OpenRISC Talk

Stafford Horne

The Linux Foundation CE Workgroup

Japan Technical Jamboree

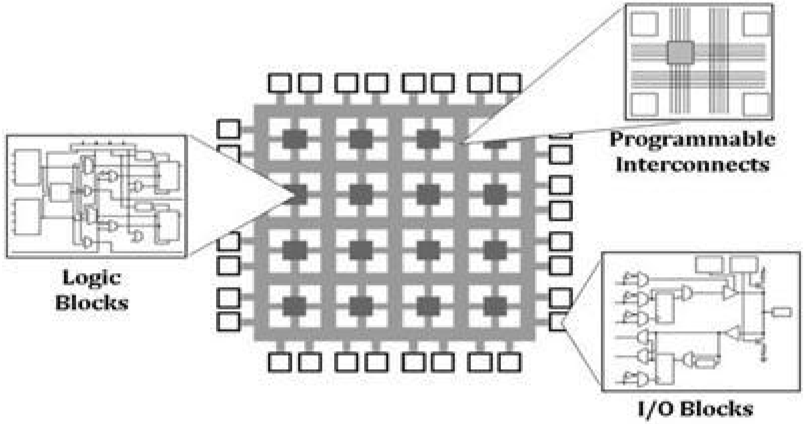
What is OpenRISC?

FPGA, IP cores

OpenCores

FuseSOC

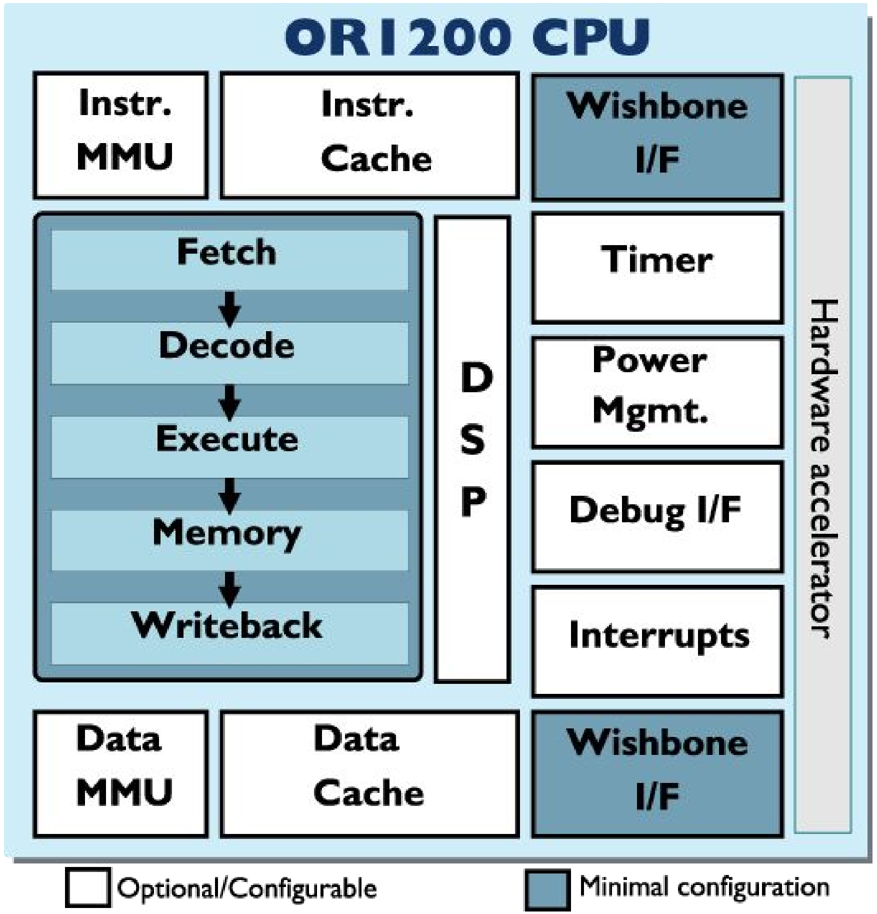
FOSSi



What is OpenRISC?

Officially OpenRISC 1000 is an open source RISC architecture:

- 32-bit / 64-bit
- 32 General Purpose Registers
- Delay Slot
- Instruction & Data MMU
- Linux support since 2010
  - 50mhz, 5 secs



Read More: https://raw.githubusercontent.com/openrisc/doc/master/openrisc-arch-1.1-rev0.pdf

OpenRISC implementations

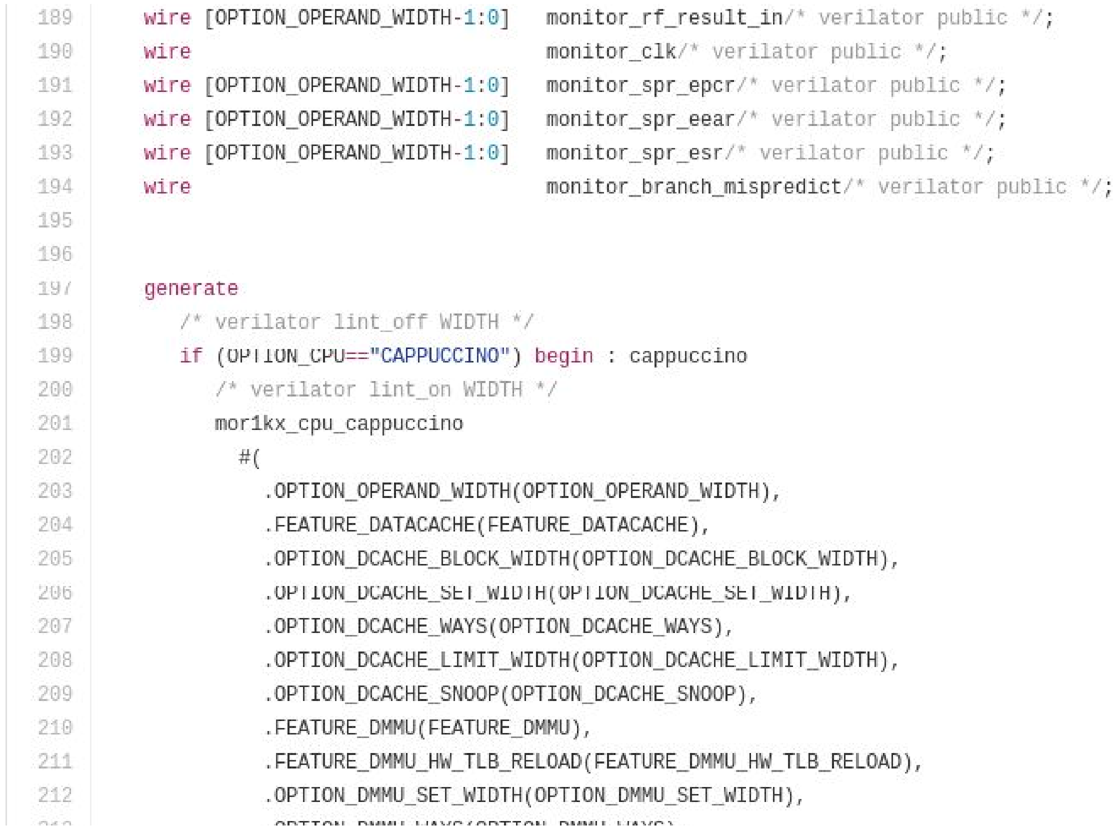
No 64-bit implementations

OR1200 - https://github.com/openrisc/or1200 - original (~2000)

mor1kx - https://github.com/openrisc/mor1kx - recommended (~2012)

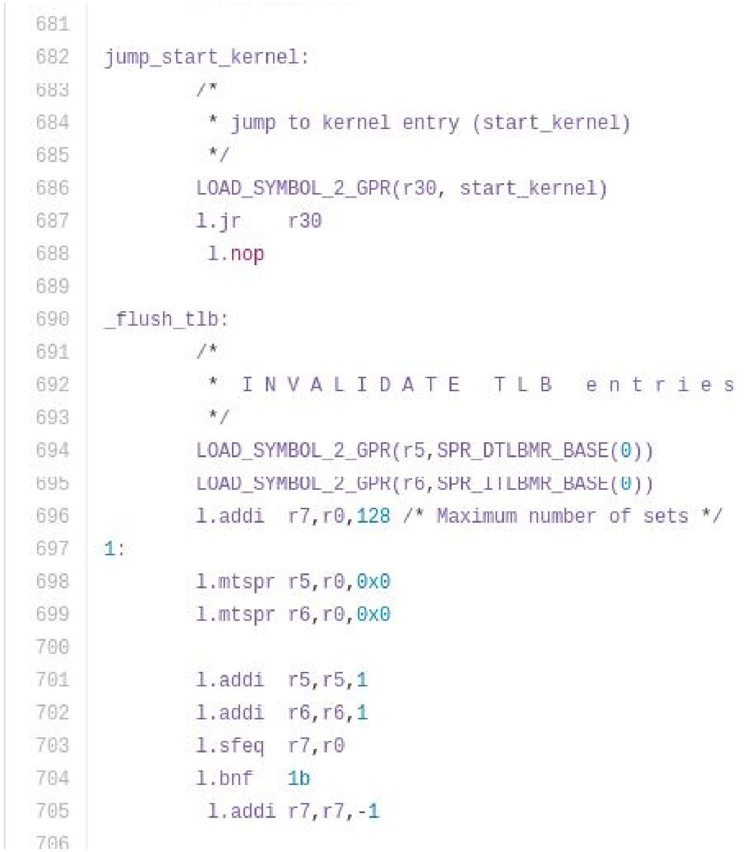
- Modular
  - Cappuccino - 5 stage, caching + mmu
  - Espresso - 2 stage, no caching
  - Pronto Espresso - 3 stage no delay slot, no caching

OpenRISC modular



```
wire [OPTION_OPERAND_WIDTH-1:0]   monitor_rf_result_in/* verilator public */;
wire                              monitor_clk/* verilator public */;
wire [OPTION_OPERAND_WIDTH-1:0]   monitor_spr_epcr/* verilator public */;
wire [OPTION_OPERAND_WIDTH-1:0]   monitor_spr_eear/* verilator public */;
wire [OPTION_OPERAND_WIDTH-1:0]   monitor_spr_esr/* verilator public */;
wire                              monitor_branch_mispredict/* verilator public */;


generate
   /* verilator lint_off WIDTH */
   if (OPTION_CPU=="CAPPUCCINO") begin : cappuccino
      /* verilator lint_on WIDTH */
      mor1kx_cpu_cappuccino
        #(
          .OPTION_OPERAND_WIDTH(OPTION_OPERAND_WIDTH),
          .FEATURE_DATACACHE(FEATURE_DATACACHE),
          .OPTION_DCACHE_BLOCK_WIDTH(OPTION_DCACHE_BLOCK_WIDTH),
          .OPTION_DCACHE_SET_WIDTH(OPTION_DCACHE_SET_WIDTH),
          .OPTION_DCACHE_WAYS(OPTION_DCACHE_WAYS),
          .OPTION_DCACHE_LIMIT_WIDTH(OPTION_DCACHE_LIMIT_WIDTH),
          .OPTION_DCACHE_SNOOP(OPTION_DCACHE_SNOOP),
          .FEATURE_DMMU(FEATURE_DMMU),
          .FEATURE_DMMU_HW_TLB_RELOAD(FEATURE_DMMU_HW_TLB_RELOAD),
          .OPTION_DMMU_SET_WIDTH(OPTION_DMMU_SET_WIDTH),
```

OpenRISC assembly

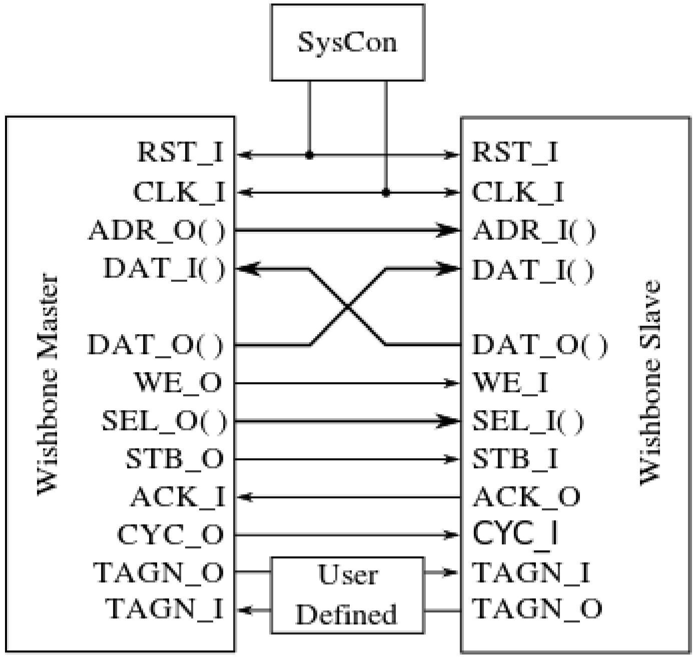


```
jump_start_kernel:
        /*
         * jump to kernel entry (start_kernel)
         */
        LOAD_SYMBOL_2_GPR(r30, start_kernel)
        l.jr    r30
         l.nop

_flush_tlb:
        /*
         * I N V A L I D A T E   T L B   e n t r i e s
         */
        LOAD_SYMBOL_2_GPR(r5,SPR_DTLBMR_BASE(0))
        LOAD_SYMBOL_2_GPR(r6,SPR_ITLBMR_BASE(0))
        l.addi  r7,r0,128 /* Maximum number of sets */
1:
        l.mtspr r5,r0,0x0
        l.mtspr r6,r0,0x0

        l.addi  r5,r5,1
        l.addi  r6,r6,1
        l.sfeq  r7,r0
        l.bnf   1b
         l.addi r7,r7,-1
```

OpenRISC & wishbone bus

OpenRISC / opencores

- Standard SOC interconnect
- Supports common topologies
  - Master - Slave
  - Shared bus (multiple master)
  - Crossbar
- Different pipeline burst options



Read More: https://opencores.org/cdn/downloads/wbspec_b3.pdf

OpenRISC vs Other soft codes

	Open	MMU	LInux	Upstream	Silicon
OpenRISC	✓	✓	32-bit	✓	Limited
RISC-V	✓	Kinda	64-bit	Soon	Multi
Nios2	✗	✓	32-bit	✓	✓
Microblaze	✗	✓	32-bit	✓	✓

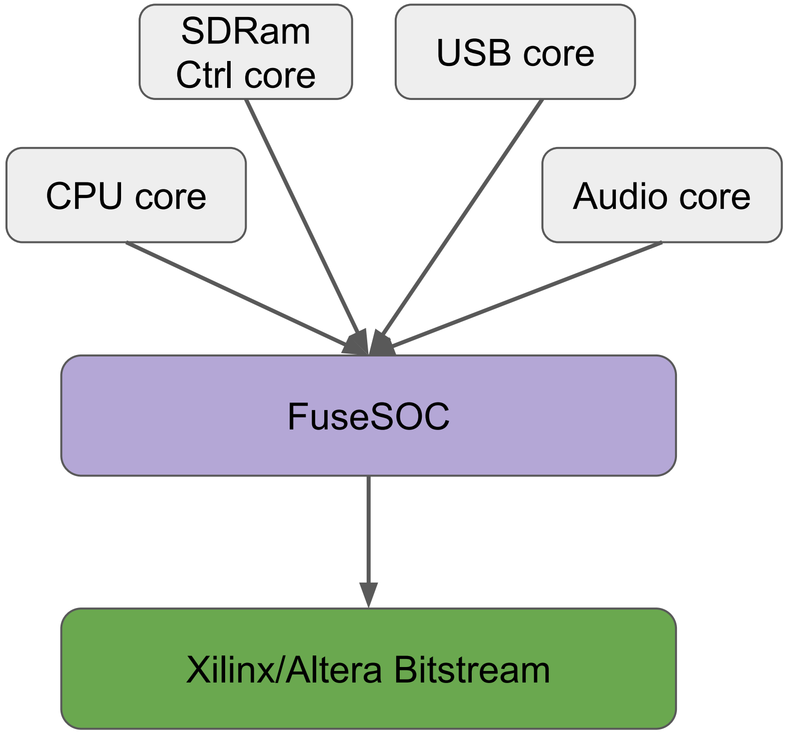
FuseSOC

Software world

- GCC, make
- Classpath, LD_LIBRARY_PATH
- import, include

Hardware world

- Vendor proprietary



A brief history

2000 - OpenRISC

2001 - opencores.org

2005 - OpenRISC founders create Beyond Semi

2007 - opencores.org changes hands ORSoC AB

2015 - FOSSi create fossi-foundation.org

- librecores.org
- freecores.github.io
- openrisc.io

FOSSi

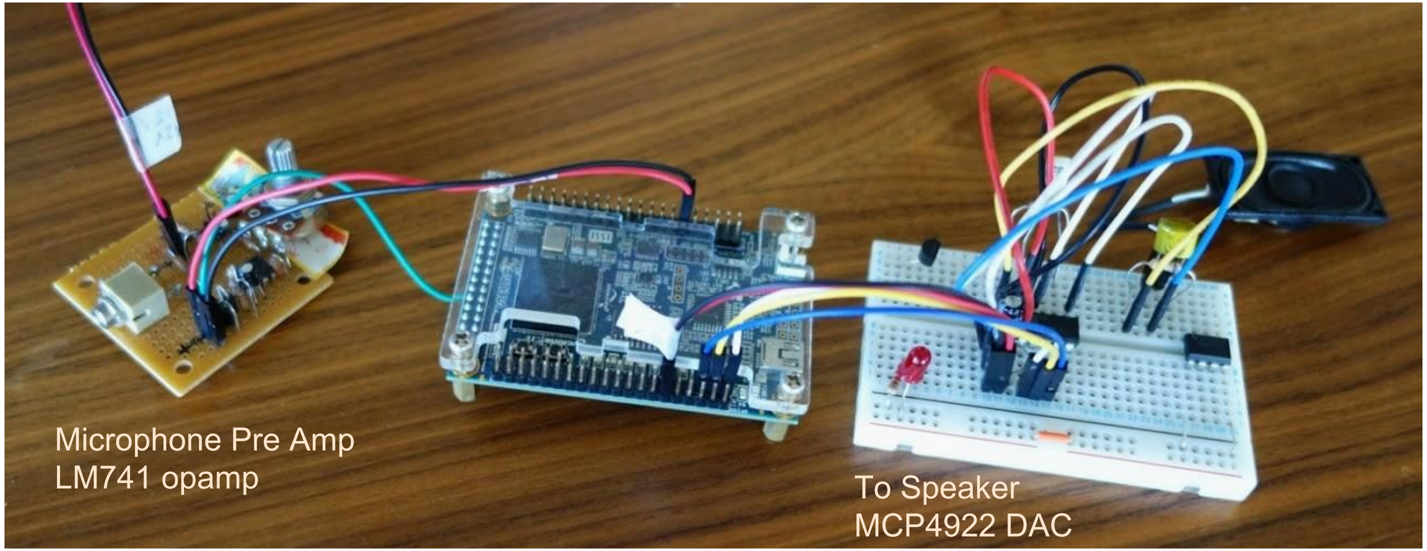
Help support and promote free and open source silicon

- Frontend Design - ip, simulation, analysis
- Backend Design - cell libraries
- Fabrication - cheaper processes
- Licencing
- Conferences - orconf

Getting Started

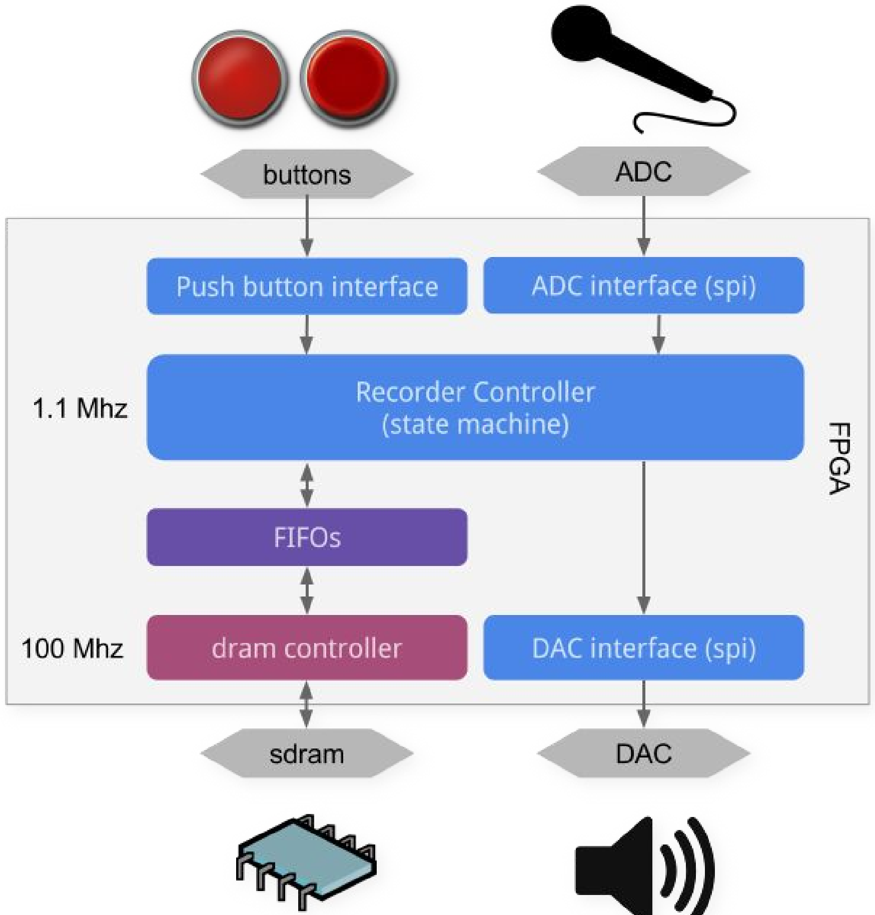
2015

I got back into fpga with a simple sound project. Designed some simple analog circuits and tied them together with a De0 Nano.



Project

- Developed all by hand
- Spent many hours with an ISSI datasheet to learn how to write the sdram controller



Project - next steps

- After that I wanted to do more but
  - Shareable IP
  - Better development lifecycle (not locked into quartus)
  - Run linux on the de0 nano

Toolchain GCC

or1k-gcc

- 5.3.0 released
- 5.4.0 (needed to build or1k-linux-musl toolchain) in git and testing

Upstream status - Pending

- 7.0.0 - in development (2016 Apr)
- 6.2.1 - latest 6 release (2016 Aug)
- 5.4.1 - latest 5 release (2016 Jun)

Toolchain Newlib (libc)

newlib

- 2.3.0 (Mar 2016)
- 2.4.0 (in github)

Upstream Status

- (~20 patches pending)
- 2.4.0 (Mar 2016)

Toolchain Binutils/GDB

- Release 7.11 (upstream at 7.12)
- 2016 Work
  - Testing Effort (Dejagnu)
  - Native Simulator Support (cgen)
- Upstream status (patches sent)
  - Working on fixups right now
    - Remove doxygen
    - Code style
    - ChangeLog merge (since 2008)
    - Implement pseudo registers

Toolchain Binutils/GDB - testing

```
$ export DEJAGNU=../../or1k-src/boards/or1k-elf-sim.exp

$ make check

# of expected passes            18667
# of unexpected failures        404     (450)
# of expected failures          28
# of known failures             52
# of unresolved testcases       34
# of untested testcases         163
# of unsupported tests          263
```

Toolchain Binutils/GDB - sim

```c
struct data {
  char * str;
};

struct data gdata = {
 .str = "global"
};

char * foo(struct data tdata) {
  return tdata.str;
}

int main() {
  struct data tdata;
  tdata.str = "hello";
  foo(tdata);
  return 0;
}
```

```
$ or1k-elf-gcc -g -o test test.c

$ or1k-elf-gdb test

(gdb) target sim

(gdb) load

(gdb) break main

(gdb) run

(gdb) print foo(gdata)

$1 = 0x3bd0 “global”
```

Toolchain Binutils/GDB - dummy call

Supporting OpenRISC abi:

- Structs and unions be passed as pointers
- Variadic arguments are passed on the stack

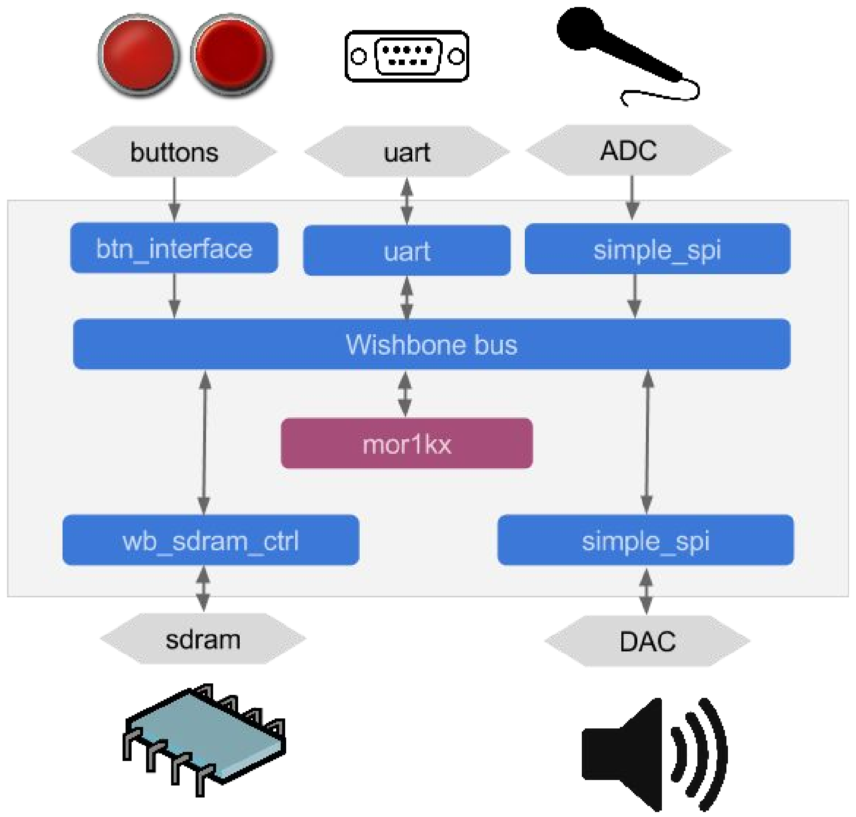
Linux

- No upstream patches since Feb 2012
- My work so far
  - `memcpy` - decrease boot time from 7 to 5 seconds
  - `bootmem` to `memblock` conversion
- Upstream effort
  - Prioritized and submitted patches for review
  - Maintaining commit queue
  - Patches in linux-next
  - Stafan for Me got pgp key signature

Memcpy Routine	Cycs
Word Copies + Loop Unrolls + Non Aligned	1882
Word Copies + Loop Unrolls	1887
Word Copies	2441
Byte Copies + Loop Unrolls	6467
Byte Copies	7600

Project

- Use openrisc soc with wishbone bus
- Reuse as much as possible
- re-write interface modules to work with wishbone bus
- Run Linux
  - DMA sound card driver
  - Controllable via UART



On the web

github.com/openrisc - projects hosted here

#openrisc on freenode (I'm shorne)

openrisc@lists.librecores.org

openrisc.io - work in progress, moving from (opencores.org)

- Downloads
- Tutorials

freecores.github.io - Opencores svn cores moved to github

Questions

?