Assignment-6 submission report

M.Nikhil(EE17B138) P.S.V.Kushal(EE17b141)

For assignment 6 we are working following areas. - Error detection and handling it - setting up done signal for DMEM

Error detection and handling

Our goal for error detetection was to be able to detect any error in the instruction or any memory misallignment in case of dmem and any out range address for both IMEM and DMEM

Error detection

1.wrong instruction

we made a module which takes in idata directly and then gives the an output errorbits[4:0] with errorbits[0] for opcode errors,errorbits[1] for funct3 errors,errorbits[2] for other errors in instruction, errorbits[3] for DMEM related errors and errorbits[4] out of range errors.

2.Memory misallignment

Here we addressed error in giving wrong memory address particularly wrong for load(or store)word(or half word).We implemented this in memc module since it deals with the we for DMEM by using last bits. errorbits[3] corresponds to this type

3. Out of range memory accsess

This one we just made some additions in 1. and 2. since we can associate out of range iaddr as wrong instruction and out of range daddr as wrong memory.

Handling

our goal for handling is jump to a given address stored at erroraddr, store errorbits at scause register, iaddr for wrong instruction at at saddr and flush out any remains of wrong instruction and following instructions in the pipelined registers.

Error at Instruction fetch phase

For wrong instruction and out of range iaddr we are storing the iaddr,errorbits and then jump to erroraddr for any testing we are giving errorbits and wrongaddr as output and we are also flushing out any IF_ID as if we dont it will have wrong error instruction in it.

Error at DMEM phase or Mem read or store phase

Any error at DMEM part is realized from EX_MEM registers so error signal is used to clear all stages before MEM_WB and change PC to required address.

Ideas to load error address, DMEM size and IMEM size

One idea is to load these values in some part of dmem as use some spealized instructions to load them into some set of registers which can be used to compare thse values.These instructions work in the similar to load instructions.

Instructions for running the Demo

1. Add all the files to a new project and copy all the .data file into the project folder then simulate cpu_pipe_tb.v
2. for hardware implementation genearate two ip core files ICON named 'icon0' with 1 control port and VIO named 'vio0' with 2 ASYN_OUT and 201 SYNC_IN ports and synthesize top.v
3.