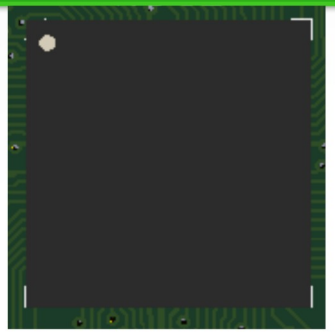
Lesson learned from Retro-uC and search for ideal HDL for open source silicon
3/2/2019

Staf Verhaegen
Chips4Makers

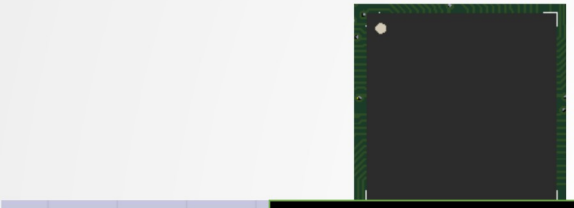
Retro-uC



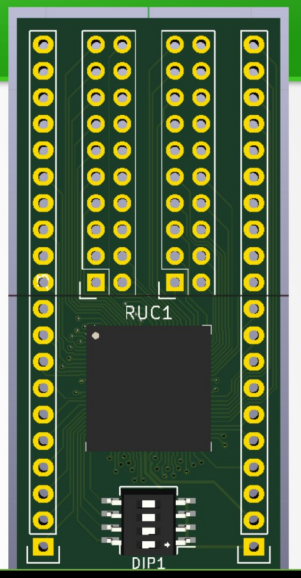
- Open source microcontroller with a Z80, MOS 6502 and Motorola 68000 core
- 4 kB of on-chip RAM
- 72 5 V digital general purpose I/O pins
- JTAG interface for programming the device
- Optionally bootable from external I2C flash memory
- I/O pins that can select the enabled core during reset
- One or more UART, I2C and PWM controllers

Retro-uC

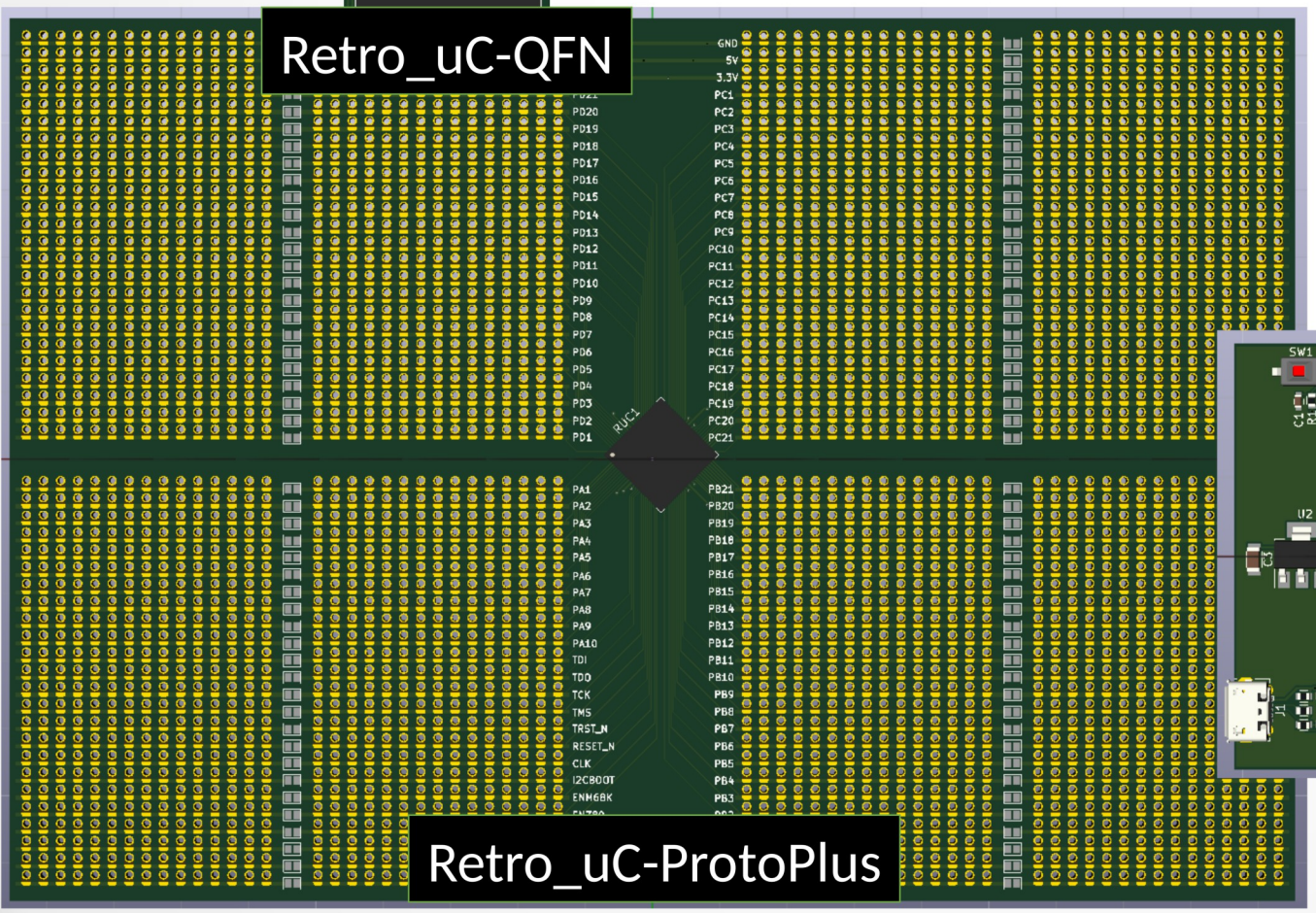
- Planned Product Options



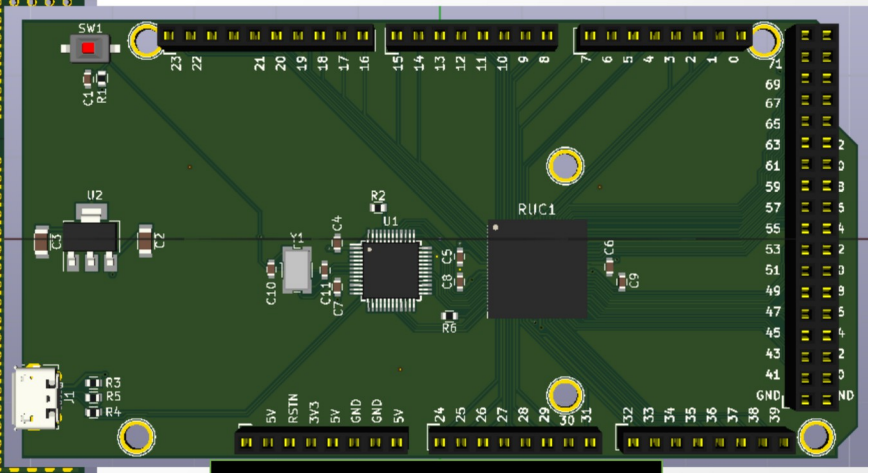
Retro_uC-QFN



Retro_uC-Breadboard



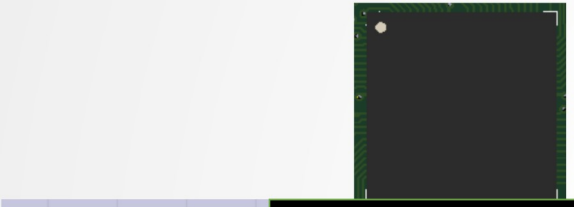
Retro_uC-ProtoPlus



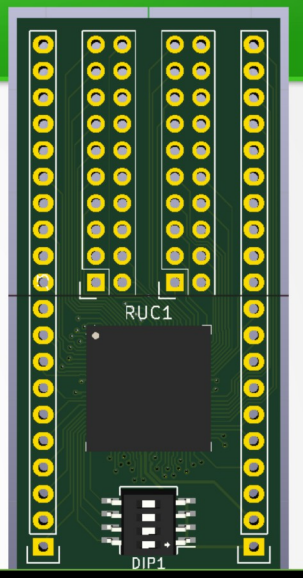
Retro_uC-Retrino

Retro-uC

- Planned Product Options

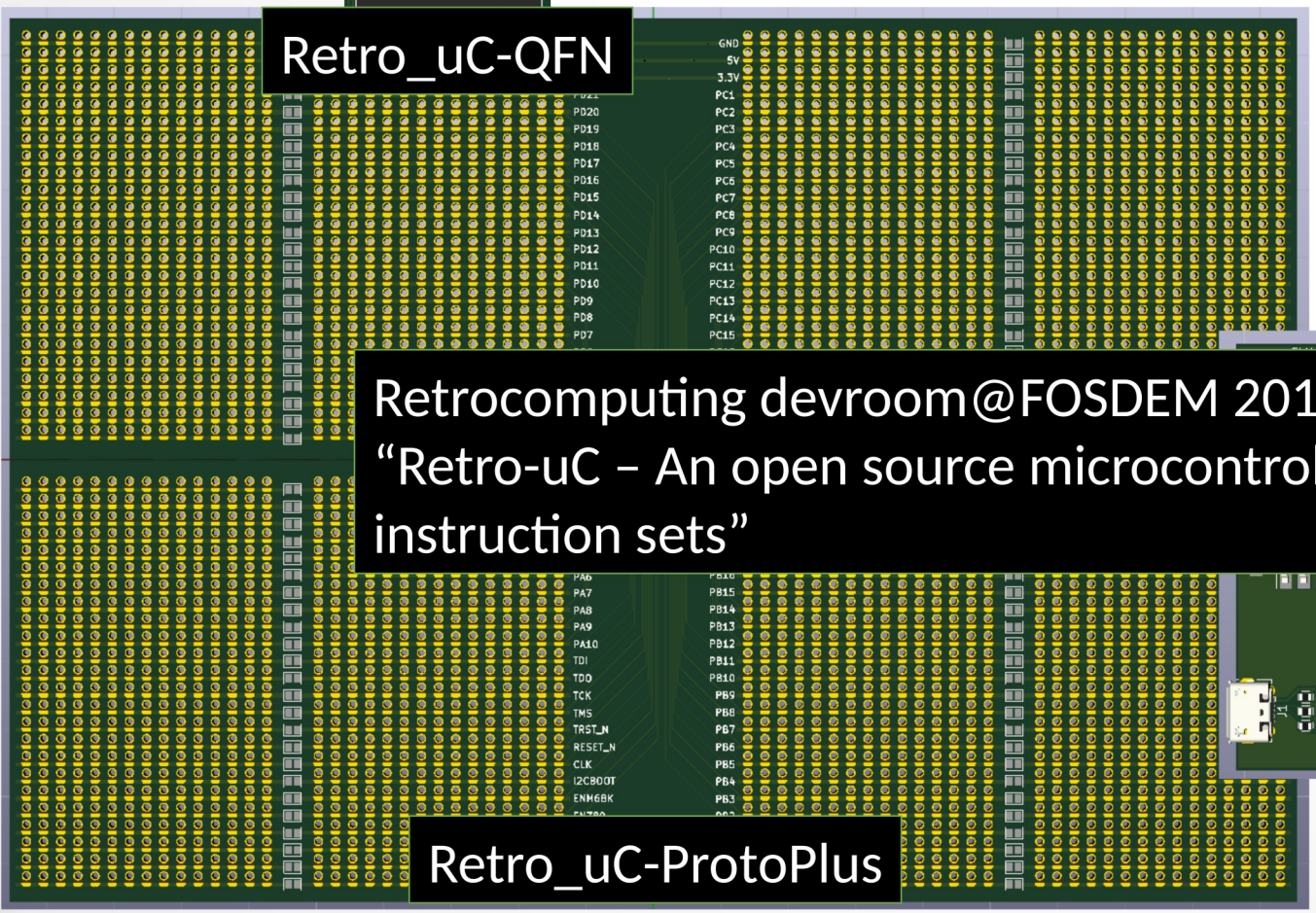


Retro_uC-QFN

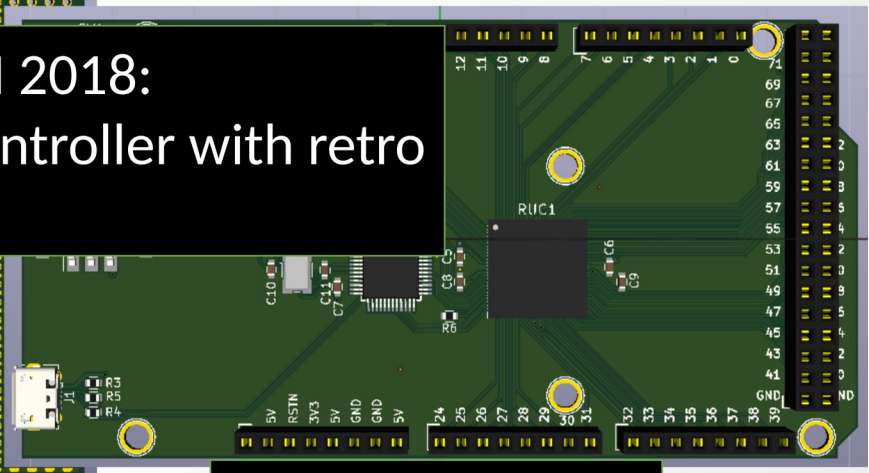


Retro_uC-Breadboard

Retrocomputing devroom@FOSDEM 2018:
“Retro-uC – An open source microcontroller with retro instruction sets”



Retro_uC-ProtoPlus



Retro_uC-Retrino

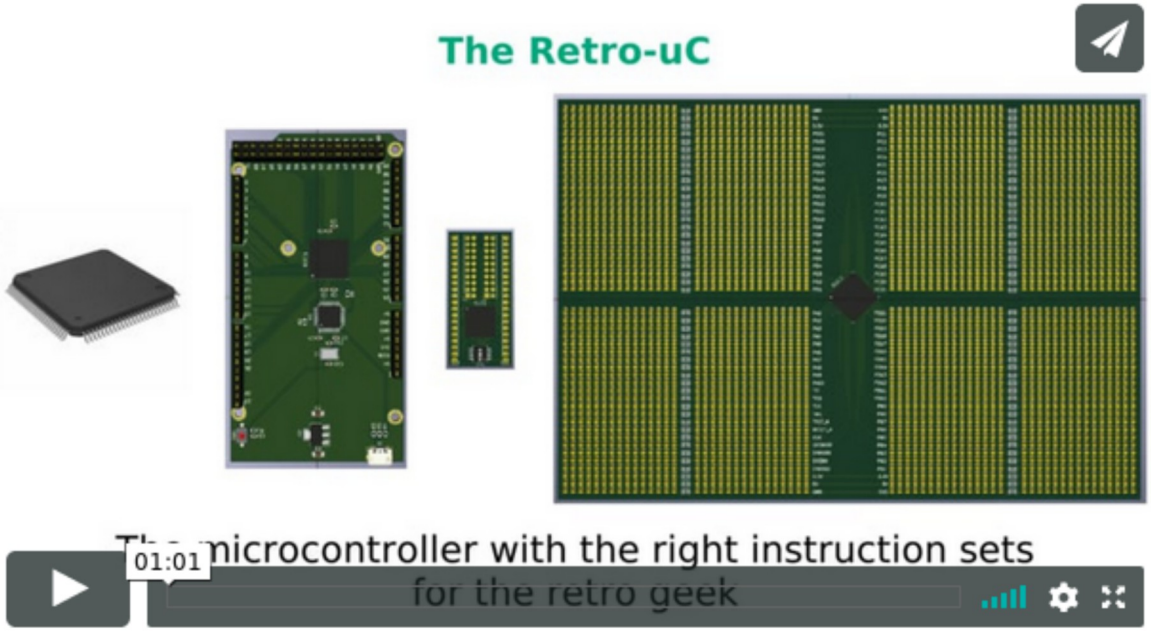
Retro-uC

CROWD SUPPLY BROWSE LAUNCH ABOUT US Search

Education

Retro-uC by Chips4Makers

An open silicon microcontroller with a Z80, MOS6502, and M68K - start the open silicon revolution



$4,034 raised

of $22,000 goal

18% Funded | Time Expired

8	Oct 21	40
updates	ended	backers

Last update posted Oct 24, 2018

me@examp Subscribe to Updates

Recent Updates View all 8 updates.

Campaign End Oct 24, 2018

Support Chips4Makers! $5

Thanks for supporting to get the open

Retro-uC

CROWD SUPPLY BROWSE LAUNCH ABOUT US Search

Education

Retro-uC by Chips4Makers

An open silicon microcontroller with a Z80, MOS6502, and M68K - start the open silicon revolution

Arduino guy:

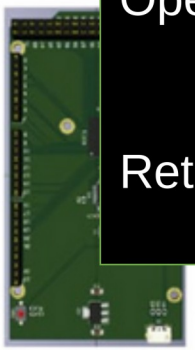
"Costs more than Arduino and has less features"

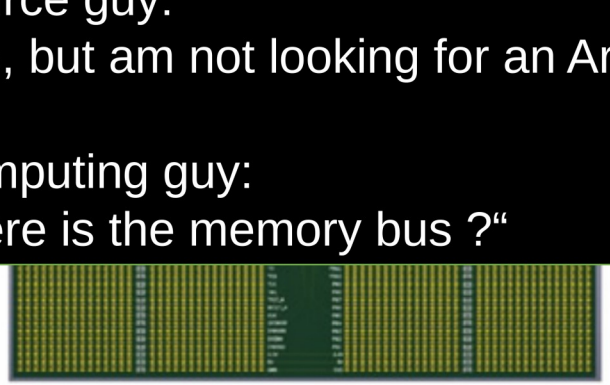
Open source guy:

"Nice, but am not looking for an Arduino board"

Retro-computing guy:

"Where is the memory bus ?"





Time Expired

t 21 nded

40 backers

Last update posted Oct 24, 2018

me@examp Subscribe to Updates

01:01 for the retro geek

Recent Updates View all 8 updates.

Oct 24, 2018

Support Chips4Makers! $5

Thanks for supporting to get the open

Retro-uC

CROWD SUPPLY BROWSE LAUNCH ABOUT US Search

Education

Retro-uC by Chips4Makers

An open silicon microcontroller with a Z80, MOS6502, and M68K - start the open silicon revolution

Arduino guy:

"Costs more than Arduino and has less features"

Open source guy:

"Nice, but am not looking for an Arduino board"

Retro-computing guy:

"Where is the memory bus ?"

First time right

Time Expired

40 backers

Last update posted Oct 24, 2018

me@examp Subscribe to Updates

01:01

Recent Updates View all 8 updates.

Oct 24, 2018

Support Chips4Makers! $5

Thanks for supporting to get the open

Overview

- Retro-uc
- First time right
- Would-be solutions
- Right direction
- I want more

First time right

- IMHO: the main RTL faults
  - Clock is logic signal
    event based on rising and falling edge
  - Need to know what is inferred
    if <> flip-flop or mux
  - Synthesizable versus non-synthesizable
  - Blocking versus non-blocking statement
  - FPGA vs ASIC
  - RTFLRM

Overview

- Retro-uc
- First time right
- Would-be solutions
- Right direction
- I want more

Would-be solutions

- RTL improvements
  - Verilog: -1995, -2001, -2005, SystemVerilog
    VHDL: -1987, -1993, -1900, -2002, -2008
  - New constructs
    - Signed
    - @*, process (*)
    - generate
    - Record interface
  - But IMO no fix of fundamental problems

Would-be solutions

- TL-Verilog (https://www.redwoodeda.com/)
  - Improved for defining pipelines easily:

```
\TLV_version 1a: tl-x.org
\SV
   // SystemVerilog module definition could go here.

\TLV  // enables TL-Verilog constructs
   |calc   // a pipeline, called "calc"
      ?$valid   // condition under which |calc transaction is valid

         // c = sqrt(a^2 + b^2), computed across 3 pipeline stages
         @1
            $aa_squared[31:0] = $aa * $aa;
            $bb_squared[31:0] = $bb * $bb;
         @2
            $cc_squared[31:0] = $aa_squared + $bb_squared;
         @3
            $cc[31:0] = sqrt($cc_squared);
```

  - But IMO no fix of fundamental problems

Would-be solutions

- SystemC/TLM
  - C++ class library; event driven or TLM (transaction-level modeling)

```
#include "systemc.h"
SC_MODULE(nand2)           // declare nand2 sc_module
{
  sc_in<bool> A, B;        // input signal ports
  sc_out<bool> F;          // output signal ports

  void do_nand2()          // a C++ function
  {
    F.write( !(A.read() && B.read()) );
  }

  SC_CTOR(nand2)           // constructor for nand2
  {
    SC_METHOD(do_nand2);   // register do_nand2 with kernel
    sensitive << A << B;   // sensitivity list
  }
};
```

  - Seems to have similar shortcomings
  - AFAIK TLM mainly for prototyping
    needs to be manually converted to event driven code

Would-be solutions

- C/C++/SystemC for ESL (electronic system-level) design
  - Mainly proprietary tools now (Vivado Hlx, Catapult,...)
    would like to see more love for PandA/BAMBU (https://panda.dei.polimi.it/)
    should investigate GAUT (http://www.gaut.fr/)
  - IMO: mainly for compute/data path application.
    Not good fit for implementing Retro-uC but would still enlarge FOSS EDA tool portfolio.

Would-be solutions

- MyHDL
  - RTL with nice python syntax
    should increase productivity in RTL writing

```
from myhdl import *

def dffa(q, d, clk, rst):

    @always(clk.posedge, rst.negedge)
    def logic():
        if rst == 0:
            q.next = 0
        else:
            q.next = d

    return logic
```

  - Based on same event driven principles as other RTLs; by design by author

Overview

- Retro-uc
- First time right
- Would-be solutions
- Right direction
- I want more

Right direction

- Chisel

```
import chisel3._
import scala.collection.mutable.ArrayBuffer

/** Four-by-four multiply using a look-up table.
*/
class Mul extends Module {
  val io = IO(new Bundle {
    val x   = Input(UInt(4.W))
    val y   = Input(UInt(4.W))
    val z   = Output(UInt(8.W))
  })
  val muls = new ArrayBuffer[UInt]()

  // ------------------------------- \\
  // Calculate io.z = io.x * io.y by
  // building filling out muls
  // ------------------------------- \\

  for (i <- 0 until 16)
    for (j <- 0 until 16)
      muls += (i * j).U(8.W)
  val tbl = Vec(muls)
  io.z := tbl((io.x << 4.U) | io.y)
}
```

- SpinalHDL

```
class Counter(width : Int) extends Component{
  val io = new Bundle{
    val clear = in Bool
    val value = out UInt(width bits)
  }
  val register = Reg(UInt(width bits)) init(0)
  register := register + 1
  when(io.clear){
    register := 0
  }
  io.value := register
}
```

  - Both based on Scala

Right direction

- Migen/MiSoC/nmigen

```
from migen.fhdl.std import *
from migen.fhdl import verilog

class Blinker(Module):
    def __init__(self, led, maxperiod):
        counter = Signal(max=maxperiod+1)
        period = Signal(max=maxperiod+1)
        self.comb += period.eq(maxperiod)
        self.sync += If(counter == 0,
                        led.eq( ~led),
                        counter.eq(period)
                    ).Else(
                        counter.eq(counter - 1)
                    )
```

  - Hardware generation next to description
  - Currently my preference but need still development

Overview

- Retro-uc
- First time right
- Would-be solutions
- Right direction
- I want more

I want more

- Compiling and debugging on the higher level
  - Now most of the time HDL converted in RTL and need to be debugged there
    Chisel allows debugging on FIRRTL level
    migen allows to simulate/verify in python (with restrictions)
  - Need compiler and debugger on the same level as the HDL next to testbenches
    like gcc/g++/gnat/... and gdb for programming code