Helpful notes from chat about ULX3S

t=0x8bf3ec8 [0,0]

Contents: [Dobrica PavlinuÅ¡iÄ 's random unstructured stuff]

i2c

https://gitter.im/ulx3s/Lobby?at=5dfcf29acf771f7708ff69e9

It should not be that hard. I have i2c in SaxonSoc in other projects such as https://github.com/SpinalHDL/SaxonSoc/blob/dev/hardware/scala/saxon/board/blackice/BlackiceSocArduino

The generated board support packages include a generated dts, but it is not used yet and the simple i2c generator that I wrote does not generate the dts.

There is a problem building SaxonSoc Linux, that some build randomly do not work. It seems to be something to do with SDRAM access.

@Dolu1990 is about to redo the SDRAM access for the Ulx3s, which should make it more reliable and faster, as he plans to support double frequency access.

There is a lot of information on the development of the u-boot version here , which might be useful to you - SpinalHDL/SaxonSoc#7

https://github.com/SpinalHDL/SaxonSoc/pull/7

I2c Linux support will also need a spinal.lib driver here - https://github.com/SpinalHDL/linux/tree/linux-5.0.y/drivers/i2c

The terasic De1Soc version of SaxonSoc Linux has a dts entry for an led for disk access - https://github.com/SpinalHDL/buildroot/blob/saxon/board/spinal/saxon_default/spinal_saxon_default_de1_so

We really need a better GPIO mapping for the Ulx3s. That might involve including adding a second gpio peripheral to the hardware (gpioB) and doing a better lpf file mapping to pins. It might mean increasing the niumbers of pins that support interrupts. It would be good to include access to the buttons and switches and to make it easy to add Pmods that need interrupt support like the enc28j60 one.

ecp5pll

> Mario Hoffmann @hoffma_gitlab Aug 05 16:11

Hello, I wanted to have a 90 degree shifted clock. I was trying out the ecp5pll from @emard. I get it running and the clock adjustment seems to work, but somehow it does not get shifted for me. Is this currently not working with yosys and so forth, is there anything I have to take care of, or might it just be a bug on my side? Tried to search this chat a bit and just searched a bit on the internet but the things I found about it were quite old and not necessarily about my problem. Any ideas? I am using the vhdl version btw

> emard @emard Aug 05 16:59

There's some order of signals and small delays between them for dynamic shifting to work. https://github.com/emard/ulx3s-misc/blob/master/examples/sdram/memtest_mister/hdl/top/top_memtest.v#L here is module to control phase shifts by pressing of BTN for testing of SDRAM with variable phase shift of the clock to chip

> Mario Hoffmann @hoffma_gitlab Aug 05 17:01

alright, ill look at that. thanks

> Mario Hoffmann @hoffma_gitlab Aug 05 17:53

Oh nice it works. My thinking was just a bit wrong and there was a little bug too. Thanks

> emard @emard Aug 05 18:07

ecp5pll has possibility for fine-precision phase adjustment but this simple BTN module doesn't generate proper phaseloadreg signal
https://github.com/emard/ulx3s-misc/blob/master/examples/sdram/memtest_mister/hdl/btn_ecp5pll_phase.v

display

https://gitter.im/ulx3s/Lobby?at=5f3b7df7ee58011680b36cb0

> emard @emard Aug 18 09:06

SSD1331 96x64 is full featured OLED beautiful contrast, but currently best buy is ST7789 1.3" 240x240 LCD less contrast but more pix and less $. Do not solder directly the display. Solder female 7-pin header and plug the display. Mechanical stabilization can be 3D printed https://github.com/emard/ulx3s/tree/master/box it will "work" properly only if whole box is printed. There is no problem with heat normally to FPGA but SSD1331 with wrong software commands can become hot to fry fingers and smoke, it happened to me once, colors faded and I just replaced display. Onboard USB-SERIAL can provide 2nd JTAG channel for openocd debugger of RISCV processor it will work but it will be VERY slow (for debug single step ok, but for transferring Mbps no).

rtc

https://gitter.im/ulx3s/Lobby?at=5f3b86a1a8c178017657caf6

> emard @emard Aug 18 09:43

Yes for RTC I first saw it "works" but after few reboots I found out that it resets to compile time or something. Core could consult onboard RTC using i2c and then set unix time integer counter which will continue ticking afterwards. For setting RTC we have esp32 and other options. Btw import ntptime; ntptime.settime() will set ESP32 localtime() from a pool of network NTP servers and then esp32 can initialize onboard RTC here I have played with this
https://github.com/emard/ulx3s-misc/tree/master/examples/rtc/micropython-mcp7940n/esp32

https://gitter.im/ulx3s/Lobby?at=5f3dbb34750a2741302ea427

> emard @emard Aug 20 01:52

@pnru_gitlab here is 8-bit master interface. I haven't tested it very simple by reading seconds and seems ok
https://github.com/emard/ulx3s-misc/blob/master/examples/rtc/i2c_master/proj/hdl/i2c_master_8bit.v

https://gitter.im/ulx3s/Lobby?at=5f3e2d1c07c30d132a9c4f58

> emard @emard Aug 20 09:58

when we are at verilog, I have found on the net several i2c bidirectional bridges. Only one code worked at our board but it works only if compiled by diamond. trellis compiles but doesn't work. Can someone take a look what I have missed: bridge: https://github.com/emard/ulx3s-misc/blob/master/examples/rtc/micropython-mcp7940n/proj/hdl/i2c_bridge.v toplevel usage: https://github.com/emard/ulx3s-misc/blob/master/examples/rtc/micropython-mcp7940n/proj/top/top_i2c_brid

emard @emard Aug 20 22:04

for me it works for any gpdi_scl/sda PULLMODE=UP, DOWN, or NONE so it's not pull setting. I can prepare a bitstream and micropython to ulx3s-bin example that initializes the clock setting time from NTP

emard @emard Aug 20 22:18

https://github.com/emard/ulx3s-bin/tree/master/esp32/micropython/rtc can you try this RTC esp32 example for NTP setting? >>> rtcdemo.mcp.time should advance seconds

emard @emard Aug 21 00:04

RTC traffic also appears at GPDI connector (if there's i2c chip 3.3V->5V adapter soldered on board PCA9306D). Some monitors may hold i2c lines. Also gpdi connector can be used to monitor traffic if e.g. you have hdmi breakout board
If you don't have battery, RTC will keep setting until board is powered off
If you have 2 ULX3S boards and mini-display OLED/LCD, you can connect 2 ULX3S together and on one run scopeio, which will monitor the other i2c traffic
I can prepare scope stuff for such setup because scopeio is vhdl, advanced so much that ghdl won't have chance in near future to compile it

emard @emard Aug 21 00:09

UPS but we have problem there will be 2 RTC chips colliding address :)
about i2c master 8bit. First write highest bytes 3,2,1 (order not important) and last write byte 0, this should initiate i2c transaction. byte 3=0x80 is READ, byte 3=0x00 is write
If you write 0x00 to register 0x00 (seconds) it will stop RTC. To start, it needs 0x80 written to 0x00 (MSB bit must be set) then the RTC should start "ticking"

emard @emard Aug 21 18:14

I fixed i2c_bridge.v to work with both trellis and diamond.

Paul Ruiz @pnru_gitlab Aug 21 23:35

Which way does the battery go? I think with the + side (cap) up and the - side (ribbed) to the PCB? @emard: thanks for all the links, but your code already appears to work. The long one is interesting, it uses the same two level state machine idea that my non-working i2c controller uses.

emard @emard Aug 21 23:59

Battery goes + up (larger part of battery should be in contact with metallic holder soldered) - down (smaller part of battery in contact to big circular pad on the PCB
Glad to hear good news that my code works - it hasn't been tested on real CPU but I made some BTNs toplevel and a read and write to register 0 worked

scopeio

https://gitter.im/ulx3s/Lobby?at=5f3c27ffa05e464346d2f6ea

emard @emard Aug 18 21:11

@gkankanh MAX11125 is 1Msa/s total so e.g. if you use 4ch then each channel will be 0.25MSa/s per channel. For oscilloscope it, we have ready solution at hdl4fpga/ULX3S/scopeio you will see traces on monitor. For analysis, onboard USB-serial can do 3Mbps so it could be nearly useable. For faster ADC, yes 100 Mbit ethernet ETH8720 from ebay, module for 2.2$ and for example ebay's AN108 AD/DA module 32MSa/s input, 125MSa/s output
https://www.ebay.com/itm/ADDA-Module-Data-Signal-Acquisition-High-speed-Directly-pluggable-connector-
also scopeio supports it

how to solder headers

https://gitter.im/ulx3s/Lobby?at=5f3cd88378f4a801801336cb

emard @emard Aug 19 09:45

To have GP/GN not be swapped from "default" design, Either solder 90Â° FEMALE headers on top side of board (nice for PMODs directly) or straight 0Â° MALE pins down on bottom side of board. PMODs can also plug to other end of flat cable and pinout will be identical as if 90Â° was soldered onboard.

nmingen

https://gitter.im/ulx3s/Lobby?at=5f3cfbce8b8d4f633effcea7

Lawrie Griffiths @lawrie Aug 19 12:15

I am using Ubuntu 20.04, so I need to use pip3 not pip. (You can't get python2 pip on 20.04 easily).
The installation instructions for nmgen-boards says "Todo", so I installed it like the m_labs version said.
I changed the blinky example to use ULX3S_85F_Platform and ran that. It complained that tool {} was missing.
It seemed that it needed OpenFpgaLoader for upload, so I installed that, and then the blinky worked.

```
from nmigen import *
from nmigen_boards.ulx3s import *


class Blinky(Elaboratable):
    def elaborate(self, platform):
        led   = platform.request("led", 0)
        timer = Signal(26)

        m = Module()
        m.d.sync += timer.eq(timer + 1)
        m.d.comb += led.o.eq(timer[-1])
        return m


if __name__ == "__main__":
    platform = ULX3S_85F_Platform()
    platform.build(Blinky(), do_program=True)
```

https://github.com/GuzTech/ulx3s-nmigen-examples

https://github.com/greatscottgadgets/luna

SDRAM memtest

https://gitter.im/ulx3s/Lobby?at=5f4ea96249a1df0a12c0d83e

> Lawrie Griffiths @lawrie Sep 01 22:04

@pnru_gitlab @Dolu1990 asked these questions about using the SDRAM, which I thought you might know something about from all your recent work on SDRAM drivers:
I'm thinking about the SDRAM
currently, the soc is at 50 mhz, and the sdram run at 100 Mhz using DDR io
but maybe we should quad pump the SDRAM, and doing some bootloader calibration to ajust read delays
i'm just currently not sure what is the critical path of the SDRAM chip themself
in other words "why they are specified to X frequancy and not more"

> Dolu1990 @Dolu1990 Sep 01 22:05

Moaaaar powaaaaaaaa

> emard @emard Sep 01 22:36

oooh yeea :)) if you want to push SDRAM near the edge and give it some heat, on selected designs it can push 133MHz chips to 180-200 MHz, fmax depends on each board. 12F performs better than 85F. Here's memtest
https://github.com/emard/ulx3s-misc/tree/master/examples/sdram/memtest_mister shows results on DVI monitor and with BTNs can adjust phase shift dynamically and watch for errors.

Dolu1990 @Dolu1990 Sep 01 23:01

<3
Nice thanks :)
So this test controle the shift of the clock sent to the DRAM ?
(it doesn't use the programable input delay ?)

emard @emard Sep 01 23:08

Yes this has a classic sdram driver that normally needs 90Â° phase offset to chip hardware, but as fmax is getting higher the actual phase shift which makes it really work moves. PLL is used to provide phase shift. Paul has made a better sdram driver with cool vendor-independent delay solution with a number of NOT gates.Only ns delay per NOT gate remains vendor dependent

Dolu1990 @Dolu1990 Sep 01 23:09

ok :D

Paul Ruiz @pnru_gitlab 00:02

@lawrie @Dolu1990 I am not sure I understand the SDRAM questions. In any case, my latest version is here: https://gitlab.com/pnru/cortex/-/blob/master/sdram.v
In particular note new lines 45-47 - I am not sure why, but this mod generally pushes Fmax up to about 200MHz (it depends a bit on the NextPNR seed).

I am not sure what you mean by "using DDR io" - does the SDRAM chip on the ULX3S support DDR? Maybe you mean by DDR that it runs at twice the speed of the CPU or that it uses burst size 2?

I don't know what the critical path in the SDRAM chip is, but I do have a hypothesis. When working with a CAS delay of 3 clocks, the data really arrives after 2 clocks plus 6-7ns (spending on the speed grade). If you clock a grade 6 chip (PC166) faster, a clock cycle will take less than 6ns and the data will only arrive after the third rising clock edge. My guess is that the 6-7ns is related to the speed of the sense amplifiers or something like that.

resource utilization

https://gitter.im/ulx3s/Lobby?at=5f4f8c30d4f0f55ebbf93007

Dolu1990 @Dolu1990 Sep 02 14:12

Is somebody aware of a way to get hearchical ressource utilisation report out from yosys/next-pnr for ECP5 ?
Basicaly, trying to nail down the ressource usage

David Shah @daveshah1 Sep 02 14:18

You can get a hierarchical report with Yosys by passing -noflatten to synth_ecp5

ps2 keyboard

https://gitter.im/ulx3s/Lobby?at=5f57e59c59ac794e02f71d14

Kid CUDA @KidCUDA_gitlab Sep 08 22:12

anyone used a PS2 keyboard with ULX3S?
like a proper PS2 keyboard with the pins adapted to USB?
is a level shifter needed or is the USB port 5V-data-tolerant?
from the schematic it doesn't look 5V tolerant but I'm not sure how else it would work with just a pure PS2 adapter as suggested in the docs

emard @emard Sep 08 22:52

@KidCUDA_gitlab US2 pins are 5V tolerant, limited by R and Zener diodes. Still some PS/2 keyboards don't accept 3.3V levels. Best is to obtain combo PS/2+USB they usually work in both modes for ULX3S
PS/2 is normally used over OTG connector for most of our retro-computing cores, apple1-2, ti99, zx, vic20, QL just to name a few

cortex

source: https://gitlab.com/pnru/cortex

https://gitter.im/ulx3s/Lobby?at=5f500a7eec534f584fdbeb15

Paul Ruiz @pnru_gitlab Sep 02 23:11

8s is super-comfortable! Btw I wonder how did cortex start, before it ever booted they need some filesystem to hold files. Is cortex filesystem mountable by modern linux? How did they made it in early times?

The Cortex was a traditional home computer with Basic in its day. Running Unix on it was my project some 6-7 years ago. It was a long journey: porting a C compiler and tool chain, building simple kernels with a linked in user program (downloaded to the H/W via something similar to S-records), etc. When the time for disk access came, I used a tool to create & manage disk images.

For the original Unix, the file system was almost the first thing that was built, after the assembler (that is how a.out got its name: assembler output). An empty disk image was written by a custom format program. Files were then loaded from paper tape. Some 1969/1970 Unix code can be found here:
https://www.tuhs.org/cgi-bin/utree.pl
In its first incarnations it was all assembler, but many of the core ideas were already there. Some more background is here:
https://www.bell-labs.com/usr/dmr/www/hist.html

emard @emard Sep 02 23:14

Ahaaaa so unix was not all the time available on cortex hardware. Still I'd wanted to know how did you initially start with populated filesystem. Normally e.g. if we have linux on riscv, we can mount the same partition on x86 PC, copy files and and it will work on riscv, but how was this done on cortex?

So If I understood, you have a tool that from a directory creates disk image, but there's currently no support to actually mount cortex fs on linux for example. linux has some possibility to write a user-space fs driver like "fuse" but I ghess thats difficult and fragile

Paul Ruiz @pnru_gitlab Sep 02 23:29

You can follow my journey here, in 315 commits:
https://www.jslite.net/cgi-bin/9995/timeline?n=400&y=all&v=0

I use a program ("ufs") which creates a disk image from scratch and then adds files to it. The source code is here:
https://www.jslite.net/cgi-bin/9995/dir?ci=84b2a75947eb76db&name=fsutil

Even on the mini Cortex hardware, the CF Card uses FAT formatting and has an image file on it. I make sure the image is contiguous and the boot loader lets the Unix disk driver know in which sector the image starts. This way I can simply copy disk images to the CF card without needing to use special tools.

Actually, the card also has disk images for other OS's as well - MDEX and NOS, which are somewhat similar to CP/M and MS-DOS 3 respectively.

emard @emard Sep 02 23:44

There has been a lot of concentrated effort! The idea to use contiguous file in FAT is very good, so the CF itself can be easily written from laptop.

Paul Ruiz @pnru_gitlab Sep 02 23:53

Thank you. The ulx3s Cortex has it even easier: because of the ESP32, now I don't even have to worry about things being contiguous and I can ftp disk images without having to handle the SD card.

emard @emard Sep 03 00:34

yeees it was a piece of luck that for esp32 appeared good micropython support with almost all important things working and that spi-jtag adventure turned out successful. I have ulx3s with SD in a box and once inserted SD I never move out, just ftp files. Things will be even better when WROVER-E prototype starts working, 4MB RAM, 16MB FLASH no more out of memory. Bitstreams could be unzipped on-the-fly, even larger FLASH chips supported with 64K and 256K erase blocks (esp32 must buffer data size of erase block and now we are struggling with 4K buffers)

saxonsoc audio

https://gitter.im/ulx3s/Lobby?at=5f6482c3603d0b37f43d3ec0

Lawrie Griffiths @lawrie Sep 18 11:49

I have a 4-cpu 85F SaxonSoc version with music, working now.

It is now inSmp/bitstreams/ulx3s_85f_blue_4core_saxonsoc.bit
I renamed images as oldimages and the new one are in Smp/images.
You need dtb, uImage and you need to untar the new rootfs.tar.
You will also need:

```
root@buildroot:~# cat .asoundrc
pcm.!default {
    type            plug
    slave.pcm       "softvol"   #make use of softvol
}

pcm.softvol {
    type         softvol
    slave {
        pcm         "hw:0,0"      #redirect the output to dmix (instead of "hw:0,0")
    }
    control {
            name        "PCM"       #override the PCM slider to set the softvol volume level globa
        card     0
    }
}
```

To play music do: mpg123 -T -f 4096 -m file.mp3.
Or to play in the background nohup mpg123 -T -f 4096 -m file.mp3 &
It is set up for a 64MB blue 85f.

https://gitter.im/ulx3s/Lobby?at=5f648d8cf51808513b4f7db5

olu1990 @Dolu1990 Sep 18 12:35

.asoundrc isn't necessary, it just add volume controles in alsamixer app
I would suggest to not add the .asoundrc for single core versions, as it add quite a bit of overhead
the -m of mpg123 is for mono, if the mp3 bit rate isn't to high, it might be fine in stereo
(for single core)

saxonsoc rtc

https://gitter.im/ulx3s/Lobby?at=5f69eb5d6a6e094525ac61f5

https://gitter.im/ulx3s/Lobby?at=5f69f087e1dd7c19548aad12

Dolu1990 @Dolu1990 Sep 22 14:39

got the rtc to start counting seconds and read it via :

```
i2cset -y 0 0x6F 0x00 0x80
sleep 4
i2cget -y 0 0x6F 0x00
```

https://gitter.im/ulx3s/Lobby?at=5f6a4562e1dd7c19548b96f0

Lawrie Griffiths @lawrie Sep 22 20:41

```
oot@buildroot:~# cat date.sh
R6=`i2cget -y 0 0x6f 0x06`
R5=`i2cget -y 0 0x6f 0x05`
R4=`i2cget -y 0 0x6f 0x04`
R2=`i2cget -y 0 0x6f 0x02`
R1=`i2cget -y 0 0x6f 0x01`

YY="${R6:2:2}"
MON="$((${R5:2:2}-20))"
DD="${R4:2:2}"
HH="${R2:2:2}"
MM="${R1:2:2}"

echo "20$YY-$MON-$DD $HH:$MM"
root@buildroot:~# ./date.sh
2020-9-22 19:40

date -s "`./date.sh`"
```

https://gitter.im/ulx3s/Lobby?at=5f6a69588fe6f11963554984

emard @emard Sep 22 23:15

```
#include <stdio.h>
#include <stdlib.h>

#define I2C_SLAVE 0x703
#define O_RDWR 2

int i2c_rtc;

void rtc_open(int addr)
{
  i2c_rtc = open("/dev/i2c-0", O_RDWR);
  ioctl(i2c_rtc, I2C_SLAVE, addr);
}

void rtc_read(unsigned char *buf, int reg, int n)
{
  buf[0] = reg;
  write(i2c_rtc, buf, 1);
  read(i2c_rtc, buf, n);
}

void i2cdemo(void)
{
  int i;
  unsigned char buf[7];
  // mask for BCD          SEC   MIN   HOUR  WKDAY DAY   MONTH YEAR
  unsigned char mask[7] = {0x7F, 0x7F, 0x3F, 0x07, 0x3F, 0x1F, 0xFF};

  rtc_read(buf, 0, sizeof(buf));
  for(i = sizeof(buf)-1; i >= 0; i--)
    printf(" %02x", buf[i] & mask[i]);
  printf("\n");
}

int main(int argc, char *argv[])
{
  int i;
  rtc_open(0x6F);
  for(i = 0; i < 60; i++)
  {
    i2cdemo();
    sleep(1);
  }
  return 0;
}
root@buildroot:/home/root/rtc# ./a.out
 20 09 22 02 21 14 27
 20 09 22 02 21 14 28
 20 09 22 02 21 14 29
 20 09 22 02 21 14 30
 20 09 22 02 21 14 31
```

saxonsoc jtag

https://gitter.im/ulx3s/Lobby?at=5f78c209dfe47e4d57464c10

Lawrie Griffiths @lawrie Oct 03 20:25

The pins to connect to are these - https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.1/hardware/synthesis/radiona/ulx3s/smp/ulx3s_v20_linu

emard @emard Oct 03 20:26

I tried to upload SVF file and it works, (fast). Yes, this pins connection is important. Is there a simple JTAG openocd scan command I can test it to make sure I connected all
Some command similar to

```
jtag newtap lfe5 tap -expected-id 0x21111043 -irlen 8 -irmask 0xFF -ircapture 0x5
```

Dolu1990 @Dolu1990 Oct 03 20:29

hoo when openocd run it scan everything

emard @emard Oct 03 20:29

yes yes I see it on above linked script

Dolu1990 @Dolu1990 Oct 03 20:30

i'm not sur there a command to rescan, just rerun it ^^

emard @emard Oct 03 20:31

OK I need to make a bit on the solder and pins to board, then I will test it and when I get vexrisc scanned by openocd jtag I will put it online for remote access

Dolu1990 @Dolu1990 Oct 03 20:31

you should get a scan like 0x10001FFF

Lawrie Griffiths @lawrie Oct 03 20:33

@Dolu1990 Doesn't @emard need to build your version of openocd - https://github.com/SpinalHDL/openocd_riscv

Dolu1990 @Dolu1990 Oct 03 20:34

Yes
so :
You can source the source.sh, and then do a saxon_clone; saxon_openocd
this will build it
Then modify
https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.1/bsp/radiona/ulx3s/smp/openocd/usb_connect.cfg#L2
to match the jtag you have installed, and then saxon_openocd_connect

Dolu1990 @Dolu1990 Oct 03 21:02

sudo apt-get install libtool automake libusb-1.0.0-dev texinfo libusb-dev libyaml-dev pkg-config for the full depedency on debian

emard @emard 00:00

```
Escape character is '^]'.
Open On-Chip Debugger
> init
> scan_chain
   TapName             Enabled  IdCode     Expected   IrLen IrCap IrMask
-- ------------------- -------- ---------- ---------- ----- ----- ------
 0 fpga_spinal.bridge     Y     0x10001fff 0x10001fff     4 0x01  0x0f
```

Got openocd to connect

https://gitter.im/ulx3s/Lobby?at=5f7c33646e0eb844696af0a2

Lawrie Griffiths @lawrie 11:05

You then need a usb JTAG device, preferably an FT2232 one, and you connect the pins to gn0 - gn3. I have not used openocd with SaxonSoc for a while, but @Dolu1990 uses it all the time.

micropython socks

https://gitter.im/ulx3s/Lobby/archives/2020/10/10?at=5f820bfb07361f0cc6430489

kost @kost Oct 10 19:31 UTC

since you guys around SaxonSoc made such a great progress. I had to do my part of the promise.
Just released socks server for micropython at https://github.com/kost/micropython-socks
that means you can tunnel any SOCKS5 connection over ESP32
since micropython does not come with NAT support, that means you can go to the internet over ESP32 using SOCKS server.
Installation is simple if you have connected ESP32 already to the internet:
You have to run this on ulx3s repl shell:

```
import upip
upip.install('micropython-socks')
```

and then you can just simply say:

```
import socks
socks.start()
```

it will start listening on 0.0.0.0:1080 for SOCKS5 connections.
Then you can simply from SaxonSoc test it with the following (or any other host):

```
curl --socks5 192.168.4.1:1080 http://ifconfig.co
```

nmigen - OV7670 with st7789

https://gitter.im/ulx3s/Lobby?at=5f949a4e6c8d484be2a6df34

Lawrie Griffiths @lawrie Oct 24 23:19

@goran-mahovlic I now have a version of the OV7670 camera application working in nmigen with the st7789 display - https://github.com/lawrie/ulx3s-nmigen-examples/tree/master/ov7670

oberon

https://gitter.im/ulx3s/Lobby?at=5f949d40270d004bcfea9716

Charles Perkins @charlesap Oct 24 23:31

@pnru_gitlab @emard I have created a new disk image based on the www.projectoberon.com sources, only adding some more LED output in the inner core boot process.

Here's a zip file of a disk image that you should be able to directly write to an SD card: https://github.com/io-core/io/blob/main/images/ledboot.img.zip ... this image has the Oberon partition at the correct offset but I did not bother to format the first part with a FAT partition. It boots on my ulx3s with the earlier Oberon .bit file.

```
Here's the meaning of the LEDs on booting:
LEDs: 7------- In BootLoad Firmware
LEDs: 7-----1- In BootLoad Firmware
LEDs: 7----2-- In BootLoad Firmware
LEDs: -------0 Control transferred to Modules
LEDs: ------1- Control transferred to Files
LEDs: -----2-- Control transferred to Kernel
LEDs: ----3--- In Kernel - Temporary Trap installed
LEDs: ---4---- In Kernel - Stack and heap origins and limits configured
LEDs: --5----- Kernel SecMap initialized, control transferred to FileDir
LEDs: -6------ Directory traversal complete
LEDs: 7------- Sectors marked, control transferred to Modules which has loaded Oberon.
LEDs: -------- Icons are defined, display subsystem initialized, ready to load System.
LEDs: --5----0 GC in the Oberon Loop
LEDs: --5---10 GC in the Oberon Loop
LEDs: --5--210 GC in the Oberon Loop
LEDs: --5-----
```

If there is an early trap before the system introduces a nicer printing trap the leds will fast-blink now with the trap value (0-7).
If the system gets an error trying to load modules after the boot loader successfully loads the kernel (which Oberon needs to actually display graphics and text to the screen) then this code fast-blinks bit zero.

The Oberon source code modified with the LED output is here:

https://github.com/io-orig/projectoberon/blob/main/Kernel.Mod.NL
https://github.com/io-orig/projectoberon/blob/main/FileDir.Mod.NL
https://github.com/io-orig/projectoberon/blob/main/Files.Mod.NL
https://github.com/io-orig/projectoberon/blob/main/Modules.Mod.NL
https://github.com/io-orig/projectoberon/blob/main/Oberon.Mod.NL

oled pins

https://gitter.im/ulx3s/Lobby?at=5fa2bcb374152347c213d4ae

> emard @emard Nov 04 11:59

@sthornington yes bitstream projects do rearrange pins. At least GND and 3.3V must match, FPGA is flexible about pins, practical is to plug display directly without wires. Original markings are for SSD1331. ST7789 7-pin is similar but I think pin has BL (backlight) function instead of CS.
wifi_gpio17 refers to the same thing and also ESP32 is flexible about SPI pinout so you can match practicaly any combination.
esp32 Channels 1 or 2 itself are the same but if you mount SD card from ESP32 it will always use channel 1 so channel 2 remains free. Without SD channel 1 or 2 are for display the same.

@sthornington of course GND and VCC can't be swapped by FPGA, because OLED draws current and must connect to hard power supply, can't be powered from FPGA 16mA pins (so low power devices actually could swap even GND/VCC). new board will have 8-pin LCD header instead of 7-pin but there's really crowded with routing so additional 2 pins are nearly impossible. But if display has GND and VCC swapped, then it will fit to external connector GN/GN 0-27 so there's still a possibility to plug such display directly on board on the side

rtc - i2c master example

https://gitter.im/ulx3s/Lobby?at=5fa7ba79c6fe0131d4e19d31

emard @emard Nov 08 10:29

There is example https://github.com/emard/ulx3s-misc/tree/master/examples/rtc/i2c_master/proj which makes i2c master in verilog, talks to RTC and displays time as HEX on DVI and LCD. There can't be interference with ESP32 because ESP32 is not directly connected to RTC, FPGA is between them.

rt2232 second channel openocd

https://gitter.im/ulx3s/Lobby?at=5fc57a82150b213e980592d8

emard @emard 00:04

@sthornington if you have external ft2232 it is fastest and normally used as openocd jtag debugger for softcore cpus like litex or saxonsoc linux. Secondary US1 channel is possible and fully supported by openocd, but it will be unacceptably slow to transfer big files like kernel or root fs images. https://github.com/emard/ulx3s-jtagthru/blob/master/scripts/ft231x2.ocd here is some project that has openocd script to export secondard jtag channel to external pins but normally you can use it internally to soft-core too. See also the schematics for your board (v3.0.8) how FTDI is connected to FPGA

Simon Thornington @sthornington 01:04

@emard thanks, mostly what I want to get going is an on-board scope to dump traces, is there any particular recommended F2232 interface? Does that plug into the jtag header of the below the oled one?

emard @emard 02:56

any FT2232 breakout board or programmer is ok. It should connect to external pins GP/GN something, depends on where litex/saxonsoc expects them, usually pins 0-5 I guess. I does not plug to JTAG header, it doesn't need to program ECP5 but the CPU RISC5 done by FPGA. If you need onboard scope to display traces in realtime check hdl4fpga project, it has great scope for our

boards.

esp32 passthrough

https://gitter.im/ulx3s/Lobby?at=601c699dd0d32d7d4fc94dfc

liebman @liebman Feb 04 22:39

@emard this one is improved. It tristates gpio0 instead if setting it high so that it can be used elsewhere if needed. Also added an enable that can be used as a reset for the esp32.

```
module ulx3s_passthru (
  input  wire txd,
  output wire rxd,
  input  wire dtr,
  input  wire rts,
  input  wire esp_txd,
  output wire esp_rxd,
  output wire esp_en,
  output wire esp_io0,
  input  wire en,
);
  // TX/RX passthru
  assign rxd     = esp_txd;
  assign esp_rxd = txd;

  // Programming logic
  // SERIAL  ->  ESP32
  // DTR RTS -> EN IO0
  //  1   1     1   Z
  //  0   0     1   Z
  //  1   0     0   Z
  //  0   1     1   0
  assign esp_en  = (~dtr |  rts) & en;
  assign esp_io0 = ( dtr | ~rts) ? 1'bz : 1'b0; // we only want to drive this pin low

endmodule
```

can be called like

```
module top(
    input wire clk_25mhz,
    output wire ftdi_rxd,
    input wire ftdi_txd,
    inout wire ftdi_ndtr,
    inout wire ftdi_nrts,
    output wire wifi_rxd,
    input wire wifi_txd,
    inout wire wifi_en,
    inout wire wifi_gpio0,
    output [7:0] led,
    input  [6:0] btn,
    output wire shutdown,
);
    ulx3s_passthru passthru(.txd(ftdi_txd),
                            .rxd(ftdi_rxd),
                            .dtr(ftdi_ndtr),
```

```
                            .rts(ftdi_nrts),
                            .esp_txd(wifi_txd),
                            .esp_rxd(wifi_rxd),
                            .esp_en(wifi_en),
                            .esp_io0(wifi_gpio0),
                            .en(btn[0]),  // btn[0] will work as a reset for esp
    );

    // blinky for something to do so we know its operational
    assign led[0]   = btn[1];
    assign led[6:1] = 0;
    assign led[7]   = wifi_gpio0;
endmodule
```

liebman @liebman Feb 04 23:18

thats good to know, which are the (non esp) pins that are clock capable?
(that explains why some of my tests failed)

emard @emard Feb 04 23:19

They are mentioned on pdf schematics_v3.0.8 let me see https://github.com/emard/ulx3s/blob/master/doc/schematics_v308.pdf page 2 GP,GN 12 are clock capable and shared with ESP32 but small design fail is those pins are on ESP32 input only. In new board v3.1.5 I tried to fix this by routing one esp32 output capable pin to FPGA clock input capable...

PCLK .. means primary clock capable pins. they are best. GR_PCLK are second best, general routed to primary clock capable

A small fix could be possible with a jumper GN11-GN12 this will connect ESP32 pin 25 GN11 which is output capable to FPGA clock input capable at GN12

lpf documentation

https://gitter.im/ulx3s/Lobby?at=601fa45432e01b4f717e0ebb

Dave Anderson @danderson 09:27

Couldn't find any decent docs other than nextpnr source code and poorly explained technical notes from lattice, so I wrote https://github.com/danderson/ulxs/blob/main/lpf.md
Also comes with pointers to the Lattice tech notes that go into more detail about e.g. ECP5 configuration and I/O pin config.

hdmi

https://gitter.im/ulx3s/Lobby?at=6029a8349337c51bc688733e

splinedrive @splinedrive Feb 14 23:46

Hi, I have done a hdmi reimplementation for ulx3s and blackicemx (ice40) they have the same code base. I hope you like it. I learned from other projects to take the semantic (ulx3s-examples-dvi, fpga4fun, ...) . ulx3s has DDR and SRD support and blackicemx can only DDR. I used the pmod from Luke Wren. It works only with passive resistors and works with long hdmi cables without any problems. https://github.com/splinedrive/my_hdmi_device

osd

https://gitter.im/ulx3s/Lobby?at=603a1016457d6b4a948f3208

Lawrie Griffiths @lawrie 10:25

@sylefeb I am not sure that between us, @emard and I, have documented the OSD and rom loader very well. There are lots of versions of the code in different projects. This is the spi slave from my Z80 template project -
https://github.com/lawrie/ulx3s_z80_template/blob/main/src/osd/spirw_slave_v.v
The rest of the code is in that osd directory.
This is a version of the micropython code that reads and writes memory and controls the cpu remotely from the esp32 - https://github.com/lawrie/ulx3s_z80_template/blob/main/esp32/spiram.py
The rest of the esp32 code including osd.py is in that directory.
Here is a youtube video that shows the osd and loader being used -
https://www.youtube.com/watch?v=YE7pSuZiN9Y&t=8s
The latest version of the osd look a bit nicer.

This is my TRS 80 Model 1 implementation that has a short description on using the OSD -
https://github.com/lawrie/ulx3s_z80_trs80
Perhaps @emard knows of a better description of all this.

The OSD and loader is used by many Ulx3s projects including the Apple II, C64, ZX Spectrum, Mac Plus, QL, TI-99/4A, Amiga (OSD only), Vic 20, NES, SNES, Sega Master System, Orao, etc.
This is a good video by @Speccery that shows the OSD used on the TI-99/4A -
https://www.youtube.com/watch?v=zdST3wz00KU
I don't think there is a Risc-V implementation on the Ulx3s that uses the OSD yet.

emard @emard 12:16

@sylefeb @lawrie OSD loader behaves similar as SPI RAM using 32-bit byte address. FPGA behaves as SPI slave, ESP32 as SPI master. If slave needs to initiate transfer, there is additional IRQ line. Resources at SPI address space are memory mapped, RAM to upload for CPU, reset/halt control, buttons, OSD video chars, floppy disks etc. All is very simple and protocol is not too much standardized so it can be adapted to completely unusual usage. Generally for apple2 c64 vic20 mac trs80 etc we just copy-paste the same thing

saxonsoc memory map

https://gitter.im/ulx3s/Lobby?at=606dcc72bc8e6f2e0d2e5be9

> Dolu1990 @Dolu1990 Apr 07 17:14

@irvise:matrix.org

> 0x340000

That's the flash address. the CPU copy that part of the flash to the SDRAM at 0x80F80000 (global address). See :

> Memory copy :
> source address : OPENSBI_FLASH
> destination address : OPENSBI_MEMORY

https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.3/bsp/radiona/ulx3s/smp/app/bootloaderConfig.h#L99

Then it does similar things with uboot but with that set of addresses :
https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.3/bsp/radiona/ulx3s/smp/app/bootloaderConfig.h#L22

> if I generate a program that load to the address 0x380000 it should "just work"

Yes, as long you programe is complied to sit at 0x80F00000 in the global memory space

> And where can I find more info on what memory addresses are being used for MMIO

This autogenerated header file contains all the peripheral addresses :

https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.3/bsp/radiona/ulx3s/smp/include/soc.h#L64

> blink

See
https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.3/software/standalone/blinkAndEcho/src/main.c
used with
https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.3/bsp/radiona/ulx3s/smp/include/bsp.h#L15

you can compile it using the command "saxon_standalone_compile blinkAndEcho" It will sit where uboot sit (see
https://github.com/SpinalHDL/SaxonSoc/blob/dev-0.3/bsp/radiona/ulx3s/smp/linker/default.ld#L1)

nmigen

https://gitter.im/ulx3s/Lobby?at=608e7b59d5e2793379e961a8

> Lawrie Griffiths @lawrie May 02 12:13

I am currently working on my nmigen OV7670 camera application - https://github.com/lawrie/ulx3s-nmigen-examples/blob/master/image/camtest.py
That now displays the image on the HDMI monitor and I am starting to add image signal processing functions like brightness and color control, and simple edge detection.

The latest code that I have pushed calculates the mouse pointer (p_x, p_y) - https://github.com/lawrie/ulx3s-nmigen-examples/blob/master/image/image_stream.py

If you set the ImageStream filter switch then pixels with a red channel value less than some threshold are filtered out and the remaining pixels are displayed on the HDMI monitor as pure red. You can adjust the threshold with the up and down arrows, and a value of about 30 seems best. As the threshold is increased the picture turns black except where the laser pointer points, and the spot gets smaller as the threshold is increased. I then calculate the min and max x and y values for those red pixels, and calculate the mid points as p_x and p_y. It seems to be working.

saxon esp32ecp5 start

https://gitter.im/ulx3s/Lobby?at=60b93a55a10461235db71cff

> emard @emard Jun 03 22:23

I have just checked saxonsoc linux and it still works, lAN8720 ethernet (ssh) works out of the box. Tetris still compiles and works. A small hint here is the esp32 script for new esp32ecp5 to start linux

```
import os
from machine import SDCard, Pin
import ecp5

os.mount(SDCard(slot=3),"/sd")
# copy to root of SD card: dtb, rootfs.cpio.uboot, uImage
# flash with "False" without starting the bitstream
ecp5.flash("/sd/linux/smp/fw_jump.bin@0x340000",0x340000,False)
ecp5.flash("/sd/linux/smp/u-boot.bin@0x380000",0x380000,False)
ecp5.prog("/sd/linux/smp/ulx3s_12f_1core_saxonsoc.bit")
os.umount("/sd")
p12=Pin(12,Pin.IN)
p13=Pin(13,Pin.IN)
p14=Pin(14,Pin.IN)
p15=Pin(15,Pin.IN)
```

With esp32ecp5 I'd recommend micropython builds with idf3 because idf4 builds reboot ad SD card deinit (ftp> site umount)

dfu and openfpgaloader

https://gitter.im/ulx3s/Lobby?at=6213c5d29a09ab24f36e464d

I suggest new passthru with DFU integrated. Get or compile latest openFPGALoader. connect to US1 and flash this first: You don't need to unzip, openfpgaloader will detect .gz and unzip on-the-fly

```
download https://github.com/emard/ulx3s-bin/blob/master/fpga/dfu/85f-v317/multiboot.img.gz
openFPGALoader -b ulx3s -f --unprotect-flash --file-type raw multiboot.img.gz
```

If you hold BTN1 or set DIP SW1 ON and plug board to US2 (optionally press BTN0 if it doesn't power up at plug) then USB DFU compatible device should be detected and you can use much faster flashing like this

```
openFPGALoader -b ulx3s_dfu bitstream.bit
```

This is for user-bitstream, if you need to reflash bootloader DFU itself from US2, hold BTN1 and BTN2 together and use DFU alternate 5 option back to ESP32: plug to US1 again and try to erase and flash ESP32

bootloader switches

https://gitter.im/ulx3s/Lobby?at=62370af9f43b6d783f0de7d8

emard @emard 12:07

@NostosArch bootloader checks DIP SW1 state on power up. Switching it afterwards doesn't matter. 3 LEDs ON means it is DFU bootloader and passthru, and US2 should enumerate DFU with 3 LEDS it should be possible to flash esp32 with 3 LEDs on. 7 LEDs on means it is in passthru only (not DFU) mode. esp32 should be able to be flashed from passthru-only mode like from DFU+passthru mode. Apart from that I can either say esp32 is flashed with something difficult to get rid of or another PC and OS should be tried.

I didn't follow, but are you using linux microsoft or apple? Does ESP32 respond with ">>>" usbserial 115200 micropyhton prompt on US1?

If you upload blink, then passthru-only mode is overwritten. Still DFU+passthru mode (3 LEDs) is kept (write protected from fujprog, only openfpgaloader can overwrite bootloader). So DFU+passthru 3 LEDS should allow flashing esp32 but of course PCs and OSs compatibility and contents of esp32 already flashed may cause difficulty