

1. Preface
2. Design
3. Architecture
 3. 1. Registers
 3. 2. Instruction set
 3. 2. 1. RIS (register, immediate, shift)
 3. 2. 2. RRO (register, register, offset)
 3. 2. 3. I (immediate)
 3. 2. 4. nop
 3. 2. 5. set (immediate)
 3. 2. 6. load (using a register)
 3. 2. 7. store (using a register)
 3. 2. 8. halt
4. Conventions
 4. 1. Bit attributes
5. Physical memory map
6. DDR Controller
7. ROM
8. SRAM
9. Syscon
10. Interconnect
11. Interrupt controller
12. UART
 12. 1. Description
 12. 2. Initialization
 12. 3. Registers
 12. 3. 1. Register map
 12. 3. 2. UART_STAT
 12. 3. 3. UART_DATA
13. SPI
14. I2C
15. GPIO

1. Preface

This project is in it's very, very early stage, and most of the documentation below is not what is actually implemented, it's just my thoughts or ideas written down.

This documentation is written in [Typora](#) and this is the recommended tool for viewing it. A PDF version of this documentation should be available [here](#) but it's not guaranteed to be up-to-date with this source.

2. Design

The goal for Icarium is for it to be a very simple, embedded 64-bit System on Chip (SoC), which is suited for microkernels.

3. Architecture

3.1. Registers

Name	Description	Register id
r0	Reading this register will always return 0. Writing operations are ignored.	5'h00
r1	General purpose	5'h01
r2	General purpose	5'h02
r3	General purpose	5'h03
r4	General purpose	5'h04
r5	General purpose	5'h05
r6	General purpose	5'h06
r7	General purpose	5'h07
r8	General purpose	5'h08
r9	General purpose	5'h09
r10	General purpose	5'h0a
r11	General purpose	5'h0b
r12	General purpose	5'h0c
r13	General purpose	5'h0d
r14	General purpose	5'h0e
r15	General purpose	5'h0f
r16	General purpose	5'h10
r17	General purpose	5'h11
r18	General purpose	5'h12
r19	General purpose	5'h13
r20	General purpose	5'h14
r21	General purpose	5'h15
r22	General purpose	5'h16
r23	General purpose	5'h17
r24	General purpose	5'h18
r25	General purpose	5'h19
r26	General purpose	5'h1a
r27	General purpose	5'h1b

Name	Description	Register id
r28	General purpose	5'h1c
r29	General purpose	5'h1d
r30	General purpose	5'h1e
pc	Program counter.	5'h1f

3.2. Instruction set

All instructions are 64 bit wide.

General instruction types:

3.2.1. RIS (register, immediate, shift)

Opcode (7 bits)	Variant (2 bits)	Register (5 bits)	Immediate value (44 bits)	Shift (6 bits)
opcode [63:57]	variant [56:55]	reg [54:50]	imm [49:6]	shift [5:0]

3.2.2. RRO (register, register, offset)

Opcode (7 bits)	Variant (2 bits)	Destination register (5 bits)	Source register (5 bits)	Offset (45 bits)
opcode [63:57]	variant [56:55]	dst_reg [54:50]	src_reg [49:45]	off [44:0]

3.2.3. I (immediate)

Opcode (7 bits)	Variant (2 bits)	Immediate value (55 bits)
opcode [63:57]	variant [56:55]	imm [54:0]

3.2.4. nop

Format: I

Opcode: 7'h00000000

Does nothing.

3.2.5. set (immediate)

Format: RIS

Opcode: 7'b00000001

Variant: 2'b00

```
set r1, 0x80000000 shl 32
```

Will set the register `r1` to the value `0x8000000000000000`

3.2.6. load (using a register)

Format: RRO

Opcode: `7'h0000010`

Variant: `2'b00`

```
load r2, r1, 0x10
```

Will issue a read bus cycle to access memory at address which is stored in register `r1` having added the offset of `0x10` to it, and storing the result of this bus transaction into register `r2`.

3.2.7. store (using a register)

Format: RRO

Opcode: `7'h0000000`

Variant: `2'b00`

```
store r3, r1, 0x20
```

Will issue a write bus cycle with the data stored in address `r3` to the address stored in register `r1` having the offset `0x20` added

3.2.8. halt

Format: I

Opcode: `7'b1111111`

The `halt` instruction causes the CPU to halt. The only way of getting out of this state is by resetting the CPU.

4. Conventions

4.1. Bit attributes

Attribute	Meaning
RW	Read / write
RO	Read only
RsvZ	Reserved - always returns 0
RsvT	Reserved - writing a 1 will cause the CPU to trap

5. Physical memory map

Address range	Size	Device
0x0000000000000000 - 0x0000800000000000	128TiB	DDR RAM
0x0000800000000000 - 0x0000800000000400	1KiB	ROM
0x0000800080000000 - 0x0000800080000400	1KiB	SRAM
0x0000800100000000 - 0x0000800100000000	?	Syscon
0x0000800200000000 - 0x0000800200000000	?	Interconnect
0x0000800300000000 - 0x0000800300000000	?	Interrupt controller
0x0000801000000000 - 0x0000801000000000	?	UART
0x0000802000000000 - 0x0000802000000000	?	SPI
0x0000803000000000 - 0x0000803000000000	?	I2C
0x0000804000000000 - 0x0000804000000000	?	GPIO

6. DDR Controller

7. ROM

8. SRAM

9. Syscon

10. Interconnect

11. Interrupt controller

12. UART

12.1. Description

Icarium sports a very simple UART controllers, which currently support a static configuration of 1 start bit, 8 data bits, 1 stop bit, 9600 baudrate.

There are no FIFOs, no DMA, nothing fancy.

12.2. Initialization

12.3. Registers

12.3.1. Register map

Offset	Name	Description
0x00	UART_STAT	Status register
0x08	UART_CTRL	Control register
0x10	UART_DATA	Data register

12.3.2. UART_STAT

Bit(s)	Name	Reset value	Attribute	Description
2	STAT_RXD_BUSY			
1	STAT_TXD_BUSY	1'b0	RO	Transmitter busy - if 1 then the controller is currently transmitting a character, and a next character cannot be sent.
0	STAT_ENA	1'b1	RO	Enabled - shows the current state of the controller. If this bit is 1 then the controller is operating, and can send and receive data.

12.3.3. UART_DATA

Bit(s)	Name	Reset value	Attribute	Description
63:8		56'h0	RsvZ	
7:0	DATA	8'h0	RW	Writing to this register will trigger a transmit of the character, and reading from it will return any previously read character.

13. SPI

14. I2C

15. GPIO
