**********DRAFT**********

Adapting the ICOBOARD ZIPCPU to the CATBOARD

06/05/18

**********DRAFT**********

The following information is from https://github.com/ZipCPU/zipcpu/blob/master/doc/spec.pdf

Introduction

The goal of the ZipCPU was to be a very simple CPU. You might think of it as a poor manâ alternative to the OpenRISC architecture. You might also think of it as an Open Source microcontroller. For this reason, all instructions have been designed to be as simple as possible, and the base instructions are all designed to be executed in one instruction cycle per instruction, barring pipeline stalls.1 Indeed, even the bus has been simplified to a constant 32-bit width, with no option for more or less. This has resulted in the choice to drop push and pop instructions, pre-increment and post-decrement addressing modes, the integrated memory management unit (MMU), and more

For those who like buzz words, the ZipCPU is:

A 32-bit CPU: All registers are 32-bits, addresses are 32-bits, instructions are 32-bits wide, etc.

A RISC CPU. There is no microcode for executing instructions. All instructions are designed to be completed in one clock cycle.

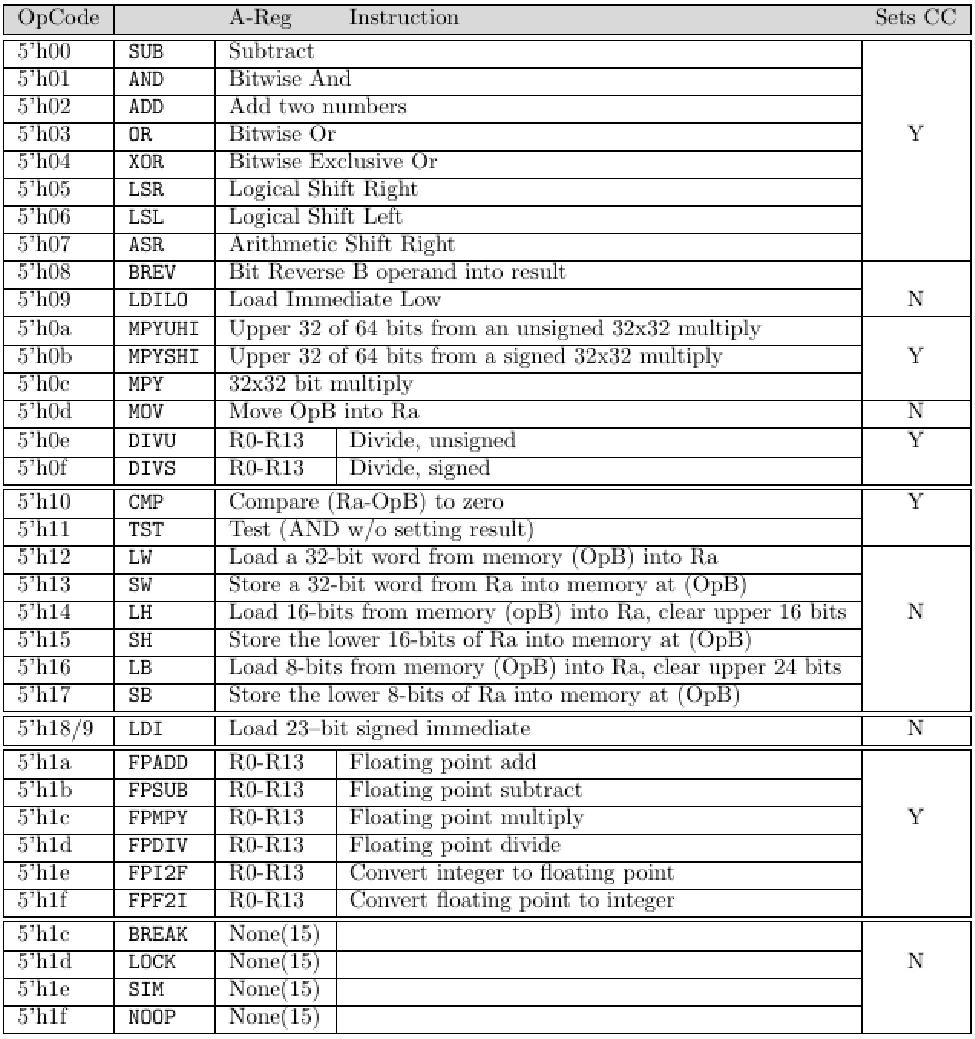
A Load/Store architecture. (Only load and store instructions can access memory.)

Wishbone compliant. All peripherals are accessed just like memory across this bus.

A Von-Neumann architecture. The instructions and data share a common bus.

A pipelined architecture, having stages for Prefetch, Decode, Read-Operand, a combined stage containing the ALU, Memory, Divide, and Floating Point units, and then the final Write-back stage. See Fig. 1.1 for a diagram of this structure.

Completely open source, licensed under the GPL.3



OpCode		A-Reg	Instruction	Sets CC
5'h00	SUB	Subtract		Y
5'h01	AND	Bitwise And		
5'h02	ADD	Add two numbers		
5'h03	OR	Bitwise Or		
5'h04	XOR	Bitwise Exclusive Or		
5'h05	LSR	Logical Shift Right		
5'h06	LSL	Logical Shift Left		
5'h07	ASR	Arithmetic Shift Right		
5'h08	BREV	Bit Reverse B operand into result		N
5'h09	LDILO	Load Immediate Low		
5'h0a	MPYUHI	Upper 32 of 64 bits from an unsigned 32x32 multiply		Y
5'h0b	MPYSHI	Upper 32 of 64 bits from a signed 32x32 multiply		
5'h0c	MPY	32x32 bit multiply		
5'h0d	MOV	Move OpB into Ra		N
5'h0e	DIVU	R0-R13	Divide, unsigned	Y
5'h0f	DIVS	R0-R13	Divide, signed	
5'h10	CMP	Compare (Ra-OpB) to zero		Y
5'h11	TST	Test (AND w/o setting result)		
5'h12	LW	Load a 32-bit word from memory (OpB) into Ra		N
5'h13	SW	Store a 32-bit word from Ra into memory at (OpB)		
5'h14	LH	Load 16-bits from memory (opB) into Ra, clear upper 16 bits		
5'h15	SH	Store the lower 16-bits of Ra into memory at (OpB)		
5'h16	LB	Load 8-bits from memory (OpB) into Ra, clear upper 24 bits		
5'h17	SB	Store the lower 8-bits of Ra into memory at (OpB)		
5'h18/9	LDI	Load 23–bit signed immediate		N
5'h1a	FPADD	R0-R13	Floating point add	Y
5'h1b	FPSUB	R0-R13	Floating point subtract	
5'h1c	FPMPY	R0-R13	Floating point multiply	
5'h1d	FPDIV	R0-R13	Floating point divide	
5'h1e	FPI2F	R0-R13	Convert integer to floating point	
5'h1f	FPF2I	R0-R13	Convert floating point to integer	
5'h1c	BREAK	None(15)		N
5'h1d	LOCK	None(15)		
5'h1e	SIM	None(15)		
5'h1f	NOOP	None(15)		

Table 2.2: ZipCPU OpCodes

Initial testing

06/05/18

```
cd catzip/rtl/catzip
```

```
sudo config_cat catzip.bin
```

```
cd catzip/sw/host
```

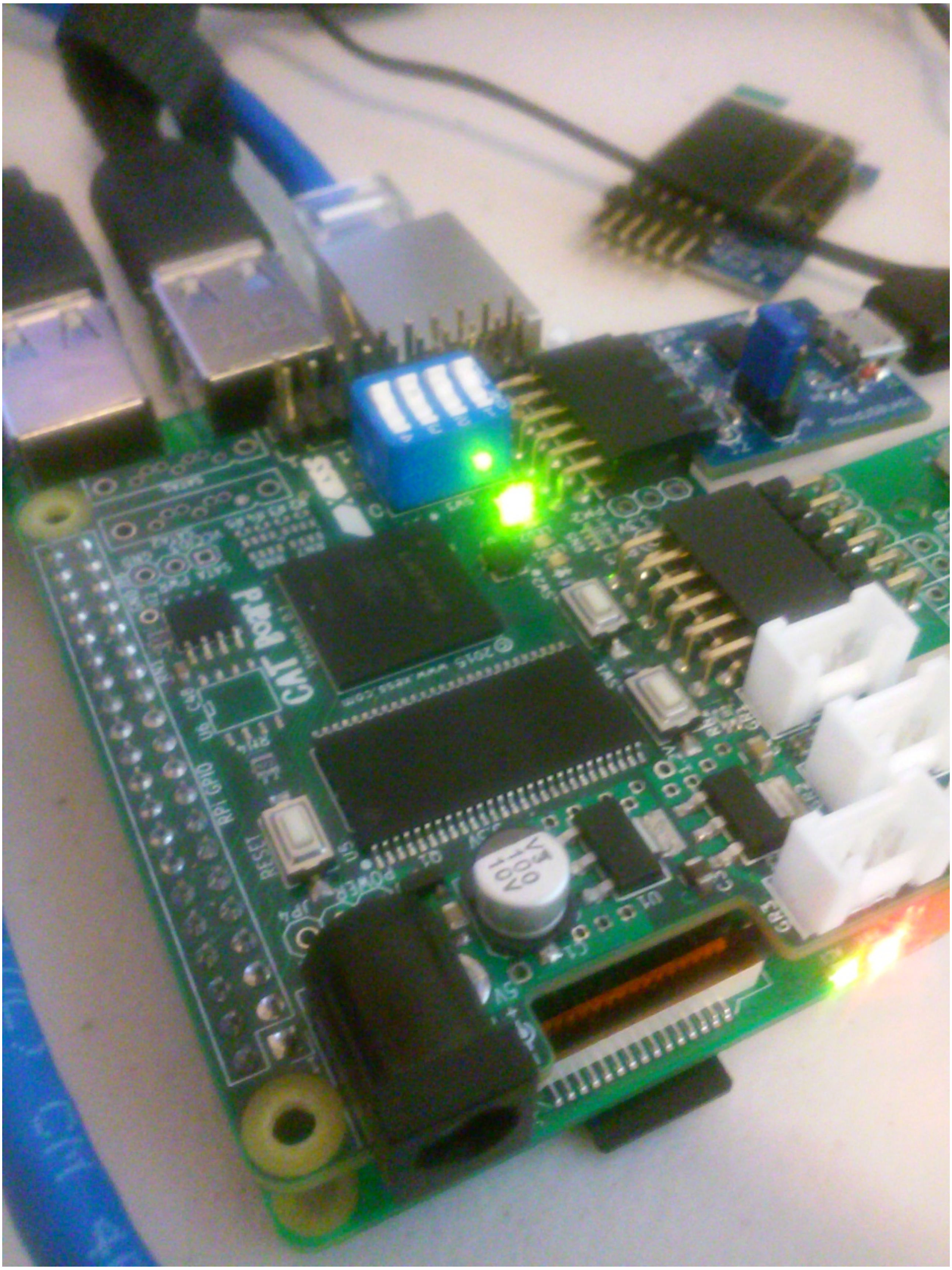
```
./arm-netpport
Listening on port 8363
Listening on port 8364
```

```
cd catzip/sw/host
```

```
./arm-wbregs version
00001010 ( VERSION) : [....] 20180605
```

1st Led on

```
/arm-wbregs gpio 0x00010001
00001008 (    GPIO)-> 00010001
```

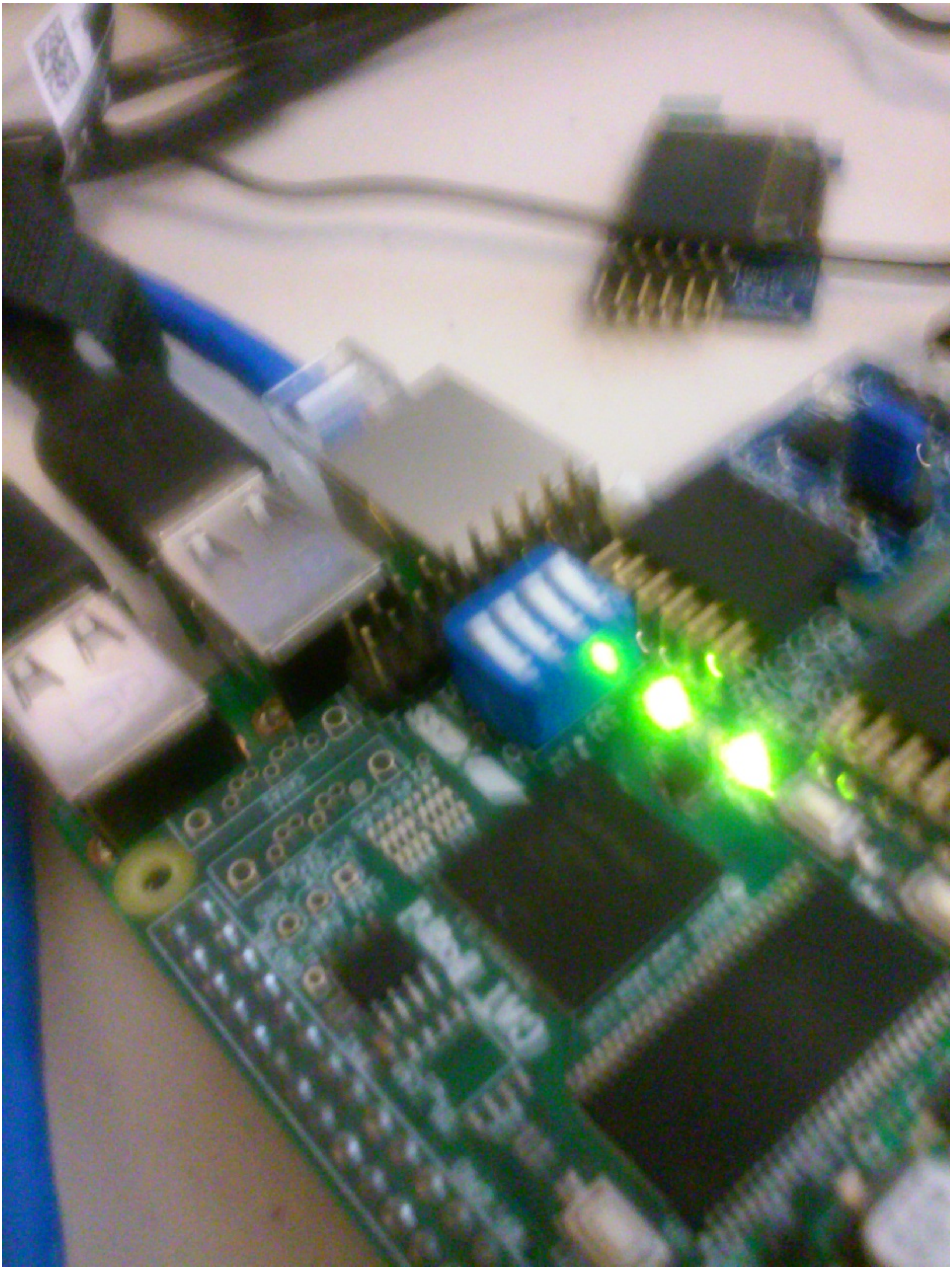

CAT Board
2015
RESET
RPi GPIO
100
10V

1st Led off

```
./arm-wbregs gpio 0x00010000
00001008 (    GPIO)-> 00010000
```

2n led on

```
./arm-wbregs gpio 0x00020002
00001008 (    GPIO)-> 00020002
```

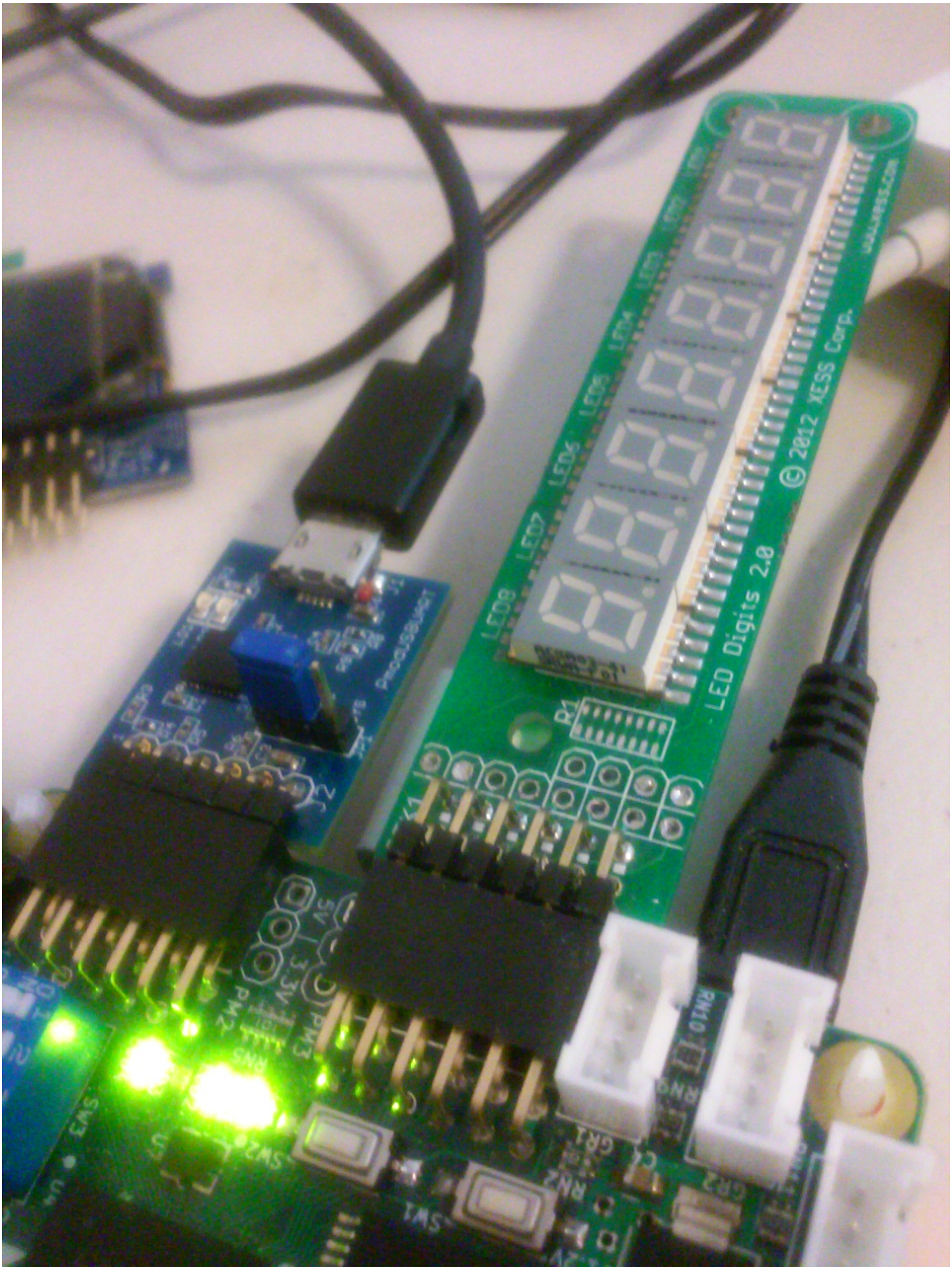


2nd led off

```
./arm-wbregs gpio 0x00020000
00001008 (    GPIO)-> 00020000
```

3rd led on

```
./arm-wbregs gpio 0x00040004
00001008 (    GPIO)-> 00040004
```


LED Digits 2.0
© 2012 XESS Corp.
www.xess.com
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
R1
PmodUSBUART
J1
J2
PM2
PM3
3.3V
5V
SW1
SW2
SW3
RN2
RN5
RN9
RN10
GR1
GR2
U4
U7

3rd led off

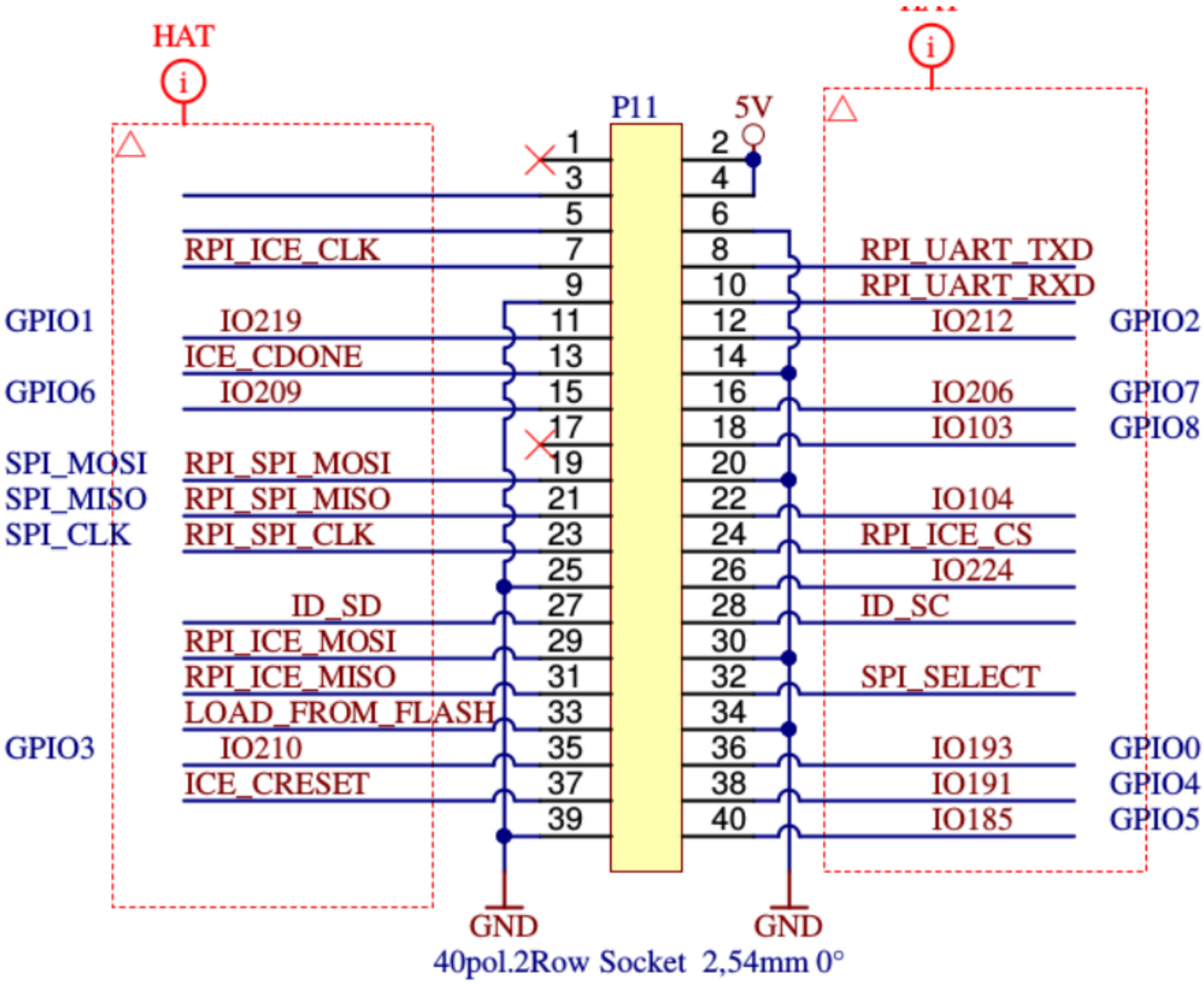
```
./arm-wbregs gpio 0x00040000
00001008 (    GPIO)-> 00040000
```

```
./arm-wbregs 0x2000 0x01
00002000 (     RAM)-> 00000001
pi@pi3-5:~/catzip/sw/host $ ./arm-wbregs ram
00002000 (     RAM) : [....] 00000001
pi@pi3-5:~/catzip/sw/host $ ./arm-wbregs 0x2000 0x02
00002000 (     RAM)-> 00000002
pi@pi3-5:~/catzip/sw/host $ ./arm-wbregs ram
00002000 (     RAM) : [....] 00000002
```

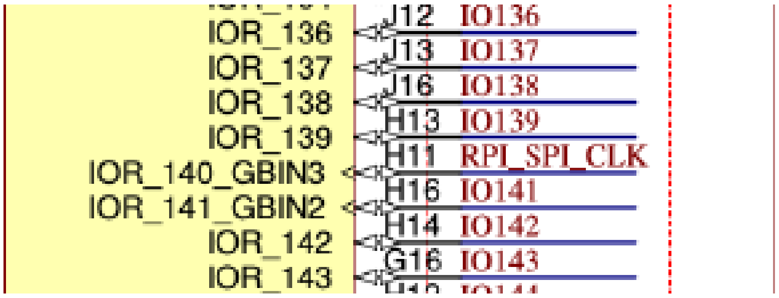
```
./arm-wbregs pic
00001004 (     PIC) : [....] 00000003
```

```
./arm-wbregs ufifo
00000804 (   UFIFO) : [@?@.] 403f4000
```

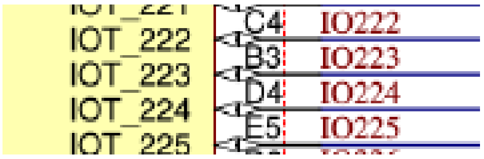
ICOBOARD RPi



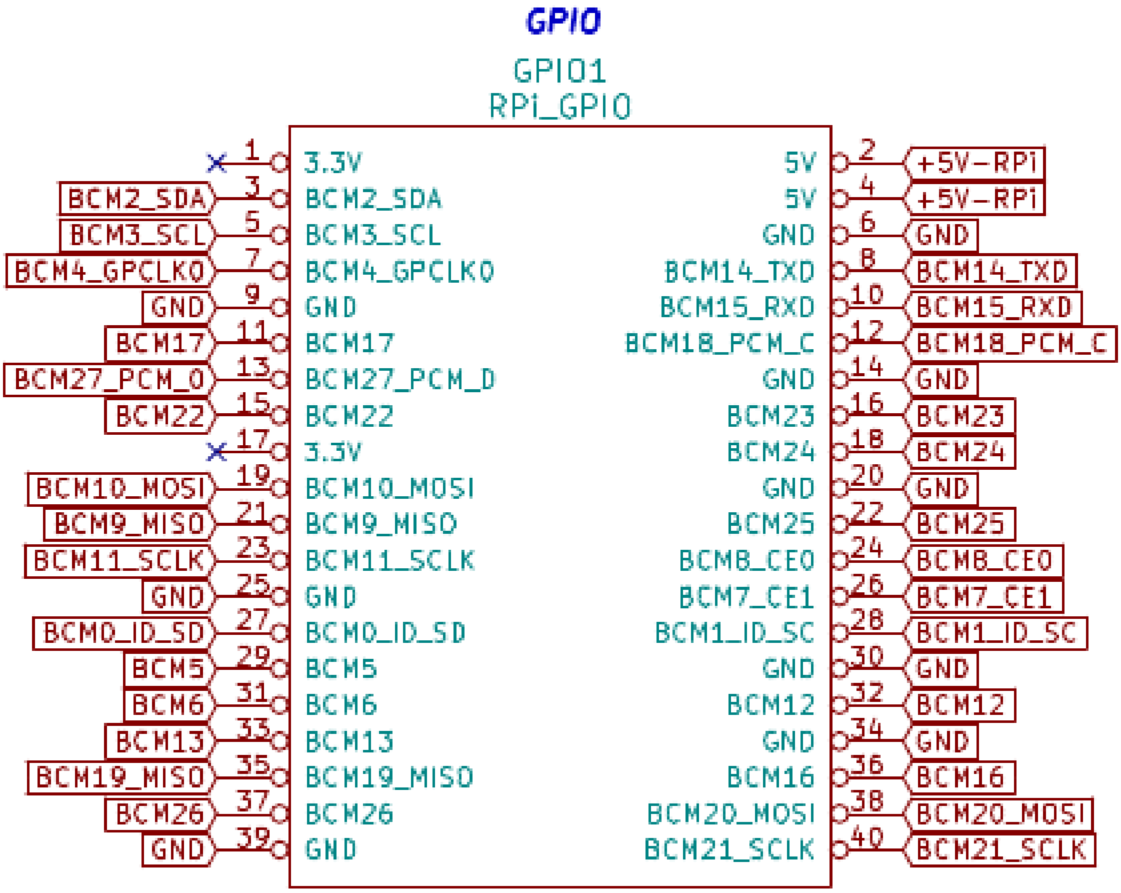
RPI_SPI_CLK H11 Pin 23 Pi icoboard



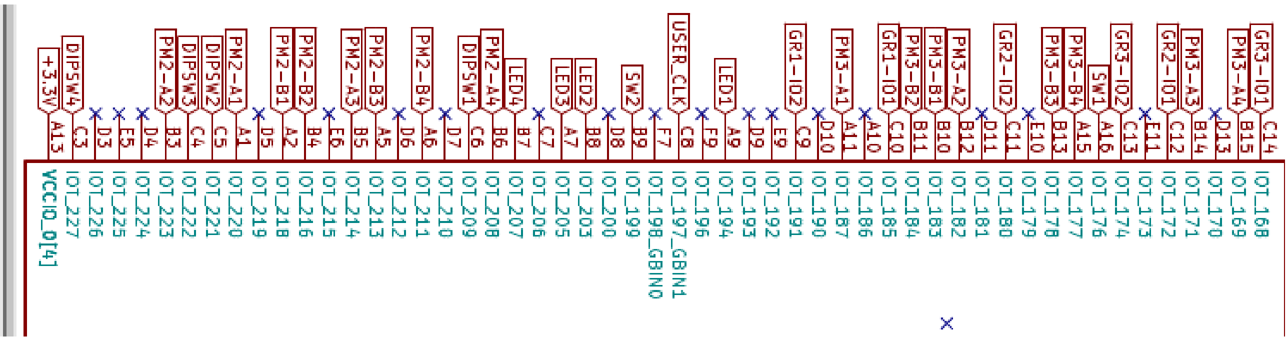
rpi_cs D4 IOT_224 Pin 26 Pi icoboard



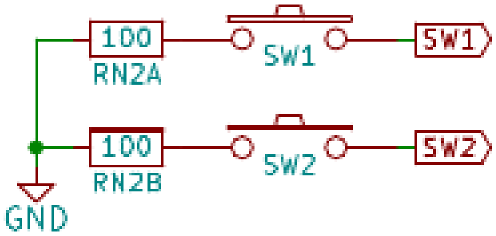
CATBOARD RPi



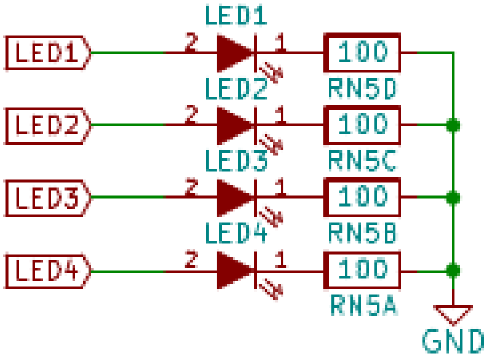
CATBOARD connection to FPGA pins PMOD 2 & PMOD 3 push button switches, dip switch, and leds.



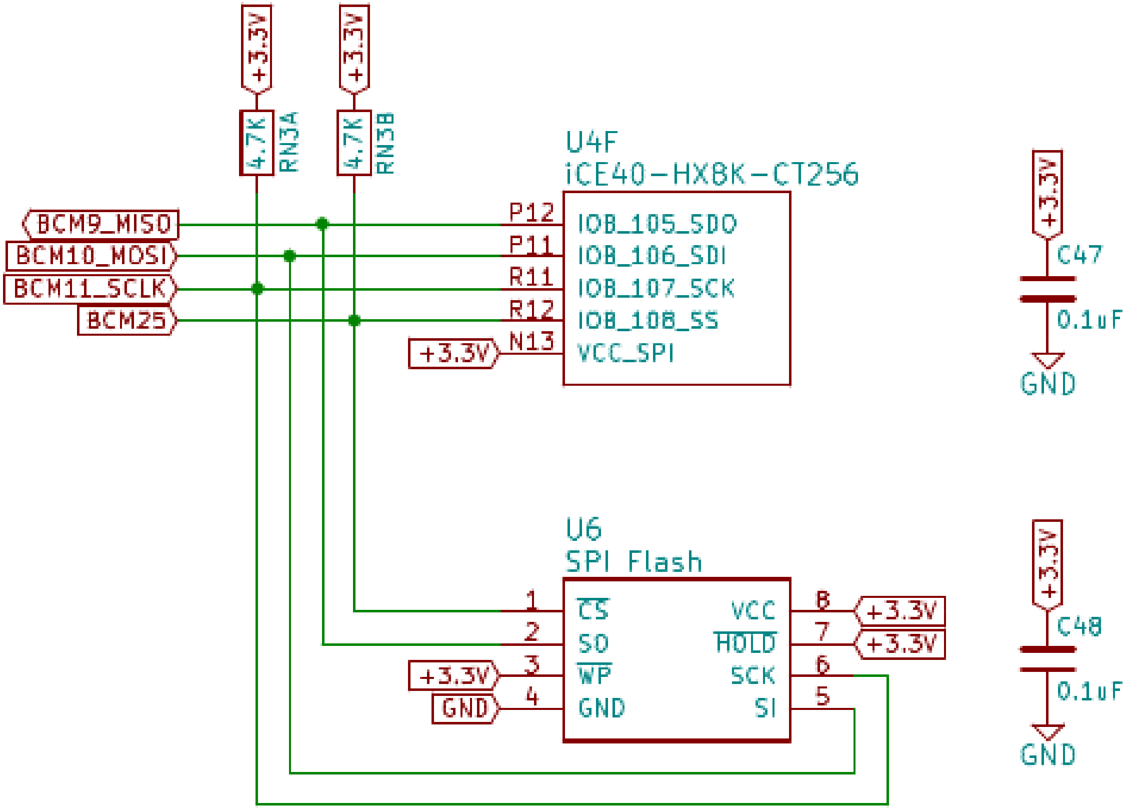
CATBOARD sw1 & sw2



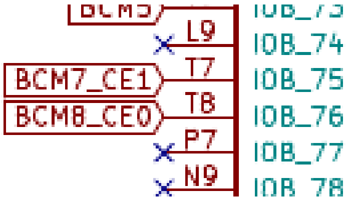
CATBOARD leds



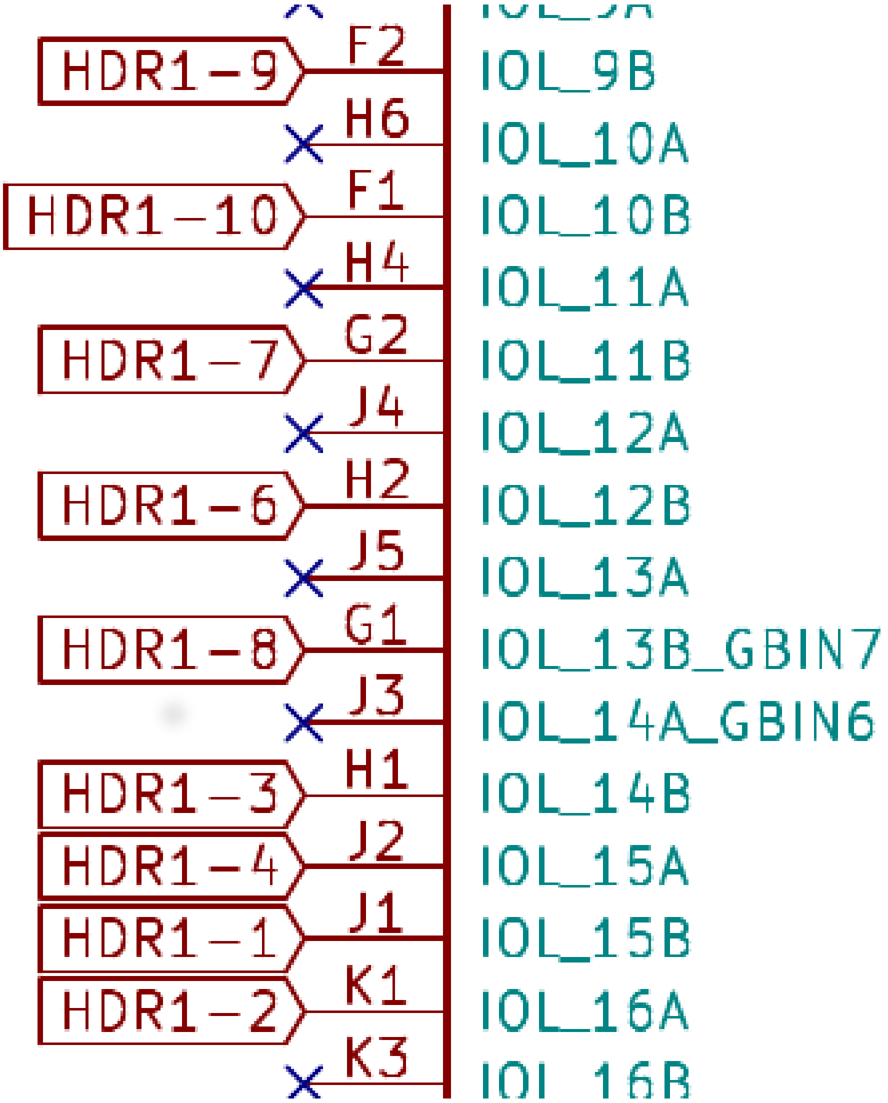
BCM11_SCLK Pin 23 CATBOARD



BCM7_CE1 Pin 26 CATBOARD


L9
IOB_74
BCM7_CE1
T7
IOB_75
BCM8_CE0
T8
IOB_76
P7
IOB_77
N9
IOB_78

CATBOARD


HDR1-9
F2
IOL_9B
H6
IOL_10A
HDR1-10
F1
IOL_10B
H4
IOL_11A
HDR1-7
G2
IOL_11B
J4
IOL_12A
HDR1-6
H2
IOL_12B
J5
IOL_13A
HDR1-8
G1
IOL_13B_GBIN7
J3
IOL_14A_GBIN6
HDR1-3
H1
IOL_14B
HDR1-4
J2
IOL_15A
HDR1-1
J1
IOL_15B
HDR1-2
K1
IOL_16A
K3
IOL_16B

2.) The 2nd issue is the PMOD connections to FPGA are different.
3.) Third, I do not have a Diglient PMOD 4 push button switch module.
4.) The 4th issue is the PHASE LOCK LOOP difference.

Post on #yosys

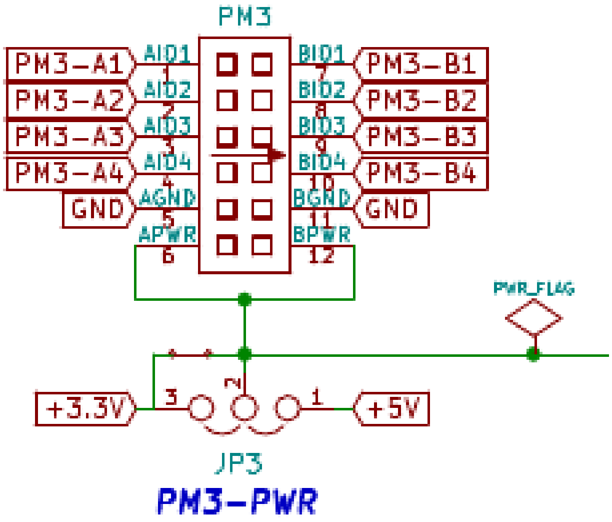
Pin C8 is my USER_CLK comes from a 100MHz osc. It is connected to IOT_197_GBIN1 on HX8K. When I try using it for as an input to PLL I get the fatal error: bad constraint on `i_clk': no PLL at pin C8.

Can only certain pins be used as inputs to PLL?

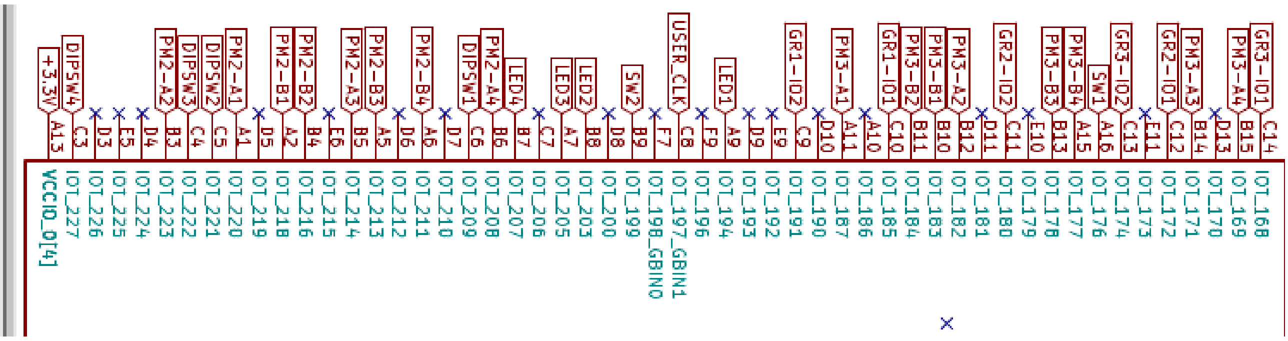
daveshah

develonepi3: use the SB_PLL40_CORE instead of SB_PLL40_PAD variant (and REFERENCECLK in instead of PACKAGEPIN)

```
set_io clk_100mhz C8    #R9

set_io pmod1_1 A11      #D8
set_io pmod1_2 B12      #B9
set_io pmod1_3 B14      #B10
set_io pmod1_4 B15      #B11
#   654321      catboard    #   654321  icoboard
#       xxxxxx  PMOD3 A         #       xxxxxx  PMOD1 A
#       xxxxxx  PMOD3 B         #       xxxxxx  PMOD1 B
#   654321                      #   654321
#
set_io pmod1_7 B10      #B8
set_io pmod1_8 B11      #A9
set_io pmod1_9 B13      #A10
set_io pmod1_10 A15     #A11
```



CATBOARD connection to FPGA pins PMOD 2 & PMOD 3 push button switches, dip switch, and leds.

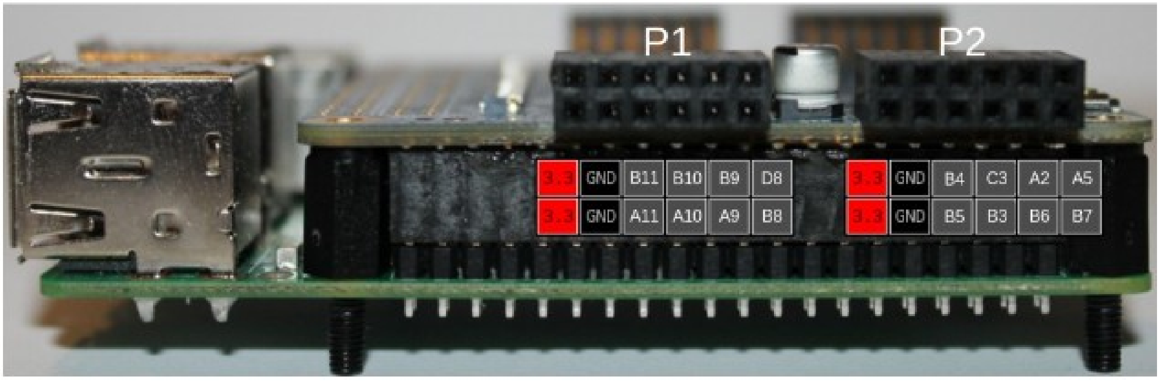


In top.v

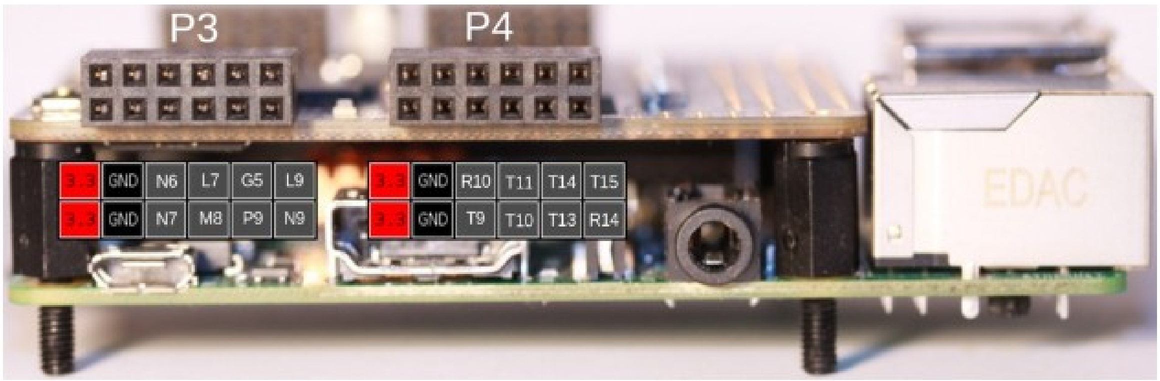
```
module top(clk_100mhz, pmod1_1, pmod1_2, pmod1_3, pmod1_4, pmod1_7, pmod1_8,
  pmod1_9, pmod1_10, pmod2_7, pmod2_8, pmod2_9, pmod2_10, rpi_sck, rpi_cs,
  rpi_mosi);

input rpi_sck, rpi_cs, rpi_mosi;
rpi_sck
rpi_cs
rpi_mosi

spi_ram_slave spi_ram_slave(clk, rpi_sck, rpi_cs, rpi_mosi,
  ram_addr, ram_data, ram_wr);
module spi_ram_slave(clk, sck, cs, mosi, ram_addr, ram_data, ram_wr);
```

PMOD pin out on icoboard

Pinout Pmod P1 and P2



Pinout PMOD P3 and P4



"lrwxrwxrwx 1 root staff 34 May 18 20:10 /usr/local/bin/config_cat -> /home/pi/catboard_yosys/config_cat"

```
#!/bin/bash

#   A script to configure Lattice iCE40 FPGA by SPI from Raspberry Pi
#
#   Copyright (C) 2015 Jan Marjanovic <jan@marjanovic.pro>
#
#   This program is free software: you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation, either version 3 of the License, or
#   (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.


echo ""
if [ $# -ne 1 ]; then
   echo "Usage: $0 FPGA-bin-file "
   exit 1
fi

if [ $EUID -ne 0 ]; then
   echo "This script must be run as root" 1>&2
   exit 1
fi


if [ ! -d /sys/class/gpio/gpio25 ]; then
   echo "GPIO 25 not exported, trying to export..."
   echo 25 > /sys/class/gpio/export
   if [ ! -d /sys/class/gpio/gpio25 ]; then
         echo "ERROR: directory /sys/class/gpio/gpio25 does not exist"
         exit 1
   fi
else
   echo "OK: GPIO 25 exported"
fi

if [ ! -d /sys/class/gpio/gpio17 ]; then
   echo "GPIO 17 not exported, trying to export..."
   echo 17 > /sys/class/gpio/export
   if [ ! -d /sys/class/gpio/gpio17 ]; then
         echo "ERROR: directory /sys/class/gpio/gpio17 does not exist"
         exit 1
   fi
else
   echo "OK: GPIO 17 exported"
fi

if [ ! -d /sys/class/gpio/gpio22 ]; then
   echo "GPIO 22 not exported, trying to export..."
   echo 22 > /sys/class/gpio/export
   if [ ! -d /sys/class/gpio/gpio22 ]; then
         echo "ERROR: directory /sys/class/gpio/gpio22 does not exist"
         exit 1
   fi
else
   echo "OK: GPIO 22 exported"
fi
```

```
echo ""
if [ -e /dev/spidev0.0 ]; then
    echo "OK: SPI driver loaded"
else
    echo "spidev does not exist"

    lsmod | grep spi_bcm2708 >& /dev/null

    if [ $? -ne 0 ]; then
            echo "SPI driver not loaded, try to load it..."
            modprobe spi_bcm2708

            if [ $? -eq 0 ]; then
                echo "OK: SPI driver loaded"
            else
                echo "Could not load SPI driver"
                exit 1
            fi
    fi
fi

echo ""
echo "Setting GPIO directions"
echo out > /sys/class/gpio/gpio25/direction
cat /sys/class/gpio/gpio25/direction
echo out > /sys/class/gpio/gpio22/direction
cat /sys/class/gpio/gpio22/direction
echo in > /sys/class/gpio/gpio17/direction
cat /sys/class/gpio/gpio17/direction

echo "Setting output to low"
echo 0 > /sys/class/gpio/gpio25/value
cat /sys/class/gpio/gpio25/value

#echo ""
#echo "Please reset the iCE40 FPGA board"
#echo "Press any key..."
#read

echo "Reseting FPGA"
echo 0 > /sys/class/gpio/gpio22/value
cat /sys/class/gpio/gpio22/value
echo 1 > /sys/class/gpio/gpio22/value
cat /sys/class/gpio/gpio22/value

echo "Checking DONE pin"
cat /sys/class/gpio/gpio17/value

echo "Continuing with configuration procedure"
dd if=$1 of=/dev/spidev0.0
```

```
echo -e "\x0\x0\x0\x0\x0\x0\x0" > /dev/spidev0.0

echo "Setting output to high"
echo 1 > /sys/class/gpio/gpio25/value
cat /sys/class/gpio/gpio25/value

echo "Checking DONE pin"
cat /sys/class/gpio/gpio17/value


“cd otl-icoboard-pmodoledrgb-demo/stream-tool/”

“ffmpeg -f v4l2 -i /dev/video0 -s 96x64 -f rawvideo -pix_fmt rgb565 - | ./stream-tool”
```