به نام خدا

فرزان رحمانی ۹۹۵۲۱۲۷۱

گزارش تمرین سری دوم طراحی کامپیوتری سیستم های دیجیتال

سوال اول

توضیح کد سوال:
کد فوق یک ماژول به نام "sorter" را تعریف می‌کند که برای مرتب‌سازی یک آرایه از اعداد ۸ بیتی استفاده می‌شود. ماژول دارای پورت ورودی با نام "input" و نوع داده "array_of_8bit" و پورت خروجی با نام "output" و نوع داده "array_of_8bit" است.

در آرشیتکتور ماژول "sorter" یک بلاک فرآیند (process) وجود دارد که ورودی را دریافت می‌کند. در این بلاک فرآیند، دو متغیر محلی به نام "tmp" و "result" تعریف شده‌اند. ابتدا مقدار ورودی به متغیر "result" انتساب داده می‌شود.

سپس با استفاده از حلقه های تودرتو، برای هر عضو i و j در بازه ۰ تا ۹، اعداد موجود در آرایه "result" مقایسه می‌شوند. اگر عضو i کوچکتر یا مساوی عضو j باشد، این دو عضو با یکدیگر جابجا می‌شوند. این عملیات به صورت متوالی انجام می‌شود تا زمانی که آرایه به صورت صعودی مرتب شود.

در نهایت، آرایه "result" به پورت خروجی "output" انتساب داده می‌شود تا آرایه مرتب شده به بیرون منتقل شود.

توضیح کد تست:
کد فوق برای تست و اعتبارسنجی ماژول sorter طراحی شده است. این کد شبیه‌سازی را برای ماژول sorter انجام می‌دهد تا مشخص شود آیا ماژول درست عمل می‌کند یا خیر.

در ابتدا، کتابخانه HW2 و پکیج types.all از همان کتابخانه استفاده شده است. این کتابخانه و پکیج‌ها می‌توانند تعریف‌ها و نوع داده‌های خاصی را شامل شوند که در این کد به کار رفته است.

سپس، ماژول sorter_tb تعریف شده است که به عنوان ماژول تست استفاده می‌شود.

در بخش معماریTB_ARCHITECTURE ، ابتدا کامپوننت sorter تعریف شده است که به عنوان واحدی که برای تست استفاده می‌شود، به داخل تست بند اضافه می‌شود. این کامپوننت دارای دو پورت ورودی به نام input و خروجی به نام output است.

سیگنال‌های stimulus و output تعریف شده‌اند که به ترتیب به پورت ورودی و خروجی ماژول sorter متصل می‌شوند.

در بلاکprocess ، سیگنال input با مقادیر تصادفی پر می‌شود و سپس منتظر می‌ماند. سپس، مقادیر دیگری به سیگنال input انتساب داده می‌شود و دوباره منتظر می‌ماند. در اینجا می‌توانید مقادیر input را به عنوان الگوهای تست تغییر دهید.

در بخش انتهایی، ماژول sorter با استفاده از port map به سیگنال‌های stimulus و output متصل شده است.

تنظیمات نهایی TESTBENCH_FOR_sorter نیز تعیین می‌کند که ماژول sorter_tb با استفاده از ماژول sorter تعریف شده در کارتابانک استفاده شود.

در کل، کد فوق برای تست و اعتبارسنجی ماژول sorter طراحی شده است و مقادیر stimulus به ماژول داده می‌شود تا خروجی را تولید کند و بررسی شود که آیا ماژول درست عمل می‌کند یا خیر.

سوال دوم

توضیح کد سوال:

کد فوق برای ماژول root_picker طراحی شده است. این ماژول یک عدد ۱۰ بیتی را به عنوان ورودی دریافت می‌کند و جذر مربع این عدد را محاسبه می‌کند. سپس خروجی محاسبه شده را برگردانده می‌کند.

در ابتدا، کتابخانه IEEE و پکیج‌های مورد نیاز استفاده شده از آنها فراخوانی می‌شوند. همچنین پکیج types.all از کتابخانه work نیز فراخوانی شده است که می‌تواند تعریف‌ها و نوع داده‌های خاصی را شامل شود.

سپس، ماژول root_picker تعریف شده است که علاوه بر پورت‌های ورودی و خروجی، شامل یک فرآیند است.

در بخش معماریBehavioral ، فرآیندی تعریف شده است که وابسته به سیگنال ورودی input است. در این فرآیند، متغیرهای x و resultتعریف شده‌اند. متغیر x برابر با مقدار صحیحی است که از سیگنال input به دست می‌آید. متغیر result نیز یک std_logic_vectorبا طول ٦ بیتی است که از ابتدا به صفر مقداردهی شده است.

سپس، متغیر y نیز به صورت صحیح تعریف شده است. یک حلقه while تعریف شده است که تا زمانی که مقدار y به توان دوم کوچکتر از x باشد، به طور متوالی مقدار y را افزایش می‌دهد. در واقع، این حلقه به دنبال مقدار کوچکترین y است که y به توان دوم بزرگتر مساوی x باشد.

در نهایت، مقدار y به عنوان نتیجه محاسبه جذر مربع به صورت unsigned تبدیل می‌شود و سپس با استفاده از تابع std_logic_vectorبه یک std_logic_vector با طول ٦ بیتی تبدیل می‌شود. این مقدار به عنوان خروجی به سیگنال output اختصاص داده می شود.

توضیح کد تست:

کد فوق یک تست بنچ برای ماژول root_picker است. این تست بنچ برای ارزیابی عملکرد ماژول root_picker طراحی شده است.

ابتدا، کتابخانه HW2 و پکیج types.all از آن کتابخانه فراخوانی می‌شوند. همچنین کتابخانه IEEE و پکیج‌های مورد نیاز استفاده شده از آن فراخوانی می‌شوند.

سپس، ماژول root_picker_tb تعریف شده است که شامل معماری TB_ARCHITECTURE است.

در بخش معماریTB_ARCHITECTURE ، ابتدا ماژول root_picker تعریف شده است که شامل پورت‌های ورودی و خروجی است. سیگنال‌های ورودی و خروجی نیز به شکل STD_LOGIC_VECTOR تعریف شده‌اند.

سپس، ماژول root_picker به عنوان UUT (Unit Under Test) با استفاده از دستور port map به سیگنال‌های ورودی و خروجی متصل می‌شود.

در بخشprocess ، یک فرآیند تعریف شده است که به توالی مقادیر مختلفی را به سیگنال ورودی input اختصاص می‌دهد. بین هر اختصاص مقدار، با استفاده از دستورwait for ، مدت زمان ۱۰۰ nsصبر می‌کند.

در نهایت، تنظیمات تست بنچ تعریف شده است. ماژول root_picker به عنوان UUT تعریف شده است و entity work.root_pickerبا معماری behavioral برای آن استفاده شده است.

با اجرای این تست بنچ، مقادیر مختلف به عنوان ورودی به ماژول root_picker اختصاص داده می‌شوند و خروجی محاسبه شده توسط ماژول را مشاهده می‌کنیم. این تست بنچ برای ارزیابی صحت و عملکرد ماژول root_picker مفید است.

سوال سوم

توضیح کد سوال:

کد فوق یک ماژول با نام one_by_one را پیاده‌سازی می‌کند. این ماژول ورودی یک بردار بیتی به نام input با طول ۱۶ دارد و خروجی یک بردار بیتی به نام output با طول ۲ را تولید می‌کند.

در معماری one_by_one ، یک فرآیند (process) تعریف شده است که وابسته به تغییرات در ورودی input فعال می‌شود. این فرآیند شامل تعریف دو متغیر از نوع integer به نام‌های even_idx و odd_idx است.

سپس، یک حلقه (for loop) تعریف شده است که برای هر عضو در ورودی input اجرا می‌شود. در هر مرحله، بررسی می‌شود که آیا این عضو در جایگاه فردی یا زوجی قرار دارد. اگر جایگاه زوجی باشد، بررسی می‌شود که آیا مقدار آن برابر با ۱ است یا خیر، و در صورت برقراری شرط، مقدار متغیر even_idx یک واحد افزایش می‌یابد. اگر جایگاه فردی باشد، همین بررسی برای مقدار odd_idx انجام می‌شود.

سپس، با استفاده از دو عبارت if-else ، بررسی می‌شود که آیا مقدار even_idx به‌ازای ورودی‌های زوجی برابر با ۳ بخش‌پذیر است و آیا مقدار odd_idx به‌ازای ورودی‌های فردی برابر با ۵ بخش‌پذیر است. بسته به نتیجه بررسی، مقدار مناسب به خروجی output اختصاص داده می‌شود.

بنابراین، ماژول one_by_one به‌ازای ورودی‌های ۱۶ بیتی، با تحلیل جایگاه فردی و زوجی بیت‌ها، مقادیر دو بیت خروجی را تولید می‌کند که نشان‌دهنده بخش‌پذیری even_idx بر ۳ و odd_idx بر ۵ است.

توضیح کد تست:

کد فوق یک تست بنچ برای ماژول one_by_one را پیاده‌سازی می‌کند. این تست بنچ شامل ماژول تستی با نام one_by_one_tb و معماری TB_ARCHITECTURE است که برای تست واحد one_by_one ایجاد شده است.

در این تست بنچ، ابتدا یک ماژول one_by_one با استفاده از اتصالات port map به ماژول تستی UUT متصل می‌شود. سپس، سیگنال‌های stimulus و observed که به ترتیب با ورودی‌ها و خروجی‌های ماژول one_by_one مرتبط هستند، تعریف می‌شوند.

سپس، یک فرآیند process تعریف شده است که شامل چندین دستور wait است. این دستورها با فاصله زمانی ۱۰۰ ns صبر می‌کنند و سپس ورودی input ماژول one_by_one را با مقادیر مشخص شده تغییر می‌دهند. هر یک از مقادیر ورودی برای مدت ۱۰۰ ns در ورودی قرار می‌گیرند.

در نهایت، این تست بنچ با استفاده از تنظیمات configuration ، ماژول تستی UUT را با ماژول one_by_one متصل می‌کند و تنظیمات لازم را برای اجرای تست تعیین می‌کند.

بنابراین، این تست بنچ با اعمال مقادیر مشخص برای ورودی‌های ماژول one_by_one و ثبت خروجی‌های متناظر، عملکرد و صحت عملیات ماژول one_by_one را بررسی می‌کند.

سوال چهارم

توضیح کد سوال:

کد فوق یک ماژول با نام "Alarm_Clock" را پیاده‌سازی می‌کند که یک ساعت هشدار دار را نمایش می‌دهد. این ماژول شامل ورودی‌های) "clk" سیگنال ساعت(،) "reset"ریست کردن ساعت(،) "clock_set"تنظیم زمان ساعت(،) "alarm_set"تنظیم زمان هشدار(، "alarm_stop"(توقف هشدار) و "input_time" (زمان ورودی) است. خروجی ماژول نیز "on_alarm" است که وضعیت هشدار را نمایش می‌دهد.

در معماری Behavioral این ماژول، یک فرآیند با ورودی "clk" تعریف شده است. در این فرآیند، متغیرهای محلی نظیر "current_time"(زمان فعلی) و "alarm_time" (زمان هشدار) تعریف شده‌اند. همچنین، متغیر "second" برای نگه‌داری مقدار ثانیه فعلی تعریف شده است.

در هر لبه صعودی سیگنال"clk" ، مقدار متغیر "second" افزایش می‌یابد. اگر "second" برابر با ۵۹ شود، به صفر تنظیم می‌شود. سپس، در صورتی که "second" برابر با صفر باشد، زمان فعلی به درستی بروزرسانی می‌شود. ابتدا بررسی می‌شود که آیا مقدار آخرین چهار بیت زمان فعلی (مربوط به دقیقه) برابر با ۹ است یا نه. اگر برابر با ۹ باشد، این چهار بیت به صفر تنظیم می‌شوند و سپس بررسی می‌شود که آیا مقدار بیت‌های بعدی (مربوط به ساعت) نیز باید بروزرسانی شوند یا خیر. در نهایت، مقدار بیت‌های زمان فعلی به درستی بروزرسانی می‌شوند.

همچنین، در صورتی که ورودی "reset" برابر با ۱ باشد، زمان فعلی و متغیر "second" صفر می‌شوند و ساعت به حالت اولیه برگردانده می‌شود.

همچنین، اگر ورودی "clock_set" برابر با ۱ باشد، زمان فعلی با مقدار ورودی "input_time" برابر قرار می‌گیرد و متغیر "second"صفر می‌شود. همچنین، اگر ورودی "alarm_set" برابر با ۱ باشد، زمان هشدار با مقدار ورودی "input_time" برابر قرار می‌گیرد.

در نهایت، بررسی می‌شود که آیا زمان فعلی با زمان هشدار برابر است یا نه. اگر برابر باشند، خروجی "on_alarm" به ۱ تنظیم می‌شود و پس از ۱ نانوثانیه، هشدار فعال می‌شود. همچنین، اگر ورودی "alarm_stop" برابر با ۱ باشد، خروجی "on_alarm" به ۰ تنظیم می‌شود و پس از ۱ نانوثانیه، هشدار غیرفعال می‌شود.

بنابراین، این ماژول یک ساعت هشدار دار را با استفاده از سیگنال ساعت و ورودی‌های تنظیم زمان و هشدار پیاده‌سازی می‌کند و با بروزرسانی زمان فعلی و بررسی تطابق آن با زمان هشدار، وضعیت هشدار را نمایش می‌دهد.

توضیح کد تست:

کد فوق یک بنچ‌تست برای واحد "alarm_clock" را پیاده‌سازی می‌کند. این بنچ‌تست شامل موجودیت‌های "alarm_clock_tb" (موجودیت بنچ‌تست) و "alarm_clock" (موجودیت تست شده) است.

موجودیت "alarm_clock_tb" شامل اعضایی نظیر"clk" ، "reset"، "clock_set"، "alarm_set"، "alarm_stop"و "input_time"است که به ترتیب ورودی‌های سیستم ساعت هشدار دار هستند. همچنین، عضو "on_alarm" نیز خروجی موجودیت است که وضعیت هشدار را نمایش می‌دهد.

در بنچ‌تست، ابتدا موجودیت "alarm_clock" با استفاده از "port map" ورودی‌های خود را به ورودی‌های بنچ‌تست متصل می‌کند. سپس، در یک فرآیند بنچ‌تست، مقادیر مورد نیاز برای تنظیم سیگنال‌های ورودی در طول زمان تعیین می‌شود. این مقادیر شامل تنظیمات مختلف ساعت هشدار دار برای تست صحت عملکرد آن است.

در این بنچ‌تست، در دو دوره زمانی مختلف، دو ساعت هشدار دار تست می‌شود. ابتدا زمان را به ۱۳:۲۱ تنظیم می‌کنیم و سپس هشدار را در ۱۳:۲۵ تنظیم می‌کنیم. سپس، هشدار فعال می‌شود و پس از یک مدت زمان، توسط سیگنال "alarm_stop" هشدار متوقف می‌شود. سپس، زمان را به حالت اولیه بازگردانده و سپس زمان را به ۱۹:۵۵ و هشدار را به ۲۱:۰۱ تنظیم می‌کنیم. مجدداً

هشدار فعال می‌شود و پس از یک مدت زمان، متوقف می‌شود. در نهایت، زمان را به ۰۱:۰۷ و هشدار را به ۰۱:۰۷ تنظیم می‌کنیم و مجدداً هشدار فعال می‌شود و پس از یک مدت زمان، متوقف می‌شود.

همچنین، در بخش دیگری از بنچ‌تست، سیگنال "clk" با استفاده از یک فرآیند توالی سازی شده است. این فرآیند باعث تولید سیگنالی متقاطع با دوره‌های زمانی یکسان می‌شود که به عنوان ساعت به موجودیت "alarm_clock" ارسال می‌شود.

به این ترتیب، با اجرای این بنچ‌تست، عملکرد موجودیت "alarm_clock" در شرایط مختلف تست و صحت عملکرد آن بررسی می‌شود.

پایان