Multi-core RISC SoC Design & Implementation

Demonstration Viva

Ben Lancaster

201280376 ELEC5881M - Main Project

July 27, 2019

Quick Links

Main Project

B. Lancaster

Introductio

Top Leve Design

Multi-core Functionality

Results

Conclusion

• GitHub repository: https://github.com/bendl/vmicro16

• Full Report: https://github.com/bendl/vmicro16/blob/master/docs/reports/build/ELEC5881M_Ben_Lancaster_201280376_Final.pdf

• This presentation: https://github.com/bendl/vmicro16/blob/master/docs/reports/build/ELEC5881M_Ben_Lancaster_201280376_viva.pdf

About me: https://bendl.me/

Main Project

B. Lancaster

Introductio

Top Level Design

Multi-core

Functionalit

Hesuits

- 1 Introduction
- 2 Top Level Design
- Multi-core Functionality

- 4 Results
- 6 Conclusion

Main Project

B. Lancaster

Introduction Why a project on

CPUs? Why Multi-core? Why RISC?

Top Leve Design

Multi-core Functionality

Results

Conclusion

Introduction
Why a project on CPUs?
Why Multi-core?
Why RISC?

2 Top Level Design

3 Multi-core Functionality

4 Results

Why a project on CPUs?

Main Project

B. Lancaster

Why a project on

CPUs? Why Multi-core? Why RISC?

Top Leve Design

Multi-core Functionalit

Results

- · CPUs will be used for the rest of humanity
- Understand design constraints and considerations
- Prepare myself for future employment/work

Why Multi-core?

Main Project

B. Lancaster

Introduction
Why a project on
CPUs?

Why Multi-core? Why RISC?

Top Leve

Multi-core Functionality

Results

- Rate of single-core speed improvements slowing
- Future of computing = parallel

Why RISC?

Main Project

B. Lancaster

Introduction
Why a project on
CPUs?
Why Multi-core?
Why RISC?

Top Leve Design

Multi-core Functionalit

Results

- Simpler design & impl
- Smaller = fit more cores on a chip
- Previous experience + future work
- I'm a RISC purist

Introduction

Main Project

B. Lancaster

Introduction

Top Level

Design

Memory Map

Interconnect

Interrupts

Function

Results

Conclusion

2 Top Level Design

Overview Memory Map Interconnect ISA Interrupts

3 Multi-core Functionality

4 Results

Overview

Main Project

B. Lancaster

Introduction

Design Overview Memory Map Interconnect

Interconnect
ISA
Interrupts

Functio

Results

Conclusion

What this project produces:

- System-on-Chip with multi-processor functionality
- Custom 16-bit RISC CPU
- Software/Assembly compiler
- Aimed at Design Engineers, not end users

Top Level Hierarchy

Main Project

B. Lancaster

Introduction

Top Level

Overview

Memory Map

ISA

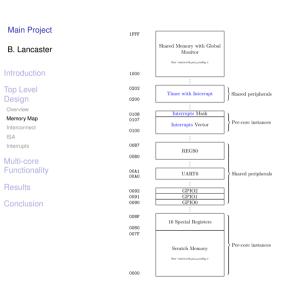
Interrupts

Functiona

Results



Memory Map



- Shared Memory*
- Timer + interrupts
- UART send/receive
- GPIO
- Scratch memory
- Extra registers
- + more

Interconnect

Main Project

B. Lancaster

Introductio

Top Level

Overview
Memory Map

Interconnect

ISA

Interrupts

Multi-cor Function

Results

- AMBA APB Bus
- Tristate & Non-tristate (mux) impl
- Originally Wishbone, now APB
- AHB too complex (limited time)
- Various schedulers

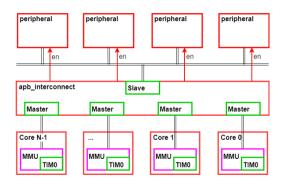


Figure: Vmicro16 interconnect

Interconnect Schematic

Main Project

B. Lancaster

Introduction

Top Leve

Overview

Memory Map

Interconnect

ISA Interrupts

Multi-core

Dogulto

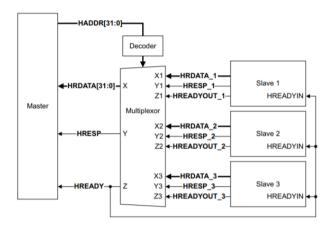


Figure: Source: ARM AHB-Lite Protocol Specification Figure 4-2.

Instruction Set Architecture

Main Project

B. Lancaster

Introductio

Top Level

Design

Memory N

ISA

Interrupts

- unction

Results

- + 16-bits, 38 instructions
- + Simple load/store arch
- + (Un)signed instructions
- + Compact
- + Optional hardware multiply instruction
- Only 8 registers
- No good compiler support

	15-11	10-8	7-5	4-0	rd ra simm5
	15-11	10-8	-8 7-0		rd imm8
	15-11	10-0			nop
	15	14:12 11:0			extended immediate
SPCL	00000	11 bits			NOP
SPCL	00000	11h'000			NOP
SPCL	00000	11h'001			HALT
SPCL	00000	11h'002			Return from interrupt
LW	00001	Rd	Ra	s5	Rd <= RAM[Ra+s5]
SW	00010	Rd	Ra	s5	RAM[Ra+s5] <= Rd
BIT	00011	Rd	Ra	s5	bitwise operations
BIT_OR	00011	Rd	Ra	00000	Rd <= Rd Ra
BIT_XOR	00011	Rd	Ra	00001	Rd <= Rd ^ Ra
BIT_AND	00011	Rd	Ra	00010	Rd <= Rd & Ra
BIT_NOT	00011	Rd	Ra	00011	Rd <= ~Ra
BIT_LSHFT	00011	Rd	Ra	00100	Rd <= Rd << Ra
BIT_RSHFT	00011	Rd	Ra	00101	Rd <= Rd >> Ra
MOV	00100	Rd	Ra	Х	Rd <= Ra
MOVI	00101	Rd	18		Rd <= i8
ARITH_U	00110	Rd	Ra	s5	unsigned arithmetic
ARITH_UADD	00110	Rd	Ra	11111	Rd <= uRd + uRa
ARITH_USUB	00110	Rd	Ra	10000	Rd <= uRd - uRa
ARITH UADDI	00110	Rd	Ra	OAAAA	Rd <= uRd + Ra + AAAA
ARITH_S	00111	Rd	Ra	s5	signed arithmetic
ARITH_SADD	00111	Rd	Ra	11111	Rd <= sRd + sRa
ARITH_SSUB	00111	Rd	Ra	10000	Rd <= sRd - sRa
ARITH_SSUBI	00111	Rd	Ra	DAAAA	Rd <= sRd - sRa + AAAA
BR	01000	Rd	i8		conditional branch
BR_U	01000	Rd	0000 0000		Any
BR_E	01000	Rd	0000 0001		Z=1
BR NE	01000	Rd	0000 0010		Z=0
BR_G	01000	Rd	0000 0011		Z=O and S=O
BR GE	01000	Rd	0000 0100		S=O
BR L	01000	Rd	0000 0101		S != O
BR LE	01000	Rd	0000 0110		Z=1 or (S != O)
BR_S	01000	Rd	0000 0111		S=1
BR_NS	01000	Rd	0000	1000	S=0
CMP	01001	Rd		Х	SZO <= CMP(Rd, Ra)
SETC	01010	Rd		m8	Rd <= (Imm8 f SZO) ? 1:0
MULT	01011	Rd		X	Rd <= uRd * uRa
HALT	01100		X		
			-		
LWEX	01101	Rd	Ra	s5	Rd <= RAM[Ra+s5]
LVVLA	01101	Nu	na	50	RAM[Ra+s5] <= Rd
SWEX	01110	Rd	Ra	s5	Rd <= 0 1 if success

Main Project

B. Lancaster

Introduction

Top Level Design Overview

Memory Map

Interconnect

. . . .

Interrupts

Functionality

Results

Conclusion

SW Example

Timer Interrupt Example

Main Project .870.000 ns 3.870.000 ns 3,000 ns 3,500 ns 4,000 ns 4,500 ns 5,000 ns 5,500 ns Name Value 2,500 ns B. Lancaster > 16 r_pc[15:0] VII VII VII VII VII 0010 0010 0010 0010 0010 > W r instr[15:0] YO YO YO YO 0 V 0 V 0 V 0 VO VO VO VO 4000 4000 4000 4000 4000 0010,0100,0008,0000,0000,000 > W regs[0:7][15:0] 0010 0100 0008 0 > COALU V TI INT > W ints[7:0] 01 Overview > W ints vector[127:0] 0000000000000000 Memory Man ints_mask[7:0] 0 f 14 w intr Interrupts M has int 14 int pending W int pending ack U reas use int > " MMU) To C1 TIMRO ₩ out M S PSELX U we

Demo: 2 Core LED toggle (GPIO0) with TIMR0 1s interrupt (interrupts_2.s)

Main Project

B. Lancaster

Introduction

Top Leve Design

Multi-core Functionality

Context Identification
Atomics
Design Challenges

Deside

Conclusion

Introduction

2 Top Level Design

3 Multi-core Functionality HW/SW Requirements Context Identification Atomics Design Challenges

4 Results

Main Project

B. Lancaster

HW/SW

Requirements

Design Challenges

Hardware:

 Bus Arbitration (scheduling: priority, rotating, etc.)

Main Project

B. Lancaster

Introduction

Design

Functionalit

HW/SW Requirements Context Identificati

Atomics
Design Challenges

Regulte

Conclusion

Hardware:

- Bus Arbitration (scheduling: priority, rotating, etc.)
- Atomic functions
 (atomic versions of load/store to prevent race conditions)

Main Project

B. Lancaster

Introduction

Design

Functionalit

HW/SW Requirements

Context Identification
Atomics
Design Challenges

Design Challer

Results

Conclusion

Hardware:

Bus Arbitration

(scheduling: priority, rotating, etc.)

- Atomic functions

 (atomic versions of load/store to prevent race conditions)
- Per-core instruction memory

Main Project

B. Lancaster

Introduction

Design

Functionali

HW/SW Requirements

Context Identification Atomics

Design Challenges

Results

Conclusion

Hardware:

- Bus Arbitration
 (scheduling: priority, rotating, etc.)
 - Atomic functions
 (atomic versions of load/store to prevent race conditions)
 - Per-core instruction memory
 - Per-core context-switching for interrupt handling

Main Project

B. Lancaster

Introduction

Design

Functionali

HW/SW Requirements

Atomics

Design Challenges

Desults

Conclusion

Hardware:

- Bus Arbitration (scheduling: priority, rotating, etc.)
- Atomic functions
 (atomic versions of load/store to prevent race conditions)
- Per-core instruction memory
- Per-core context-switching for interrupt handling

Software:

Semaphores/Mutexes
 (exclusive memory access)

Main Project

B. Lancaster

Introduction

Design

Functionali

HW/SW Requirements

Context Identification Atomics

Describe

Conclusion

Hardware:

- Bus Arbitration (scheduling: priority, rotating, etc.)
- Atomic functions
 (atomic versions of load/store to prevent race conditions)
- Per-core instruction memory
- Per-core context-switching for interrupt handling

- Semaphores/Mutexes
 (exclusive memory access)
- Thread synchronisation (memory barriers)

Main Project

B. Lancaster

Introductio

Design

Multi-core Functionali

HW/SW Requirements

Context Identification Atomics

Dogulto

Conclusion

Hardware:

- Bus Arbitration (scheduling: priority, rotating, etc.)
- Atomic functions
 (atomic versions of load/store to prevent race conditions)
- Per-core instruction memory
- Per-core context-switching for interrupt handling

- Semaphores/Mutexes
 (exclusive memory access)
- Thread synchronisation (memory barriers)
- Context identification
 What core am I?
 How many cores?
 How much memory?

Context Identification

Main Project

B. Lancaster

Introduction

Top Leve Design

Functionalit

HW/SW Requirements

Context Identification

Design Challenges

Results

Conclusion

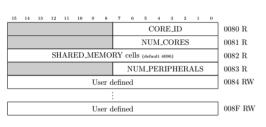


Figure: Special Registers 0x0080 to 0x008F

```
entry:
    // get core idx 0x80 in r7
    movi
             r7. #0x80
    l w
             r7, r7
       Branch away if not core 0
    cmp
             r7. r0
    movi
            ro, exit
    br
             ro. BR_NE
    // Core 0 only instructions
    nop
    nop
    nop
exit:
    halt
```

Atomic Instructions

Main Project

B. Lancaster

Introduction

Top Leve Design

Functionalit

Requirements
Context Identificatio

Atomics
Design Challenges

_ ...

- Enables semaphores, mutexes, memory barriers
- Prevent race conditions between threads/cores
- LW[EX] and SW[EX]
- Implementation in next slide

```
try_inc:
    // load and lock
    // (if not already locked)
    lwex
            r0. r1
    // do something
    // (i.e. add 1 (semaphore))
    addi
            r0, #0x01
    // attempt store
            r0. r1
    swex
      check success (== 0)
            r0. r3
    cmp
    // if not equal (NE), retry
    movi
            r4, try_inc
            r4, BR_NE
    br
critical:
    // r0 is latest value
```

Exclusive Access Flow Chart

Main Project

B. Lancaster

Introduction

Top Leve Design

Multi-core

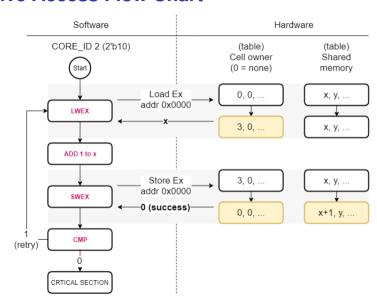
Functional HW/SW

Requirements

Atomics

Design Challenges

Results



HW - How do I know which core this lwex/swex is from?

Main Project

B. Lancaster

Introduction

Top Leve Design

Multi-core

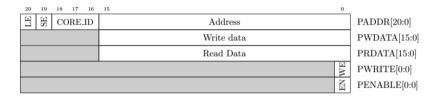
HW/SW Requirements

Context Identifica

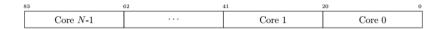
Atomics
Design Challenges

. .

Conclusion



The Core Idx is sent with each MMU request to the shared bus.



PADDR*NUMCORES-1:0 interconnect input.

Exclusive Access

```
Shared Memory (bram ex apb)
                                                        -- APB S PWRITE-
                                                                                                              S_PWRITI
Main Project
                mutex claim:
                    // load and lock
                                                        - APB_S_PWDATA-
                                                                                                              DATA[15:0]
B. Lancaster
                    // (if not already locked)
                                                        --- APB S PADDR--
                                                                        LWEX
                                                                                 CORE ID
                                                                                                             ADDR[15:0]
                              r0. r1
                    // do something
                    // (i.e. add 1 (semaphore))
                    addi
                              r0, #0x01
                    // attempt store
                              r0 r1
                     swex
                                                                                                                              (c)
                                                                                                                CORE ID MATCH
                    // check success (== 0)
HW/SW
                                                                                              we addr data
                    cmp
                              r0. r3
                    // if not equal (NE), retry
Atomice
                    mowi
                              r4. mutex_claim
Design Challenges
                                                                                                                         SWEX
                    br
                              r4. BR NE
                                                                                                                         Successe
                                                                                     64
                                                                                                         64
                                                                                                                                  -mem out-->
                                                                                   [2:0]
                                                                                                     [15:0]
                                                                                                                         UNIA
                critical:
                                                                               ex flags
                                                                                                      bram
                    nop
```

Figure: HW impl

Demo: 8 core number summation (sum.s)

Design Challenges

Main Project

B. Lancaster

Introduction

Top Leve Design

Functionali

Requirements
Context Identification
Atomics

Design Challenges

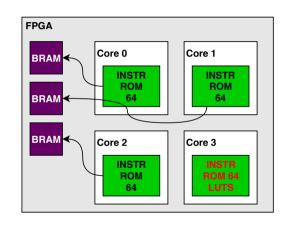
Results

Conclusion

Memory Limitations

Each core has it's own instruction memory

- + Fast fetching and branching
- Requires a dedicated BRAM (FPGA) per core
- Limited BRAM blocks available
- Reduces maximum core count



Design Challenges

Main Project

B. Lancaster

Multi-core

HW/SW

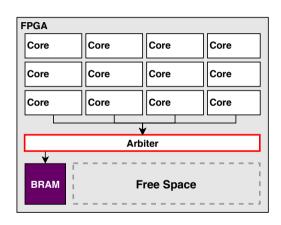
Requirements Design Challenges

Memory Limitations - Solution

Global instruction ROM

- + Single BRAM
- Smaller core size
- Slow access times





Main Project

B. Lancaster

Multi-core

Results

Summation - Shared Instruction ROM

Multi-core Functionality

Results Summation Summation - Shared Instruction ROM

Summation - Multi-core vs Single-Core

Main Project

B. Lancaster

- Each core has low work load
 - Sum subset of numbers in for loop
- Ideal scenario for parallelism
 - Highly parallelisable
 - Few inter-thread dependencies

Summation - Multi-core vs Single-Core

240 samples (@30 cores = 8 samples per core)

Main Project

B. Lancaster

Introduction

Top Leve Design

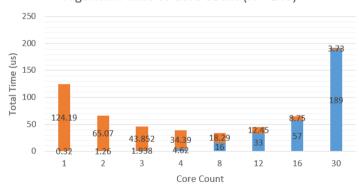
Multi-core Functionalit

Results

Summation - Sha

Instruction ROM





Multi-core vs Single-Core for Summation

240 samples (@30 cores = 8 samples per core)

Main Project

B. Lancaster

Introduction

Top Leve Design

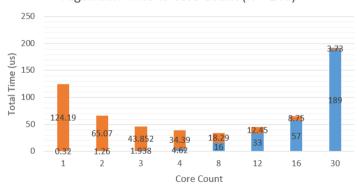
Multi-core Functionalit

Results

Summation - Shared Instruction ROM

Conclusior





Main Project

B. Lancaster

Conclusion

Conclusion Accomplishments **Future Improvements** Q&A

3 Multi-core Functionality

Accomplishments

Main Project

B. Lancaster

Introduction

Top Leve Design

Multi-core Functionalit

Result

Conclusio

Accomplishments
Future Improvement
Q&A

System-on-Chip with peripherals
 Timers, GPIO, UART, Registers, Memory

- Common multi-thread/core synchronisation primitives
 Semaphores, Mutexes, Memory Barriers, Atomic Instructions
- AMBA APB bus interconnects
- Interrupts with hardware context-switching
- Understanding of limitations and solutions

Future Improvements

Main Project

B. Lancaster

Introductio

Design

Multi-core Functionali

Results

Accomplishments

Future Improvements

Global Reset

- On-chip Programming
- Per-core gating/enabling
- Improve memory bottleneck

Main Project

B. Lancaster

Introduction

Top Level Design

Multi-core Functionality

Results

Conclusion

Accomplishments
Future Improvements
Q&A

Q&A

Q&A

Main Project

B. Lancaster

Introductio

Top Leve Design

Multi-core Functionality

Result

Conclusio

Accomplishments
Future Improveme

• GitHub repository: https://github.com/bendl/vmicro16

• Full Report: https://github.com/bendl/vmicro16/blob/master/docs/reports/build/ELEC5881M_Ben_Lancaster_201280376_Final.pdf

- Presentation tools:
 - Latex Beamer
 - \usecolortheme{orchid}
 - \useoutertheme[hideothersubsections]{sidebar}