فربد فولادی-98243045

عرفان رفیعی-98243027

مقدمه

هدف این پروژه پیاده سازی الگوریتم Restoring Division Algorithm با استفاده از زبان GEZEL و در انتها تبدیل آن به زبان VHDL است.

توضیح الگوریتم

این الگوریتم تقسیم از دسته الگوریتم های کند به شمار می رود. در ادامه روند اجرای الگوریتم توضیح داده شده است.

1. رجیستر ها را مقداردهی اولیه می کنیم.)n هم برابر با تعداد بیت های مقسوم است.(
2. محتویات رجیسترهای A و Q را شیفت به چپ می دهیم، به گونه ای که انگار این دو رجیستر با هم یک عدد هستند.
3. مقدار رجیستر M را از مقدار رجیستر A کم میکنیم و در رجیستر A می ریزیم.
4. پرارزشترین بیت A چک میشود. اگه صفر باشد، کم ارزشترین بیت Q را برابر با یک قرار میدهیم. در غیر این صورت مقدار Q را برابر با صفر قرار داده و مقدار A را به مقدارش قبل از انجامِ منها برمیگردانیم.
5. مقدار n را کم می کنیم.
6. اگر مقدار n صفر بود الگوریتم به پایان رسیده است. خارج قسمت در رجیستر Q و باقیمانده در رجیسترِ A خواهد بود. در غیر این صورت به مرحله ی ۲ برمیگردیم.

پیاده سازی

برای پیاده سازی با استفاده از GEZEL یک FSMD پیاده سازی کردیم و همچنین برای برنامه یک test bench نیز نوشتیم تا صحت آن را بررسی کنیم.

منطق پیاده سازی کد

حالات FSM تقسیم در ادامه توضیح داده شده است.

S0 <– حالت ابتدایی است. قسمت idle از data path تقسیم را اجرا می کند که مقدار رجیستر reg_start را برابر با ورودی start قرار میدهد و مقدار done را نیز صفر میکند. بعد از اجرای این حالت به حالتِ s1 میرویم.

S1 <– اگر مقدار رجیستر reg_start برابر با یک بود به حالت s2 میرویم. در غیر این صورت به s0 برمیگردیم.

S2 <– رجیسترِ مربوط به باقیمانده(r_reg) را شیفت داده و به حالت s3 میرویم.

S3 <– رجیسترِ مربوط به مقسوم (q_reg) را شیفت داده و به حالتِs4 میرویم.

S4 <– اینجا باید مقدار r_reg_m_reg را در r_reg بریزیم. چک میکنیم که کدام یک از این دو رجیستر بزرگتر هستند .رجیستر کوچکتر را منهای رجیستر بزرگتر کرده و در صورت لزوم حاصل را منفی میکنیم. در انتها به حالت s5 میرویم.

S5 <– مقدار پرارزشترین بیت r_reg را چک میکنیم و بسته به این که مقدارش صفر است یا یک، به کم ارزشترین بیت q_reg مقدار میدهیم. سپس به حالت s6 میرویم.

S6 <– مقدار n_reg را یه واحد کم میکنیم و به s7 میرویم.

S7 <– اگر مقدار n_reg به صفر رسیده بود یعنی محاسبات به اتمام رسیده است. قسمت final را از data path اجرا میکنیم تا جواب ها روی صفحه چاپ شوند و به حالت s0 برمیگردیم. اگر مقدار n_reg صفر نبود کار خاصی انجام نمی دهیم و به حالت s2 برمیگردیم.

test bench

برای تست کد هم مطابق با کد ارائه شده در صورت تکلیف عمل کردیم.