گزارش تمرین 1 و 2 طراحی کامپیوتری سیستم های دیجیتال

محمد فرهی 810198451

امیرمحمد خسروی 810198386

تابع فیبوناچی در وریلاگ براساس طراحی دیتاپس و کنترلر که در تمرین یک انجام شد، (بدون تغییر) پیاده سازی شد و اطلاعات مربوط به تمرین اول و این طراحی در پوشه doc و فایل HW1-part1 موجود است.

در پیاده سازی این تابع دو فایل fib_datapath و fib_controller داریم که نحوه پیاده سازی و اطلاعات مربوط به fib_cotroller به طور کامل در فایل HW1-part1 ذکر شده است. همچنین برای بخش fib_datapath از ماژول هایی استفاده شده که در ادامه به طور خلاصه هر کدام توضیح داده شده است:

- رجیستر های N_reg, Return_reg, arg_reg که هر کدام 128 بیتی هستند و به ترتیب برای ذخیره آرگومان فراخوانی، مقدار بازگشتی تابع در مراحل مختلف و مقدار ورودی N استفاده شده اند.
- شیفتر right_shifter که مقدار ورودی را یک بیت به سمت راست شیفت میدهد یا به عبارتی بر دو تقسیم میکند.
- ضرب کننده multiplier که به عنوان ورودی دو عدد 128 بیتی میگیرد و خروجی 256 بیتی دارد (128 بیت LSB خروجی مورد نیاز ماست)
- جمع کننده adder که دو عدد 128 بیتی ورودی گرفته و جمع 128 بیتی آنها را خروجی میدهد.
- تفریق کننده subtracter که دو عدد 128 بیتی ورودی گرفته و تفریق آنها را خروجی میدهد.
- مقایسه کننده comparator که دو ورودی 128 بیتی میگیرد و اگر ورودی اول بزرگتر باشد 1 و در غیر این صورت 0 را در خروجی قرار میدهد.
- مالتی پلکسرهای دو به یک mux1, mux2, mux_return, mux_comp که ورودی و خروجی 128 بیتی دارند و به ترتیب ورودی های ماژول های مقایسه کننده، رجیستر Return_reg و مالتی پلکسر 4 به 1 mux_sub را تعیین میکنند.
- مالتی پلکسرهای سه به یک mux_d, mux_c که به ترتیب ورودی 2 و 128 بیتی دارند. مالتی پلکسر اول برای ست کردن شماره فراخوانی (اینکه اول است یا دوم) و دیگری برای تعیین ورودی استک استفاده میشود.
- مالتی پلکسرهای چهار به یک mux_sub, mux_arg که ورودی 128 بیتی دارند و به ترتیب برای تعیین ورودی رجیستر arg_reg و برای ورودی تفریق کننده استفاده میشود.
- ماژول استک که در سطح رفتاری پیاده سازی شده است و در هر لبه بالا رونده کلاک مقادیر pop, tos, push بررسی میشوند و در صورت یک شدن هر کدام، عملیات لازم بر روی top_of_stack (که عددی 16 بیتی است و نشان دهنده روی استک یا آدرس خانه حافظه آخرین عدد push شده در استک است) و mem (حافظه استک) انجام میشود. همچنین در یک always بررسی میشود که در صورت تغییر مقدار top_of_stack ، اگر مقدار آن برابر 0 شد سیگنال empty یک میشود که نشان دهنده خالی شدن استک است.

استیت های کنترلر نیز به صورت زیر هستند:

- Idle : استیت شروع
- load : لود ورودی N و ست کردن اولین آرگومان
- call : فراخوانی تابع با آرگومانی که در استک پوش میشود
- wait : صبر برای مقایسه شدن N/2 با مقدار آرگومان تابع
- Recursive1 : پوش کردن {c=1, n} به عنوان پرچم فراخوانی بازگشتی اول و ذخیره n-1 در arg_reg
- base : پاپ کردن آرگومان صفر یا 1 (حالت پایه تابع) و 1 کردن مقدار موجود در Return_reg
- return : return کردن از تابع (مقدار بازگشتی در Return_reg است) و پاپ کردن پرچمی که آن تابع را صدا زده است.
- decide : تعیین کردن اینکه اخرین فراخوانی بازگشتی چه نوعی بوده است
- poping : ذخیره مقدار برگرداننده شده تابع بازگشتی اول ((n-1)fib(n-1) یا (n-2)fib(n-1)) در arg_reg برای آماده سازی برای جمع با مقدار برگردانده شده تابع بازگشتی دوم
- mult2 : محاسبه مقدار نهایی برگشتی تابع و پاپ کردن کامل فریم تابع از استک
- mult1 : انجام ضرب و ذخیره مقدار برگشتی تابع اول در Return_reg
- Recursive2 : پوش کردن {c=2, n} در استک و قرار دادن n-2 در arg_reg برای فراخوانی دوم

خروجی نهایی نیز به صورت زیر است. برنامه برای ورودی های 4، 5، 6 و 7 تست شده است: