گزارشِ تکلیفِ چهارمِ هم‌طرّاحی

روزبه صیّادی - امیرعلی منجر

۱ مقدمه

هدفِ این پروژه پیاده‌سازیِ الگوریتمِ «تقسیم به روشِ ترمیمی» یا به زبانِ انگلیسی «Restoring division algorithm» با استفاده از زبانِ Gezel، و در انتها تبدیلِ کد به زبانِ VHDL است. Gezel یک زبان برای توصیفِ سخت‌افزار است که اجازه‌ی پیاده‌سازیِ مدلِ FSM + Datapath را می‌دهد. ابزارهایی برای شبیه‌سازی، هم‌شبیه‌سازی و همچنین کامپایلِ برنامه‌های این زبان به VHDLوجود دارند.

۲ توضیحِ الگوریتم

الگوریتمِ «تقسیم به روشِ ترمیمی» جزوِ دسته‌ی الگوریتم‌های کند به شمار می‌رود. روندِ اجرای این الگوریتم در جدولِ ۱ توضیح داده شده است.

مرحله	توضیح
۱	رجیسترها را مقداردهیِ اوّلیّه می‌کنیم. Q = مقسوم، M = مقسوم‌علیه، A = صفر و n هم برابر با تعدادِ بیت‌های مقسوم است.
۲	محتویات رجیسترهای A و Q را شیفت به چپ می‌دهیم، به گونه‌ای که انگار این دو رجیستر با هم یک عدد هستند.
۳	مقدار رجیستر M از رجیستر A کم شده و نتیجه در رجیستر A ریخته می‌شود.
۴	پرارزش‌ترین بیت A چک می‌شود. اگه صفر باشد، کم‌ارزش‌ترین 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.
۵	مقدار n را کم می‌کنیم.
۶	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 مرحله‌ی ۲ برمی‌گردیم.

جدولِ ۱: روندِ اجرای الگوریتمِ «تقسیم به روشِ ترمیمی»

۳ پیاده‌سازی

برای پیاده‌سازیِ این الگوریتم، با استفاده از زبانِ Gezel یک FSMD پیاده‌سازی کردیم. همچنین برای برنامه یک test benchنیز نوشتیم که در ادامه هردوی این‌ها توضیح داده شده‌اند.

۳,۱ منطقِ کد

ماشین حالت متناهی[۱] مربوط به تقسیم شاملِ چند حالت است که آن‌ها را توضیح می‌دهیم.

s0

حالتِ ابتدایی. قسمت idle **از داده‌مسیر[۲] تقسیم را اجرا می‌کند (که مقدار رجیستر** start_reg **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 start_reg **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 حالتِ** s3**می‌رویم.**

s3

رجیسترِ مربوط به مقسوم (q_reg**) را شیفت داده و به حالتِ** s4**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. رجیسترِ کوچک‌تر را منهای رجیسترِ بزرگ‌تر کرده و در صورتِ لزوم حاصل را منفی می‌کنیم. در انتها به حالت** s5 **می‌رویم.**

s5

مقدارِ پرارزش‌ترین بیتِ r_reg **را چک می‌کنیم و بسته به این که مقدارش صفر است یا یک، به کم‌ارزش‌ترین بیتِ** q_reg**مقدار می‌دهیم. سپس به حالتِ** s6 **می‌رویم.**

s6

مقدار n_reg **را یه واحد کم می‌کنیم و به** s7 **می‌رویم.**

---

[۱] Finite state machine

[۲] Datapath

s7

اگر مقدار n_reg به صفر رسیده بود یعنی محاسبات به اتمام رسیده است. قسمتِ final را از داده‌مسیر اجرا می‌کنیم تا جواب‌ها روی صفحه چاپ شوند و به حالتِ s0 برمی‌گردیم. اگر مقدار n_reg صفر نبود کار خاصی انجام نمی‌دهیم و به حالتِ s2 برمی‌گردیم.

۳,۲ تستِ کد

برای تستِ کد مطابقِ کدِ ارائه شده در تکلیف عمل کردیم.

۴ نتیجه‌گیری

ما در این تکلیف با نحوه‌ی نوشتنِ یک برنامه‌ی ساده با Gezel، بایدها و نبایدهای Gezel هنگامِ شبیه‌سازی و همچنین طرزِ کار با ابزارهای این زبان را یاد گرفتیم. همچین با الگوریتمِ «تقسیم به روشِ ترمیمی» آشنا شدیم.