توضیحات و مستندات

در این مطلب نحوه عملکرد دو ماژول "STACK_BASED_ALU" و "tb" را به طور کامل و مفصل توضیح می‌دهیم. سپس چندین نکته پایانی را که باید به آن‌ها توجه شود ذکر می‌کنیم.

ابتدا به سراغ ماژول "STACK_BASED_ALU" رفته و نحوه عملکرد آن را به صورت مختصر توضیح داده و سپس به سراغ ماژول "tb" می‌رویم:

- STACK_BASED_ALU:

این ALU با استفاده از پشته برای ذخیره عملوندها و انجام عملیاتی مانند جمع، ضرب، پوش و پاپ کردن استفاده می‌شود. در زیر توضیح دقیقی در مورد هر قسمت از کد آورده شده است:

1. پارامترها:

N : پهنای بیت داده ها است و به طور پیش فرض روی 8 تنظیم شده است.

STACK_SIZE: اندازه پشته است و به طور پیش فرض روی 1000 تنظیم شده است.

2. ورودی‌ها:

input_data: داده‌هایی که باید روی پشته پوش شوند.

opcode: کد عملیاتی که مشخص می کند کدام عملیات باید انجام شود.

clk: سیگنال ساعت.

reset: سیگنال را برای مقداردهی اولیه یا تنظیم مجدد ALU تنظیم کنید.

3. خروجی‌ها:

output_data: نتیجه عملیات یا داده های بیرون آمده از پشته.

overflow: پرچمی که نشان می دهد سرریزی در طول عملیات رخ داده است یا خیر.

4. رجیسترها و متغیرهای داخلی:

stack: آرایه ای که پشته را با تعداد عناصر STACK_SIZE، هر یک از n بیت نشان می دهد.

sp: پوینتر پشته، که به بالای فعلی پشته اشاره می کند.

temp_a و temp_b: رجیسترهای موقت برای نگهداری عملوندها برای عملیات.

temp_result: یک ثبات موقت با n+1 بیت برای ذخیره نتیجه عملیات و تشخیص سرریز.

5. بلاک always برای توالی منطقی:

وضعیت بازنشانی: اگر سیگنال reset بالا باشد، نشانگر پشته (sp)، output_data و سرریز به حالت اولیه خود بازنشانی می‌شوند.

کدهای عملیاتی (opcode):

b100'3 (افزودن): اگر حداقل دو عنصر وجود داشته باشد، دو عنصر بالای پشته را اضافه می کند. با مقایسه بیت های علامت نتیجه، سرریز را بررسی می کند.

b101'3 (ضرب): اگر حداقل دو عنصر وجود داشته باشد، دو عنصر بالای پشته را ضرب می کند. به روشی مشابه سرریز را بررسی می کند.

3'b110 (Push): در صورت وجود فضا، داده های ورودی را به پشته فشار می دهد.

3'b111 (Pop): اگر عنصر بالایی پشته خالی نباشد، آن را از پشته خارج می کند و output_data را روی این مقدار تنظیم می کند.

پیش‌فرض: بدون عملیات (NOP) - داده‌های خروجی و سرریز را روی حالت امپدانس بالا (bz'1) تنظیم می‌کند.

6. چندین نکته کلیدی:

6.1. نشانگر پشته (sp) برای مدیریت عملیات پشته استفاده می شود و از عملیات پوش و پاپ مناسب اطمینان حاصل می کند.

6.2. از ثبات temp_result با یک بیت اضافی برای تشخیص سرریز با مقایسه بیت‌های علامت نتیجه استفاده می‌کند.

6.3. منطق بازنشانی تضمین می کند که پشته و خروجی ها به درستی مقداردهی اولیه شده اند.

6.4. از امپدانس بالا (bz'1) برای نشان دادن عدم وجود داده معتبر در output_data و سرریز استفاده می کند.

- tb :

ماژول testbench ارائه شده، tb، برای آزمایش عملکرد ماژول STACK_BASED_ALU با عرض داده های مختلف طراحی شده است. در اینجا توضیح مفصلی در مورد تست بنچ آورده شده است:

1. رجیسترها و سیم‌ها:

clk and reset: سیگنال های ساعت و تنظیم مجدد برای ماژول های ALU.

opcode: کد عملیاتی که عملیاتی که باید انجام شود را مشخص می کند.

input_data1,input_data2,input_data3, input_data4: داده های ورودی برای ماژول های ALU با عرض بیت های مختلف.

output_data1,output_data2,output_data3,output_data4: داده‌های خروجی از ماژول‌های ALU با پهنای بیت‌های مختلف صادر می‌شود.

overflow1, overflow2, overflow3, overflow4: پرچم های سرریز از ماژول های ALU با عرض بیت های مختلف.

2. نمونه سازی ماژول های ALU:

نمونه‌سازی: چهار نمونه از ماژول STACK_BASED_ALU با پهنای داده های مختلف (4، 8، 16 و 32 بیت) و اندازه پشته 8 ایجاد می شود.

```
STACK_BASED_ALU #(.n(4) , .STACK_SIZE(8)) alu_4 (.input_data(input_data1), .opcode(opcode), .clk(clk), .reset(reset), .output_data(output_data1), .overflow(overflow1));

STACK_BASED_ALU #(.n(8) , .STACK_SIZE(8)) alu_8 (.input_data(input_data2), .opcode(opcode), .clk(clk), .reset(reset), .output_data(output_data2), .overflow(overflow2));

STACK_BASED_ALU #(.n(16), .STACK_SIZE(8)) alu_16 (.input_data(input_data3), .opcode(opcode), .clk(clk), .reset(reset), .output_data(output_data3), .overflow(overflow3));

STACK_BASED_ALU #(.n(32), .STACK_SIZE(8)) alu_32 (.input_data(input_data4), .opcode(opcode), .clk(clk), .reset(reset), .output_data(output_data4), .overflow(overflow4));
```

3. تولید پالس ساعت:

سیگنال ساعت: یک سیگنال ساعت با دوره زمانی 10 واحد زمانی (5 واحد زمانی بالا و 5 واحد زمانی پایین) تولید می کند.

```
always #5 clk = ~clk;
```

4. Initial Block:

بلوک اولیه دنباله ای از عملیات را برای آزمایش عملکرد ALU تعریف می کند:

clk را روی 0 تنظیم می کند و روی 1 بازنشانی می کند، سپس پس از 6 واحد زمانی بازنشانی می شود.
سپس یک سری عملیات پوش، جمع، ضرب و پاپ را با ورودی های داده مختلف انجام می دهد تا ALU ها را با پهنای بیت های مختلف آزمایش کند.

5. Monitor Block:

مانیتورینگ: وظیفه سیستم monitor$ برای نمایش مداوم وضعیت متغیرهای testbench در هر مرحله زمانی استفاده می شود. کد عملیاتی فعلی، وضعیت پشته‌ها برای هر نمونه ALU، پرچم‌های سرریز و داده‌های خروجی را در قالب‌های هگزادسیمال و اعشاری نشان می‌دهد.

6. چندین نکته کلیدی:

6.1. تست بنچ عملکرد ALU را با پهنای داده های مختلف (4، 8، 16 و 32 بیت) با استفاده از اندازه پشته ثابت 8 بررسی می کند.
6.2. سیگنال ساعت با یک دوره زمانی 10 واحد تولید می شود تا نمونه های ALU را هدایت کند.
6.3. سیگنال ALU reset ها را مقداردهی اولیه می کند و یک سری عملیات برای آزمایش عملکرد پوش، پاپ، جمع و ضرب انجام می شود.
6.4. وظیفه monitor$ یک نمای جامع از عملیات testbench ارائه می‌کند و ردیابی تغییرات و اشکال‌زدایی را آسان می‌کند.