MicroZed APO

This page contains a detailed description of hardware and peripherals ready for exercises and semester assignments. For instructions on how to connect to the product, see the MicroZed APO Help page .

Description of the hardware of the used MicroZed processor board

- MICROZED EVALUATION KIT
- ADSAES-Z7MB-7Z010-G
- Xylinx Zynq 7Z010

- Base Chip: Xilinx Zynq-7000 All Programmable SoC
- Type: Z-7010, part XC7Z010
- CPU: Dual ARM Cortex ™ -A9 MPCore ™ @ 866 (NEON ™ & Single/Double Precision Floating Point)
- 2x L1 32 kB data + 32 kB instruction, L2 512 KB
- FPGA: 28K Logic Cells (~ 430K ASIC logic gates, 35 kbit)
- Computing units in FPGAs: 100 GMACs
- FPGA memory: 240 KB
- Memory on MicroZed board: 1GB
- Operating system: GNU/Linux
- GNU LIBC (libc6) 2.19-18 + deb8u7
    - Linux kernel 4.9.9-rt6-00002-ge6c7d1c
    - Distribution: Debian Jessie
- More information at http://microzed.org/product/microzed

Interfaces accessible directly on MicroZed board

- 1G ETHERNET,
- USB Host, A connector
- serial port UART1 via converter to USB, USB micro-B
- micro SD card
- on the board is Flash, one user LED, user

button and reset button

Description of the MZ_APO development kit interface

The following peripherals are served in the logical design on the training module

- small parallel LCD display, so far access only via command and data register
    - in FPGA it is possible to implement automaton for display from framebuffer
- 32 LEDs for displaying 32-bit words (connection to FPGA via SPI)
- two RGB LEDs (SPI connection)

- three rotary control inputs () with output phase-shifted signals and a push-on contact (connection via SPI)
- four outputs for modeling servos, signal and power supply
- audio input to the converter in Zynqu, which has an interface to the ARM part, but it can be connected via bus to FPGA Programmable Logic. The board is equipped with a small microphone, when connecting an external to 3.5 mm red JACK, it switches to an external input, the jumper can be set from amplified to the classic 1V line standard
- audio output, only PWM directly from FPGA PL, but with appropriate modulation, the melody can be played. There is a small speaker on the board with the possibility of switching to headphones after connecting to the JACK (mono output only)
- 2x PMOD connector, it is a de-facto standard for connecting slow (max. Tens) peripherals to FPGA, each has 8 FPGA PL signals, + 3.3V power supply plus pins for + 5V power supply, with + 5V can be disconnected by jumper. In the event of a short circuit to 3.3 logic signals, there is a risk of destruction. The PMOD connector standard, which uses a female on the FPGA side, is not directly used. The reason is the possibility of easy connection to flat cable. With a short connector it is possible to arrange a reduction directly to interfaces compatible with available PMOD peripherals.
- 1x 40 pin connector with 36 FPGA 3.3V signals. It corresponds to the signal distribution on the Altera DE2 boards. Some of the signals are routed with respect to usability for fast LVDS connections. The other half is shared with the PMOD outputs. The output is + 3.3V and + 5V connected by a number.
- 2x interface for connecting 10-bit parallel cameras.
- two channels for CAN bus connection, drivers up to 5 MBd connection via FPGA PL either to integrated controllers or to controllers in FPGA, these could be implemented in a project with CAN-FD support
- power supply for 5.5 mm JACK from 12 to 24 VDC.
- the USB B connector is connected to an FTDI chip that provides access to the serial console routed to the ZARTQ UART0 circuit. The actual signals of the Zynq circuit and the power supply of the kit are galvanically separated from the connector and the USB signals. With the choice of console settings on the MIO10 and MIO11 pins, a custom version of the U-Boot bootloader has been modified and compiled . The kernel of the Linux operating system was compiled to measure.
- when using UART0 via FTDI, it is possible to reset the board with a break signal applied for more than 1 second.
- 2-pin connector for external reset, such as a relay when used for remote access and application debugging
- hardware design Ing. Petr Porazil at PiKRON sro

The complete mechanical design and electronics design of the kit is available in the GIT repository https://gitlab.com/pikron/projects/mz_apo/microzed_apo

To view and edit mechanical design, you need to install FreeCAD https://freecadweb.org/

For display and editing of diagrams and printed circuit, then the design system Ing. Peter Porazil - PEDA https://sourceforge.net/projects/peda/

Table of base addresses of individual peripherals

Physical address	Range	Symbolic designation	Description
0x43c40000	0x4000	SPILED_REG_BASE_PHYS	Block of peripherals, rotary selectors, matrix keyboards and RGB LEDs
0x43c00000	0x4000	PARLCD_REG_BASE_PHYS	Parallel LCD display
0x43c50000	0x4000	SERVOPS2_REG_BASE_PHYS	Control of up to four model servos, alternatively PS2
0x43c60000	0x4000	AUDIOPWM_REG_BASE_PHYS	Simple audio output (so far only PWM)
0x43c20000	0x4000	DCSPDRV_REG_BASE_PHYS_0	Peripherals for driving the first DC motor
0x43c30000	0x4000	DCSPDRV_REG_BASE_PHYS_1	Peripherals for controlling a second DC motor

Block of rotary selectors, matrix keyboards and RGB LEDs

The register set starts at the physical address SPILED_REG_BASE_PHYS

Register offset	Symbolic designation	Bits	Description
0x004	SPILED_REG_LED_LINE_o	31 .. 0	A row of 32 yellow LEDs mapped directly to memory
0x010	SPILED_REG_LED_RGB1_o	23 .. 0	Writing RGB values to PWM registers for RGB LEDs 1
		23 .. 16	Red component R
		15 .. 8	Green component G
		7 .. 0	Blue component B
0x014	SPILED_REG_LED_RGB2_o	23 .. 0	Write RGB values to PWM registers for RGB LED 2
		23 .. 16	Red component R
		15 .. 8	Green component G
		7 .. 0	Blue component B
0x018	SPILED_REG_LED_KBDWR_DIRECT_o	31 .. 0	Direct write to LED and keyboard output/scan
		2 .. 0	Direct output (or) to R, G and B RGB LEDs 1
		5 .. 3	Direct output (or) to R, G and B RGB LEDs 2
		6	Separate status LED
		7	Separate status LED
		11 .. 8	Keyboard Series Selection
0x020	SPILED_REG_KBDRD_KNOBS_DIRECT_o	31 .. 0	Keyboard and rotary selector feedback
		3 .. 0	Keyboard feedback
		16	Unfiltered state of channel A of blue

Register offset	Symbolic designation	Bits	Description
			selector B
		17	Blue B selector channel B unfiltered state
		18	Blue selector button B status not filtered
		19 Dec	Green state of G selector channel A unfiltered
		20 May	Unfiltered state of channel B of green selector G
		21	Green state of G selector button unfiltered
		22nd	Filter state A of the red selector R is unfiltered
		23	Unfiltered state of channel B of red selector R
		24	Unfiltered status of the red selector button R
0x024	SPILED_REG_KNOBS_8BIT_o	31 .. 0	Filtered selector values as 8 bit numbers
		7 .. 0	Relative rotation of the blue selector B
		15 .. 8	Relative rotation of the green G dial
		23 .. 16	Relative rotation of the red dial R
		24	Filtered Blue Dial B Value
		25	Filtered value of the green selector button G
		26	Filtered value of the red selector R

Additional information for those interested in custom peripheral design for integrated gate array processor

The peripheral implementation is located in /system/ip/spi_leds_and_enc_1.0/hdl . Because LEDs and rotary selectors are slow peripherals and the number of pins usable for more interesting and faster interfaces such as cameras is large, GPIO expanders connected in series (SPI bus) are used to transfer input and output signals to and from slow peripherals. A sequence of 48 bits is transmitted in both directions. The transfer of the old instance spi_leds_and_enc_v1_0_spi_fsm_inst components spi_leds_and_enc_v1_0_spi_fsm . The component spi_leds_and_enc_v1_0_S00_AXI then takes care of the implementation of the set of registers connected to the CPU by the AXI bus . The referenced section at the end of the source code clearly shows how the individual logic signals and their groups are mapped to bits and bit fields in the individual registers of the AXI peripherals.

Parallel LCD display

The register set starts at the physical address PARLCD_REG_BASE_PHYS

Register offset	Symbolic designation	Bits	Description
0x000	PARLCD_REG_CR_o	31 .. 0	Peripheral control register
		1	Generating a reset signal for the display
0x008	PARLCD_REG_CMD_o	7 .. 0	Generate control cycle/write command to LCD controller
0x00C	PARLCD_REG_DATA_o	15 .. 0	Generate 16-bit data cycle/command write to LCD controller
0x00C	PARLCD_REG_DATA	31 .. 0	Generating two data cycles for the controller (15 .. 0) and

Register offset	Symbolic designation	Bits	Description
	_o	(31 .. 16)	

The peripheral implementation is located in the /system/ip/display_16bit_cmd_data_bus_1.0/hdl directory .

Output to model servos or for PS2 keyboard/mouse

The register set starts at the physical address SERVOPS2_REG_BASE_PHYS

Register offset	Symbolic designation	Bits	Description
0x000	SERVOPS2_REG_CR_o	31 .. 0	Peripheral control register and direct control
		0	Setting the idle value (H/L) for LED/SERVO1
		1	Idle value setting (H/L) for LED/SERVO2
		2	Setting the idle value (H/L) for LED/SERVO3 overlaps with PS2 Clock
		3	Idle value setting (H/L) for LED/SERVO4 overlaps with PS2 Data, the third state by default
		8	signal direction control SERVO4/PS DATA, 0 .. in, 1 .. out
0x00C	SERVOPS2_REG_PWMPER_o	23 .. 0	PWM cycle period in step 10 ns
0x010	SERVOPS2_REG_PWM1_o	23 .. 0	Filling the PWM signal SERVO1 in step 10
0x014	SERVOPS2_REG_PWM2_o	23 .. 0	Filling the PWM signal SERVO2 in step 10
0x018	SERVOPS2_REG_PWM3_o	23 .. 0	Filling the PWM signal SERVO3 in step 10
0x01C	SERVOPS2_REG_PWM4_o	23 .. 0	Filling the PWM signal SERVO4 in step 10

The peripheral implementation is located in the /system/ip/servo_led_ps2_1.0/hdl directory .

PWM audio output

The register set starts at the physical address AUDIOPWM_REG_BASE_PHYS

Register offset	Symbolic designation	Bits	Description
0x000	AUDIOPWM_REG_CR_o	31 .. 0	Peripheral control register
0x008	AUDIOPWM_REG_PWMPER_o	23 .. 0	PWM cycle period in step 10 ns
0x00C	AUDIOPWM_REG_PWM_o	23 .. 0	Filling the PWM signal in step 10

The peripheral implementation is located in the /system/ip/audio_single_pwm_1.0/hdl directory .

Peripherals for DC motor control

This peripheral was originally designed for the subject Programming Real-Time Systems (B3M35PSR, B4B35PSR). It is optionally included in the FPGA design for the subject Computer Architecture.

The peripheral design is included twice. One for the power stage and position reading to the PMOD1 connector, for which the registers are mapped from DCSPDRV_REG_BASE_PHYS_0, and the second for the motor connected through the PMOD2 connector, the DCSPDRV_REG_BASE_PHYS_1 address is based.

Register offset	Symbolic designation	Bits	Description
0x0000	DCSPDRV_REG_CR_o	31 .. 0	Peripheral control register

Register offset	Symbolic designation	Bits	Description
		4	Direct PWMA output control
		5	Direct PWMB output control
		6	Enable PWM output by fill and period
		8	Reset reading position in
0x0004	DCSPDRV_REG_SR_o	31 .. 0	Peripheral status register
		8	Logic level on the IRCA pin
		9	Logic level on the IRCB pin
		10	Logic level on the IRQ pin
0x0008	DCSPDRV_REG_PERIOD_o	31 .. 0	PWM Period Setting After 10 ns (100)
		29 .. 0	PWM length setting (suitable 50000, 20 kHz)
0x000C	DCSPDRV_REG_DUTY_o	31 .. 0	Regist to set the required level of PWM performance
		29 .. 0	Mask of implemented bits, setting of pulse length/filling in the period
		30	Voltage direction applied to the motor, 10 ns unit
		31	Voltage selection for the opposite direction of rotation
0x0010	DCSPDRV_REG__o	31 .. 0	Calculated number of increments/position/motor rotation

The peripheral implementation can be found in /system/ip/dcsimpledrv_1.0/hdl .

Other peripherals

User button on the processor module

The button is connected to the PS_MIO51_501 SoC Zynq pin. On the Linux kernel, offset 906 is set for MIO.

To access the user button, you need to access the GPIO pin 957 (= 906 + 51).

```
echo 957>/sys/class/gpio/export
cat /sys/class/gpio/gpio957/value
```

User LED on the processor module

The LED is connected to pin PS_MIO47_501 SoC Zynq. On the Linux kernel, offset 906 is set for MIO.

It is therefore necessary to access GPIO pin 953 to access the user LED.

```
echo 953> /sys/class/gpio/export
echo out>/sys/class/gpio/gpio953/direction
echo 1>/sys/class/gpio/gpio953/value
```

Links

- GNU/Linux and FPGA in Real-time Control Applications , Installfest 2017 lecture . Presentation in PDF format .
- Use of MZ_APO boards for distributed motor control using CAN industrial bus . Video from the Linux Days 2017 conference . Presentation in PDF format
- Programming notes SoC Zynq .
- GIT repository with Linux kernel configuration and U-boot loader with modifications for MZ_APO kits
- Martin Jeřábek - diploma thesis Open-source and Open-hardware CAN FD Protocol Support