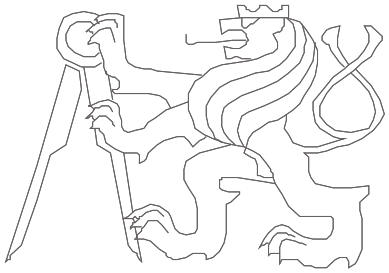
Computer Architectures

I/O Operation + MZAPO

Richard Šusta, Pavel Píša

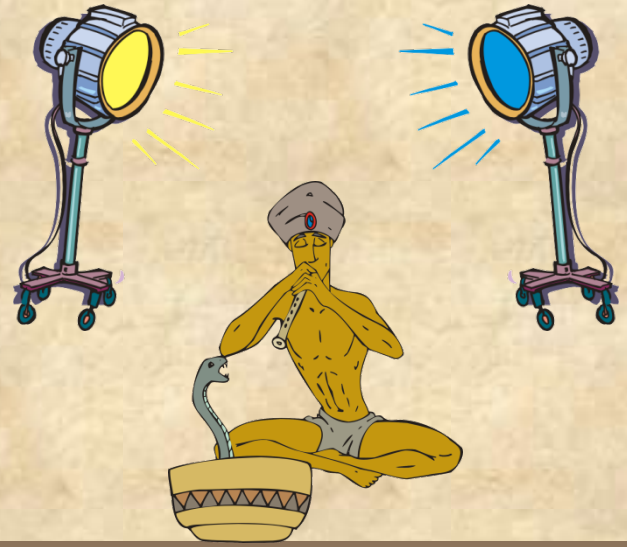


Czech Technical University in Prague, Faculty of Electrical Engineering

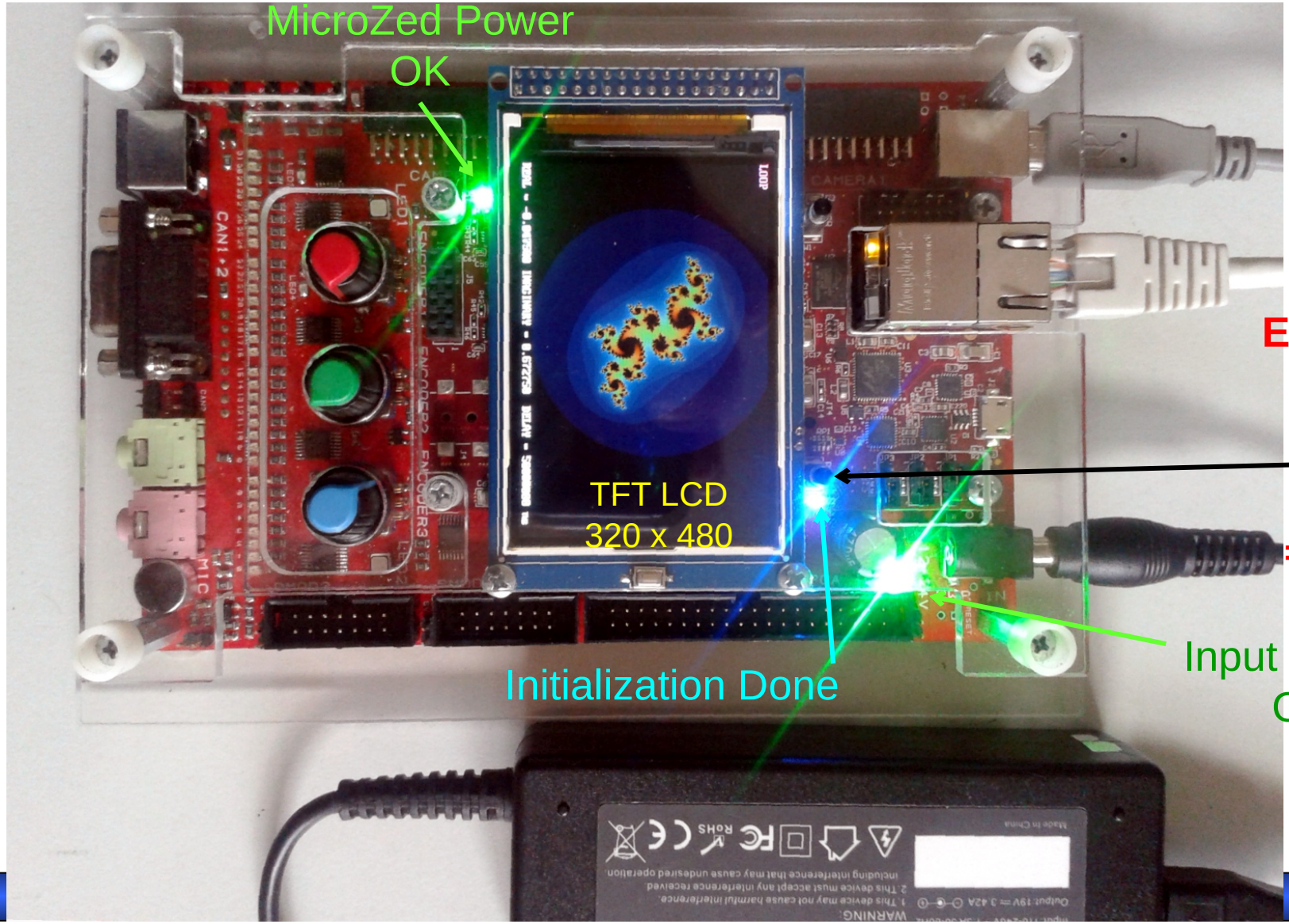
Task Description

Program the MicroZed controller to the two RGB reflectors.

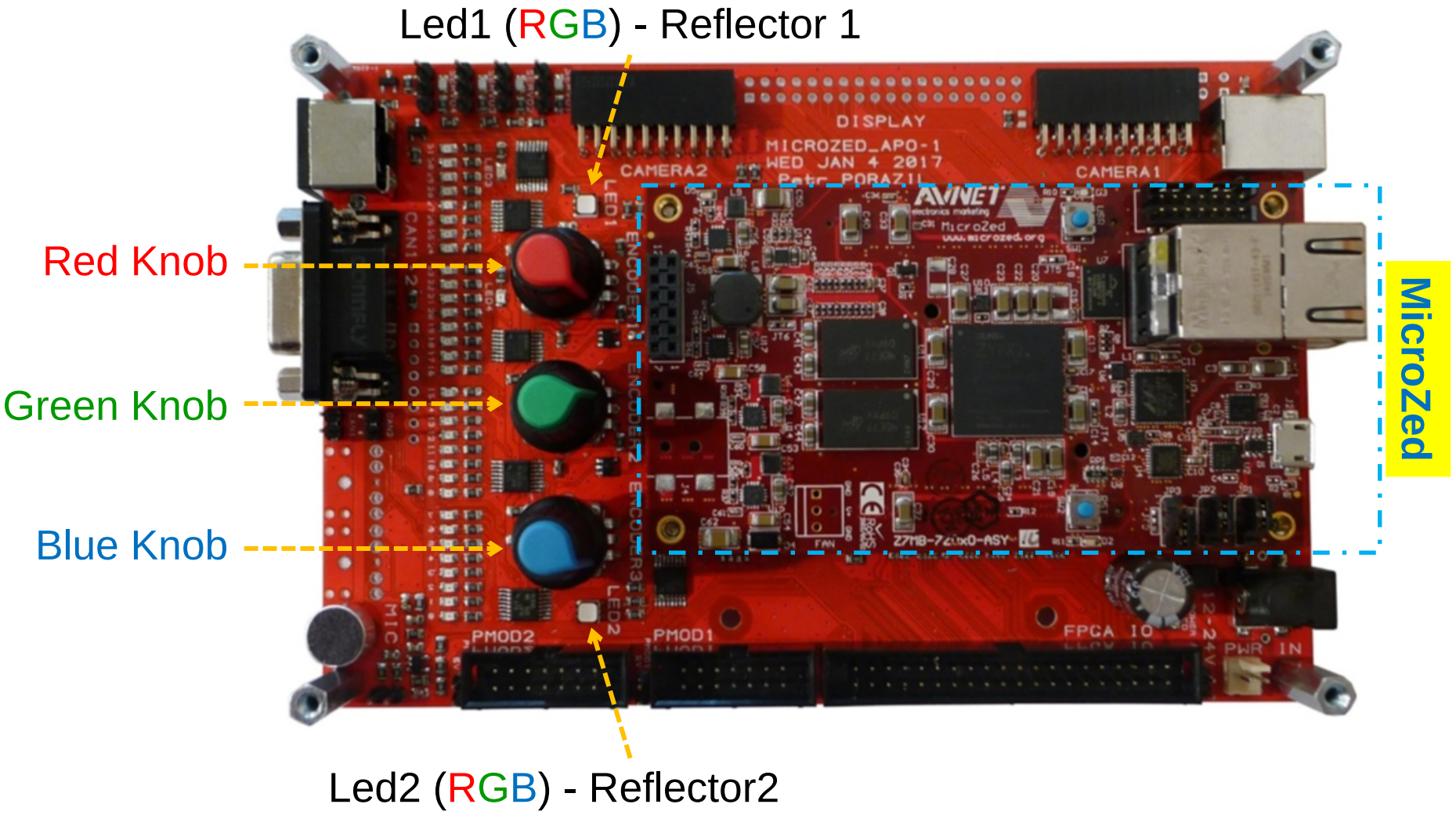
The color of light is selected in the HSV / HSB color model.



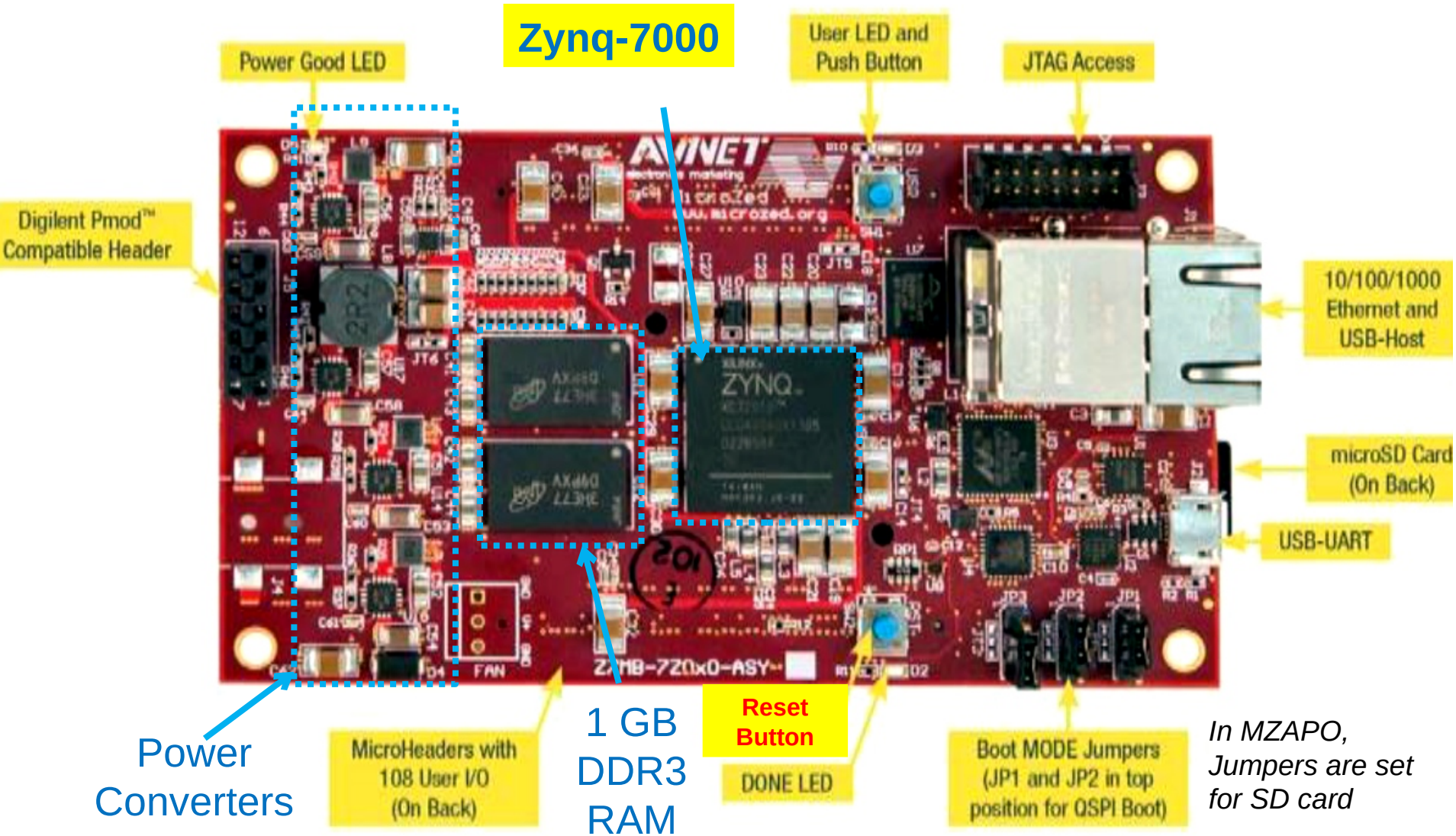
MZAPO - MicroZed_APO board



Demounting LCD -> MicroZed_APO on Carry card



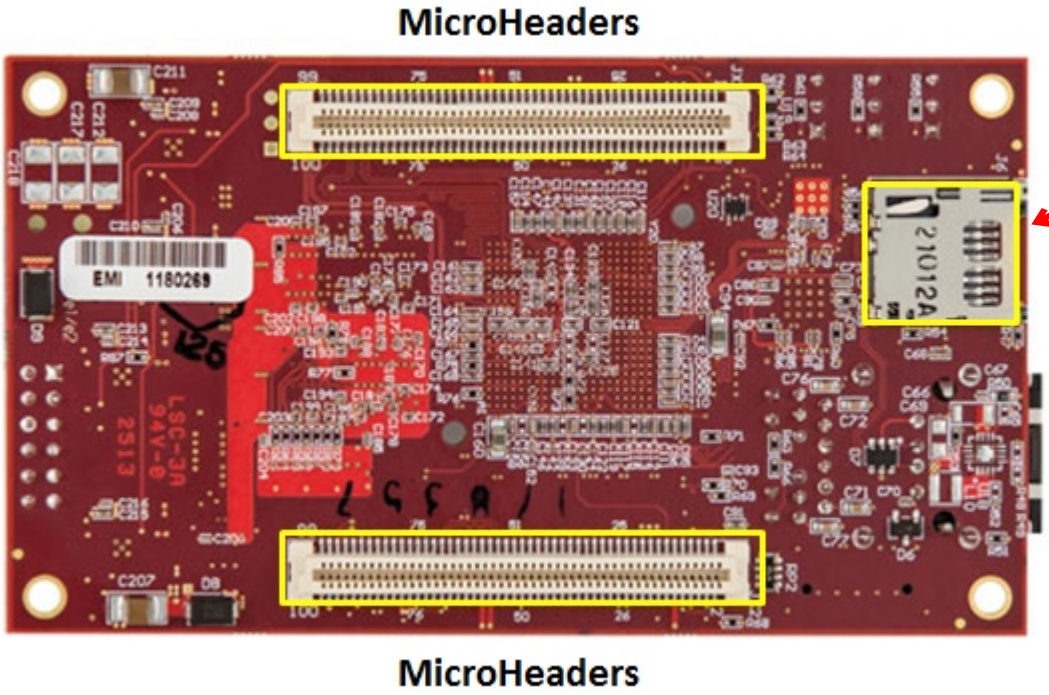
MicroZed - Top View



In MZAPO, Jumpers are set for SD card

MicroZed - Bottom View

MicroZed Evaluation Kit - ADSAES-Z7MB-7Z010-G



Micro SD Card with initialization program

108-pin micro header connectors allowing mount MicroZed™ Evaluation Kit to a carrier card.

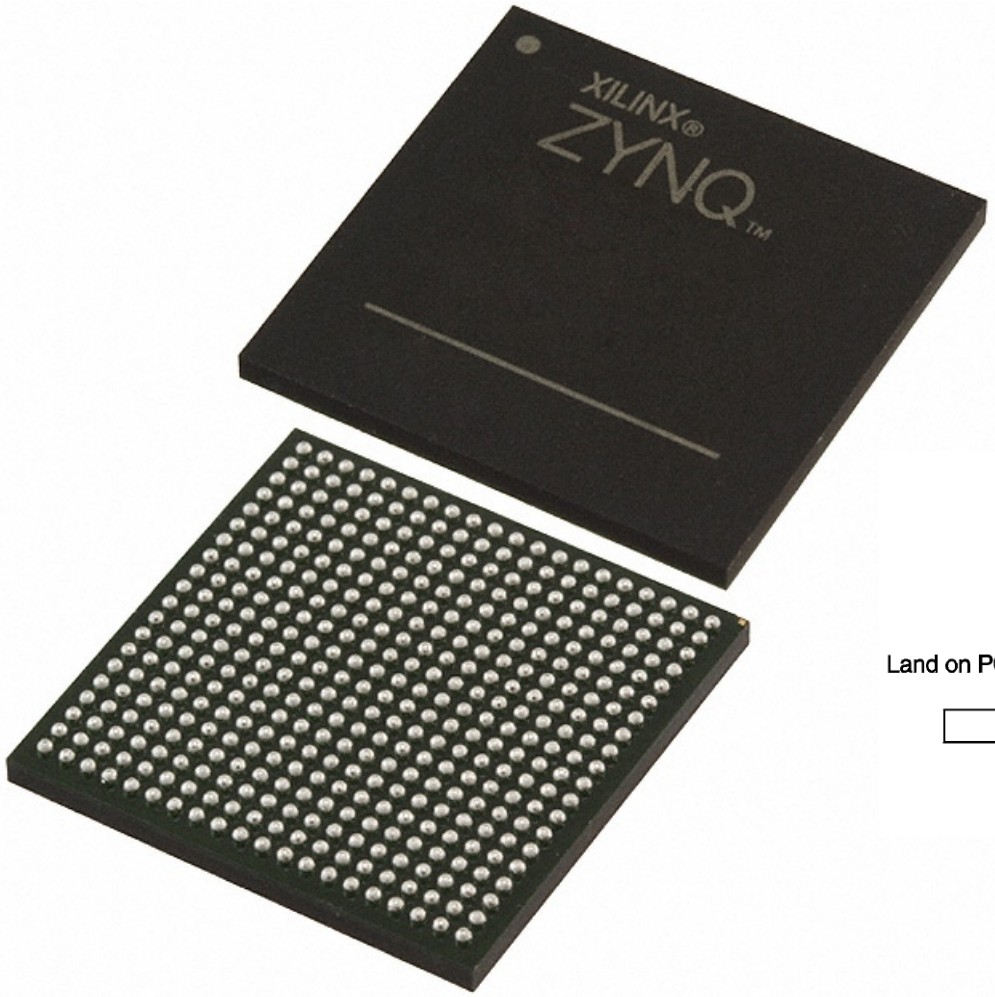
Price of 1 pcs of used Zynq: ~ $45 **,** *price of MicroZed: ~ $180 (April 2019)*

Data sheet of MicroZed

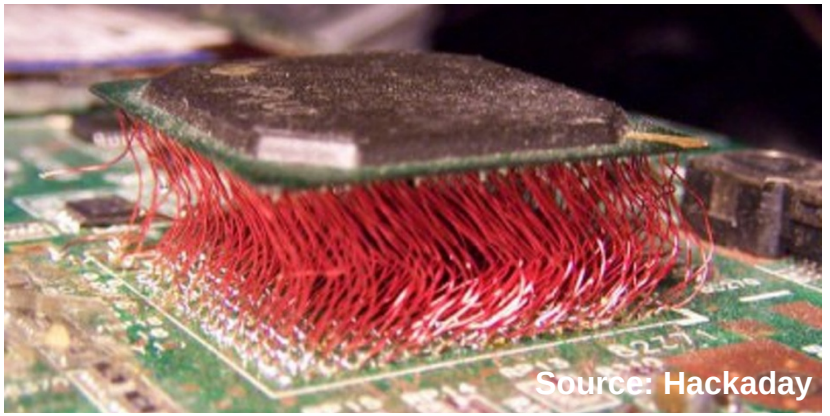
- **FPGA chip** – **Zynq™-7000** AP SoC (XC7Z010-CLG400-1)
  - CPU: Dual **ARM® Cortex™-A9** MPCore™ @ **866 MHz**
  - fast internal static memory 256 kB
  - 4400 slices - *each slice is small configurable logic circuit.*
    It can create up to 8 flip-flops and 4 logic functions with 6-inputs.
    User can freely configure them and mutually interconnect.
- External dynamic memory – **1 GB DDR3**
- Communication – 10/100/1000 **Ethernet**
- MicroSD card **4 GB**. *In APO board, it contains the loader of Linux system for Ethernet network.*
- **USB** Host 2.0 and USB-UART
- Quad-SPI Flash 128 Mb for power-up initialization.
  In APO, it is not used.

Zynq™-7000 in FBGA Package

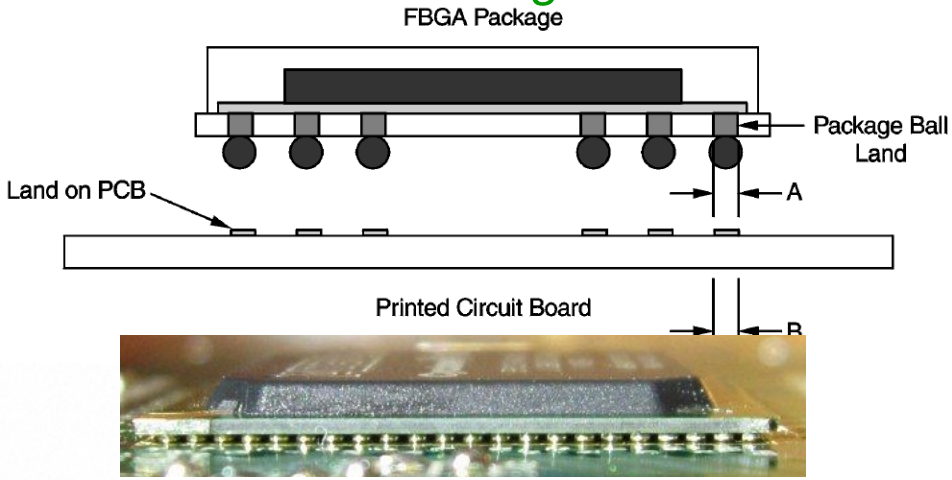
FBGA = Fine-Pitch Ball Grid Array



Clumsy method of FBGA soldering

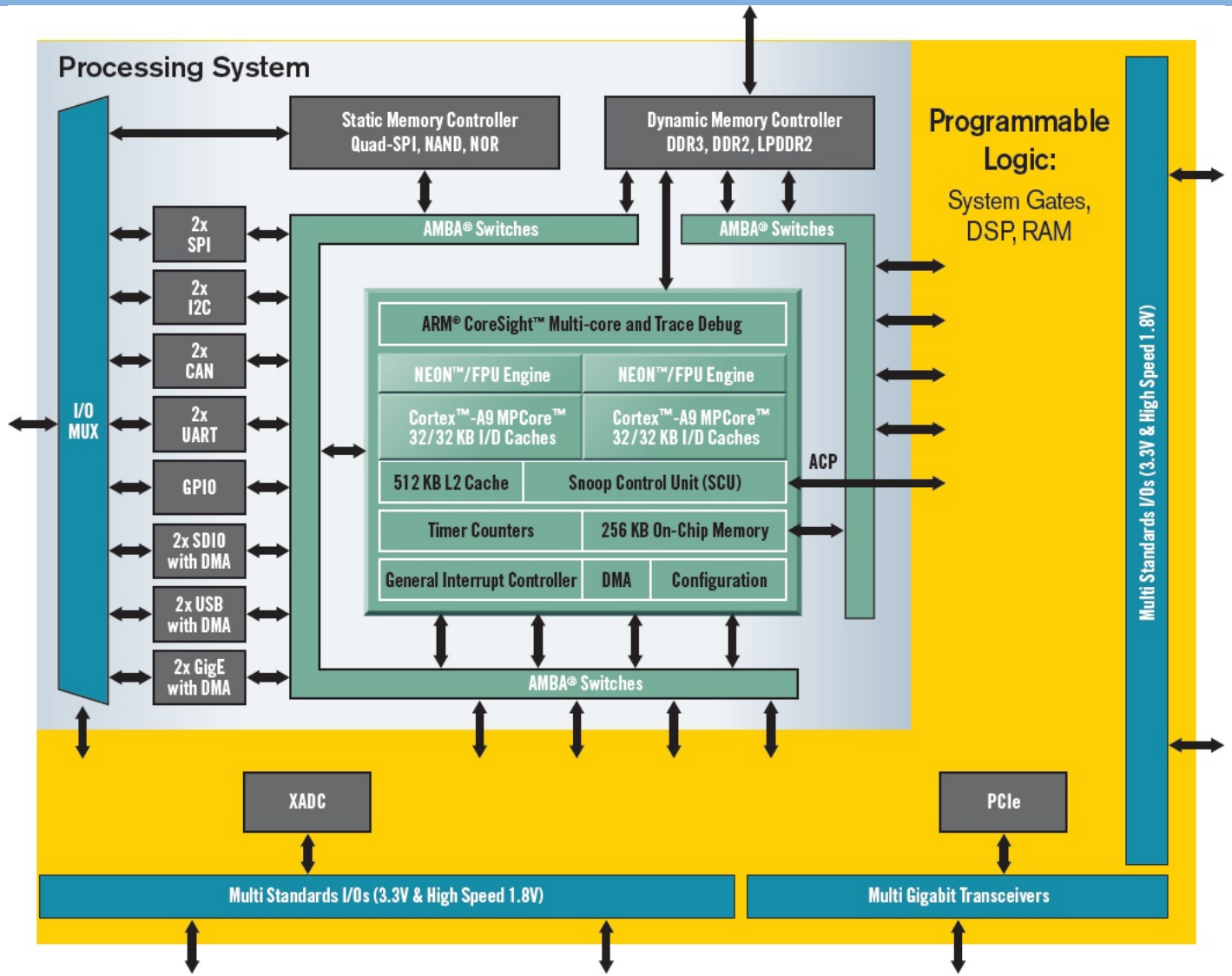


Intended soldering

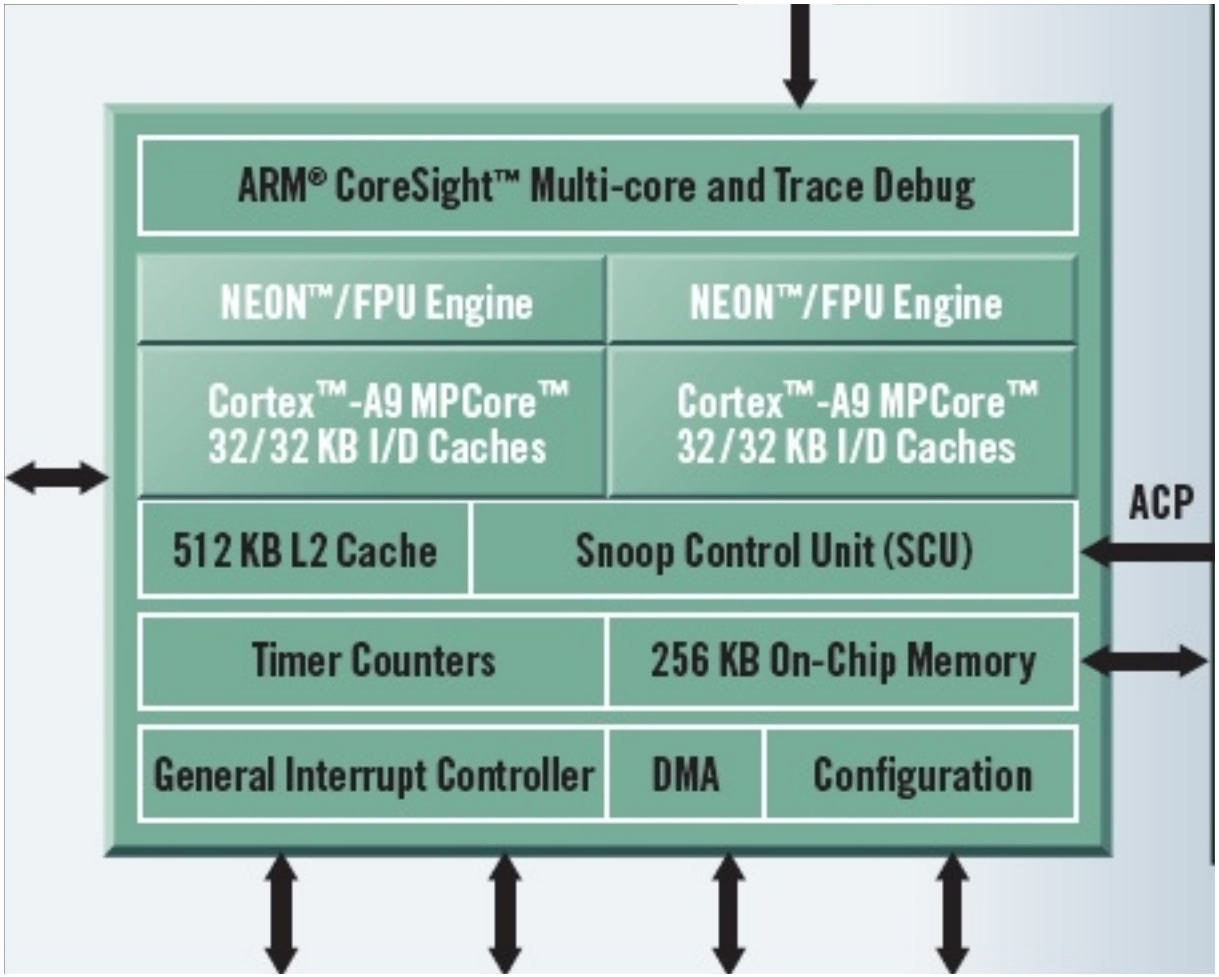


Source: Fairchild Semiconductors

Inside of **Xilinx Zynq™-7000**



Enlarged Zynq Processor Cores



We have already discussed in the lectures:

1. FPU Engine
2. Processor core
3. L1 Instruction/Data caches
4. L2 cache
5. Snoop Control Unit (cache coherency)
6. On-Chip Memory (static RAM)
7. DMA (direct memory access control)

Where is Cortex-A9 used? Some samples

Asus Transformer Pad Infinity (TF700T)

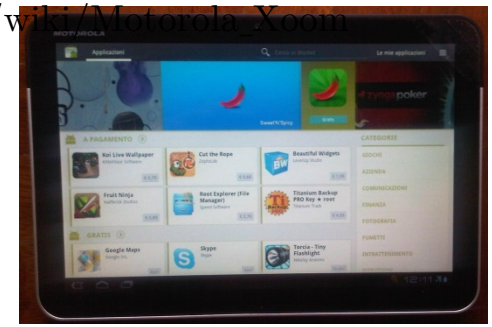
https://en.wikipedia.org/wiki/Asus_Transformer_Pad_Infinity

Apple A5 (iPhone 4S, iPad 2, iPad mini)

http://en.wikipedia.org/wiki/Iphone_4s

NVIDIA Tegra 2 (Motorola Xoom, Droid X2)

http://en.wikipedia.org/wiki/Motorola_Xoom



Full list see: https://en.wikipedia.org/wiki/ARM_Cortex-A9#Implementations

CortexA9 Microarchitecture

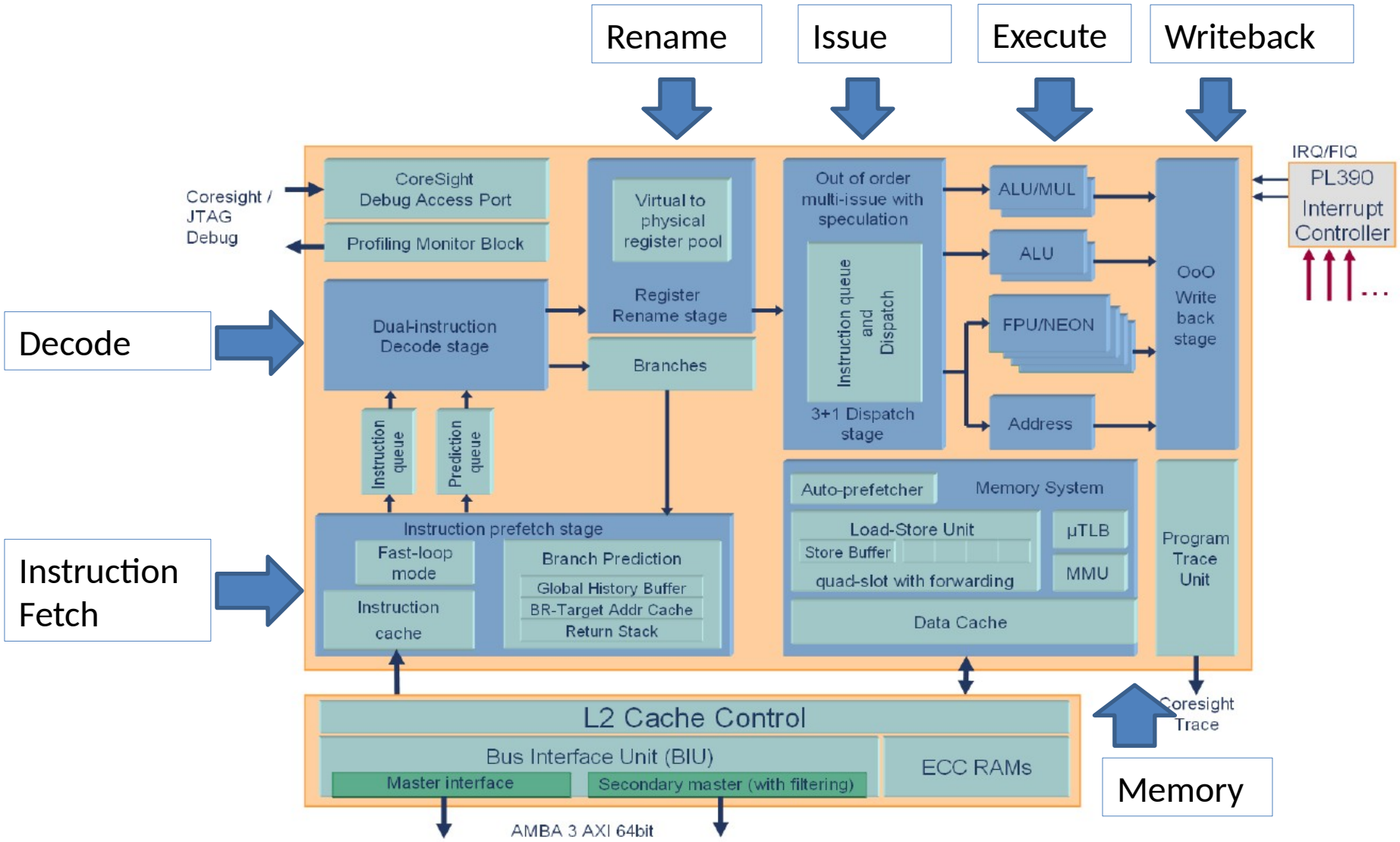


Fig. 1 Cortex-A9 microarchitecture structure and the single core interfaces.

Source: www.arm.com

Our Cortex A9 MP Core properties

- **32 bit RISC Little Endian**, 16 registers integer
- 2.5 DMIPS/MHz
  - -> 866 MHz *2.5 DMIPS/MHz = **2165 DMIPS**

  Note: DMIPS is the result of Dhrystone synthetic computing benchmark intended to represent integer programming.
- Most Integer Instructions finish within 1 cycle, integer multiply needs 4 ~ 5 cycles.
- Float point instruction last on ALU from 4 cycles addition, subtraction (FADD, FSUB), 5 cycles multiply FMUL, 15 cycles divide FDIV (3times longer than multiply!), 17 cycle square root FSQRT.
- Branch prediction
  - 4 K entries table of 2 bit predictors.
- **Virtual memory with 2 level paging tables**

L1 and L2 Caches

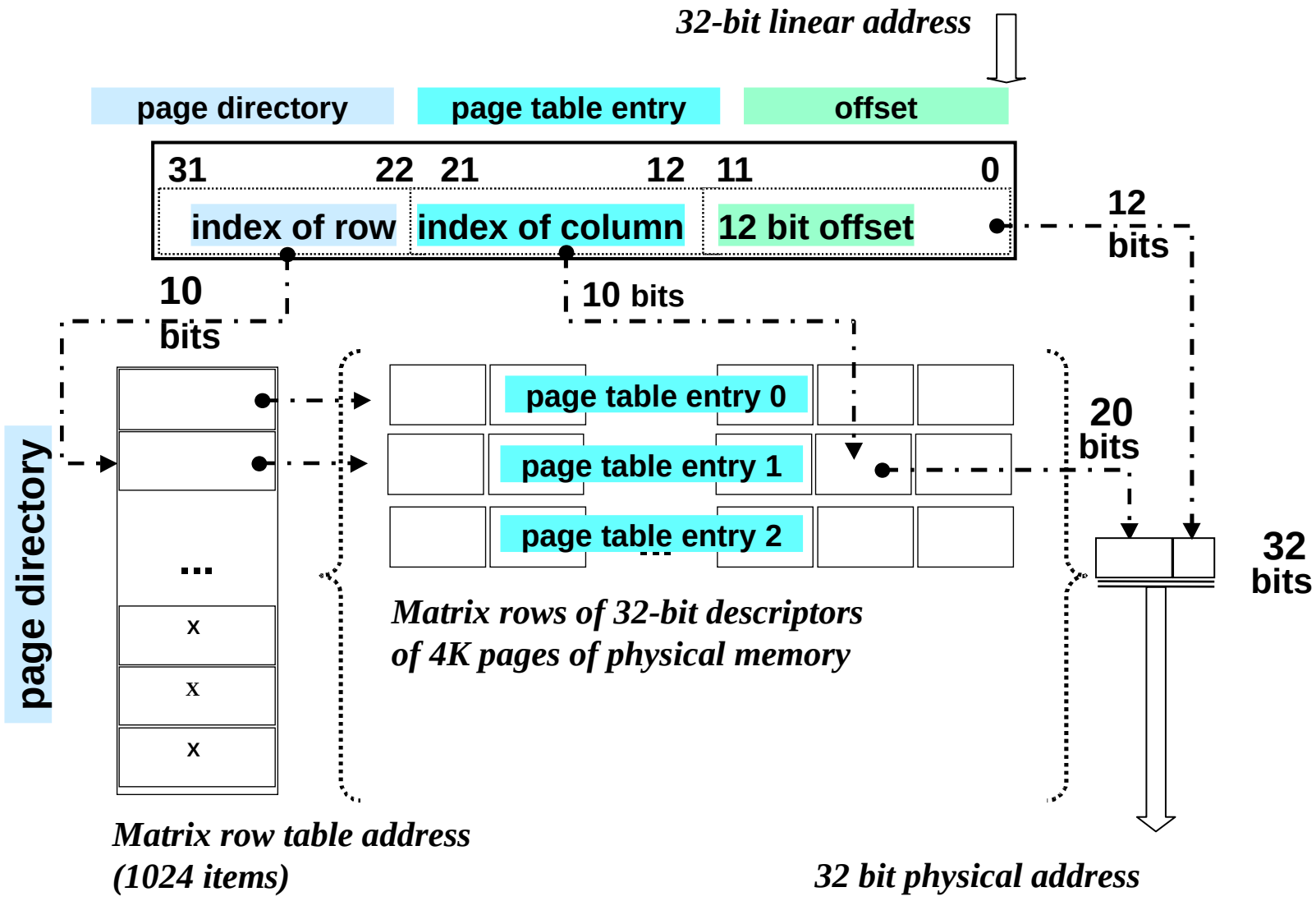
2 separate L1 caches, for I-cache instructions and D-cache for data.
Both L1 have properties:

- **32 kB size**,
- 4-way set associative,
- 32 byte block length,
- replacement policy is pseudo-random or pseudo round-robin.
- **D-Cache** only supports write-back/write-allocate policy.

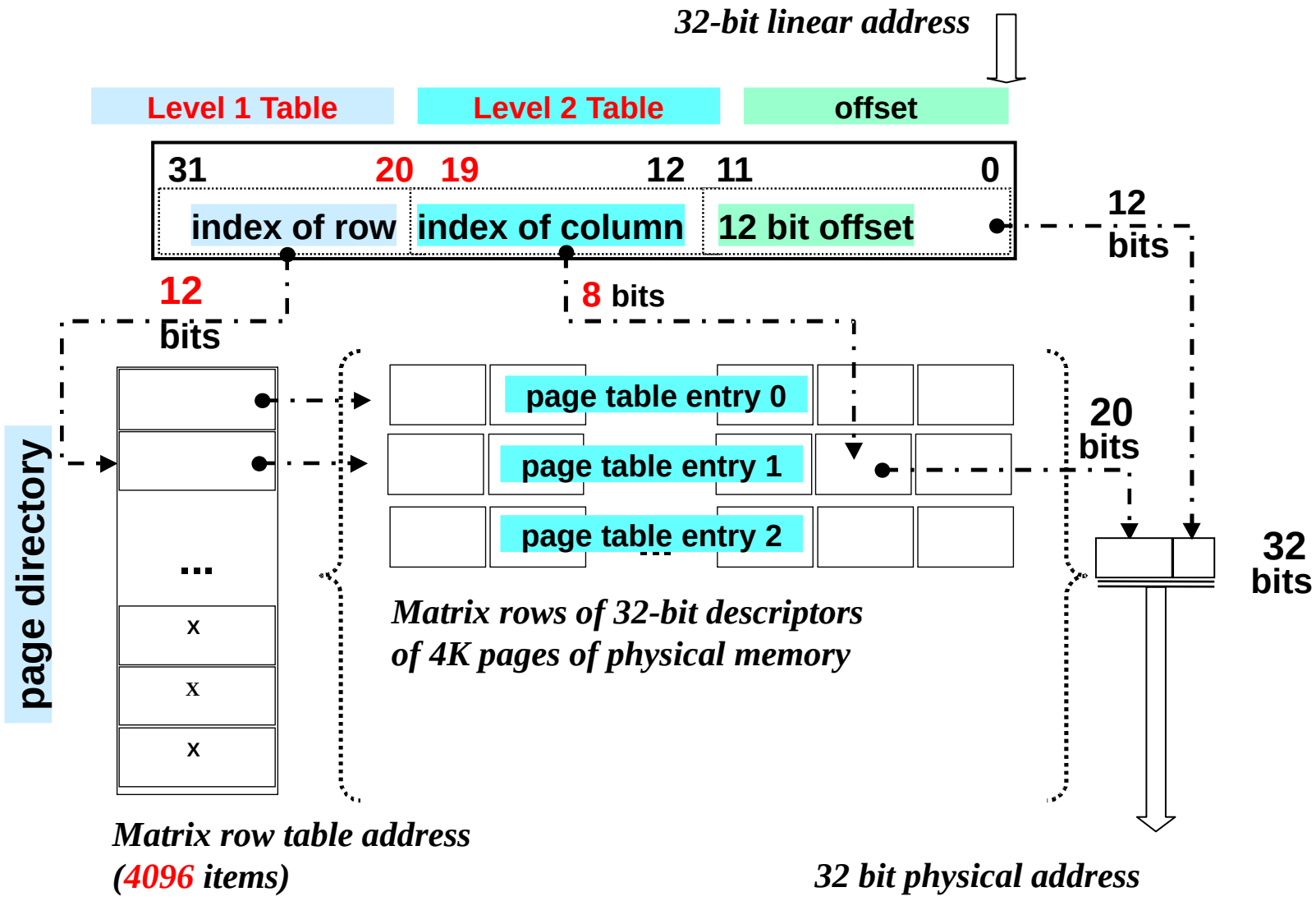
L2 cache is shared by dual Cortex-A9 cores. Its properties:

- **512 KB size,**
- 8-way set-associative,
- 32 byte block (line) length,
- replacement policy is pseudo-random,
- supports Write-back,
  and Write-through with Read allocate, Write allocate.

Recall Virtual Pages from 4th Lecture



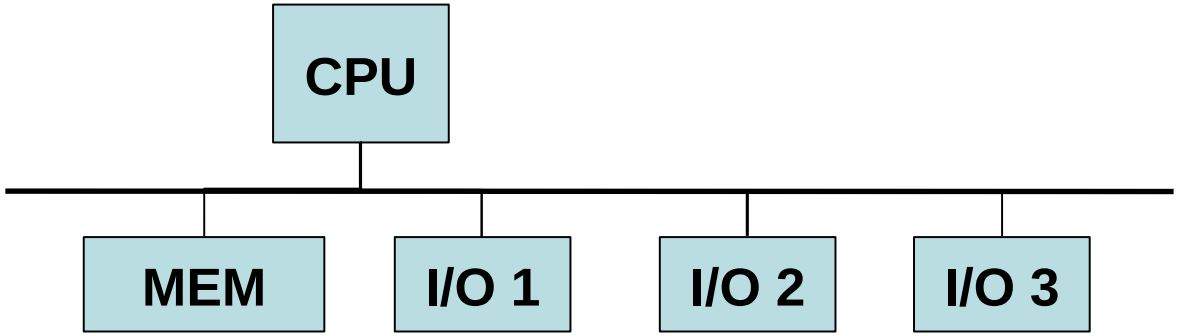
Cortex A9

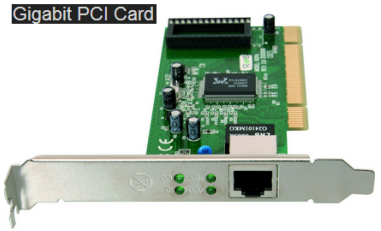

32-bit linear address
Level 1 Table
Level 2 Table
offset
31
20
19
12
11
0
index of row
index of column
12 bit offset
12 bits
8 bits
12 bits
page directory
page table entry 0
page table entry 1
page table entry 2
20 bits
32 bits
x
x
x
Matrix rows of 32-bit descriptors of 4K pages of physical memory
Matrix row table address (4096 items)
32 bit physical address

*I/O Pheripherials

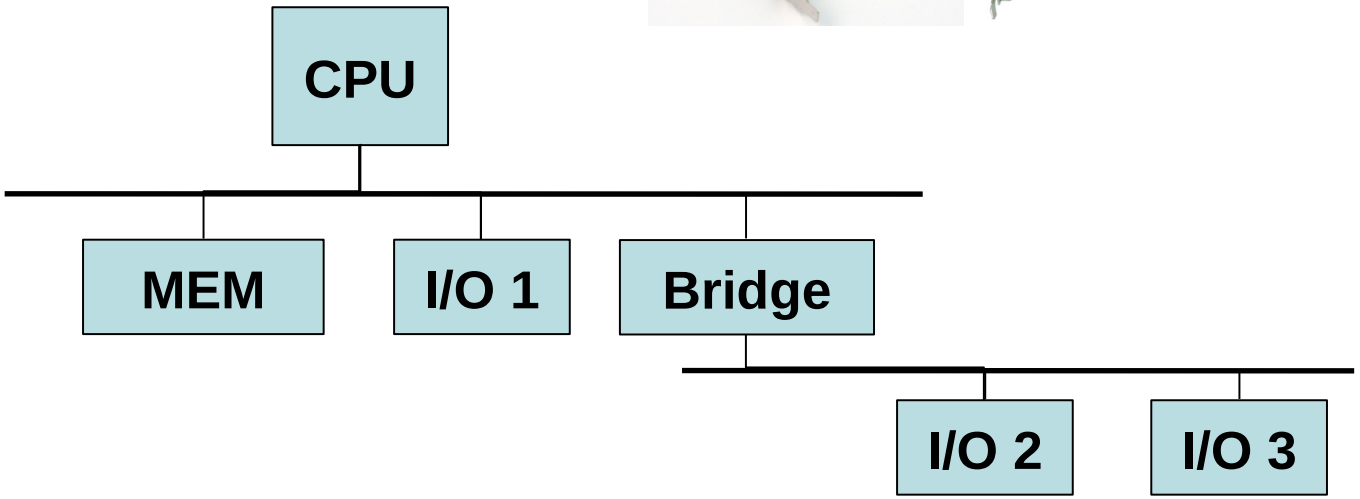
Address Decoder - Idea

- Logical Structure:
- (Illusion)



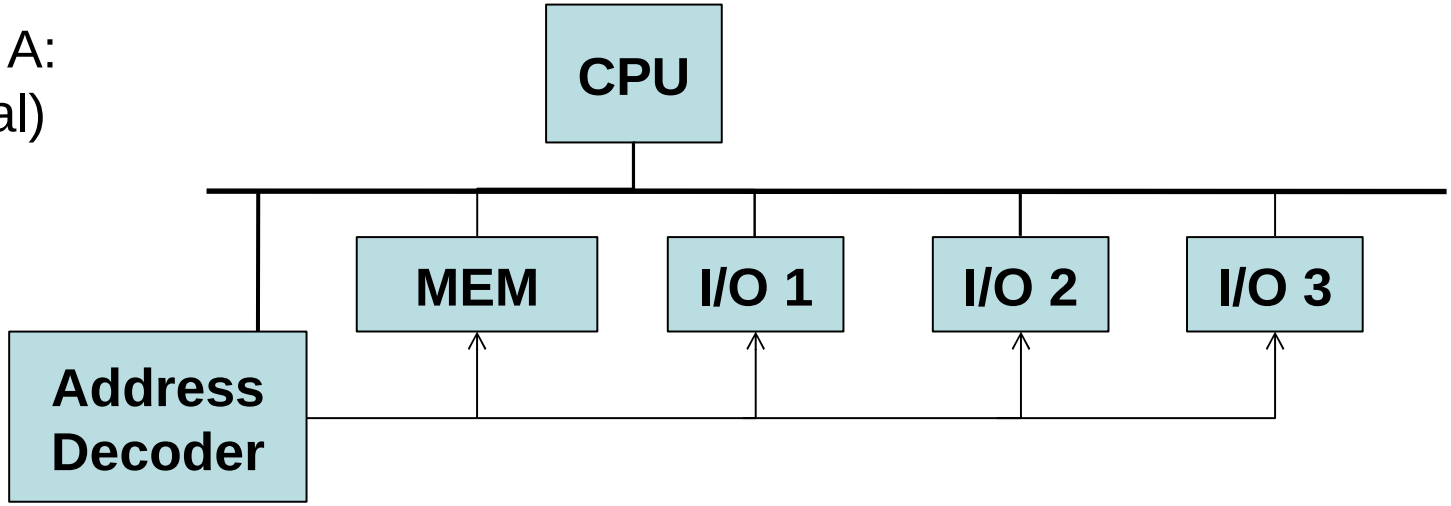


- Possible physical arrangement :

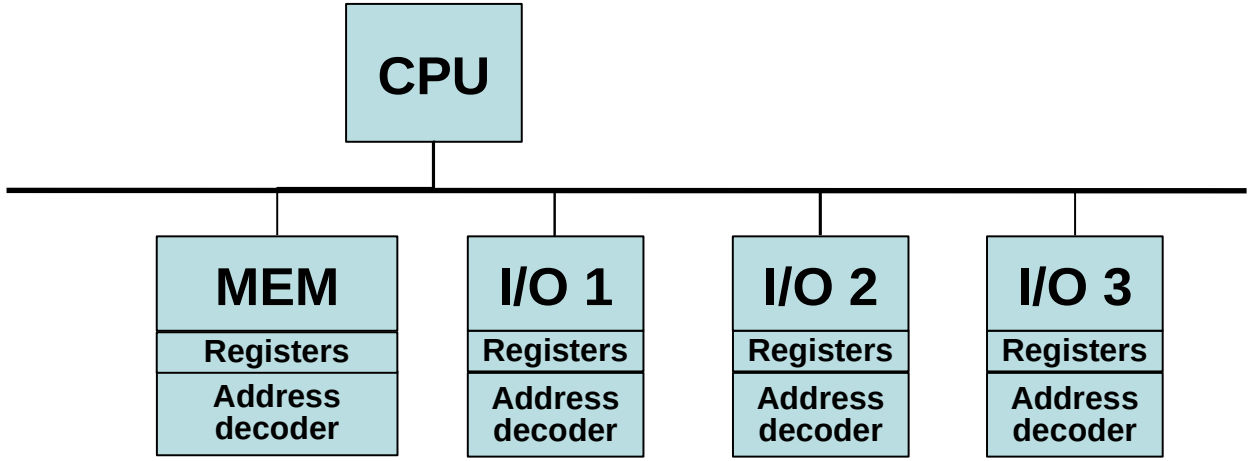


Address Decoder - Idea

- Option A:
  (Central)



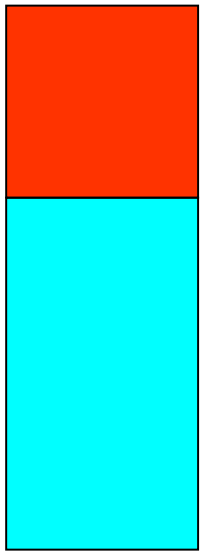
- Option B:
- (Autonomous)



Memory mapped I/O

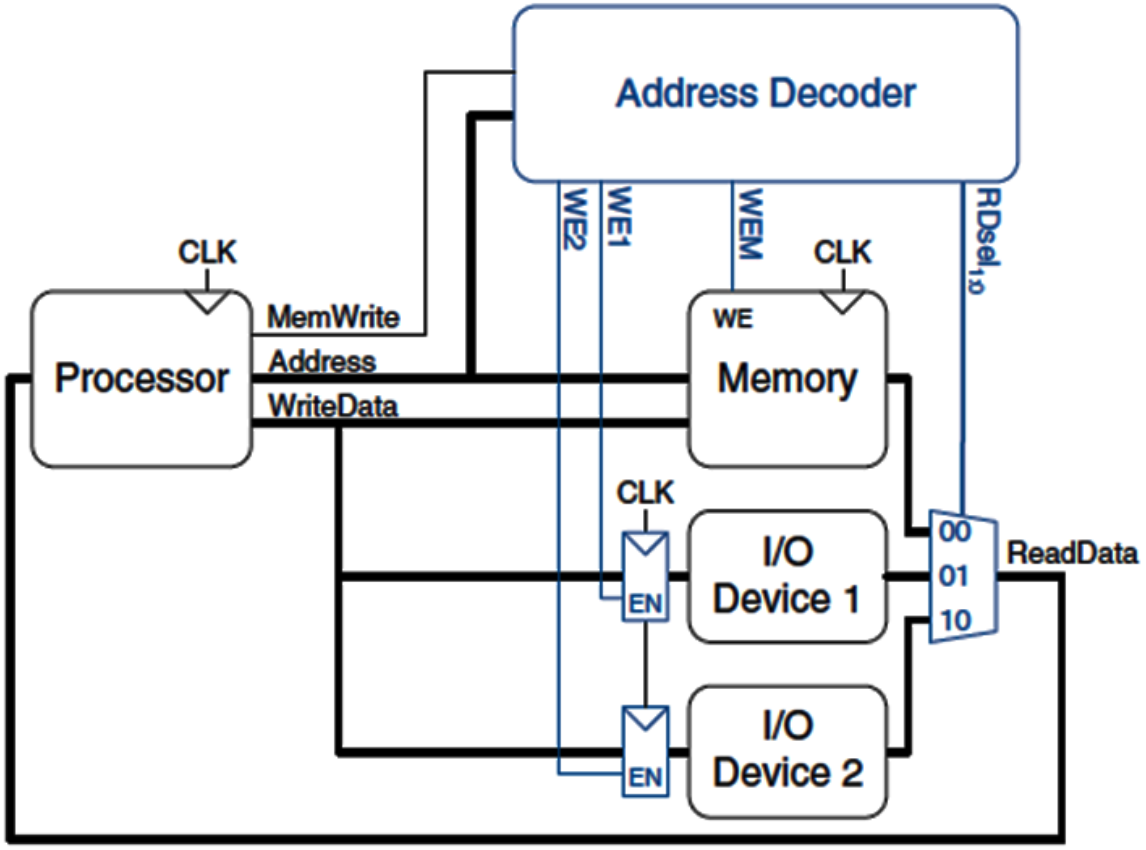
- Idea: To communicate with I / O peripherals (keyboard, monitor, printer) we can use the same interface as for memory communication (MIPS: lw, sw instruction).

Common address space for I/O and memory

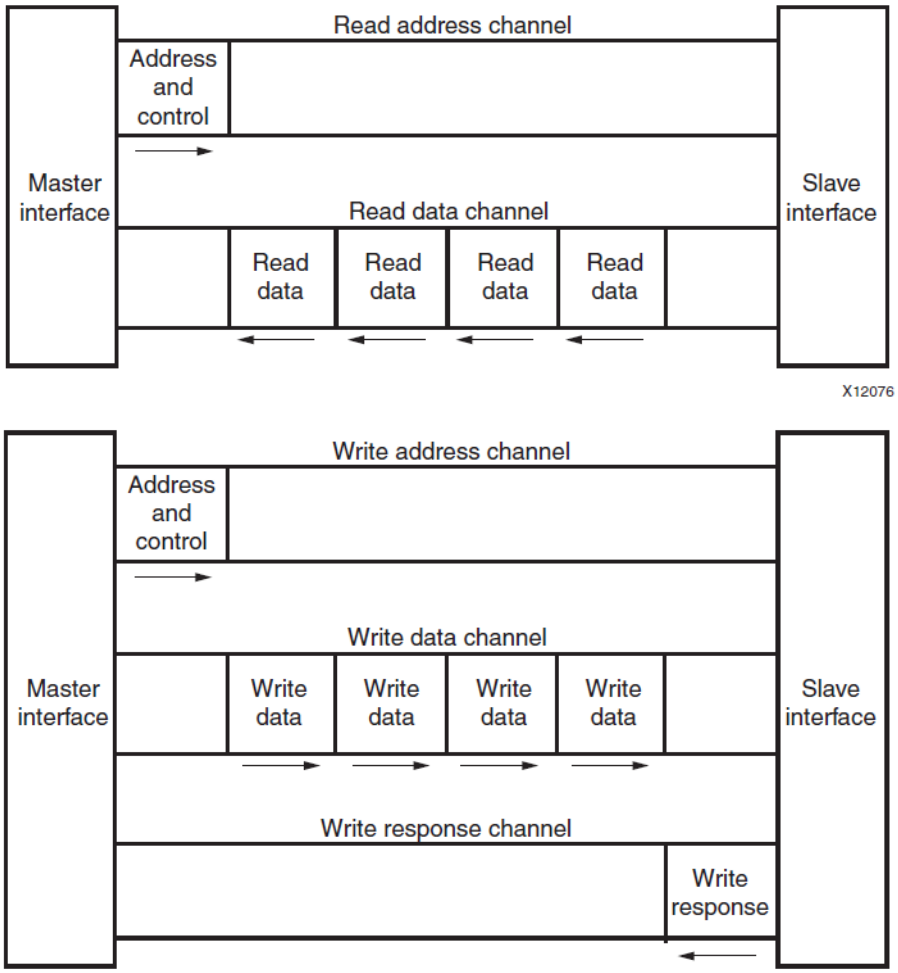


I/O ports are mapped into memory

Memory



MicroZed uses on-chip bus AXI !



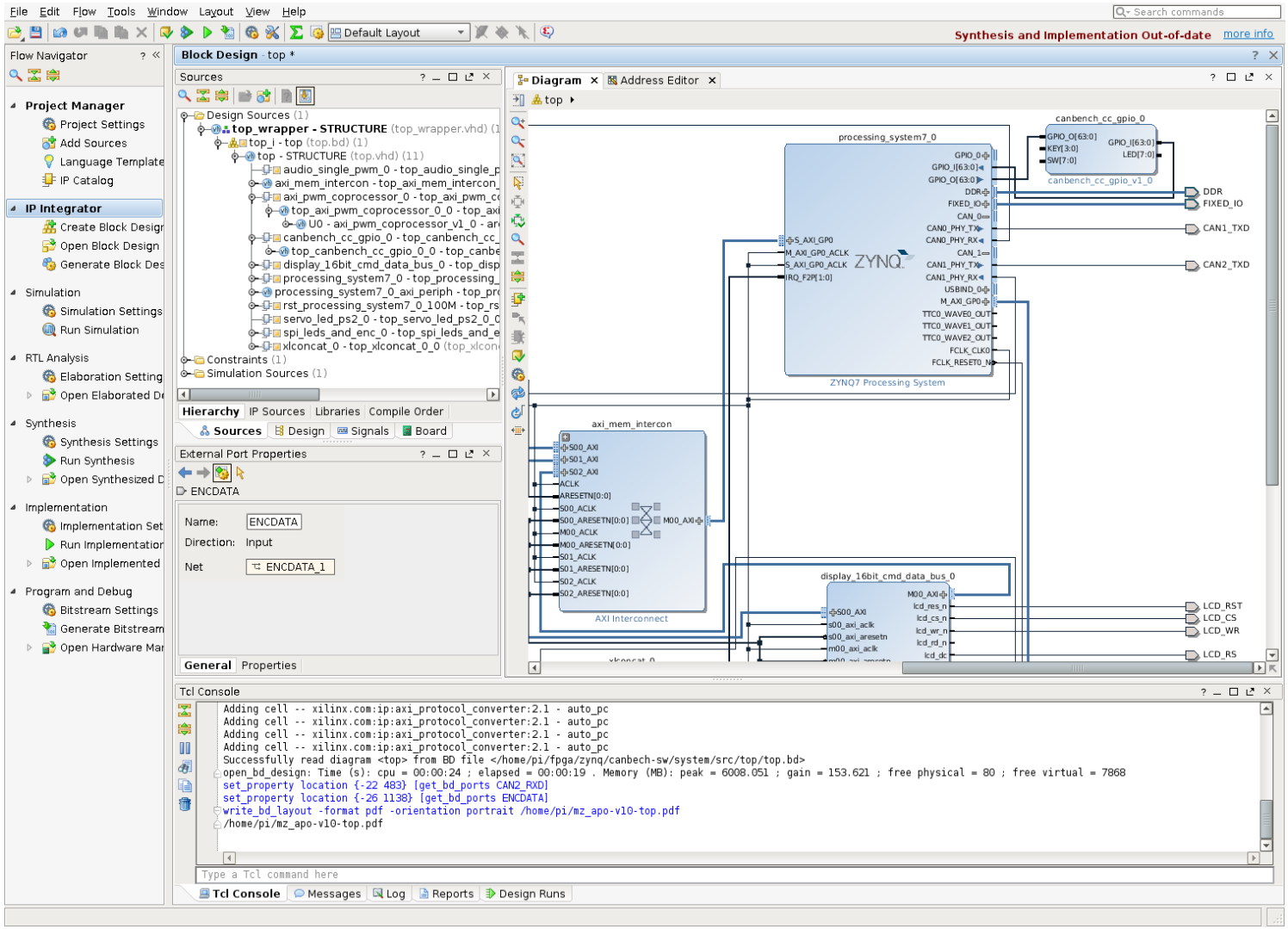
Advanced e**X**tensible **I**nterface (AXI) consists of five different channels:

- Read Address Channel
- Read Data Channel
- Write Address Channel
- Write Data Channel
- Write Response Channel

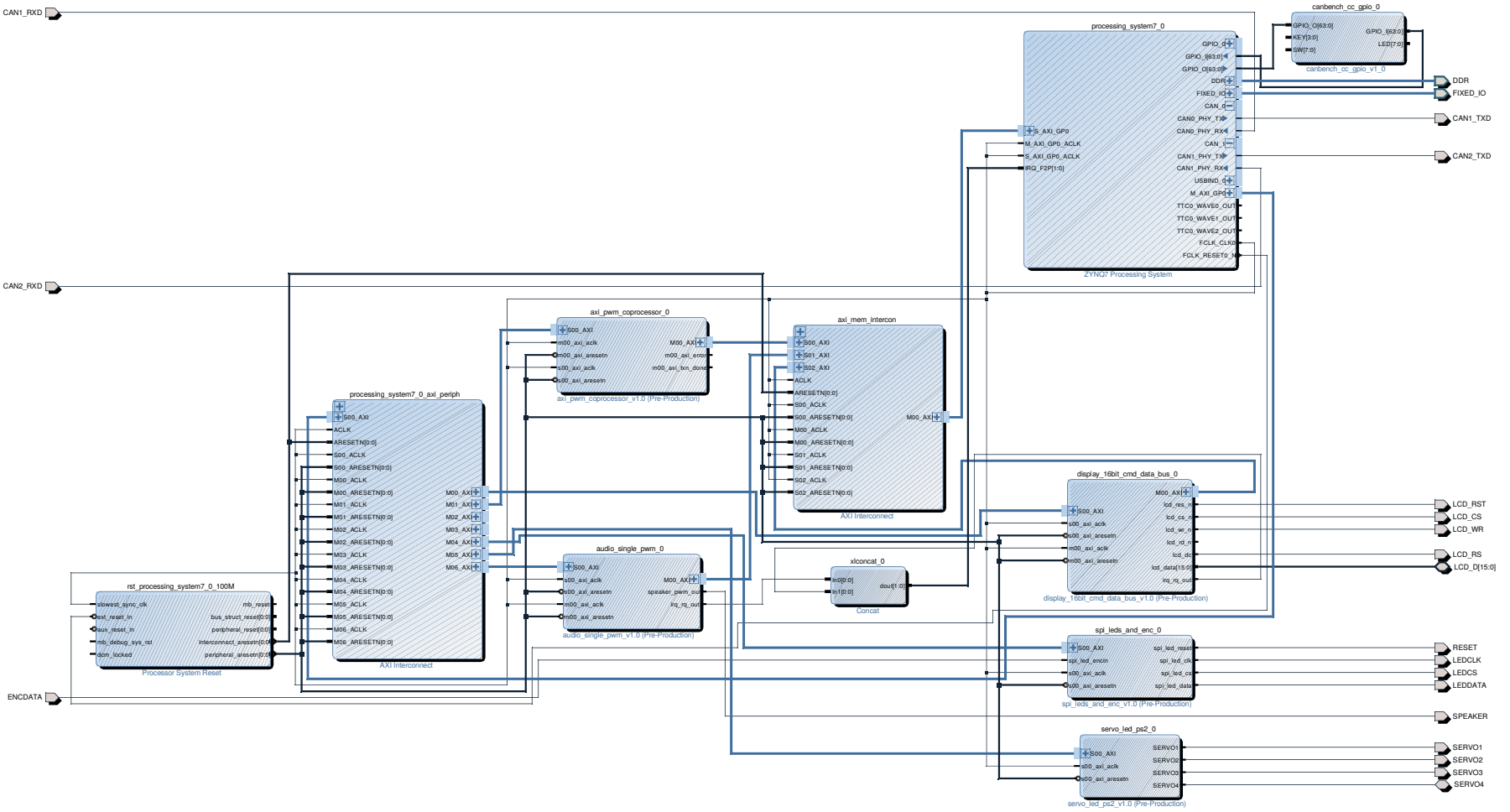
On-chip buses are used only inside integrated circuits and utilize switching multiplexors that allow simultaneous reading and writing operations. They will be described by LSP course (the next semester).

***External PC buses** (ISA, PCI, SATA, PCIe) **have different operation!** See preceding lecture.*

Design of Peripherals in Xilinx Vivado



Vivado Design of Peripherals for APO



Compete design sources: https://cw.fel.cvut.cz/b182/courses/b35apo/documentation/mz_apo/start

MicroZed Physical Memory

Address start	Length	
0x0000 0000	1 GB	DRAM
0x4000 0000	1 GB	AXI bus port 0 to FPGA
0x43c00000	16 bytes	APO – LCD display
0x43c20000	32 bytes	DC Motor on PMOD1
0x43c30000	32 bytes	DC Motor on PMOD2
0x43c40000	48 bytes	APO – Peripherals
0x43c50000	32 bytes	4× Servo or PS2
0x43c60000	32 bytes	Simple audio
0x8000 0000	1 GB	AXI bus port 1 to FPGA
0xE000 0000		Reserved for system
0xFFFC 0000	256 kB	On chip static memory

But Cortex A9 runs with GNU/Linux OS that is configured for paging (MMU), thus, we cannot use directly physical addresses!

Mapping Hardware into User Process on Linux

It is simplified part of the code that you use in your semester project

```
int fd = open("/dev/mem",  /* we ask for physical memory addresses */
              O_RDWR /* with read and write access */
              | O_SYNC /* and non-cached for /dev/mem */
              );
unsigned char *mem = (unsigned char *) mmap(
              NULL, /* kernel selects virtual address */
              0x4000 /* our required size = MicroZed physical mem view */,
              PROT_READ | PROT_WRITE, /* allow read and write*/
              MAP_SHARED, /* visible to other processes*/
              fd,  /* handle of an already opened file */
              0x43c40000 /* offset in file, here I/O physical base address*/  );
```

Note: For simplification, we have supposed that the size and offset are already align to page size.
Full template: https://gitlab.fel.cvut.cz/b35apo/mzapo_template

SPI Connected Knobs and LEDs Registers and Keyboard

base of SPILED port region

SPILED_REG_BASE_PHYS 0x43c40000

SPILED_REG_SIZE 0x00004000

RGB LED 1 color components – 8 bits each

SPILED_REG_LED_RGB1 0x43c40010

SPILED_REG_LED_RGB1_o 0x0010

RGB LED 2 color components – 8 bits each

SPILED_REG_LED_RGB2 0x43c40014

SPILED_REG_LED_RGB2_o 0x0014

Three 8 bit knob values

SPILED_REG_KNOBS_8BIT 0x43c4024

SPILED_REG_KNOBS_8BIT_o 0x0024

line of 32 LEDs for binary value representation

SPILED_REG_LED_LINE 0x43c4004

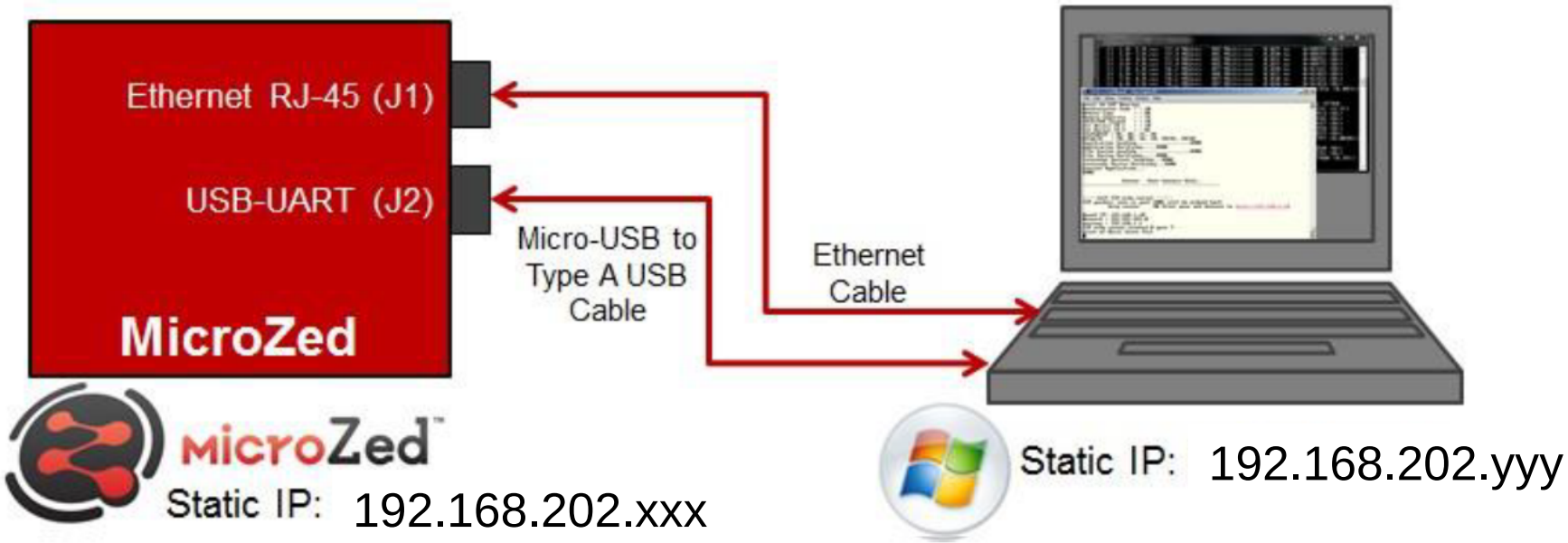
SPILED_REG_LED_LINE_o 0x0004

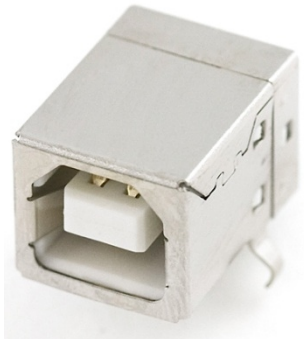
There are even more registers.

See: https://gitlab.fel.cvut.cz/b35apo/mzapo_template/-/blob/master/mzapo_regs.h

*Connecting to the board

Connection to Board

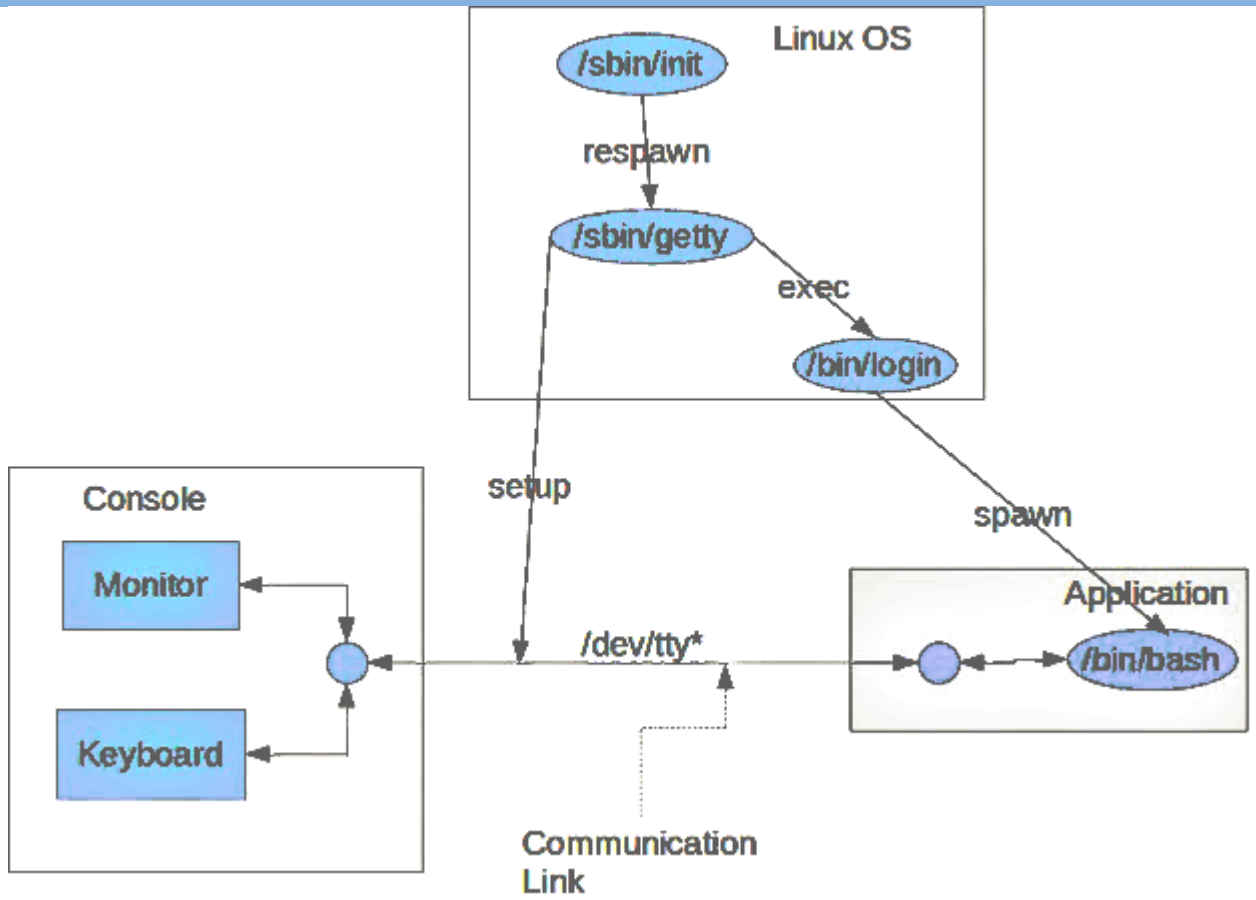




Notes:

1. *Micro USB is replaced by more robust USB type B on our carry board.*
2. *In Linux,*
   - *GtkTerm allows USB connection*
   - *SSH utilizes Ethernet*

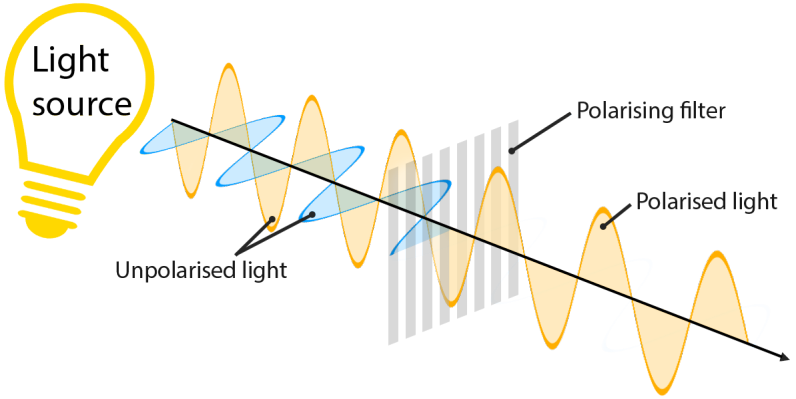
Linux Console Concept



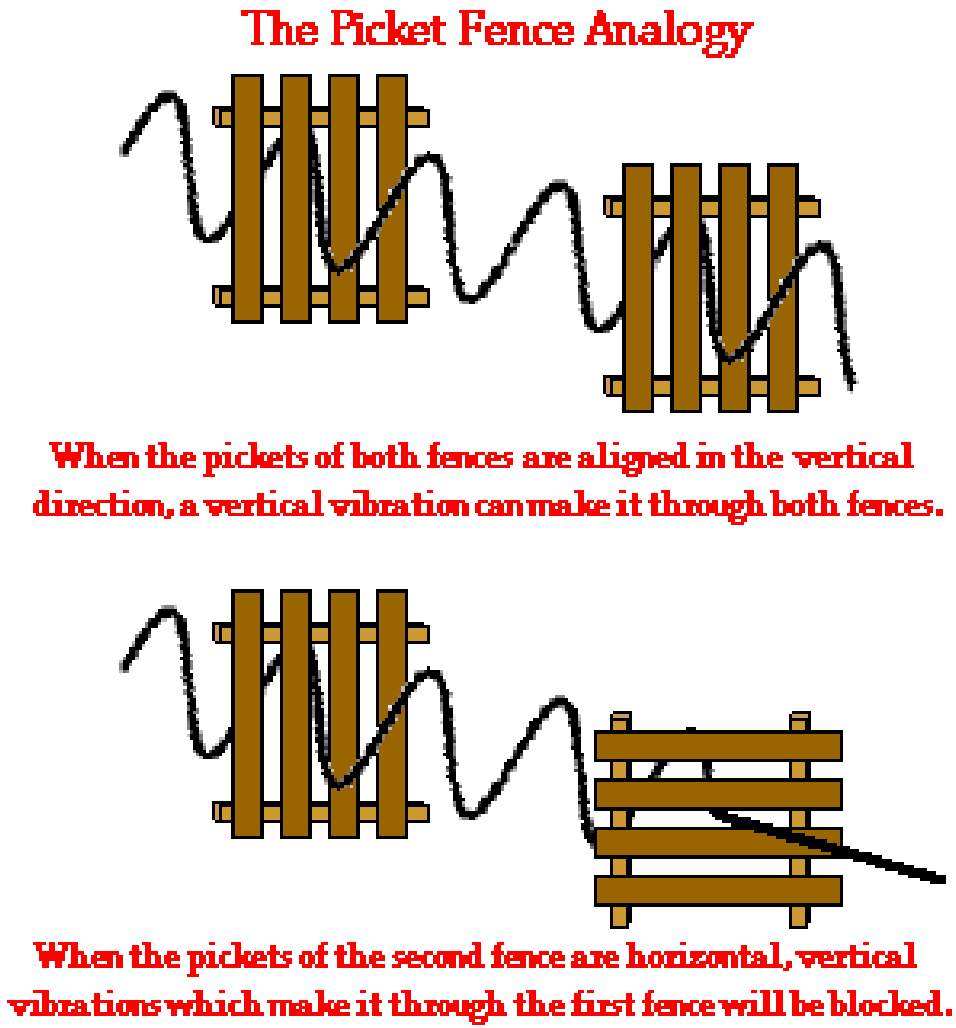
- GtkTerm uses /**dev/ttyUSB0**
- SSH connect through Ethernet and uses **/dev/ttyp****

*LCD Display

Polarization of Light

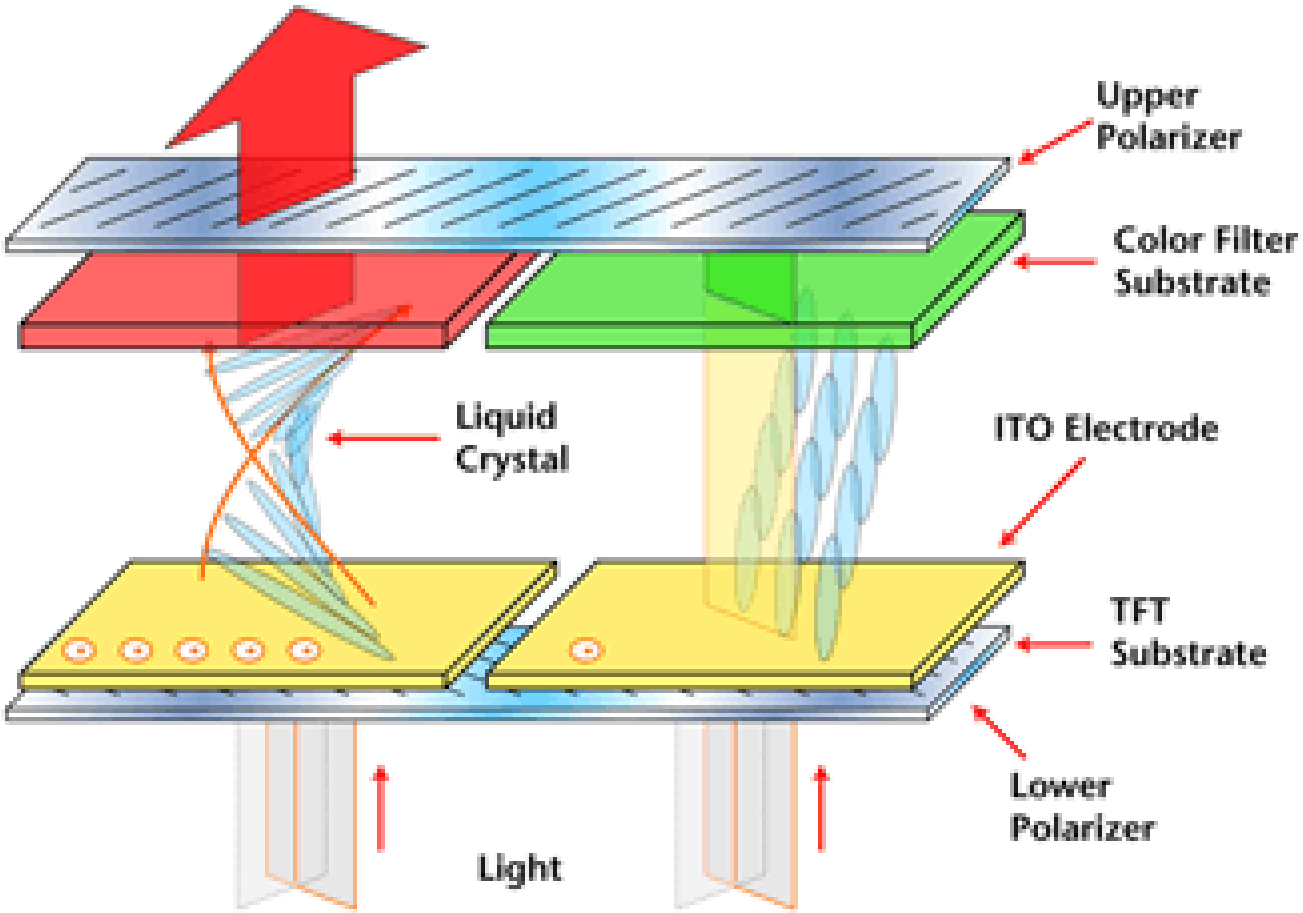


Source: physics.stackexchange.com



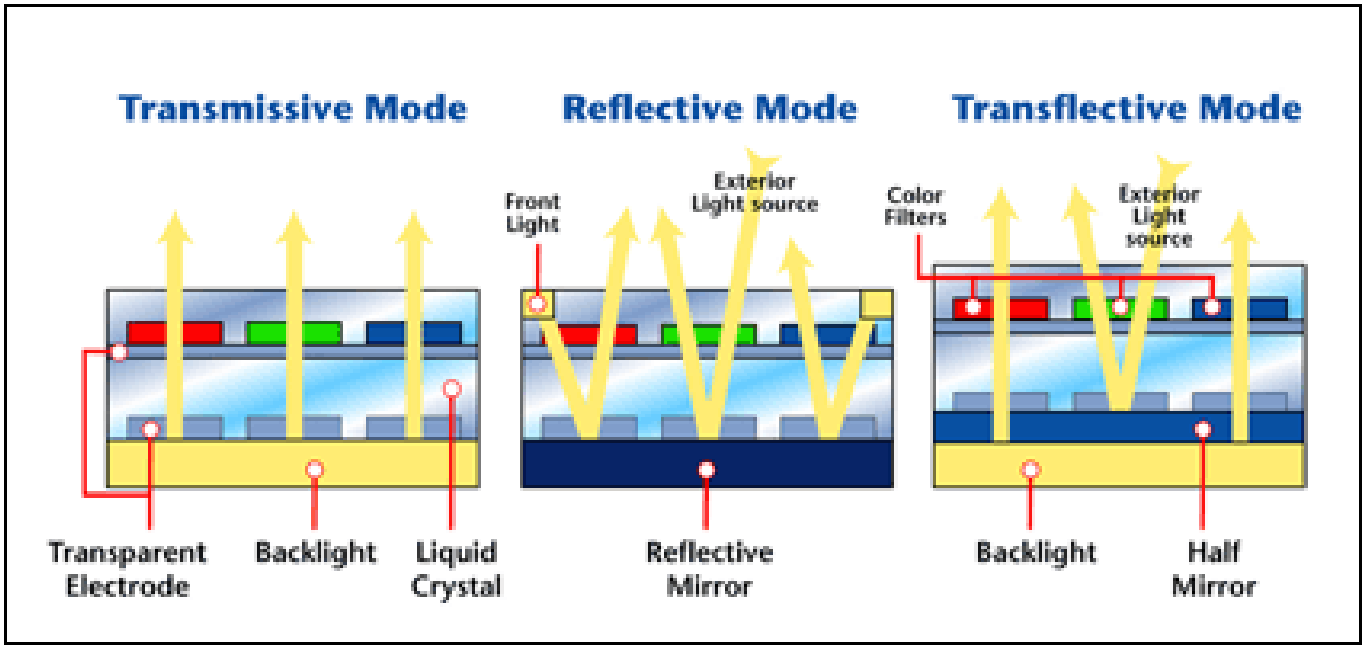
Source: www.physicsclassroom.com

LCD Lighting Theory



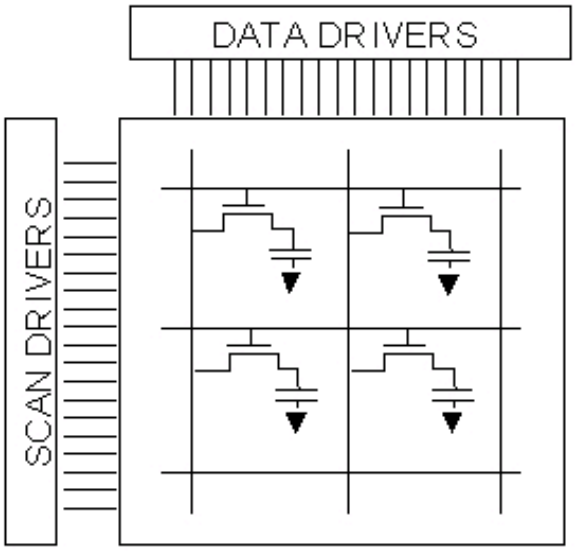
Source: Matrix

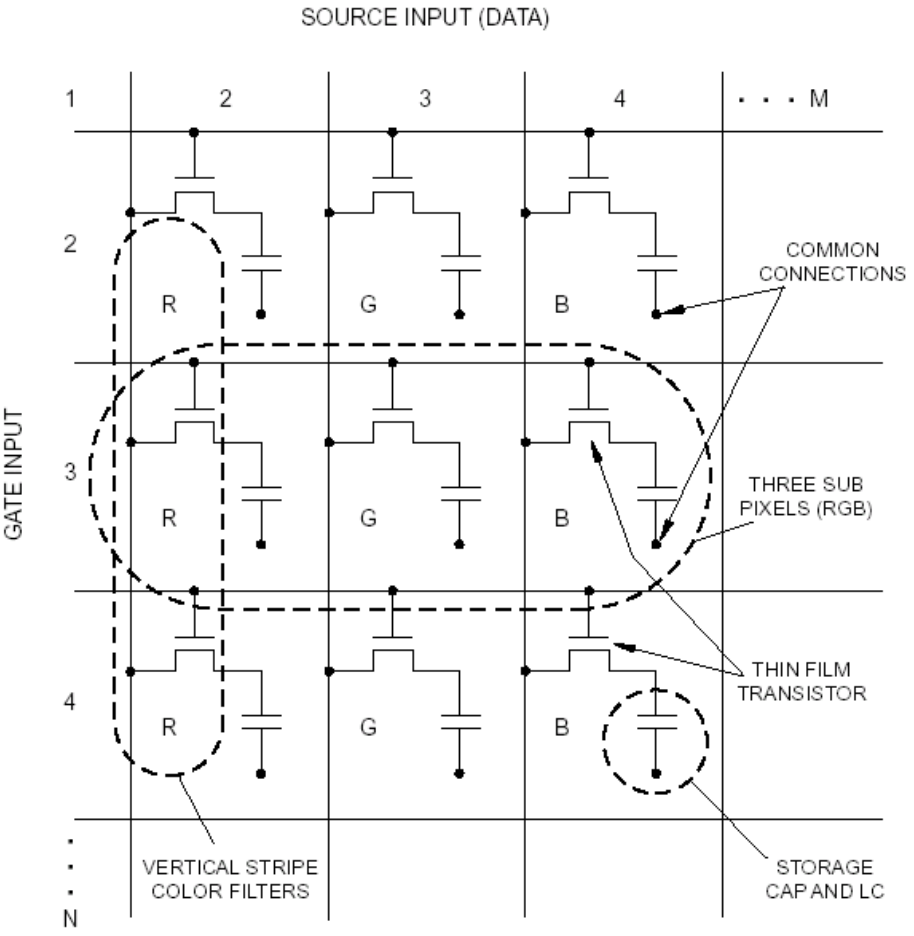
LCD Lighting Theory

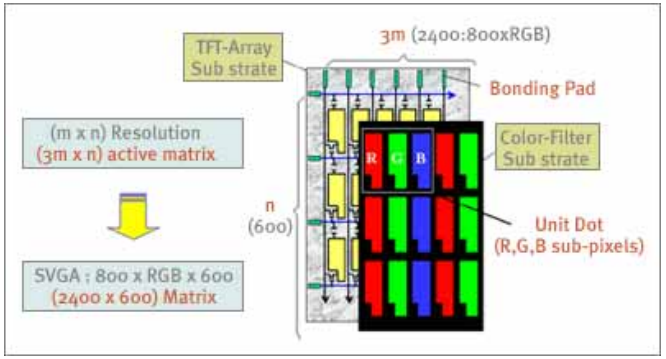


Source:

TFT with Active Matrix

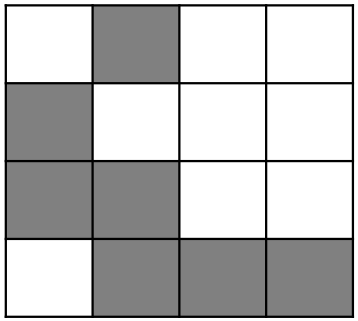






Source: Matrix

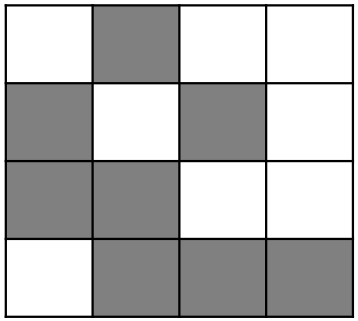
Compare with Memory Matrix from 4th lecture



TFT display only writes and usually selects only 1 pixel !

New value = 1

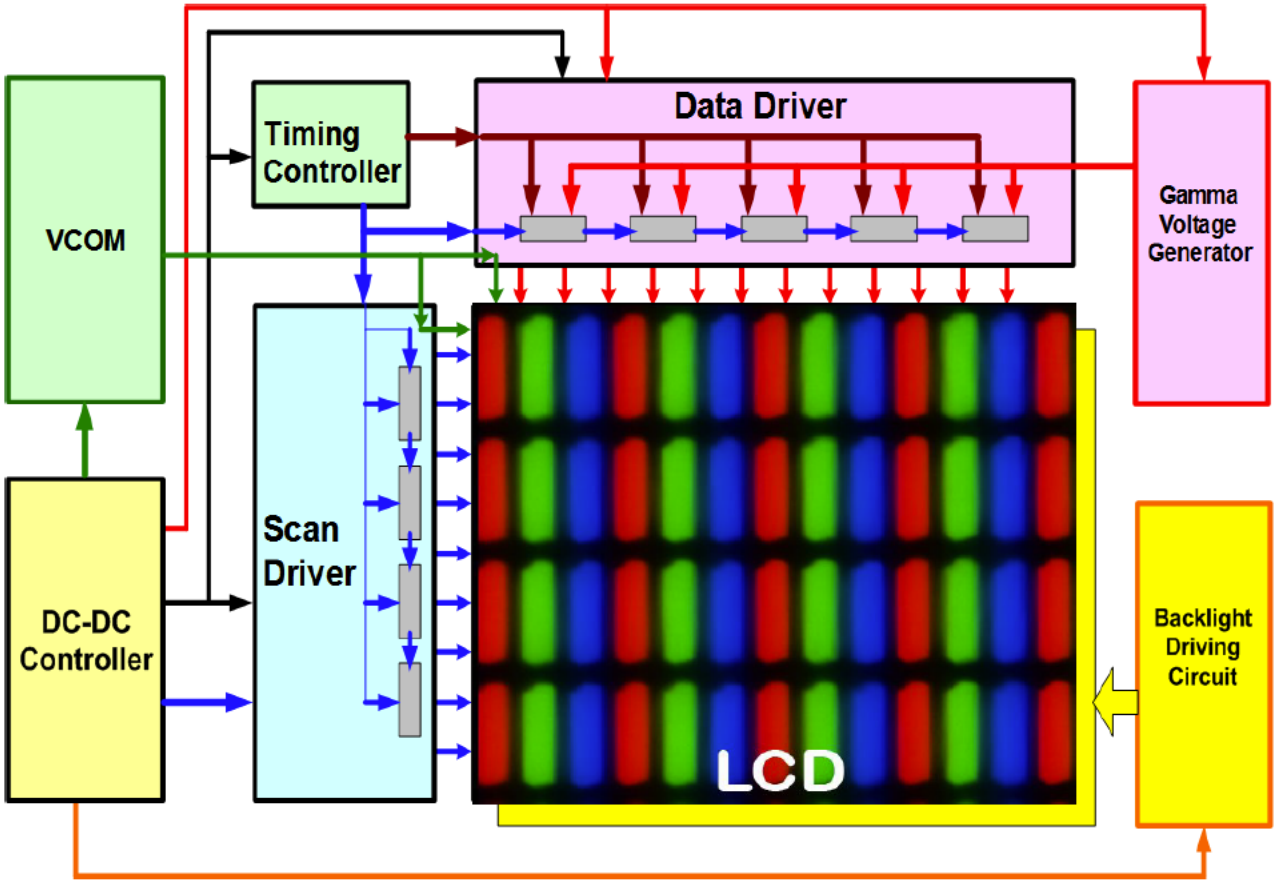
Compare with Memory matrix 4th lecture



TFT display only writes and select 1 pixel !

New value = 0

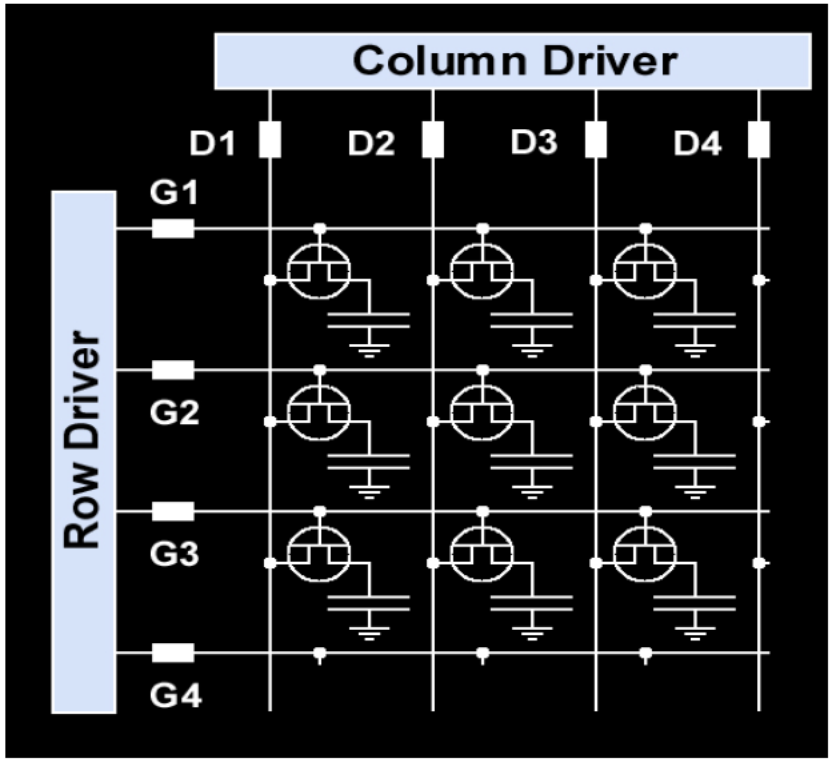
LCD Control



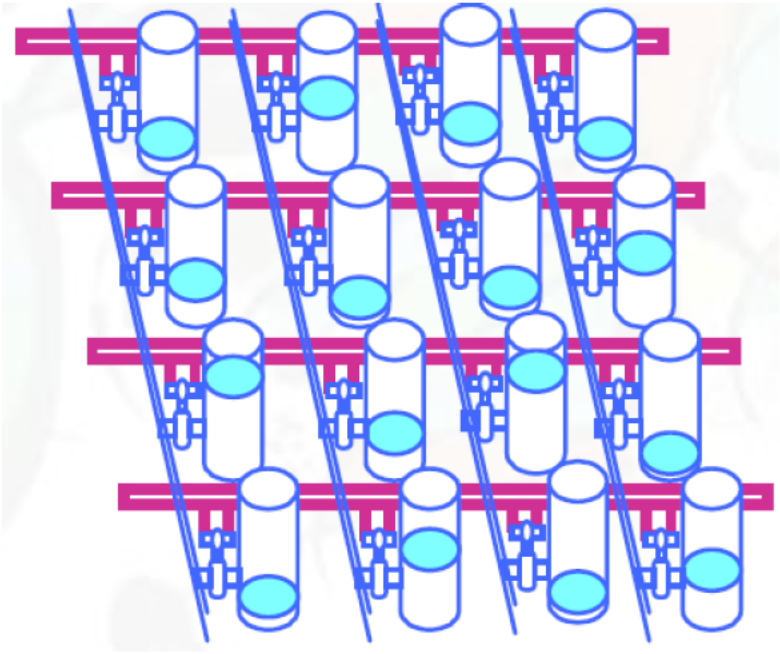
Source: Dr. Zhibing Ge, College of Optics and Photonics

LCD Pixel has More Levels !

TFT (switch) + LC cell (capacity)

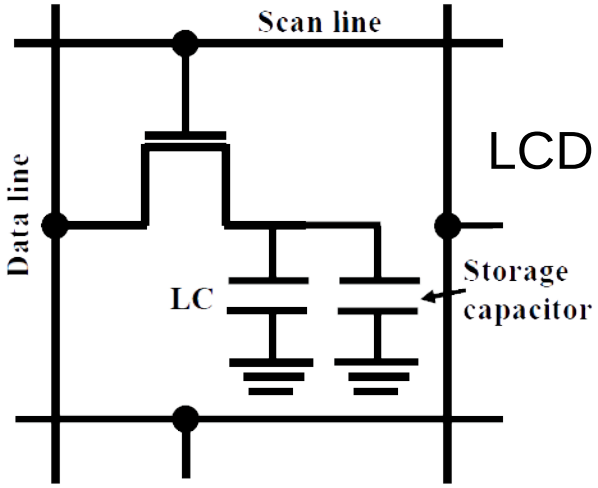


Hydrant (switch)+ Bucket (capacity)

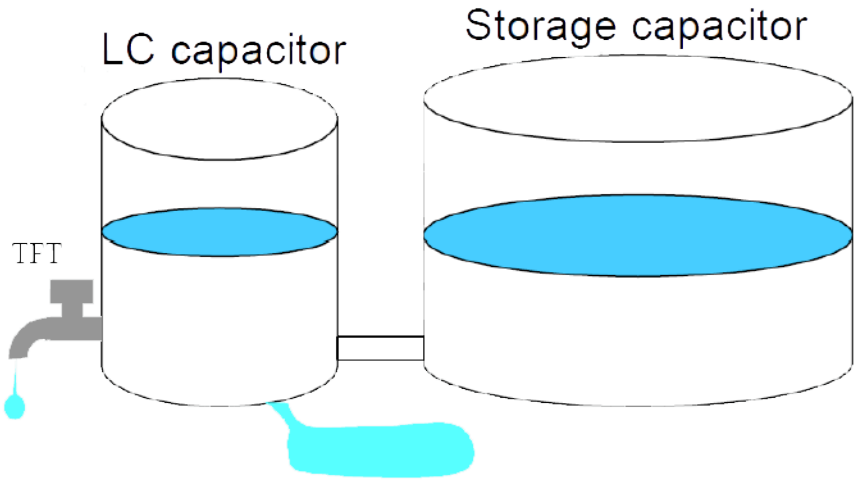


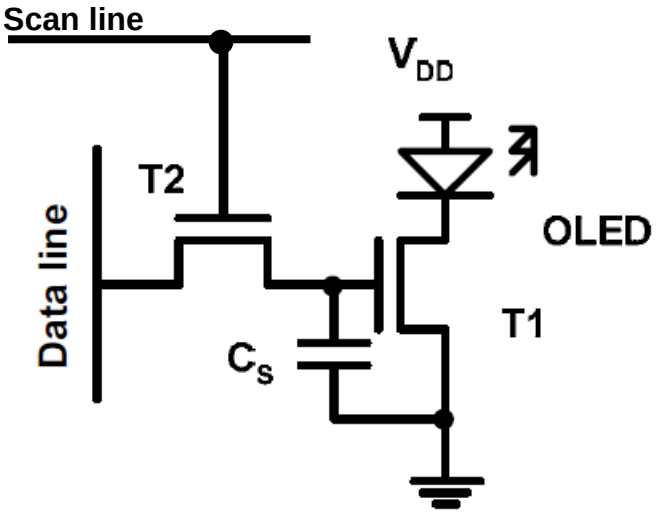
Source: Dr. Zhibing Ge, College of Optics and Photonics

LCD Refreshing



LCD TFT





LCD display needs periodic refresh as DRAM, but in slower rate.
Typical values of C_s are from 100 fF (=0.1pF) to 2000 fF (=2pF), in DRAM from 10 to 50 fF.

LCD read/write

Our program contains assembler instruction of compiled C code

register1 = * address;

or

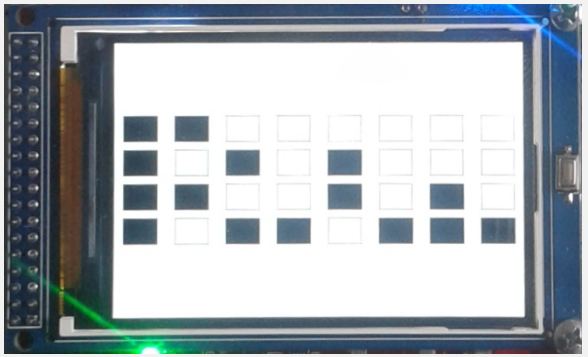
*** address = register2;**

Program in DDR3

rd/wr virtual address

APO MicroZed

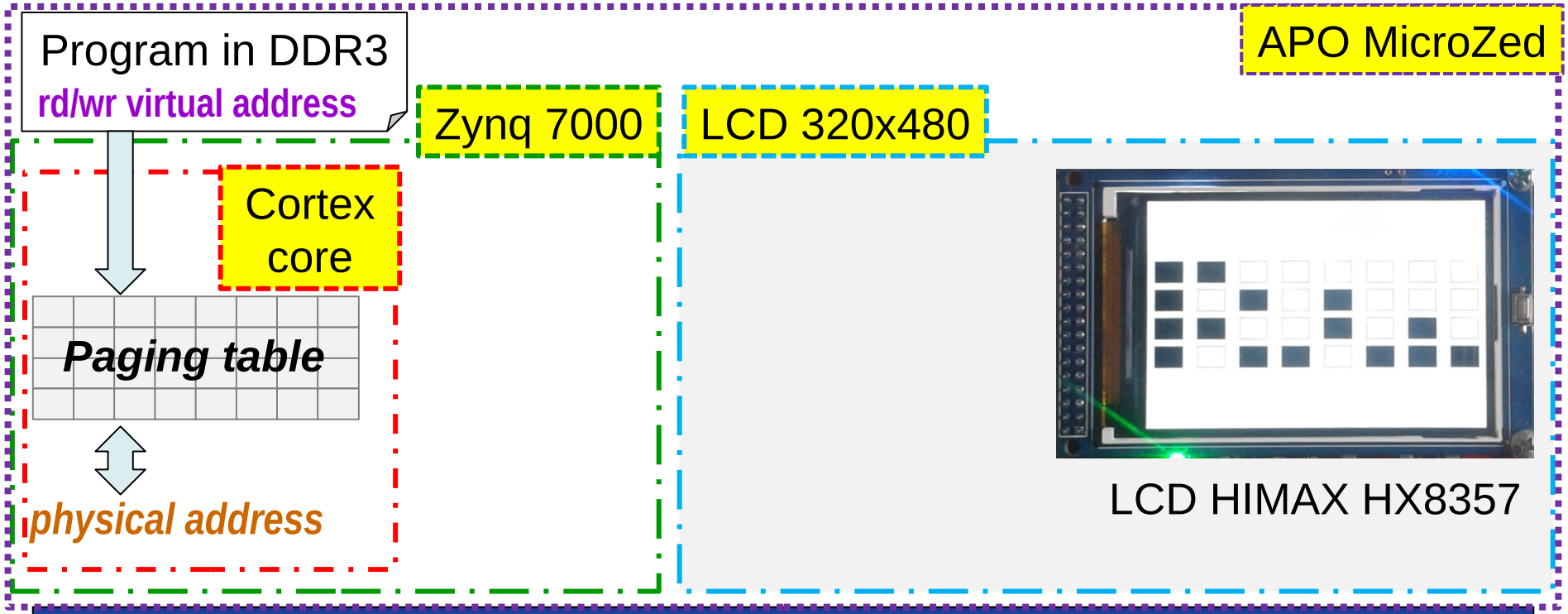
LCD 320x480



LCD HIMAX HX8357

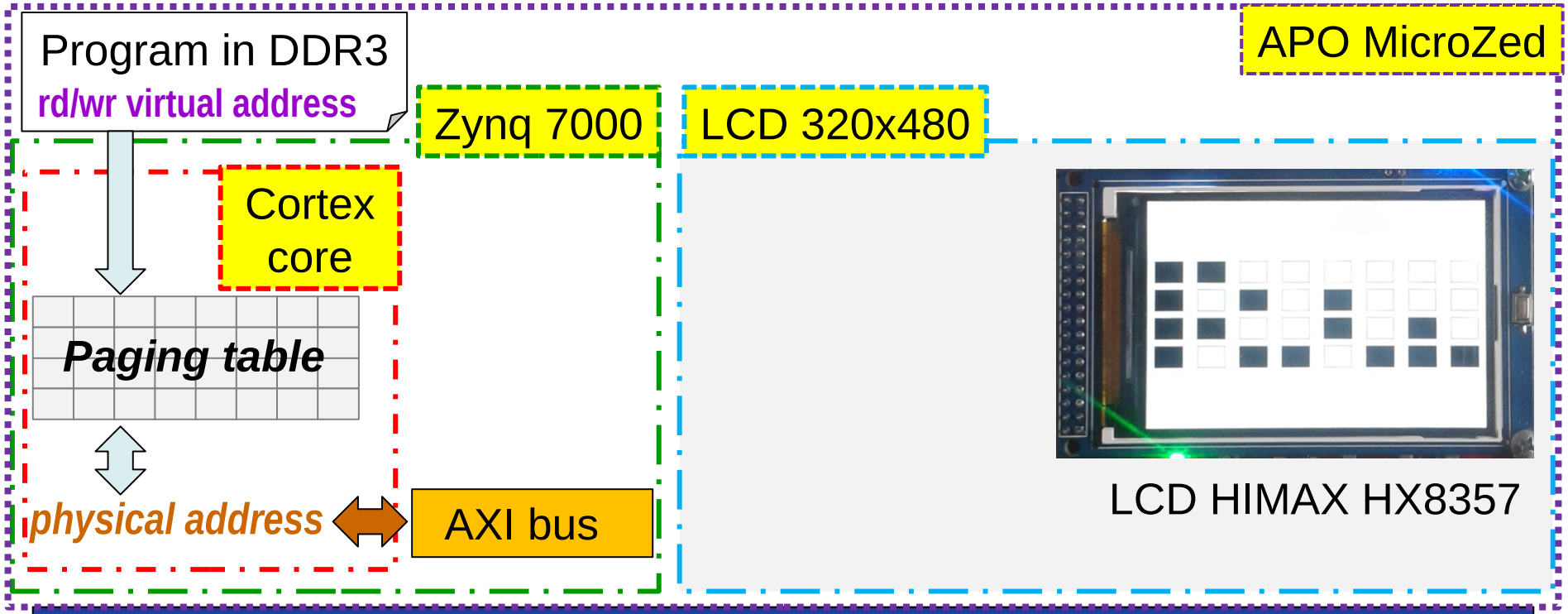
LCD read/write

Virtual address is transformed to physical address during FETCH phase of pipeline



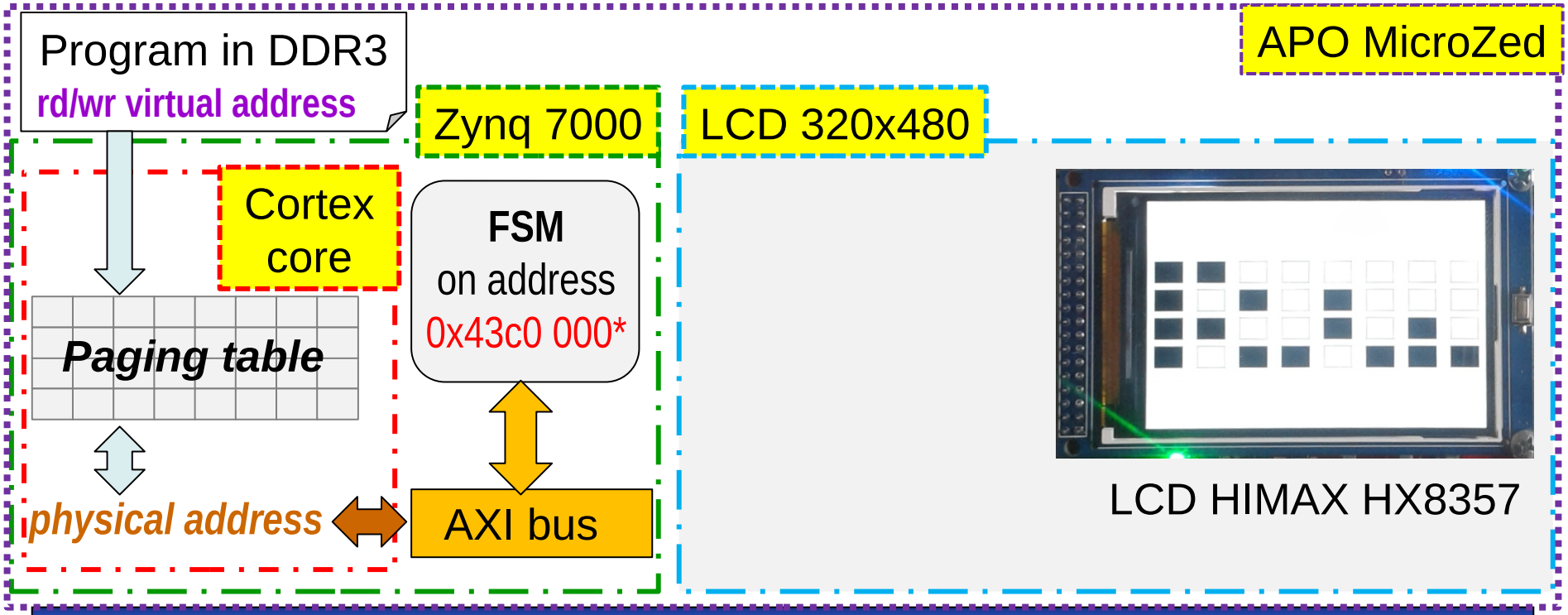
LCD read/write

Physical address and data are send to AXI bus



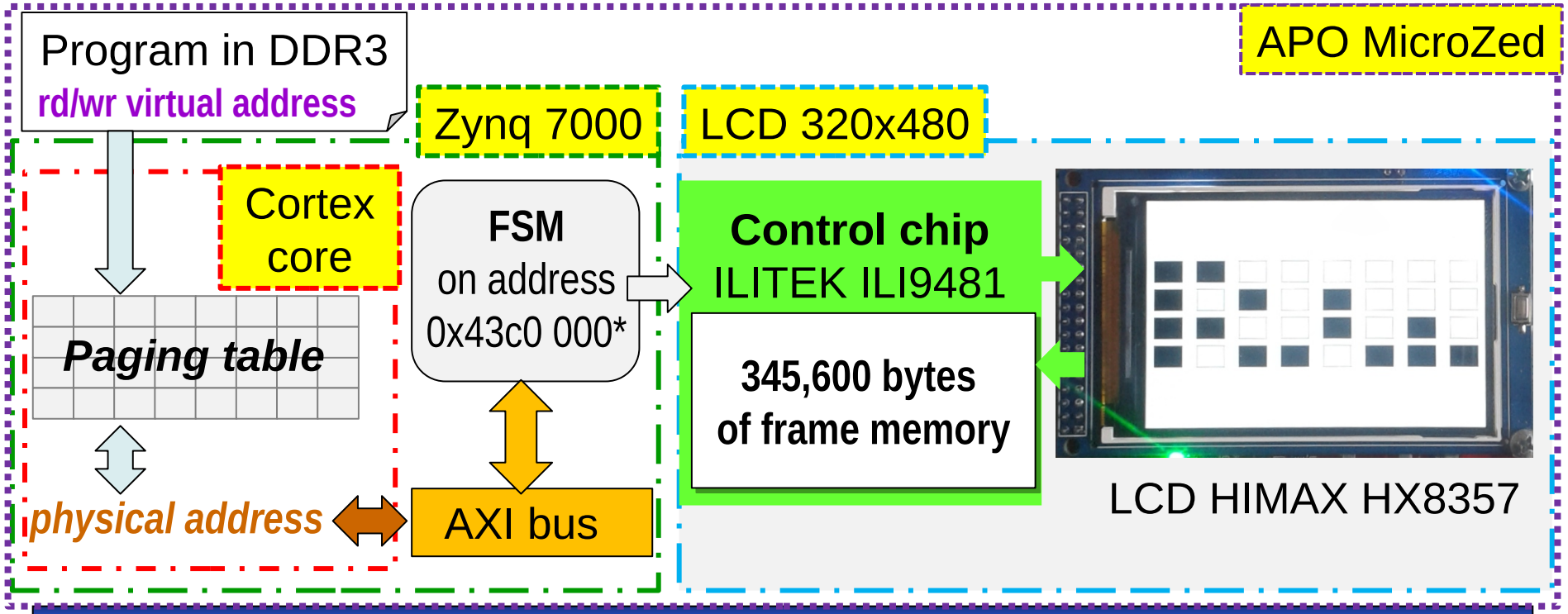
LCD read/write

FSM (Finite State Machine, cz: konečný automat) compares addresses on AXI bus.
If some address is in range from 0x43c0 0000 to 0x43c0 000F then FSM accepts corresponding data for LCD.



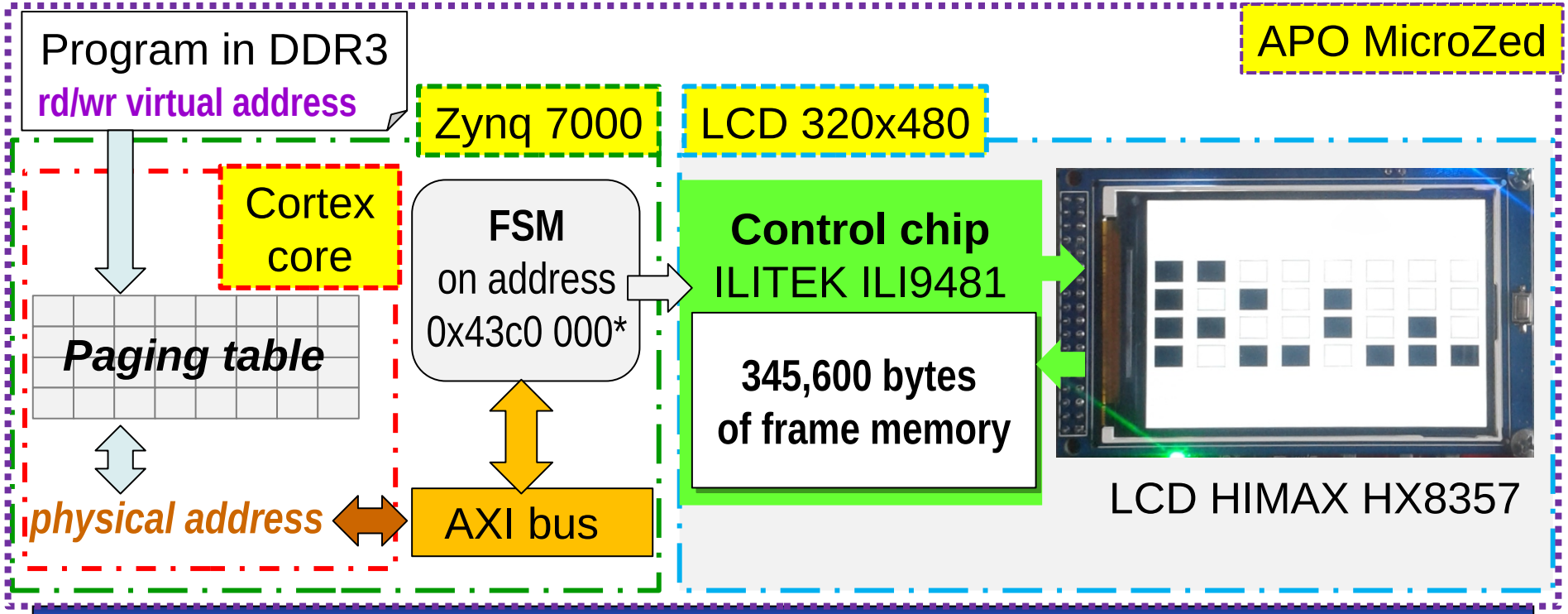
LCD read/write

FSM transmit data to LCD by generating appropriate signals for LCD control chip that periodically refreshes TFT-LCD display.
If a new request for FSM appears before the previous is done, FSM is sending WAIT to AXI bus until it can accept next data/command.



LCD read/write

Virtual Base Address	Data Type	Control Chip Operation
+0	uint16_t	0x1 - reset LCD, bit0 == 0 - no function
+8	uint16_t	LCD control command
0xC	uint16_t	write 16 bit color pixel (bits 15..0) to frame memory
0xC	uint32_t	write 2 following pixels: bits 15..0 | bits 31..0 to mem.



Parallel LCD Display 480 × 320

The LCD display it connected to 16-bit port is not mapped into address space as a framebuffer in the actual FPGA design. Display requires relatively complex setup sequence to setup controller to drive actual screen “glass”. The setup sequence consists of series sending command word to **CMD** register and then corresponding value to **DATA** port. The setup is provided in the file https://gitlab.fel.cvut.cz/b35apo/mzapo_template/-/blob/master/mzapo_parlcd.c

base of PARLCD port region

```
PARLCD_REG_BASE_PHYS 0x43c00000
PARLCD_REG_SIZE      0x00004000
```

RGB LED 1 color components – 8 bits each

```
PARLCD_REG_CMD_o          0x0008
PARLCD_REG_DATA_o         0x000c
```

When display controller is setup then the content can be updated by writee command 0x2c to **CMD** port followed by 480 × 320 16-bit RGB565 words representing pixel color written to **DATA** port. Two consecutive pixels can be written by 32-bit write to **DATA** port.

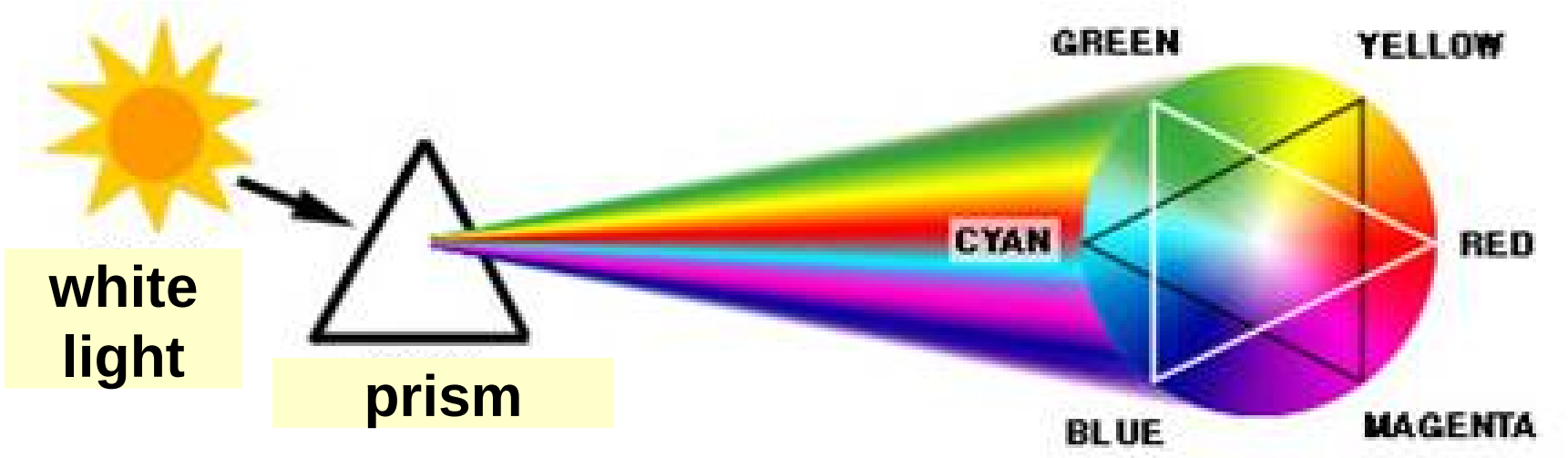
See: https://gitlab.fel.cvut.cz/b35apo/mzapo_template/-/blob/master/mzapo_regs.h

What is HSV ?

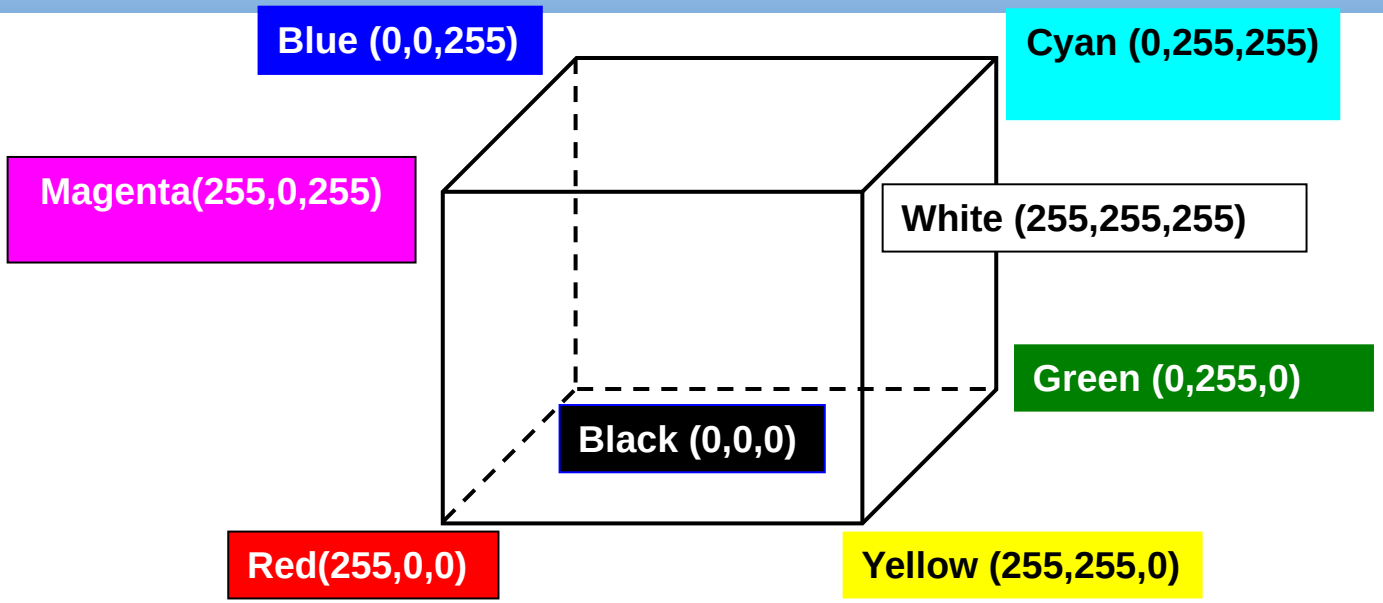
reality versus palette

[www.realcolorwheel.

Additive color mixing

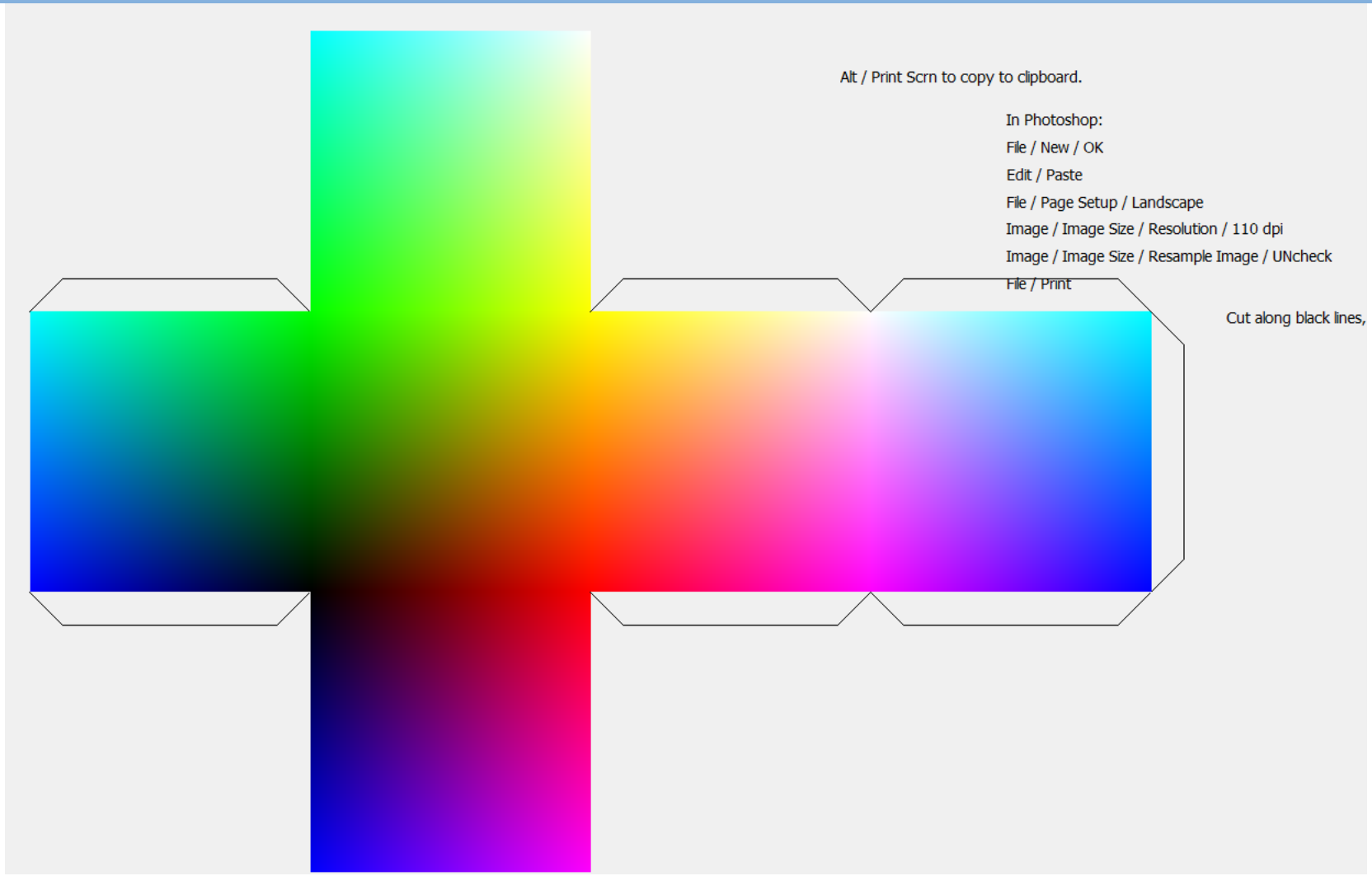


RGB color cube



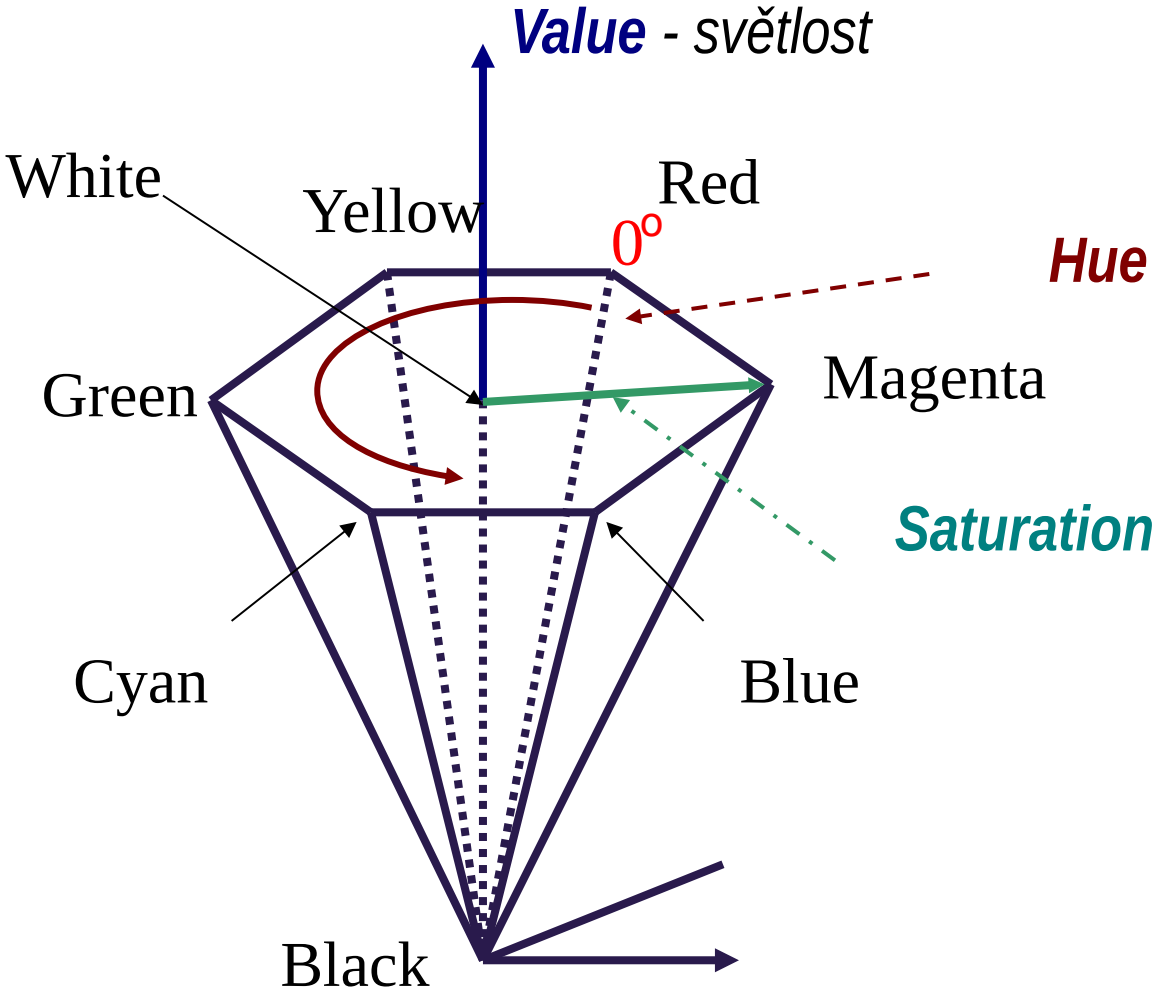
- *Additive coloring is used for example on monitors, the effects of individual colors are added as each one behaves as a light source.*

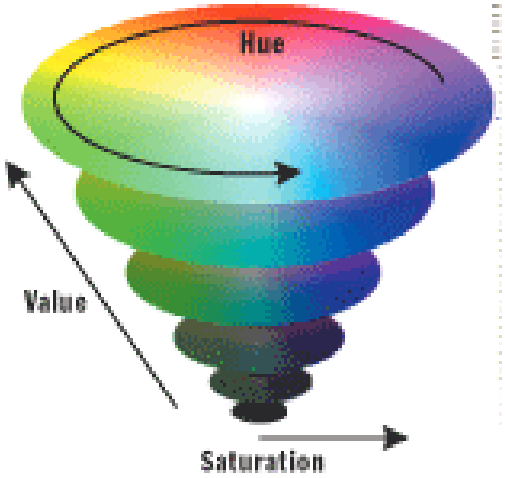
RGB Cube with Unintuitive Distribution of Color



Source: http://people.duke.edu/~ng46/borland/graphics.htm

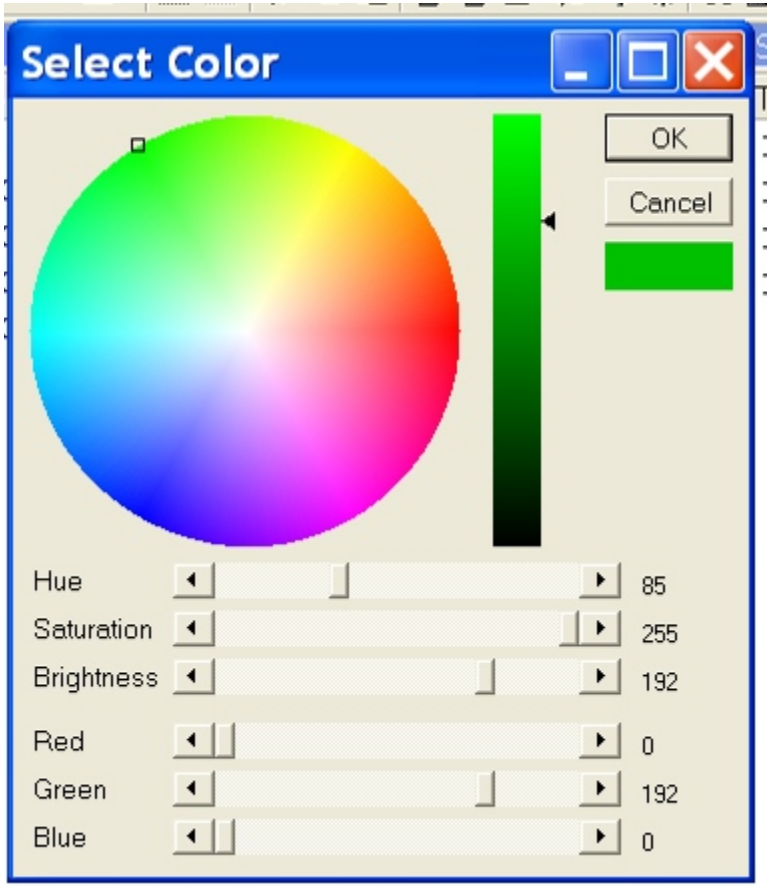
HSV system for intuitive color selection





Example of HSV Color Picker

Converting between RGB and HSV is quite complex, but there are C codes on the web.



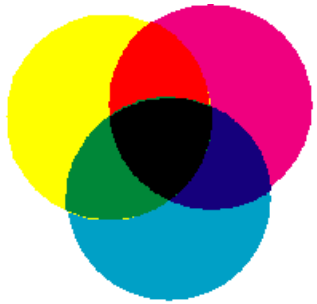
CMYK vs RGB/HSV

- *Subtractive color mixing occurs when printing - the effects of each color are subtracted as each component acts as a white light filter.*

- *The CMY color cube is obtained by swapping the opposite vertices of the RGB cube, i.e., black ↔ white, yellow ↔ blue, etc..*

$$\begin{matrix}\text{Cyan}\\ \text{Magenta}\\ \text{Yellow}\end{matrix}\begin{bmatrix}C\\M\\Y\end{bmatrix} = \begin{bmatrix}255\\255\\255\end{bmatrix} - \begin{bmatrix}R\\G\\B\end{bmatrix}$$

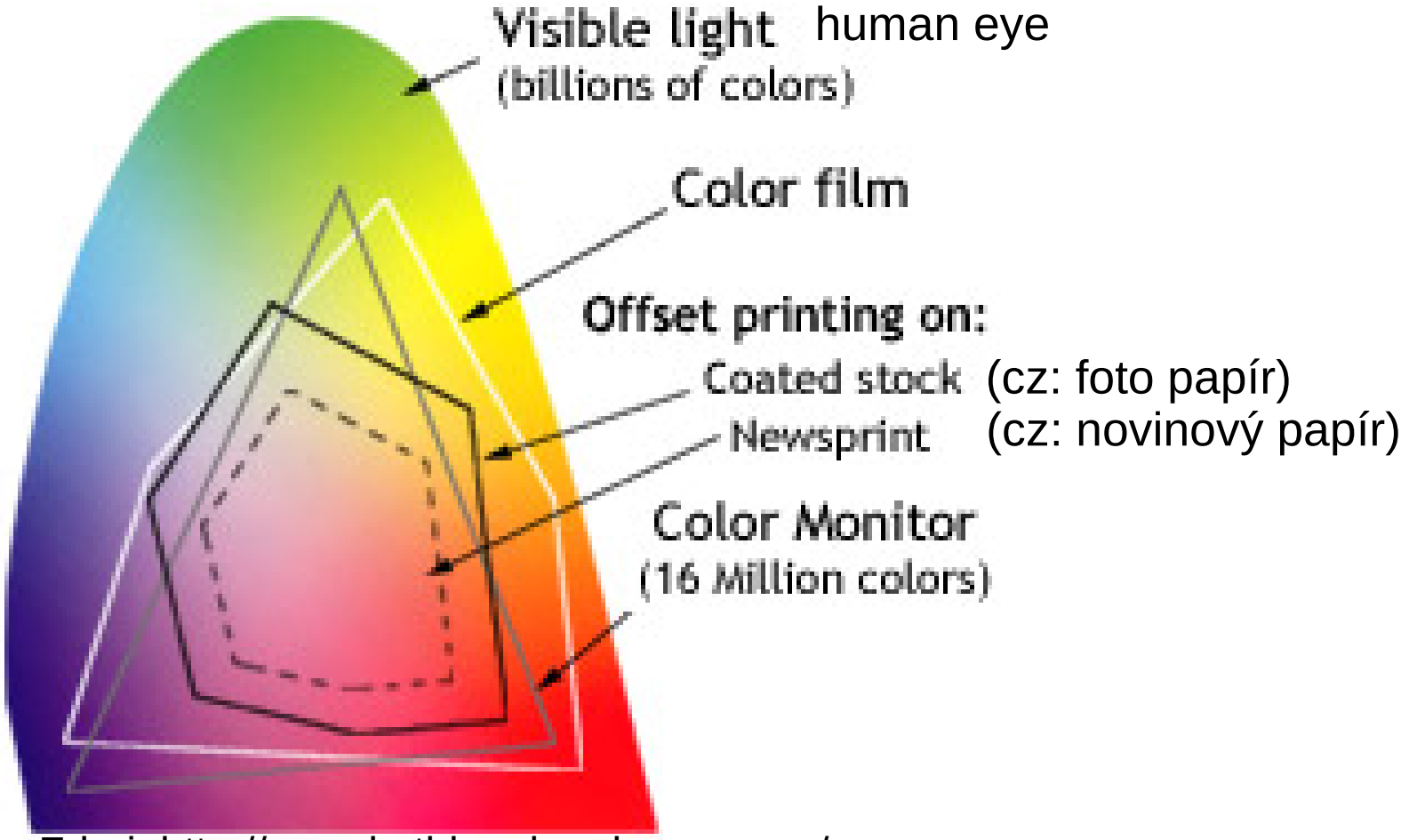
RGB



CMYK

- *By adding blacK, we obtain CMYK:*
  K = min(C,M,Y); C=C-K; M=M-K; Y=Y-K;

But will we get the same gamut?



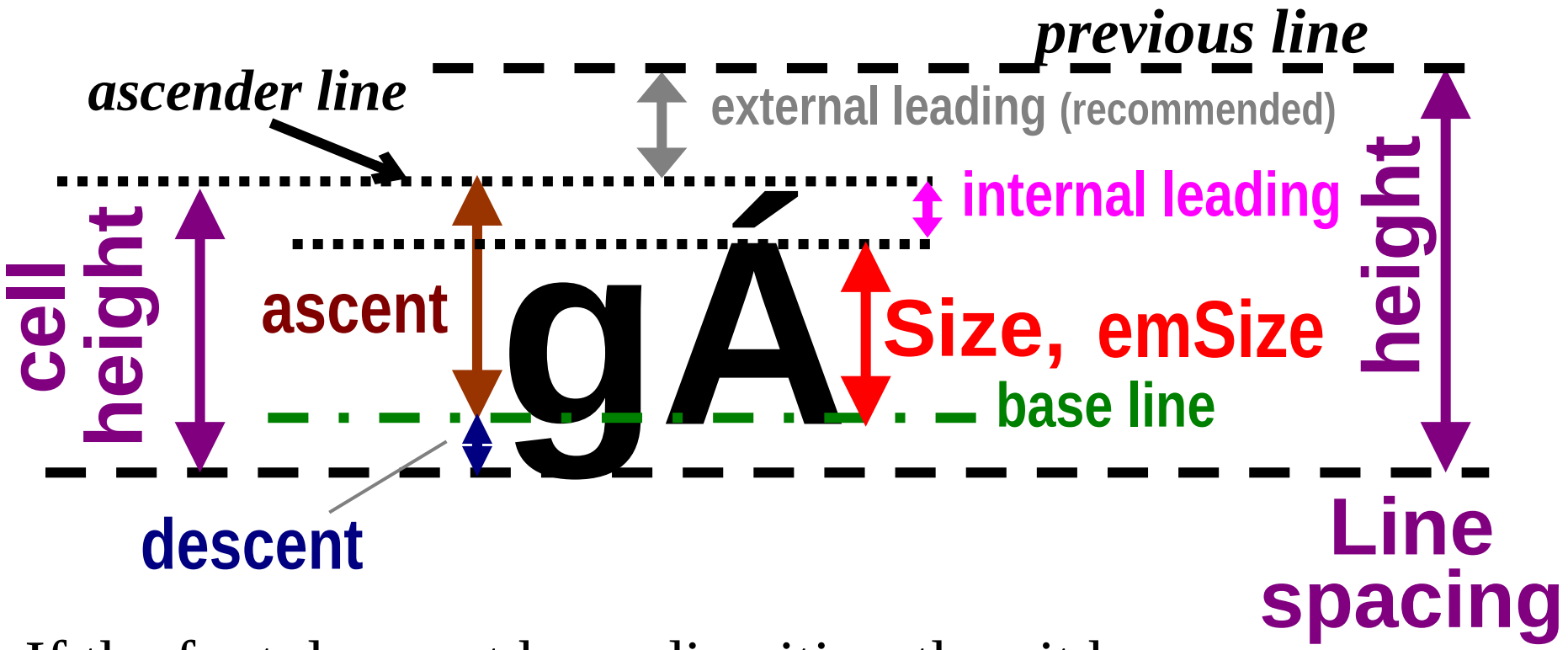
Zdroj: http://www.kathleenhmahoney.com/

gamut = the subset of colors which can be accurately represented in a given circumstance

Text to Pixels ?

LCD has no character generator
and MicroZed does not contains any graphic card!

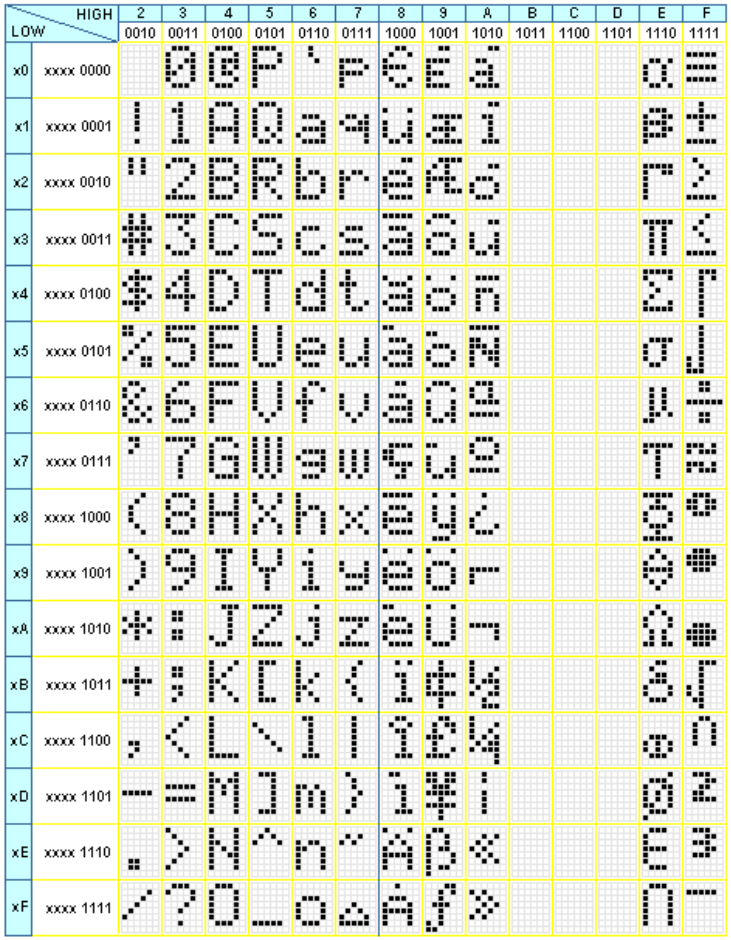
The Font Size



- If the font does not have diacritics, then it has a zero internal leading and its ascent matches emSize
- More at Scribus typography in Czech

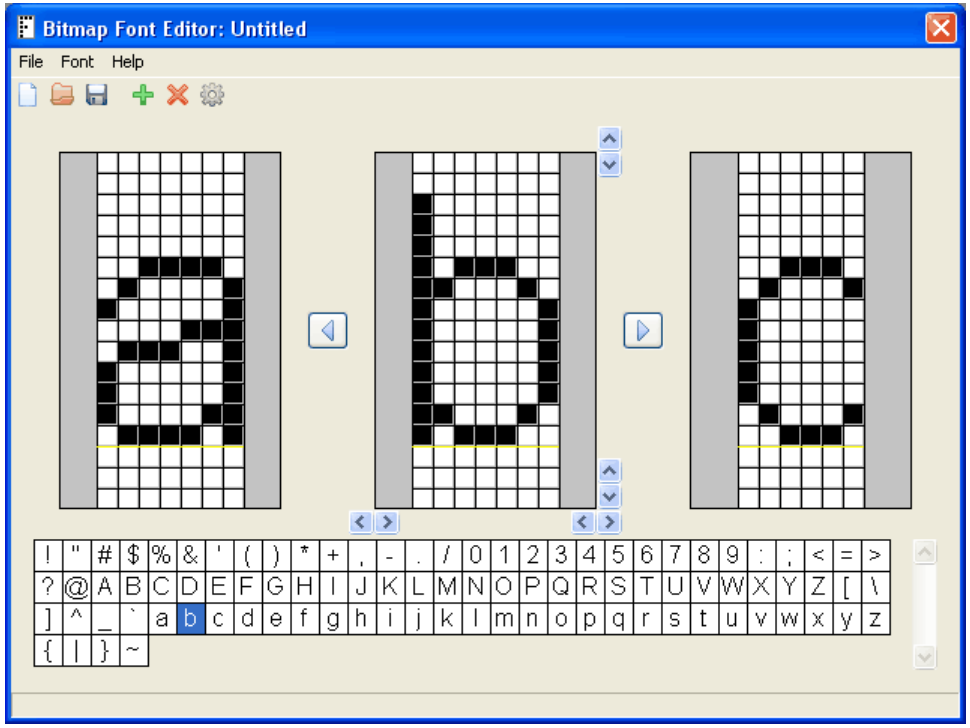
Bitmap Fonts

Bitmaps font are **faster and easier to use** and therefore suitable for low cost computer systems.



http://growboxbox.org/doku.php/lcd

http://mobilefonts.sourceforge.net/



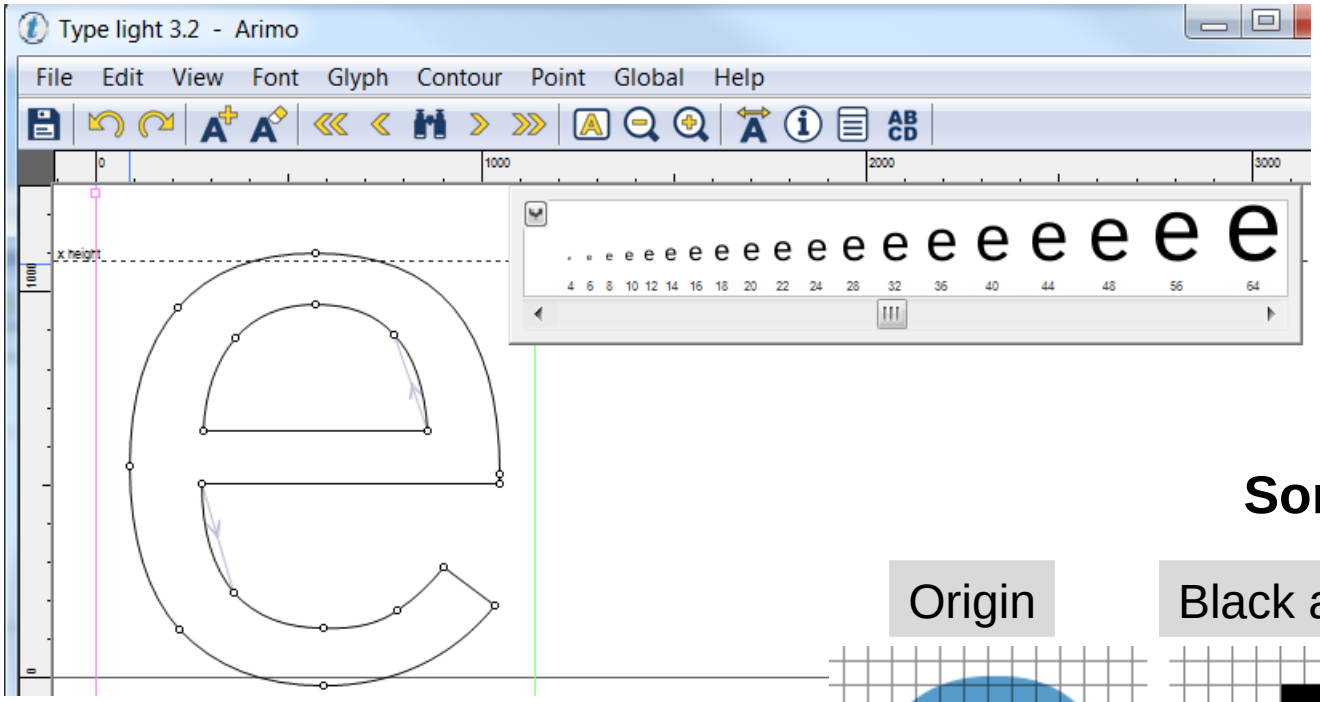
Note: In 1968 ,the first bitmap font was created by German inventor Rudolf Hell for his Digiset typesetting machine .

The Scalable Font Wars

- Apple and Microsoft developed (1980) the **TrueType** methodology that is a system of scalable outline fonts, and can draw characters at low resolution.
- Adobe **PostScript** (1984) is another method of describing an image in terms of ***mathematical constructs*** (Bézier curves), so it is used not only to describe the individual characters of a font but also to describe entire illustrations and whole pages of text.
- **Open Type** (1996) digital font format was developed jointly by Apple and Microsoft to put an end to the PostScript/TrueType war. Like TrueType, a single file contains all the outline and bitmap data for an OpenType font, but it also contains either PostScript data or additional TrueType data within the font, which in the PostScript case, makes the font truly scalable and exacting.

Source: wordpress.com

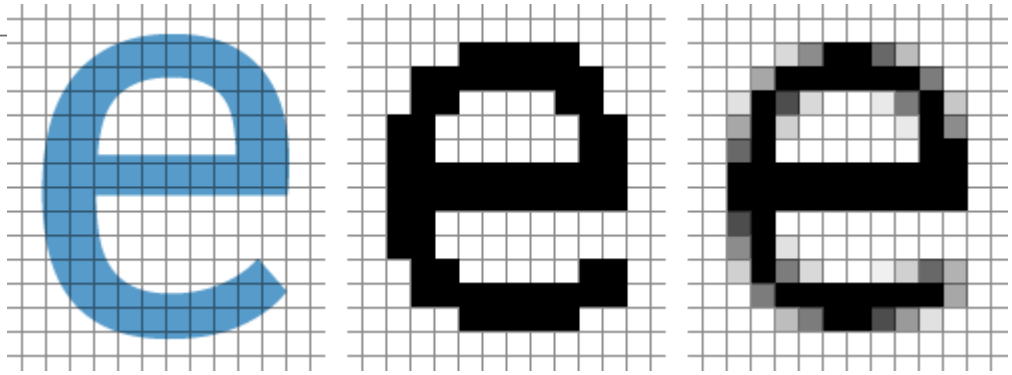
Rendering of Fonts



Upper picture:
program
Type Light V3.2

Some rasterizations

Origin | Black and white | Grayscale



Any scalable font must be finally rendered (rasterized) to pixels of display !

Image source: https://www.smashingmagazine.com/

Black and White Rastered Fonts

File: font_rom8x16.c / .h

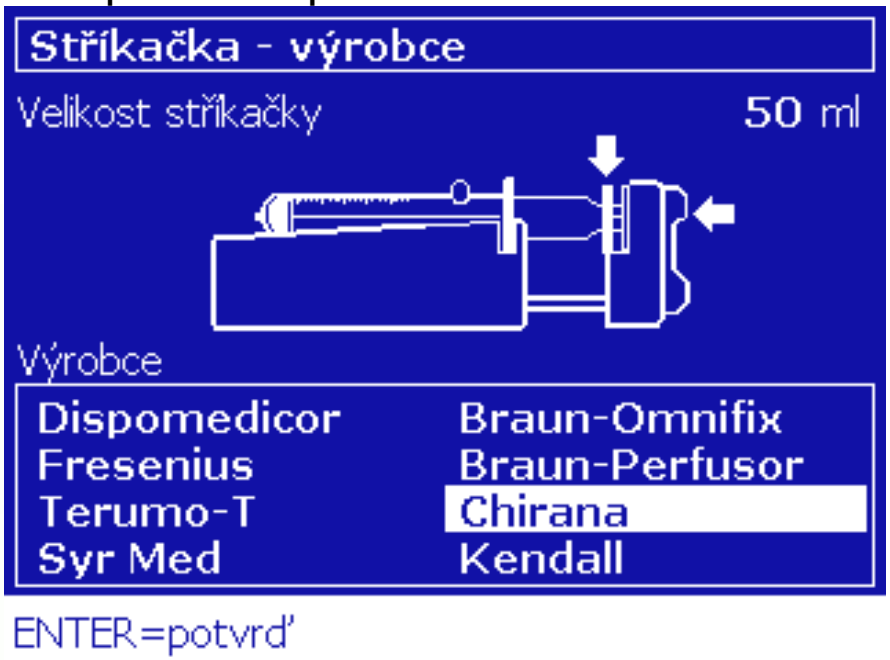
```
static font_bits_t rom8x16_bits[] =
{ ...
/* Character e (0x65):   ht=16, width=8
   +--------+
   |        |
   |        |
   |        |
   |        |
   |        |
   | *****  |
   |**   **  |
   |**   **  |
   |*******  |
   |**       |
   |**   **  |
   | *****  |
   |        |
   |        |
   |        |
   |        |
   +--------+ */
0x0000,0x0000,0x0000,0x0000,
0x0000,
0x7c00,
0xc600,
0xc600,
0xfe00,
0xc000,
0xc600,
0x7c00,
0x0000,0x0000,0x0000,0x0000,
```

		x-column								
		+0	+1	+2	+3	+4	+5	+6	+7	y-row
0x00	0000 0000...									+0
0x00	0000 0000...									+1
0x00	0000 0000...									+2
0x00	0000 0000...									+3
0x00	0000 0000...									+4
0x7c	0111 1100...		■	■	■	■	■			+5
0xc6	1100 0110...	■	■				■	■		+6
0xc6	1100 0110...	■	■				■	■		+7
0xfe	1111 1110...	■	■	■	■	■	■			+8
0xc0	1100 0000...	■	■							+9
0xc6	1100 0110...	■	■				■	■		+10
0x7c	0111 1100...		■	■	■	■	■			+11

Raster Fonts for Use in Semestral Work

The raster fonts bitmaps and widths are provided for semestral work. The fonts are defined in the format used by Microwindows/Nano-X library by Greg Haerr
https://github.com/ghaerr/microwindows
The simplified structure to describe font is provided in file
https://gitlab.fel.cvut.cz/b35apo/mzapo_template/-/blob/master/font_types.h
The example of the device where PiKRON has used the Microwindows, RTEMS and this font definitions from which provided example Is simplified

TECHNIC-I



https://devel.rtems.org/wiki/TBR/UserApp/AMV_Technic_I

See: https://gitlab.fel.cvut.cz/b35apo/mzapo_template/

Structure font_descriptor_t

```
typedef struct {
  char          *name;       /* font name*/
  int           maxwidth;    /* max width in pixels*/
  unsigned int  height;      /* height in pixels*/
  int           ascent;      /* ascent (baseline) height*/
  int           firstchar;   /* first character in bitmap*/
  int           size;        /* font size in characters*/
  const font_bits_t *bits;   /* 16-bit right-padded bitmap data*/
  const uint32_t *offset;    /* offsets into bitmap data*/
  const unsigned char *width; /* character widths or 0 if fixed*/
  int           defaultchar; /* default char (not glyph index)*/
  int32_t       bits_size;   /* # words of MWIMAGEBITS bits*/
} font_descriptor_t;
```


See: https://gitlab.fel.cvut.cz/b35apo/mzapo_template/

*More about MZ_APO development kit

MZ_APO Education Kit Sources

- The MZ_APO education kit has been designed by

  Petr Porazil (porazil@pikron.com) and

  Pavel Pisa (pisa@cmp.felk.cvut.cz, ppisa@pikron.com) on request of Department of Control Engineering (K13135) now gone Industrial Informatics Group head to propagate his group
- The complete design files for base board and mechanics are available from the repository

  https://gitlab.com/pikron/projects/mz_apo/microzed_apo
- Petr Porazil's electronic design automation PEDA has been used for electronics design

  https://sourceforge.net/projects/peda/
- FPGA design by Pavel Pisa and Martin Jeřábek is available from CTU FEE CAN bus project repositories

  https://gitlab.fel.cvut.cz/canbus/zynq/zynq-can-sja1000-top

  the actual design is available on microzed_apo_psr branch

MZ_APO Education Kit Sources cont

- The Linux kernel 4.19 with fully preemptive patches is actually running on the boards. The branch linux-4.19.y-pi with integrated patches is available from repository

  https://github.com/ppisa/linux-kernel
- The build is available

  https://github.com/ppisa/zynq-rt-utils-and-builds

- More MZ_APO documentation prepared for B35 Computer Architectures is available

  https://cw.fel.cvut.cz/wiki/courses/b35apo/documentation/mz_apo/start

- MicroZed SBC documentation

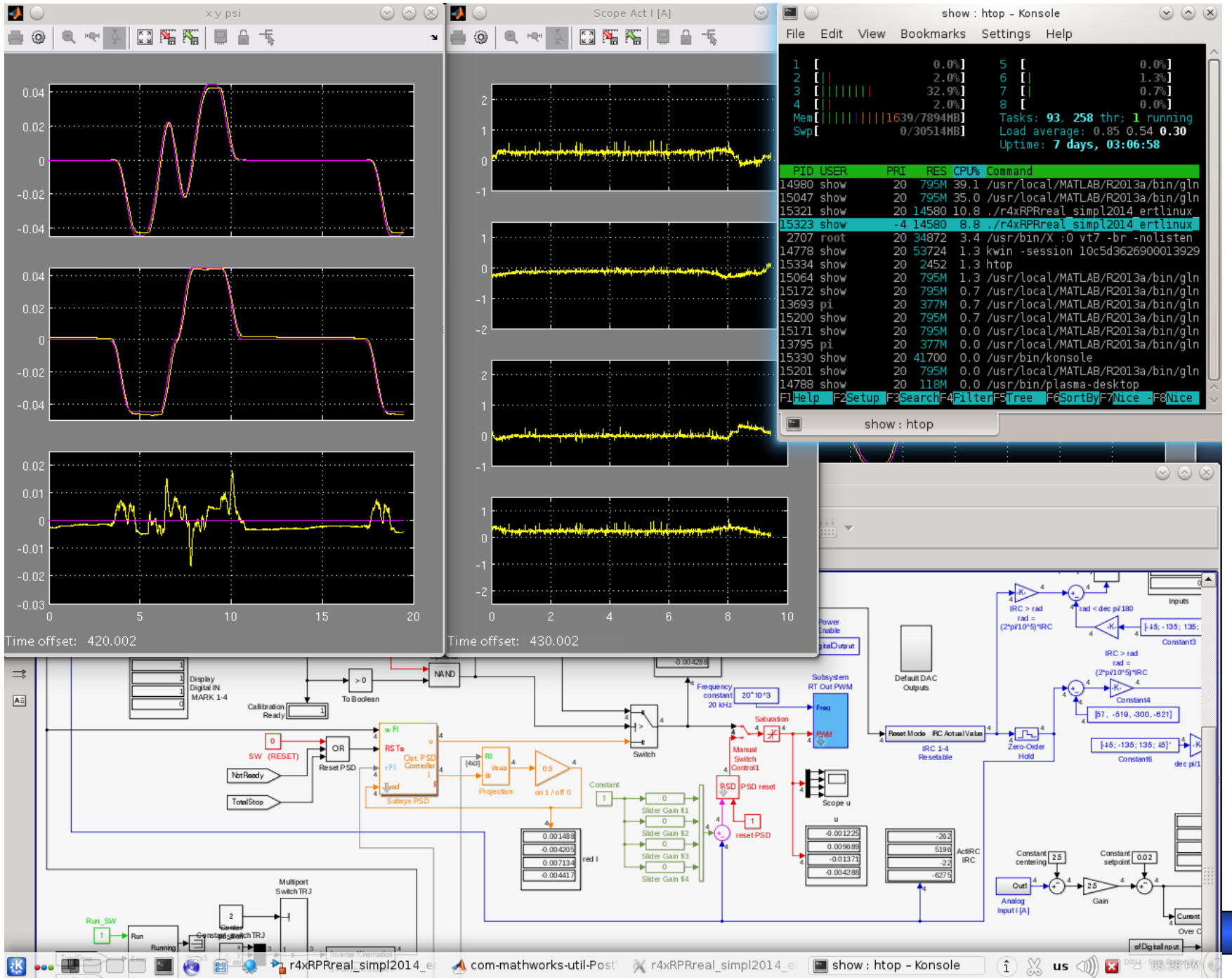
  http://zedboard.org/product/microzed

MZ_APO Education Kit Support Files and Projects

- Matlab/Simulink support
  https://github.com/aa4cc/ert_linux
- The Simulink and other projects for the boards
  https://github.com/aa4cc/zynq-rt-control
- mz_apo-2dc/zynq_dc_motor_control.slx
  support library designed for DC motor kits
  support library contributions Martin Gurtner, Lukáš Černý
  electronic design Tomas Nepivoda, Martin Szabó
  Theses: DC Motor Control Peripheral Module for Zynq Platform
  mechanical design Oxana Kovbasjuková
- simulink/mz_apo-2dc/mz_apo_dc_motor_pid_control_with_knob.slx
  control position of DC motor servo by knobs
  simulink/mz_apo-2dc/steer_by_wire.slx
  control one motor to follow other one used as IRC encoder

MZ_APO Education Kit Support Files and Projects

- The Simulink and other projects for the boards - more
  https://github.com/aa4cc/zynq-rt-control
- PMSM control project using Matlab/Simulink
  simulink/mz_apo-3pmdrv/zynq_pmsm_motor_control.slx
  The hardware design Petr Porazil (porazil@pikron.com)
  Design files hw/prj/fel/motor-driver-1.tdb from the repository
  https://gitlab.com/pikron/projects/mz_apo/microzed_apo
- FPGA design Pavel Pisa, Marek Peca, Martin Prudek
  Theses: Brushless motor control with Raspberry Pi board and Linux
- The more description of Matlab/Simulink code generation for Linux
  http://lintarget.sourceforge.net/
- Alternative/Matlab/Simulink independent robotic motion control project
  https://gitlab.com/pikron/projects/pxmc-linux
  It based on PXMC library http://pxmc.org/


x y psi
Time offset: 420.002
Scope Act I [A]
Time offset: 430.002
show : htop - Konsole
File Edit View Bookmarks Settings Help
PID USER PRI RES CPU% Command
14980 show 20 795M 39.1 /usr/local/MATLAB/R2013a/bin/gln
15047 show 20 795M 35.0 /usr/local/MATLAB/R2013a/bin/gln
15321 show 20 14580 10.8 ./r4xRPRreal_simpl2014_ertlinux
15323 show -4 14580 8.8 ./r4xRPRreal_simpl2014_ertlinux
2707 root 20 34872 3.4 /usr/bin/X :0 vt7 -br -nolisten
14778 show 20 53724 1.3 kwin -session 10c5d3626900013929
15334 show 20 2452 1.3 htop
15064 show 20 795M 1.3 /usr/local/MATLAB/R2013a/bin/gln
15172 show 20 795M 0.7 /usr/local/MATLAB/R2013a/bin/gln
13693 pi 20 377M 0.7 /usr/local/MATLAB/R2013a/bin/gln
15200 show 20 795M 0.7 /usr/local/MATLAB/R2013a/bin/gln
15171 show 20 795M 0.0 /usr/local/MATLAB/R2013a/bin/gln
13795 pi 20 377M 0.0 /usr/local/MATLAB/R2013a/bin/gln
15330 show 20 41700 0.0 /usr/bin/konsole
15201 show 20 795M 0.0 /usr/local/MATLAB/R2013a/bin/gln
14788 show 20 118M 0.0 /usr/bin/plasma-desktop
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice
show : htop
Display Digital IN MARK 1-4
Calibration Ready
To Boolean
NAND
SW (RESET)
OR
Reset PSD
NotReady
TotalStop
Subsys PSD
Projection
Switch
Frequency constant 20 kHz
Saturation
Manual Switch Control1
PSD reset
Slider Gain 1
Slider Gain 2
Slider Gain 3
Slider Gain 4
Subsystem RT Out PWM
Default DAC Outputs
Scope u
Zero-Order Hold
IRC 1-4 Resetable
Multiport Switch TRJ
Run_SW
Run
Running
Analog Input I [A]
Gain
Constant centering
Constant setpoint
Inputs

Presentations of Projects Done on MZ_APO System

- LinuxDays 2015, Linux, RPi and other HW for DC and Brushless/PMSM Motor Control, Slides, Video in Czech
- InstallFest 2017, GNU/Linux and FPGA in Real-time Control Applications, Slides, Video in Czech
- LinuxDays 2017, GNU/Linux, CAN and CANopen in Real-time Control Applications, Slides, Video in Czech
- InstallFest 2020, Embedded Linux, FPGA and Motion Control Hands-On https://pretalx.installfest.cz/installfest-2020/talk/HSNJCM/
- Dion Beqiri thesis
  Open Rapid Control Prototyping, Education and Design Tools
  pysimCoder (https://github.com/robertobucher/pysimCoder) includes examples and blocks to control DC on MZ_APO
- Articles about GNU/Linux use for RT control on Root.cz in Czech
  - GNU/Linux pro řízení a rychlost jeho odezvy - link
  - Linux pro řízení: minimalistické řešení řízení stejnosměrného mo toru - link