گزارش پروژه شبیه‌سازی CPU

مقدمه

در این پروژه، یک پردازنده ساده (CPU) با استفاده از زبان Verilog پیاده‌سازی شده است. این پروژه شامل تمام اجزای اصلی یک پردازنده شامل واحد کنترل، مسیر داده، واحد محاسباتی و منطقی، حافظه و رجیسترها می‌باشد.

معماری کلی سیستم

پردازنده پیاده‌سازی شده از معماری RISC ساده با ۱۶ بیت داده و آدرس استفاده می‌کند. ساختار کلی شامل:

واحد کنترل (Control Unit)

مدیریت حالت‌های مختلف پردازنده (Fetch, Decode, Execute, Memory, Write Back)

تولید سیگنال‌های کنترلی برای تمام بخش‌ها

مدیریت دستورالعمل‌ها و رمزگشایی آنها

مسیر داده (Datapath)

اتصال تمام اجزای پردازنده

مدیریت جریان داده بین بخش‌های مختلف

کنترل سیگنال‌های ورودی و خروجی

واحد محاسباتی و منطقی (ALU)

عملیات جمع و تفریق با استفاده از CSA (Carry Select Adder)

عملیات ضرب با الگوریتم Karatsuba

عملیات تقسیم با الگوریتم تقسیم تکراری

جزئیات پیاده‌سازی

واحد کنترل (CU.v)

```
   // State encoding
   localparam FETCH = 3'd0, DECODE = 3'd1, EXEC = 3'd2, MEM = 3'd3, WB =
3'd4, HALT = 3'd5;
```

واحد کنترل از یک ماشین حالت محدود برای مدیریت چرخه دستورالعمل استفاده می‌کند:

Fetch: خواندن دستورالعمل از حافظه

Decode: رمزگشایی دستورالعمل و استخراج فیلدهای مختلف

Execute: اجرای عملیات محاسباتی

Memory: دسترسی به حافظه (برای دستورات Load/Store)

Write Back: نوشتن نتیجه در رجیستر

واحد محاسباتی و منطقی (ALU.v)

این واحد چهار عملیات اصلی را پشتیبانی می‌کند:

جمع (ADD): استفاده از CSA برای سرعت بالا

تفریق (SUB): استفاده از متمم دو

ضرب (MUL): الگوریتم Karatsuba برای اعداد ۱۶ بیتی

تقسیم (DIV): الگوریتم تقسیم تکراری

حافظه (M.v)

حافظه ۱۶ × ۶۴K بیت

قابلیت خواندن و نوشتن همزمان

آدرس‌دهی مستقیم

فایل رجیستر (RF.v)

۴ رجیستر ۱۶ بیتی

قابلیت خواندن دو رجیستر و نوشتن یک رجیستر در هر سیکل

الگوریتم‌های پیاده‌سازی شده

Carry Select Adder (CSA)

برای عملیات جمع و تفریق سریع استفاده می‌شود:

تقسیم اعداد ۱۶ بیتی به ۴ بخش ۴ بیتی

محاسبه موازی نتایج با Carry-in صفر و یک

انتخاب نتیجه نهایی بر اساس Carry خروجی

الگوریتم Karatsuba برای ضرب

برای ضرب اعداد ۱۶ بیتی:

تقسیم اعداد به بخش‌های ۸ بیتی

محاسبه سه ضرب کوچک‌تر

ترکیب نتایج برای تولید حاصل‌ضرب نهایی

الگوریتم تقسیم

برای عملیات تقسیم:

استفاده از روش shift-and-subtract

پشتیبانی از اعداد علامت‌دار

محاسبه خارج‌قسمت

فرمت دستورالعمل

دستورالعمل‌ها ۱۶ بیتی هستند و شامل:

3-bit Opcode

2-bit rd

2-bit rs

9-bit address or 2-bit rt

انواع دستورالعمل

1. R-Type: عملیات محاسباتی (ADD, SUB, MUL, DIV)

2. M-Type: دسترسی به حافظه (LOAD, STORE)

اسکریپت‌های اجرایی

run.sh - اسکریپت اصلی اجرا

این اسکریپت مسئول اجرای تمام اسکریپت‌های تست و شبیه‌سازی است:

عملکردهای اصلی:

تبدیل تمام فایل‌های .sh به حالت قابل اجرا

اجرای ترتیبی تمام اسکریپت‌های تست

مدیریت خطاها و گزارش‌دهی

پاک‌سازی فایل‌های موقت

SD.sh - اسکریپت سنتز دیجیتال

این اسکریپت مسئول سنتز کامل پردازنده با استفاده از ابزار Yosys است:

عملکردهای اصلی:

خواندن تمام فایل‌های Verilog

سنتز به گیت‌های منطقی

بهینه‌سازی طراحی

تولید فایل‌های خروجی

فایل‌های ورودی:

CU.v (واحد کنترل)

RF.v (فایل رجیستر)

M.v (حافظه)

ALU.v (واحد محاسباتی)

CSA.v (جمع‌کننده)

K.v (ضرب‌کننده)

RD.v (تقسیم‌کننده)

D.v (مسیر داده)

فایل‌های خروجی:

- SD.v (شبکه سنتز شده)

- SD.json (فرمت JSON)

- R.txt (گزارش سنتز)

تست و اعتبارسنجی

هر بخش از پردازنده دارای فایل تست جداگانه است:

ALUTB.v: تست واحد محاسباتی

CUTB.v: تست واحد کنترل

DTB.v: تست مسیر داده

RFTB.v: تست فایل رجیستر

MTB.v: تست حافظه

نکات برجسته

پیاده‌سازی کامل چرخه دستورالعمل ۵ مرحله‌ای

استفاده از الگوریتم‌های بهینه برای عملیات محاسباتی

سیستم تست جامع و خودکار

قابلیت سنتز کامل با ابزارهای استاندارد

فایل‌های پروژه

فایل‌های اصلی:

Datapath/D.v: مسیر داده اصلی

Control.Unit/CU.v: واحد کنترل

Arithmetic.and.Logical.Unit/ALU.v: واحد محاسباتی

Register/RF.v: فایل رجیستر

Memory/M.v: حافظه

فایل‌های محاسباتی:

Summation/CSA.v: جمع‌کننده CSA

Multiply/K.v: ضرب‌کننده Karatsuba

Division/RD.v: تقسیم‌کننده

اسکریپت‌های اجرایی:

run.sh: اسکریپت اصلی اجرا

SD.sh: اسکریپت سنتز دیجیتال

فایل‌های تست مربوط به هر بخش