FPGArduino: a cross-platform RISC-V IDE for masses

Marko Zec
University of Zagreb
zec@fer.hr

Davor Jadrijević
radiona.org
vordah@gmail.com

4th RISC-V workshop, July 2016.

Arduino: a HW + SW ecosystem

- Hardware
  - Cheap microcontrollers: 8-bit AVR, 16 MHz
  - Today: 32-bit ARM, MIPS, 50..100 MHz
  - Few KB of RAM for data, more for programs
  - Peripherals: DAC, ADC, displays, sensors, actuators, comms...
- Software
  - Language: C++ subset (no exceptions, no STL...)
  - Libraries for talking to peripherals
  - IDE: java-based; GNU toolchain underneath

Main attraction: simplicity, quick results

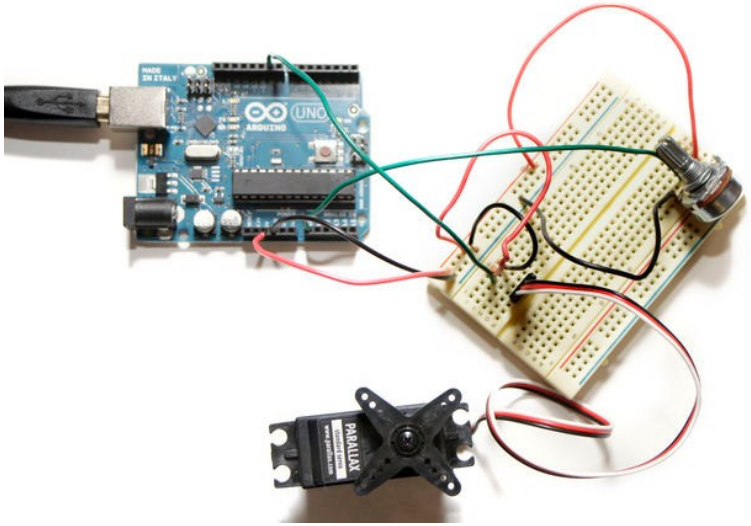
Blink | Arduino 1.5.3-Intel.1.0.4

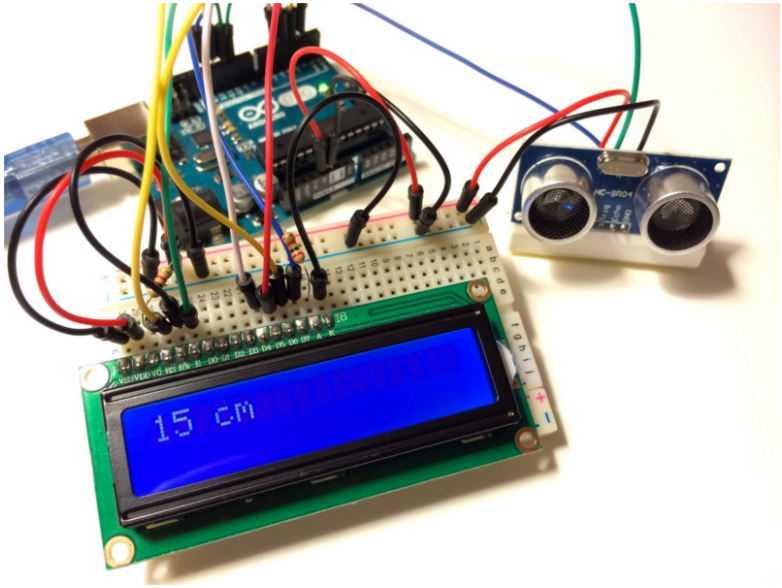
File Edit Sketch Tools Help

Blink

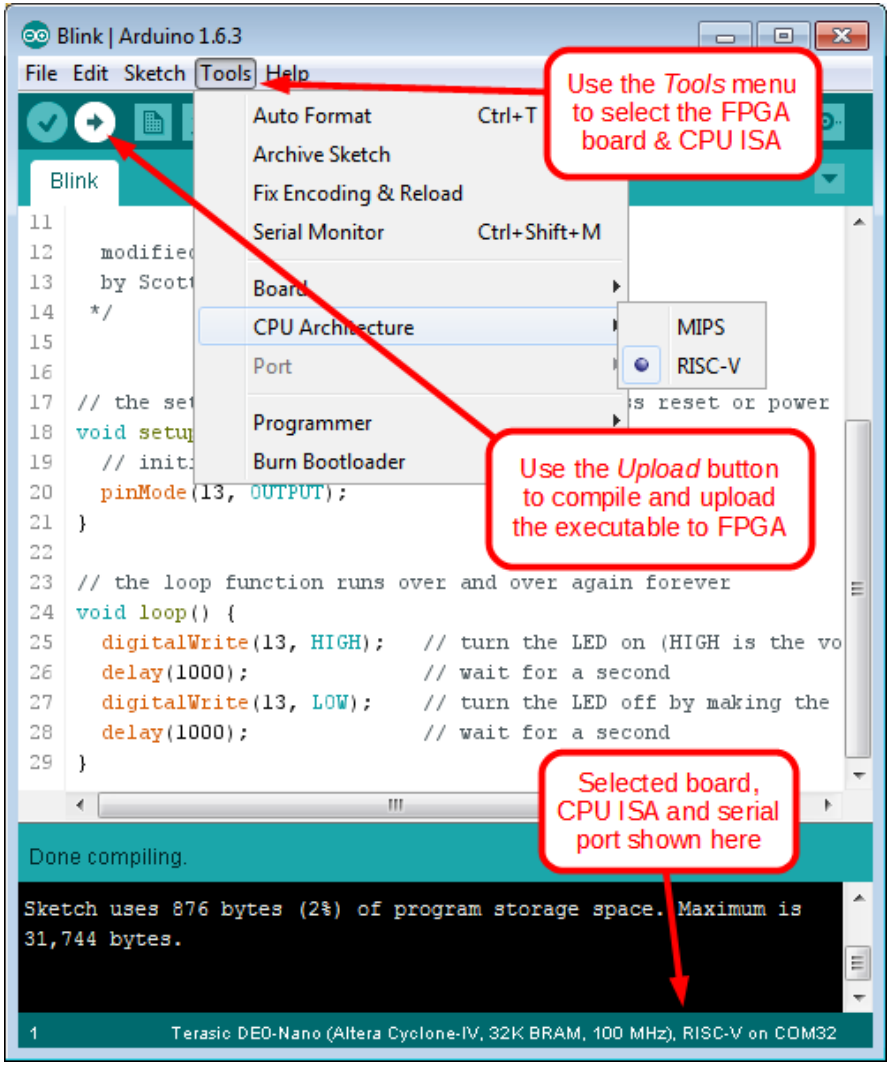
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

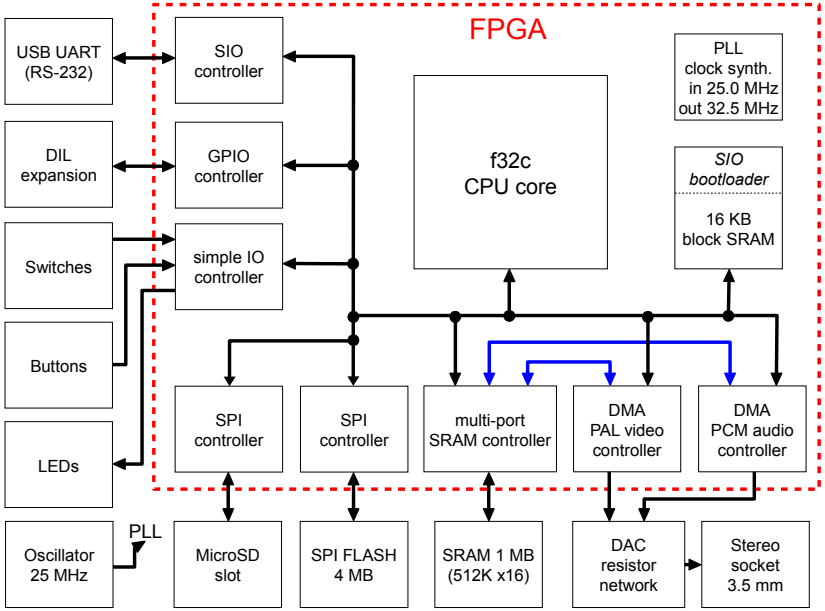
1 Intel® Edison on COM1

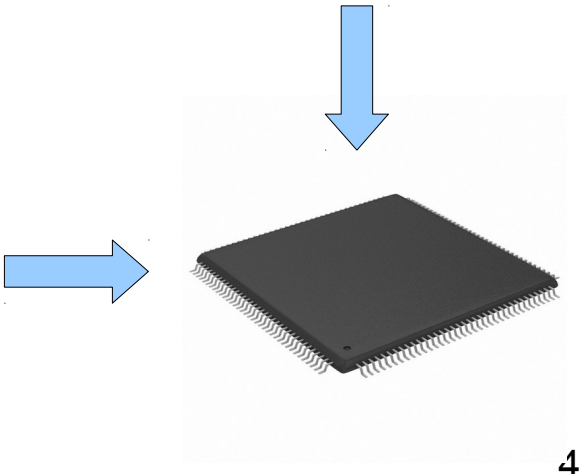




Arduino IDE + f32c SoC (RISC-V or MIPS)







FPGArduino: IDE extensions

- Pre-compiled toolchains for OSX, Win & Lin
  - RISC-V: gcc-4.9.2; MIPS: gcc-4.8.4
- C libraries, includes: mainly from FreeBSD
  - Also LLVM, NuttX parts inside (FP)
  - Startup: clear BSS, set GP, call ctors, j main
- Leverage Arduino 1.6.x extension framework
- *platform.txt* instead of Makefiles
  - compile, link, objcopy rules
  - cc -ffunction-sections -fdata-sections
  - ld -gc-sections

Code footprint

- FPGArduino
  - init + text + initialized data sections
- Others
  - init + text sections only

	Uno	Mega	**FPGArd.**	FPGArd.	Due	Galileo
	8-bit Atmega328	8-bit Atmega2560	**RV32I**	32-bit MIPS subset	32-bit Cortex-M3	32-bit i586
Blink LED	1066	1554	**1468**	1416	22116	83459
Fade LED	1324	2056	**1633**	1609	24844	83775
ASCII table	2414	2664	**2164**	2200	22540	81418
StringSubstring	3790	3964	**4394**	4549	23572	82232
StringStartsW...	5229	4516	**5058**	5229	24316	83579
LCD	2366	2748	**3408**	3504	23128	85846
TFT			**13696**	13352		

FPGArduino: IDE extensions

- *boards.txt* defines IDE menu entries, options
  - ISA (RV32I or MIPS)
  - Memory size / mappings
  - FPGA board-specific details
  - SoC features (video framebuffers, peripherals...)
- Pre-built FPGA bitstreams: upload from IDE
  - openocd / xc3sprog / ujprog / ...
  - Linux: OK; Windows: may work; OSX: ujprog only
- Program upload: async serial
  - Motorola SREC hex, or raw binary (ujprog)

FPGArduino: IDE extensions

- Automated installation
  - Install Arduino 1.6.x (OS-X, Windows, Linux)
  - File->Preferences, Additional Boards Manager URLs
  - http://www.nxlab.fer.hr/fpgarduino/pack.json
  - Tools->Board->Boards Manager
  - scroll down, find FPGArduino, click *Install*
- FPGA boards appear in *Tools/Boards* menu
  - Choose FPGA board, serial port
  - Burn bootloader → FPGA bitstream download
- Done!
  - *Upload* button: compile, upload program to FPGA

f32c: a retargetable scalar core, SoC

- Executes either RV32 or MIPS ISA subsets
- Architectural details → poster / demo session
- SoC
  - Multiport SRAM / SDRAM controllers
  - AXI bridges
  - Video framebuffers (HDMI, VGA, composite)
  - PID, SPI, UART, PCM audio, simple & GP IO...
- Generic, portable VHDL (SoC: some Verilog)
  - No dependencies on vendor-specific primitives
  - Inferred BRAMs, DSPs
  - Xilinx, Altera, Lattice – from ~2005 silicon onwards

f32c: a retargetable scalar core, SoC

- Porting: wire up & instantiate *glue* modules
  - glue_bram.vhd, glue_sram.vhd, glue_xram.vhd...
  - Provide / synthesize / constrain clk signal
  - C_arch generic: ARCH_RV32 or ARCH_MI32
- Bootloader
  - BRAM, mapped RO at 0x000 to 0x400 (1 KB)
  - Proof of life: serial prompt, fading LEDs
  - Parses Motorola SREC hex directly
  - Supports binary transfers for better efficiency
  - SPI flash (FAT32) bootloader also available (2 KB)
- Serial break: CPU reset

RISC-V / MIPS software / hardware Arduino FPGA stack - Mozilla Firefox

RISC-V / MIPS softwar...

www.nxlab.fer.hr/fpgarduino/

motorola srec hex

Supported FPGA boards with pre-built configuration bitstreams

Terasic DE0-Nano (Altera Cyclone-IV)

Xilinx Spartan 3E-500 Starter Kit

Xilinx Spartan 3E-1600 Dev. Board

Xilinx Spartan 3A/3AN Starter Kit

Digilent Basys-3 (Xilinx Artix-7)

Digilent Nexys-3 (Xilinx Spartan-6)

Digilent ZYBO (Xilinx Zynq)

FER ULX2S (Lattice XP2)

E2LP (Xilinx Spartan-6)

Lattice Brevia (Lattice XP2)

Lattice Brevia 2 (Lattice XP2)

No-name TB276 (Altera Cyclone-IV)

No-name TB299 (Xilinx Spartan-6)

Scarab MiniSpartan6+ (Xilinx Spartan-6)

Numato Mimas V2 (Xilinx Spartan-6)

Conclusions

- Home: http://www.nxlab.fer.hr/fpgarduino
  - It's plug and play – try it out yourself!
- If your'e into hacking:
  - SW: https://github.com/f32c/arduino
  - HW: https://github.com/f32c/f32c
- Hands-on demo → poster session
  - Bring your FPGA board for porting!
- Questions?

Thank you!

platform.txt

```
f32c.riscv.compiler.c.flags=-c -Os -m32 -msoft-float
-mno-muldiv -fpeel-loops -f freestanding -ffunction-sections
-fdata-sections -fpermissive -Wall -nostdinc
"-I{build.system.path}/include" -include sys/param.h
-include sys/stdint.h

f32c.riscv.compiler.cpp.flags=-c -Os -m32 -msoft-float
-mno-muldiv -fpeel-loops -ffreestanding -ffunction-sections
-fdata-sections -fpermissive -Wall -nostdinc
"-I{build.system.path}/include" -include sys/param.h
-fno-rtti  -fno-exceptions -include sys/stdint.h

f32c.riscv.compiler.ld.flags=-N -EL -melf32lriscv
-gc-sections "--library-path={build.system.path}/riscv/lib"
```

boards.txt

```
fpga_de0_nano_bram.name=Terasic DE0-Nano (Altera Cyclone-IV,
32K BRAM, 100 MHz)
fpga_de0_nano_bram.build.f_cpu=100000000
fpga_de0_nano_bram.build.variant=generic
fpga_de0_nano_bram.build.mcu=f32c
fpga_de0_nano_bram.build.core=f32c
fpga_de0_nano_bram.upload.tool=ujprog
fpga_de0_nano_bram.upload.wait_for_upload_port=false
fpga_de0_nano_bram.upload.native_usb=false
fpga_de0_nano_bram.upload.maximum_size=31232
fpga_de0_nano_bram.upload.protocol=hex
fpga_de0_nano_bram.upload.flags=-P{serial.port} -ra
```

boards.txt

```
fpga_de0_nano_bram.menu.cpu.riscv=RISC-V
fpga_de0_nano_bram.menu.cpu.riscv.cpu.arch=riscv
fpga_de0_nano_bram.menu.cpu.riscv.compiler.c.flags={f32c.ris
cv.compiler.c.flags}
fpga_de0_nano_bram.menu.cpu.riscv.compiler.cpp.flags={f32c.r
iscv.compiler.cpp.flags}
fpga_de0_nano_bram.menu.cpu.riscv.compiler.ld.flags={f32c.ri
scv.compiler.ld.flags}

fpga_de0_nano_bram.menu.cpu.mips=MIPS
fpga_de0_nano_bram.menu.cpu.mips.cpu.arch=mips
fpga_de0_nano_bram.menu.cpu.mips.compiler.c.flags={f32c.mips
.compiler.c.flags}
fpga_de0_nano_bram.menu.cpu.mips.compiler.cpp.flags={f32c.mi
ps.compiler.cpp.flags}
fpga_de0_nano_bram.menu.cpu.mips.compiler.ld.flags={f32c.mip
s.compiler.ld.flags}
```