به نام خدا

گزارش کار پروژه نهایی آزمایشگاه مدار منطقی و معماری کامپیوتر

متاسفانه چون برنامه که پوش کردم کار نمیکند در این گزارش کار به بخش های مختلف و نحوه عملکرد انها خواهیم پرداخت.

ابتدا به سراغ تشریح mypip.v میرویم:

در این فایل که فایل اجرایی ماست و دستور iverilog روی ان اجرا میشود ابتدا متغییر ها تعریف میشوند. سپس بخش های مختلفی مانند imembank و adder32bit تعریف میشود. در انتهای فایل رجیستر ها یا به عبارتی stall ها اورده شده است.

Adder32bit

کار این بخش اضافه کردن دو عدد in1,in2 میباشد.

ALU

در این بخش متناسب با aluoperation دستوری انجام میشود مانند جمع دو داده یا or دو داده. در ادامه gt,lt مقدار دهی میشود که یعنی اگر داده 1 بزرگ تر بود gt 1 و اگر داده 2 بزرگ تر بود lt 1 و دیگری صفر میباشد. در نهایت چک میشود که اگر result برابر با صفر باشد انگاه سیم صفر مقدار یک بگیرد در غیر اینصورت صفر مقدار دهی میشود.

ALUcontrol

در این بخش بر اساس ALUop و funct است که ALUsingal مقدار دهی میشود. خروجی شامل دستوراتی مانند add, sub,ori, branch و غیره میباشد.

concatForJump

در این بخش دو داده 4 و 28 بیتی کانکت میشوند.

Controlstall

در این بخش سه داده op1,op2,op3 به cpu داده میشود که براساس ان stall ها مشخص میشود. در اینجا باید یا bne,beq,j در داخل یکی از op1,op2,op3 باشد. در غیر اینصورت stall مقدار یک را میگیرد.

controlUnit

در این بخش با توجه به instruction WireID hazard هازاد ها چک میشود و در ادامه با توجه به opcode نوع داده و متغییرهایی مانند ALUop,register destination, register write مقدار دهی میشود.

DMemBank

Data memory bank : کار این بخش خواندن یا رایت کردن روی داده میباشد.

اگر memread درست باشد این دستور اجرا میشود.

readdata=mem\_array[finalAddress]

اگر memwrite درست باشد این دستور اجرا میشود.

mem\_array[finalAddress]= writedata;

EXMEM

این بخش یک رجیستر میباشد. کار آن این است که داده را یک پالس تاخییر بی اندازد.

IFID

این بخش یک رجیستر میباشد. کار آن این است که داده را یک پالس تاخییر بی اندازد.

IMemBank

Instruction memory bank: در این بخش آدرس داده وارد میشود و سپس مقدار داده برگردانده میشود این کار در if اخر اورده شده است.

MEMWB

این بخش یک رجیستر بین بخش mem و wb میباشد. کار آن این است که داده ها را یک پالس به تاخیر بی اندازد.

Mux2

این بخش مالتی پلکسر هست که با توجه select خروجی خود را مشخص میکند. یعنی اگر select مقدار یک داشته باشد b و در غیر اینصورت a انتخاب میشود.

Mux2A

همان mux2 هست منتهی a,b در اینجا 5 بیتی هستند نه 32 بیتی

PC

این بخش رجیستری هست که ورودی را زمانی که enable مقدار یک داشته باشد عینا به خروجی میفرستد

regFile

این بخش همان register file در مدار هست. روال کار اینگونه هست که سه رجیستر یکی برای رایت و دو تا برای رید به رجیستر وارد میشوند یک سیم برای دیتایی که قرار است رایت شود و یک سیم کلاک و یک سیم برای اینکه بفهمیم قرار است رید انجام شود یا رایت که نام آن regWrite میباشد.

اگر regWrite مقدار True داشته باشد انگاه عملیات رایت کردن انجام میشود در غیر اینصورت این عملیات انجام میشود که نشان دهنده خواندن از روی ورودی میباشد.

(Readreg1,readreg2:ورودی های تابع هستند)

  assign readdata1 = regfile[readreg1];

  assign readdata2 = regfile[readreg2];

shiftLeft32bitLeft

در این بخش داده دو بیت به سمت چپ شیفت پیدا میکند

shiftLeftForJump

در این بخش داده 26 بیتی وارد میشود و با دو صفر کانکت میشود که دو صفر در سمت راست داده قرار میگیرند که در نتیجه خروجی 28 بیتی خواهد شد.

signExt

در این بخش یک ورودی 16 بیتی داریم که بیت علامت 16 بار در سمت چپ آن تکرار میشود در نتیجه یک خروجی 32 بیتی خواهیم داشت.

Stall

در این بخش stall پیاده سازی شده است برای اینکه ممکن است برخی از instruction ها به تاخیر نیازمند باشند.