TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Fredo Erxleben

Design and Prototype Implementation of a Modular Reconfigurable Processor Architecture Kit

Status Report // Dresden, June 7, 2018

Contents of this Talk

Section 1

Introduction

Motivation

- Understanding the inner workings of a processor is hard
- Building even small processors from scratch is quite hard

How about having a building kit for learning, teaching and experimentation?

The Task

1. Research different approaches to processor construction
2. Identify common atomic components and their interaction
3. Design a set of modules that can be recombined into different processor architectures
4. Produce prototype modules, build demonstrative setups, and compare these

Quick Reminder

PCB: Printed Circuit Board

Units: $1\,\text{in} = 2.54\,\text{cm} = 1000\,\text{mil (US)}$

Caution

$1\,\text{mil (US)} = 1\,\text{thou (GB)}$

$1\,\text{mil (GB)} = 1\,\text{mm}$

Section 2

Initial Reseach

Where to begin?

- Modern architectures are terribly complex
- Details of commercial implementations are usually not public
- Many architectures are only described at the instruction level

Most promising approach

Check out the really old manuals and hobbyist projects

A Premature Bibliography

- *Intel 8080 Manual*, Sep. 1975
- *Cray-1 Reference Manual*, 1977
- *TMS 9900 Microprocessor Reference Manual*, 1976
- *UNIVAC 1107 Central Computer* in *UNIVAC 1107 Technical Bulletin*, Nov. 1961

Honorable Mention:

- Ben Eater's breadboard processor (`https://eater.net/8bit/`)

Section 3

Conceptualization and Interface Design

Didactic Considerations I

- Each module must encapsulate one well-defined functionality
- Use widely available, affordable standard components
- PCB layout must fit in a grid for easy recombination
- Design as few simple interfaces as possible

Didactic Considerations II

Conflict: Functional Purity vs. Versatility

Functional Purity: A module does exactly one thing.
$\rightarrow$ As few additional components as possible
Versatility: A module should be used in many different combinations
$\rightarrow$ Additional components are required

Design decisions must be made on a case-by-case basis.

Didactic Considerations III

Design Cornerstone

Data width: 8 bit
Address width: 16 bit (Can always scale down)
$\rightarrow$ Usable and comprehensible

Starting out small

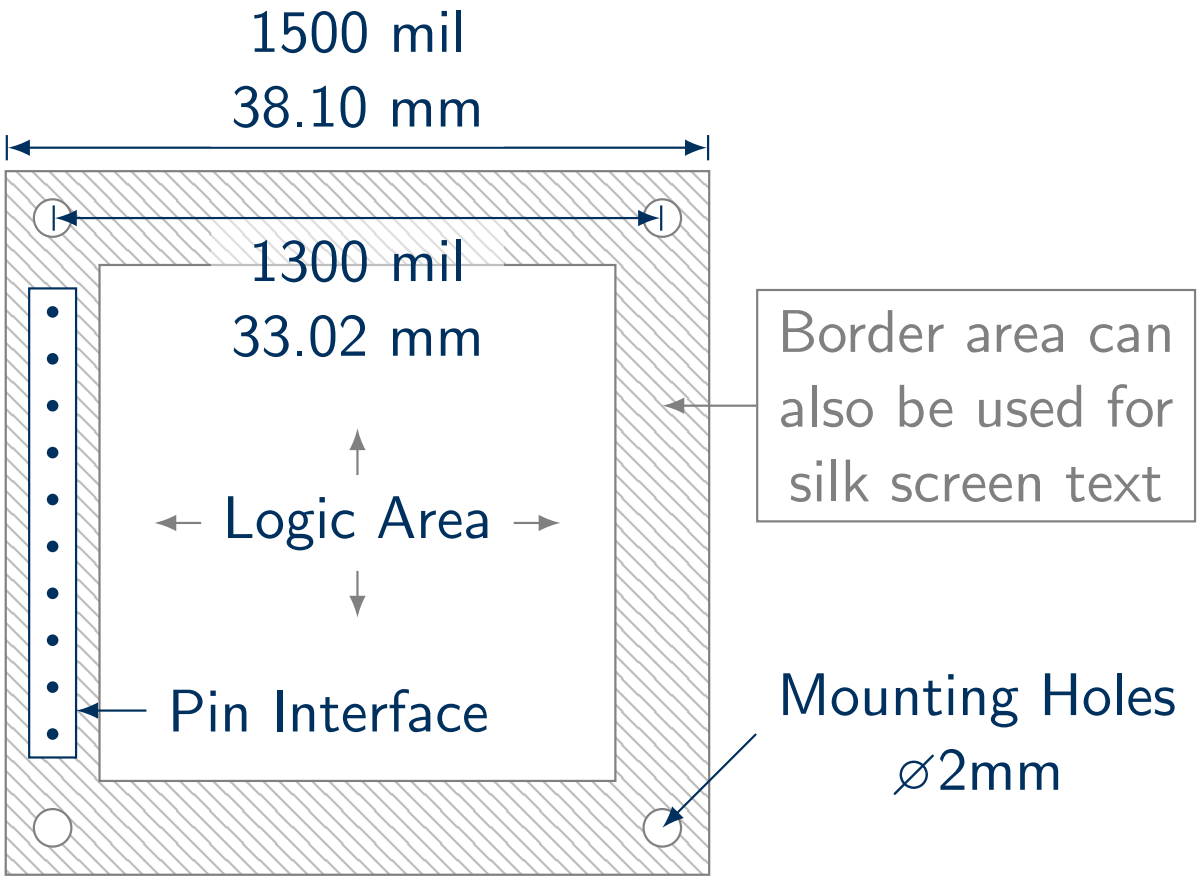
ALU reduced to an adder
Control logic mapped to ROM lookup table

And: Only very low operation frequencies required (1 . . . 100 Hz)

The Module Grid Layout I

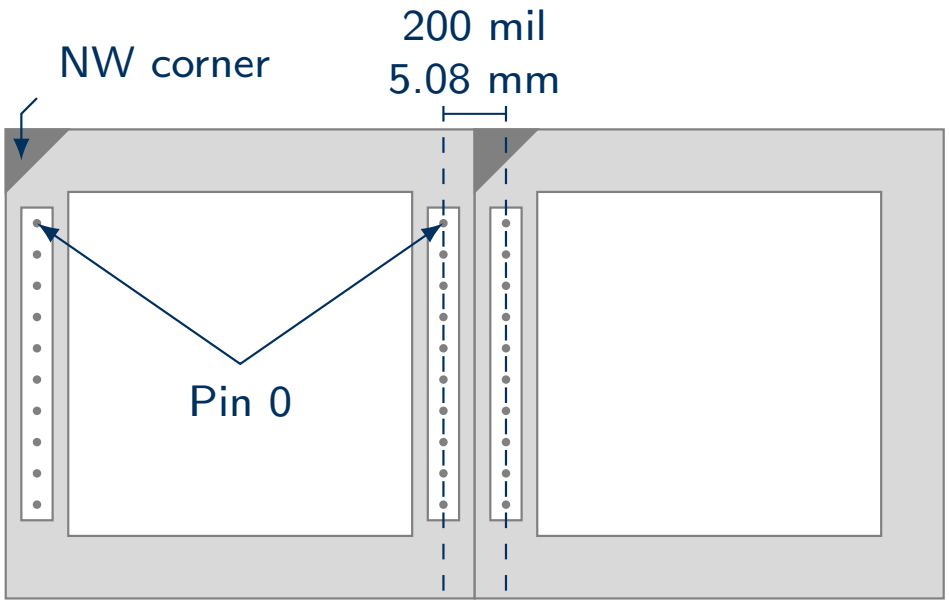
- Based on the common 2.54 mm raster
- Use through-hole mounting for easy assembly
- Modules may occupy multiple grid units
- Connection via dedicated bridge modules, ribbon cable or single jumper wires

The Module Grid Layout II



The Module Grid Layout III

- Tiling works horizontally and vertically
- Use compass directions as orientation reference



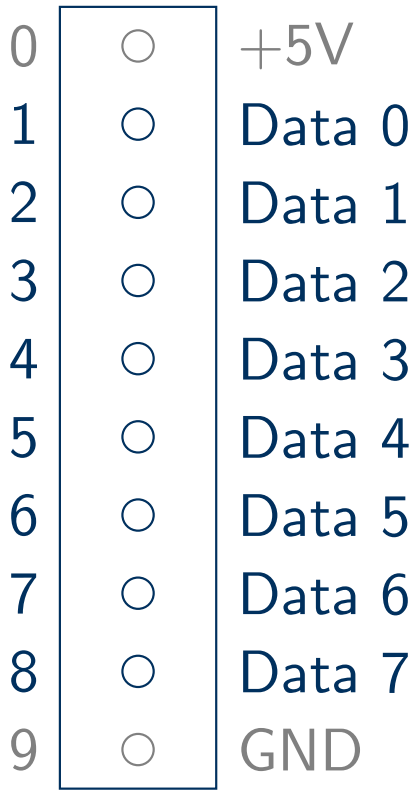
Interface Design I

- 2 interface types: *Control* and *Data*
- Each interface uses a 10×1-pin male header
- Pin 0 always is $+5\,V$, Pin 9 always is *GND*

Interface Design II

Data Interface

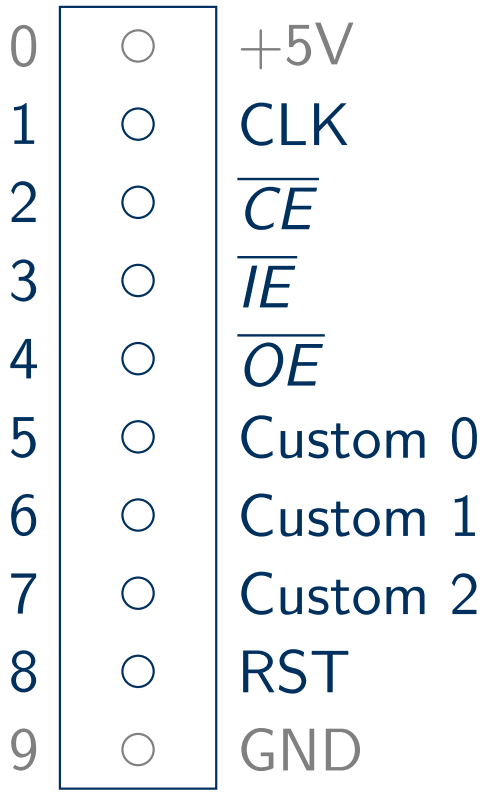
- Always on the *West* and *East* side of the chip
- Pin 0 is always oriented *North*
- Primary data flow direction is *West* $\rightarrow$ *East*



Interface Design III

Control Interface

- Always on the *North* and *South* side of the chip
- Pin 0 is always oriented *West*
- Primary control flow direction is *South* $\rightarrow$ *North*



Section 4

Modules

The Bare Necessities I

Clock Pulse Generator

- 2 modes of operation:
- Automatic (0.7 Hz . . . 480 Hz)
- Manual stepping via push-button

Binary Counter, 16 bit (WIP)

- New current value can be set anytime

The Bare Necessities II

Adder, 8 bit

- Provides a Zero- and Overflow flag
- Optional signed interpretation of second argument also allows for subtraction

Bus Interconnect (WIP)

- 3 *Data* interfaces on East/West side
- Need to be chained
  → Exception: North/South interfaces are *Data*, not *Control*

Memory I

Register, 8 bit

- Based on `74HC173`
- Provides dedicated $\overline{IE}$, $\overline{OE}$, RST

EEPROM

- Dual `(C)AT28C256`: 65 536 bytes
- Socketed, for programming via external tools

Memory II

RAM

- Dual 62256 or CY7194: 65 536 bytes
- Built-in address registers
- Address and data can be provided in parallel or byte-serial

Tricky Issue

How to easily load an initial program into the live RAM?

Solution

Added debugging / programming interface and programmer board.
→ Most complicated module so far

Section 5

Final Thoughts

Current State

Current State:

- Concept is complete
- Module design is nearly complete

Issues:

- Checking availability / alternatives for components is time-consuming
- Learning to use *KiCad* effectively took some time
- Designs need to be re-checked from time to time

Upcoming Improvements

- Increase logic area size on modules
- Externalize built-in registers of many modules
- Externalize built-in transceivers

Hot Question

Move transceivers into bus module or a separate module?

- Improved routing
- Easier to understand modules
- Improved design flexibility
- Forces user to learn when to use registers

Roadmap

The Plan:

- Finish the PCB designs
- Order PCBs, parts, and assemble them
- Build test architectures

Issues:

- PCBs and parts must be ordered in bulk for cost efficiency

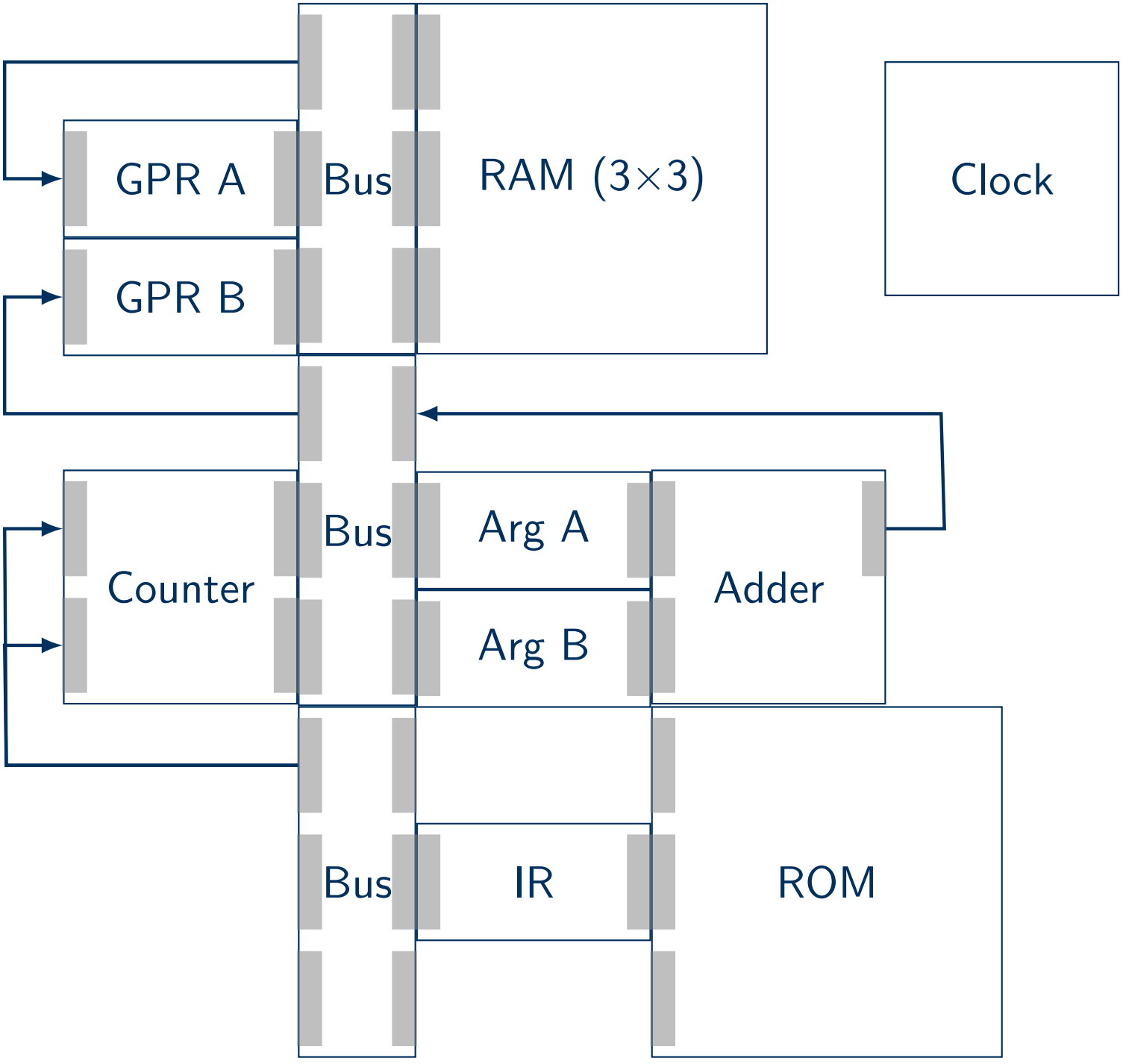
Last Slide

Thank you for your attention.
Questions?

Office: TUD, APB 1095
Phone: TUD-38456
E-Mail: fredo.erxleben@tu-dresden.de

Section 6

Appendix


GPR A
GPR B
Bus
RAM (3×3)
Clock
Counter
Bus
Arg A
Arg B
Adder
Bus
IR
ROM