction = 0.01001 → A = + ($2^3 \times 1.01001$) = + (1010. 01) = +10.25
   S = 0 10000100 00011100000000000000000
   A + B = +17.75 = 10001.11 = 1.000111×2^4

4. A = 1 10000001 01100000000000000000000

Sign = -1, exponent = 1, fraction = 0.011 → A = $-(2^1 \times 1.011) = -(10.11) = -2.75$

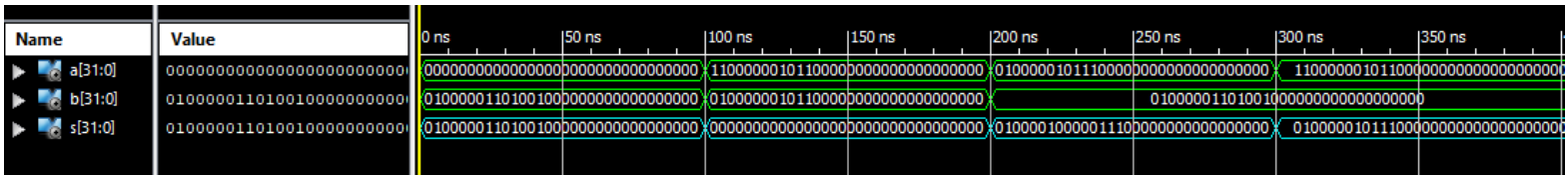
B = 0 10000011 01001000000000000000000

Sign = +1, exponent = 3, fraction = 0.01001 → A = $+(2^3 \times 1.01001) = +(1010.01) = +10.25$

S = 0 10000010 11100000000000000000000

A + B = +7.5 = 111.1 = 1.111×2^2

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.