Contents

Universal Modular Platform based on Cdm-8 processor

Platform Description

Overview

In our project we decided to build a universal platform that can be used for different purposes.

Hardware

In this section we will describe hardware part of this platorm.

Basic setup

The bare minimum for this plat form is cdm8 cpu, address decoder rom and ram

scheme here

IO Bus

To communicate with devices we need to define what IO bus looks like.

Bus lines:

- Data
- IO Address
- IO Select
- Read/Write
- Clock

image

Expanding ROM

If we need more program memory we can use ROM controller to get more address space with memmory paging technique.

scheme here

image here

Mb interrupts here

Expanding RAM

If we need more RAM we can use similar technique.The difference is that we divide RAM address space into two halfs - lower half is global and upper half is paged.

scheme here

image here

Handling Interrupts

Without ROM Controller In Cdm8 in harvard setup interrupt vectors are located in in upper 16 bytes of program memory and therefore theese vectors are constant.

In out platform you can use it as is or connect Dynamic Interrupt Controller which allows you to change theese vectors by masking their addresses with external registers.

But this device is unconpatable with ROM controller

With ROM Controller ROM Controller takes part in interrupt handling process - when interrupt occurs controller changes memory page to one that is specified on corresponding controller pins.

The easiest way to specify page to handle interrupts is to connect a constant to theese pins, but in this case you cannot change it.

Better solution is to connect a register to bus and it's output to ISR Page pins. In that case you can set page dyncamically in runtime.

Devices description

In this block we will describe each device more precisely.

Peripheral Example

Most of devices connect to IO bus and therefore have similar block and signals that are used to communicate with the bus.

images with descripiton

ROM Controller

RAM Controller

Interrupt Arbiter

Interrupt Enable Buffer

Address Decoder

Dynamic Interrupt Controller

IO Register

IO Hex Display Controller

IO Seven Segment Display Controller

IO Hardware Stack

IO Random Number Generator

Display Controller

Joystick Controller

Keypad Controller

Terminal Controller

Software

In this part we will describe software part of this platform.

As we use more than 256 bytes of program memory and need to work with a lot of code default development tool (CocoIDE) is very unconfotable to use and that's why we developed some tools to make software development process easier.

cocomake

The main application that does hard work is cocomake. It is an incremental build system desined to work with multifile projects.

It is incremental, so only modified files get recompiled. That makes compiling much faster.

There, one bank(module) is one translation unit. Each file is compiled to an 256 byte image and then theese 256 byte images glued together to produce one big image that you load straight in logisim.

So, you can have one big project with a lot of files spannig to many modyles and you just execute one command and get your project compiled in one image.

VS Code Integration

For the text editor we decied to use VS Code as it is free modern software with a lot of custimization options via extensions.

To make support for cdm8 assembler we develpoed an extension to VS Code that adds syntax highlighting for assembly and c preprocessor directives as well as code snippets.

Demonstration

In this section we will describe out demonstation setup.

Scheme Overview

image

We use this this this

Code Overview

We set up cocomake like this …

code samples

Conclusion

idk