μG4-150 embedded platform

for wire-speed network packet processing

FIT BUT Technical report

Pavol Korček

Technical report no. FIT-TR-2013-03
Faculty of Information Technology, Brno University of Technology

Last change: 18.7.2013

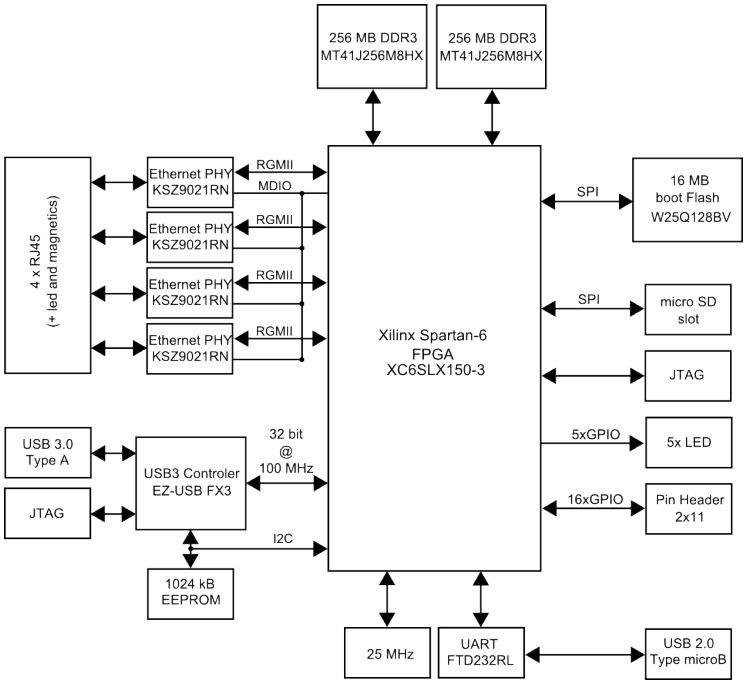
μG4-150 embedded platform for wire-speed network packet processing

Pavol Korček

Faculty of Information Technology
Brno University of Technology
Božetěchova 1/2, 612 66 Brno
ikorcek@fit.vutbr.cz

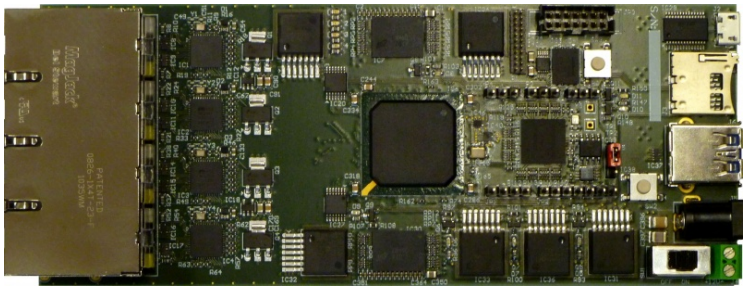
1 Introduction

Embedded development platform μG4-150 (aka micro for gigabit) is an FPGA-based board developed for wire-speed acceleration and/or processing of network packets up to 1 Gbps. It consists of a powerful Xilinx Spartan-6 FGPA of the biggest size and speedgrade, so it can be also used as an experimental platform.

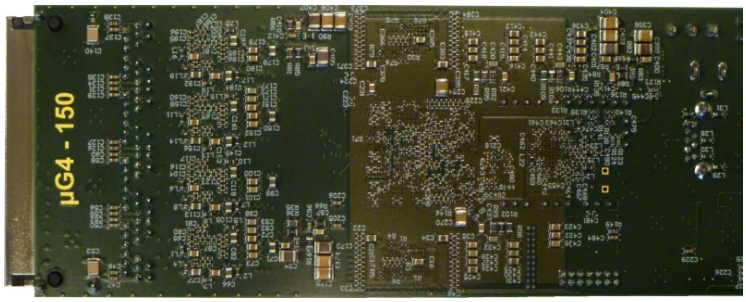


Obrázek 1. Overall concept of the embedded development platform μG4-150.

Platform uses four ports for network connectivity, DDR-type memories as a fast local data storage or a system RAM for softprocessor, a 16 MB boot flash for FPGA device which can be also used as a slow persistent local data storage, a slot for micro-SD memory cards, a powerful USB 3.0 interface for high performance transfers to a computer or an attached storage device (USB flash disc, etc.), a serial (UART) interface, a standard JTAG connector used mainly for easy debugging. It also consist of signalization LEDs and GPIO to interface other components not included on the platform. All these components are connected to the FPGA. Finally, it also consists of other components not specified in too much detail within this document (i.e. power components, crystals, passive components, etc.) but very important for the proper function. Overall concept of the embedded development platform μG4-150 is shown on the Fig. 1. Whole platform is shown on the Fig. 2 (front side) and on the Fig. 3 (back side).



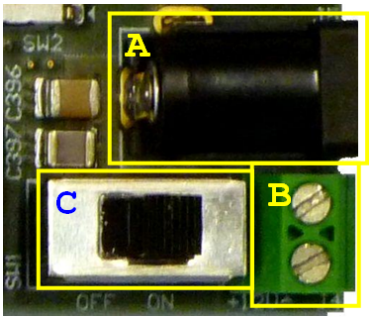
Obrázek 2. Embedded development platform μG4-150 (front side).



Obrázek 3. Embedded development platform μG4-150 (back side).

2 Power supply

The μG4-150 platform is expected to be powered by 12 V with maximum current of 1 A. Two connectors can be used for attaching power supply: the power jack and the standard ribbon cable connector (both are shown on the Fig. 4). There is no possibility to use both power supply connectors simultaneously as this situation may lead to the electrical short! Separate power switch with two positions (also shown on the Fig. 4) can be used for switching on and off.



Obrázek 4. A - Jack connector, B - Ribbon connector, C - Power switch.

There are several power supply components (suppliers) on the platform: one 5V@5A, two 3.3V@1A, one 2.5@3A, one 1.5V@3A, one 1.2V@4A and one 0.75V@3A. 5 V and both 3.3 V suppliers are fed from 12 V input. 2.5 V, 1.5 V and 1.2 V suppliers are then fed from 5 V supplier. The last supplier for deriving 0.75 V is fed from 1.5 V supplier as shown in Table 1.

Supplier for [V]	Max. current [A]	Fed from	Used for
5	5	direct 12 V input	Deriving other voltage levels, USB 3.0
3.3	1	direct 12 V input	Ethernet
3.3	1	direct 12 V input	FPGA, boot flash, SD card
2.5	3	5 V supply	USB 3.0 , FPGA, debugging LEDs
1.5	3	5 V supply	DDR memory , FPGA
1.2	4	5 V supply	USB 3.0 , FPGA, Ethernet
0.75	3	1.5 V supply	DDR memory

Tabulka 1. Different power supply components on the platform.

3 Spartan-6 FPGA

The FPGA is a Xilinx Spartan-6 of the biggest size and speedgrade from the LX family (XC6SLX150-3). This one has 147,443 logic cells, 23,038 slices (each Spartan-6 FPGA slice contains four LUTs and eight flip-flops), 184,304 Flip-Flops, can be configured for up to 1,355 kb of distributed RAM, has 180 DSP48A1 blocks and 268 of 16 kb BlockRAMs (max. 4 824 kb of memory). It also consists of 6 Clock Management Tiles (CMTs) each with two DCMs and one PLL, up to four Memory Controller Blocks (MCBs) and up to 576 user I/Os. As the μG4-150 platform is equipped with FPGA in the FG484C package, so that there are two MCBs and only 338 user I/Os available. Some other details can be found in the Spartan-6 product specification [10] or Spartan-6 FPGA Packaging and Pinouts product specification [13].

3.1 Clock distribution

There is only one clock source for the FPGA on the μG4-150 platform. It is a clock oscillator with a frequency of 25.0 MHz connected to the Y11 pin. More details about this component can be found in its corresponding datasheet [**?**]. For proper function of FPGA firmware Xilinx unified constraints file (UCF) should contain (clock input is denoted as "CLK_IN" and assuming IOSTANDARD is implicitly "LVCMOS25"):

```
NET "CLK_IN" LOC = Y11;
NET "CLK_IN" TNM_NET = "CLK_IN1";
TIMESPEC "TS_CLK_IN1" = PERIOD "CLK_IN1" 40 ns HIGH 50
 INPUT_JITTER 300.0ps;
```

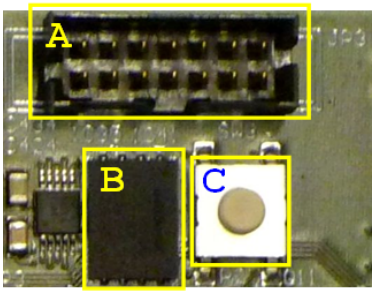
Other FPGA clock frequencies can be derived internally from DCMs or PLLs components. If it is desired, other platform components have their own clock sources. Such an example is the USB 3.0 processor with 19.2 MHz crystal or all Ethernet transceivers each with separate 25.0 MHz crystal. These clock sources are not available for use in the FPGA.

3.2 Booting subsystem

FPGA can be configured via JTAG interface shown on the Fig. 5.

The Spartan-6 FPGA device is the only device in this JTAG chain. Platform uses high-capacity on-board 128 Mb serial flash memory from Winbond (W25Q128BV) [1]. This memory can be accessed only via FPGA pins and its interconnection is shown on this UCF example (assuming IOSTANDARD is implicitly "LVCMOS25"):

```
NET "CFG_D[0]"   LOC = AB20;
NET "CFG_D[1]"   LOC = AA20;
NET "CFG_D[2]"   LOC = U14;
```



Obrázek 5. A - JTAG connector, B - Flash memory, C - Push button.

```
NET "CFG_D[3]"   LOC = U13;

NET "CFG_CS_N" LOC = T5;
NET "CFG_CLK"  LOC = Y21;
```

Signal CFG_D(0) is equivalent with flash DI/IO0, signal CFG_D(1) is equivalent with DO/IO1, signal CFG_D(2) with WP/IO2 and last signal CFG_D(3) with HLD/IO3 pin. The *Xilinx Impact* tool can be used for proper loading of FPGA bitstreams or other stuff to this flash for the first time (or processor in FPGA can access it). FPGA boot option is hard wired to master SPI [11] (M0 = 1, M1 = 0) and it enables multi-boot or golden design support with this serial flash [8]. In FPGA boot time, interface can utilize up to 26 MHz speed operation (limitation on FPGA side) and up to 4 data bits. Different FPGA boot time can be achieved with different configuration as shown in Table 2 (in seconds). All configuration values in Table 2 are possible with the *Xilinx Impact* tool.

Bit width/Frequency	2 MHz	4 MHz	6 MHz	10 MHz	12 MHz	16 MHz	22 MHz	26 MHz
1	16.8808	8.4404	5.6269	3.3761	2.8135	2.1101	1.5346	1.2985
2	8.4404	4.2202	2.8135	1.6880	1.4067	1.0550	0.7673	0.6492
3	4.2202	2.1101	1.4067	0.8440	0.7034	0.5275	0.3837	0.3246

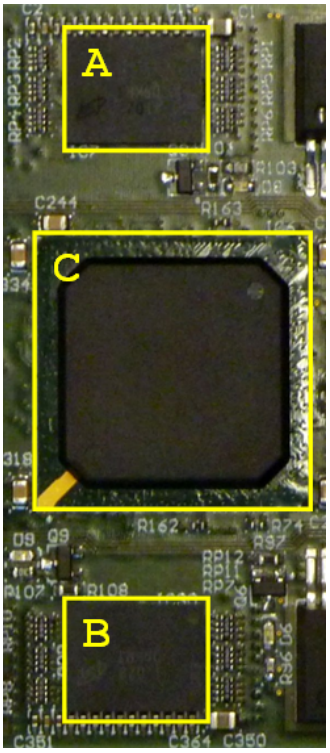
Tabulka 2. FPGA boot time (in seconds) from on-board flash memory for different configurations.

The μG4-150 platform has been successfully tested with maximum operation speed therefore the configuration time of 324,6 ms can be easily achieved with normal, uncompressed bitstream. As the FPGA (uncompressed) bitstream size is 33,761,696 bits [11], performance of 104 Mbps on the flash memory interface is achieved. Push button shown on the Fig. 5 can be used to reload bitstream

stored in flash memory at any time. LED signalisation is used for informing about successful FPGA bitstream loading (see LED indicators section).

3.3 DDR3 memory subsystem

The μG4-150 platform contains of two (DDR3A and DDR3B) 256 MB DDR3 type memories from Micron (type MT41J256M8HX) [6] shown on the Fig. 6.



Obrázek 6. A - First DDR3 memory, B - Second DDR3 memory, C - FPGA.

Both memories are connected to separate Memory Controller Block (MCB) in the Spartan-6 FPGA [9] so port mapping is given. UCF example for both memories follows:

```
NET "DDR3A_RZQ"   LOC = R22 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_ZIO"   LOC = C19 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;

NET "DDR3A_WE"    LOC = H19 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_RAS"   LOC = H21 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
```

```
NET "DDR3A_CAS"   LOC = H22 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_CK_P"  LOC = H20 | IOSTANDARD = DIFF_SSTL15_II |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_CK_N"  LOC = J19 | IOSTANDARD = DIFF_SSTL15_II |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_CKE"   LOC = D21 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_RST"   LOC = F18 | IOSTANDARD = SSTL15_II | TIG;

NET "DDR3A_A[0]"  LOC = F21 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[1]"  LOC = F22 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[2]"  LOC = E22 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[3]"  LOC = G20 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[4]"  LOC = F20 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[5]"  LOC = K20 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[6]"  LOC = K19 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[7]"  LOC = E20 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[8]"  LOC = C20 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[9]"  LOC = C22 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[10]" LOC = G19 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[11]" LOC = F19 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[12]" LOC = D22 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[13]" LOC = D19 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_A[14]" LOC = D20 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_BA[0]" LOC = J17 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_BA[1]" LOC = K17 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3A_BA[2]" LOC = H18 | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;

NET "DDR3A_DQ[0]" LOC = N20 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_DQ[1]" LOC = N22 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_DQ[2]" LOC = M21 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_DQ[3]" LOC = M22 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_DQ[4]" LOC = J20 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_DQ[5]" LOC = J22 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_DQ[6]" LOC = K21 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_DQ[7]" LOC = K22 | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3A_LDM"   LOC = L19 | IOSTANDARD = SSTL15_II |
 OUT_TERM = UNTUNED_50;
```

```
NET "DDR3A_DQS_P" LOC = L20 | IOSTANDARD = DIFF_SSTL15_II |
 IN_TERM = NONE | OUT_TERM = UNTUNED_50;
NET "DDR3A_DQS_N" LOC = L22 | IOSTANDARD = DIFF_SSTL15_II |
 IN_TERM = NONE | OUT_TERM = UNTUNED_50;
NET "DDR3A_ODT"   LOC = G22 | IOSTANDARD = SSTL15_II |
 OUT_TERM = UNTUNED_50;

NET "DDR3B_RZQ"   LOC = P1  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_ZIO"   LOC = Y2  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;

NET "DDR3B_WE"    LOC = F2  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_RAS"   LOC = K5  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_CAS"   LOC = K4  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_CK_P"  LOC = H4  | IOSTANDARD = DIFF_SSTL15_II |
OUT_TERM = UNTUNED_50;
NET "DDR3B_CK_N"  LOC = H3  | IOSTANDARD = DIFF_SSTL15_II |
OUT_TERM = UNTUNED_50;
NET "DDR3B_CKE"   LOC = D2  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_RST"   LOC = C3  | IOSTANDARD = SSTL15_II | TIG;

NET "DDR3B_A[0]"  LOC = H2  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[1]"  LOC = H1  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[2]"  LOC = H5  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[3]"  LOC = K6  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[4]"  LOC = F3  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[5]"  LOC = K3  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[6]"  LOC = J4  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[7]"  LOC = H6  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[8]"  LOC = E3  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[9]"  LOC = E1  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[10]" LOC = G4  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[11]" LOC = C1  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[12]" LOC = D1  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[13]" LOC = G6  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_A[14]" LOC = F5  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_BA[0]" LOC = G3  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_BA[1]" LOC = G1  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;
NET "DDR3B_BA[2]" LOC = F1  | IOSTANDARD = SSTL15_II | OUT_TERM = UNTUNED_50;

NET "DDR3B_DQ[0]" LOC = N3  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_DQ[1]" LOC = N1  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_DQ[2]" LOC = M2  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
```

```
NET "DDR3B_DQ[3]" LOC = M1  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_DQ[4]" LOC = J3  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_DQ[5]" LOC = J1  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_DQ[6]" LOC = K2  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_DQ[7]" LOC = K1  | IOSTANDARD = SSTL15_II | IN_TERM = NONE |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_LDM"   LOC = L4  | IOSTANDARD = SSTL15_II |
 OUT_TERM = UNTUNED_50;
NET "DDR3B_DQS_P" LOC = L3  | IOSTANDARD = DIFF_SSTL15_II |
 IN_TERM = NONE | OUT_TERM = UNTUNED_50;
NET "DDR3B_DQS_N" LOC = L1  | IOSTANDARD = DIFF_SSTL15_II |
 IN_TERM = NONE | OUT_TERM = UNTUNED_50;
NET "DDR3B_ODT"   LOC = J6  | IOSTANDARD = SSTL15_II |
OUT_TERM = UNTUNED_50;
```

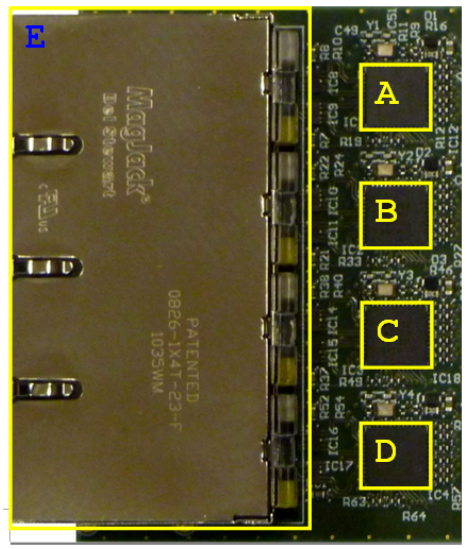
Performance analysis Xilinx Spartan-6 family of FPGAs allows up to 400 MHz performance on its pins. Using double data rate (DDR) technique, we can achieve up to 800 Mbps. Memory throughput depends also from two different power modes: normal and extended. Mode of operation specify hard requirements on the Vccint [12]. In commercial FPGAs (also used on the platform) it is 1.14 V (min) 1.2 V (typ) and 1.26 V (max). Extended mode have requirements of 1.2 V (min) 1.23 V (typ) and 1.26 V (max) and this is achievable on the μG4-150 platform. This also means possibility of use 800 Mbps performance. If Vccint power requirements are not met, only up to 667 Mbps throughput can be achieved.

Memory throughput is multiplied by the number of physical banks of the memory (x4, x8, x16, etc.). The μG4-150 platform uses x8 memory, so in theory, 6,400 Mbps should be achievable. On the other side, MCBs have 4 unidirectional and 2 bidirectional 32-bit wide physical ports [9]. Every port has its own fifo for 64 items. These ports can be configured almost soever (different direction and data wide), but every clock cycle it is decided which physical port will be serviced by specified priorities. Priorities can be also configured so that this arbitration can affect memory performance quite dramatically. The last thing that can affect memory performance is the frequency of user interface for the MCBs. The Xilinx documentation recommends to use 100 MHz [9].

3.4 Ethernet subsystem

Media access control (MAC) subsystem has to be implemented in FPGA, e.g. using Xilinx LogiCORE IP Tri-Mode Ethernet MAC [7], for proper Ethernet operation. It is due to use of Ethernet transceivers (phyters) that are connected to the FPGA by reduced GMII interface (RGMII). Platform utilizes low power

single port 10/100/1000 Mbps Micrel KSZ9021RN chip for each Ethernet port [3] shown on the Fig. 7.



Obrázek 7. A - Phyter A, B - Phyter B, C - Phyter C, D - Phyter D, E - Ethernet connectors.

By default, these four phyters are configured to advertise all capabilities (10/100/1000 Mbps operation in half or full duplex) during an auto-negotiation process. Configuration and/or status exploration of phyters can be done using shared serial MDC/MDIO bus. For example, it is possible to figure out result of auto-negotiation process. The maximum permitted frequency of MDC is 2.5 MHz. The MDIO format is defined in IEEE802.3 clause 22. Each phyter has its own address on the MDIO bus (PHYA = "001", PHYB = "010", PHYC = "011"and PHYD = "100").

Moreover, FPGA is also able to separately reset every phyter at any time using dedicated reset signals. Resetting is also recommended if given phyter is not in use. Phyters also use dedicated interrupts for different event signalling to the FPGA. On the other side, phyters are connected to the RJ45 metallic ports that are equipped with magnetics and with Green and Orange LED signalization. LEDs states are described in Table 3.

Platform utilizes 4xRJ45 Ethernet connectors (Bel Fuse) for direct phyter connection. Phyters port mapping to the FPGA can be seen in the following UCF template (assuming IOSTANDARD is implicitly "LVCMOS25"):

```
NET "MDC"   LOC = C9;
NET "MDIO"  LOC = A9;
```

Led colour	LED Definition	Link / Activity
Green	OFF	Link off
Green	ON	Link on (any speed)
Orange	OFF	No activity
Orange	Blinking	Activity (RX or TX)

Tabulka 3. Ethernet LED signalization.

```
NET "PHY_A_INT_N"        LOC = C16 | TIG;
NET "PHY_A_RST_N"        LOC = B16 | TIG;
NET "PHY_A_RGMII_TXD[0]" LOC = C13 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_A_RGMII_TXD[1]" LOC = A13 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_A_RGMII_TXD[2]" LOC = D14 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_A_RGMII_TXD[3]" LOC = C14 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_A_RGMII_TX_CTL" LOC = B14 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_A_RGMII_TX_CLK" LOC = A12 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_A_RGMII_RXD[0]" LOC = D15;
NET "PHY_A_RGMII_RXD[1]" LOC = C15;
NET "PHY_A_RGMII_RXD[2]" LOC = A15;
NET "PHY_A_RGMII_RXD[3]" LOC = A14;
NET "PHY_A_RGMII_RX_CTL" LOC = A16;
NET "PHY_A_RGMII_RX_CLK" LOC = B12;
NET "PHY_A_RGMII_RX_CLK" TNM_NET = "PHY_A_RGMII_RX_CLK1";
TIMESPEC "TS_PHY_A_RGMII_RX_CLK1" = PERIOD "PHY_A_RGMII_RX_CLK1"
 8 ns HIGH 50%;

NET "PHY_B_INT_N"        LOC = F15 | TIG;
NET "PHY_B_RST_N"        LOC = H14 | TIG;
NET "PHY_B_RGMII_TXD[0]" LOC = H12 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_B_RGMII_TXD[1]" LOC = F12 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_B_RGMII_TXD[2]" LOC = D12 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_B_RGMII_TXD[3]" LOC = E12 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_B_RGMII_TX_CTL" LOC = D13 | OUT_TERM = "UNTUNED_25" |
```

```
 SLEW = FAST;
NET "PHY_B_RGMII_TX_CLK" LOC = C12 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_B_RGMII_RXD[0]" LOC = E14;
NET "PHY_B_RGMII_RXD[1]" LOC = H13;
NET "PHY_B_RGMII_RXD[2]" LOC = G13;
NET "PHY_B_RGMII_RXD[3]" LOC = F13;
NET "PHY_B_RGMII_RX_CTL" LOC = F14;
NET "PHY_B_RGMII_RX_CLK" LOC = C11;
NET "PHY_B_RGMII_RX_CLK" TNM_NET = "PHY_B_RGMII_RX_CLK1";
TIMESPEC "TS_PHY_B_RGMII_RX_CLK1" = PERIOD "PHY_B_RGMII_RX_CLK1"
 8 ns HIGH 50%;

NET "PHY_C_INT_N"        LOC = G11 | TIG;
NET "PHY_C_RST_N"        LOC = H11 | TIG;
NET "PHY_C_RGMII_TXD[0]" LOC = G9  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_C_RGMII_TXD[1]" LOC = G8  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_C_RGMII_TXD[2]" LOC = F8  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_C_RGMII_TXD[3]" LOC = E8  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_C_RGMII_TX_CTL" LOC = F9  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_C_RGMII_TX_CLK" LOC = A11 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_C_RGMII_RXD[0]" LOC = D10;
NET "PHY_C_RGMII_RXD[1]" LOC = E10;
NET "PHY_C_RGMII_RXD[2]" LOC = F10;
NET "PHY_C_RGMII_RXD[3]" LOC = H10;
NET "PHY_C_RGMII_RX_CTL" LOC = C10;
NET "PHY_C_RGMII_RX_CLK" LOC = D11;
NET "PHY_C_RGMII_RX_CLK" TNM_NET = "PHY_C_RGMII_RX_CLK1";
TIMESPEC "TS_PHY_C_RGMII_RX_CLK1" = PERIOD "PHY_C_RGMII_RX_CLK1"
 8 ns HIGH 50%;

NET "PHY_D_INT_N"        LOC = D9  | TIG;
NET "PHY_D_RST_N"        LOC = D8  | TIG;
NET "PHY_D_RGMII_TXD[0]" LOC = D6  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_D_RGMII_TXD[1]" LOC = C6  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_D_RGMII_TXD[2]" LOC = B6  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
```

```
NET "PHY_D_RGMII_TXD[3]" LOC = A6  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_D_RGMII_TX_CTL" LOC = A7  | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_D_RGMII_TX_CLK" LOC = A10 | OUT_TERM = "UNTUNED_25" |
 SLEW = FAST;
NET "PHY_D_RGMII_RXD[0]" LOC = B8;
NET "PHY_D_RGMII_RXD[1]" LOC = A8;
NET "PHY_D_RGMII_RXD[2]" LOC = D7;
NET "PHY_D_RGMII_RXD[3]" LOC = C7;
NET "PHY_D_RGMII_RX_CTL" LOC = C8;
NET "PHY_D_RGMII_RX_CLK" LOC = B10;
NET "PHY_D_RGMII_RX_CLK" TNM_NET = "PHY_D_RGMII_RX_CLK1";
TIMESPEC "TS_PHY_D_RGMII_RX_CLK1" = PERIOD "PHY_D_RGMII_RX_CLK1"
 8 ns HIGH 50%;
```

Note that TX_CTL is same as TX_EN and RX_CTL is same as RX_DV.

RGMII specification requires clock to data skew of 1.5 ns to 2.1 ns, but as all RGMII wires on the platform are length compensated to approx. 50.5 mm the clock (or data) wires should be somehow slowed. This can be done either on the FPGA side (by adjusting IODELAYs or clock phase using DCMs/PLLs [10]) or better it can be also done by proper configuration of each phyter (as they are capable of adding positive or negative clock to data skew) [3].

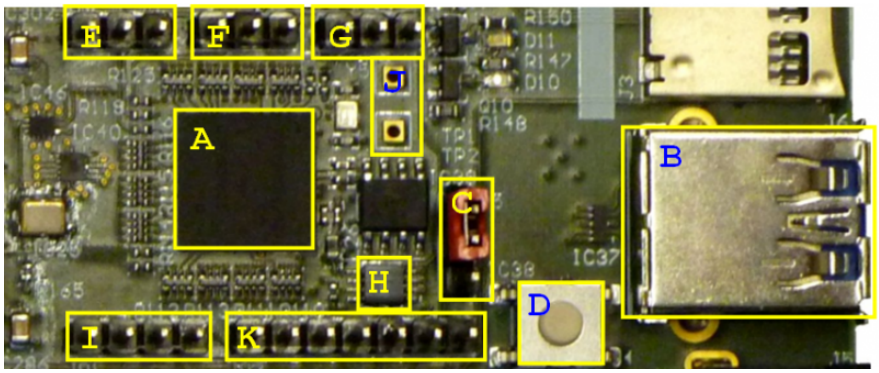
Some specific delay and/or timing constraints are needed for proper function of the RGMII interface:

```
NET "PHY_A_RGMII_RX_RXD[?]" TNM_NET = "PHY_A_RGMII_RX_DATA1";
NET "PHY_A_RGMII_RX_CTL"    TNM_NET = "PHY_A_RGMII_RX_DATA1";
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_A_RGMII_RX_CLK FALLING;
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_A_RGMII_RX_CLK RISING;

NET "PHY_B_RGMII_RX_RXD[?]" TNM_NET = "PHY_B_RGMII_RX_DATA1";
NET "PHY_B_RGMII_RX_CTL"    TNM_NET = "PHY_B_RGMII_RX_DATA1";
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_B_RGMII_RX_CLK FALLING;
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_B_RGMII_RX_CLK RISING;

NET "PHY_C_RGMII_RX_RXD[?]" TNM_NET = "PHY_C_RGMII_RX_DATA1";
NET "PHY_C_RGMII_RX_CTL"    TNM_NET = "PHY_C_RGMII_RX_DATA1";
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_C_RGMII_RX_CLK FALLING;
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_C_RGMII_RX_CLK RISING;

NET "PHY_D_RGMII_RX_RXD[?]" TNM_NET = "PHY_D_RGMII_RX_DATA1";
NET "PHY_D_RGMII_RX_CTL"    TNM_NET = "PHY_D_RGMII_RX_DATA1";
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_D_RGMII_RX_CLK FALLING;
OFFSET = IN 1.35 ns VALID 3.25 ns BEFORE PHY_D_RGMII_RX_CLK RISING;
```

These constraints can be aplied only in RX direction (i.e. from phyter to FPGA) when clock to data skew is programmed to 1.8 ns. In TX direction (i.e. from FPGA to phyter) such skew has to be adjusted with regard to phyter requirements that is between 1.0 ns and 2.6 ns [3] but no specific FPGA constraints are needed.

3.5 USB3.0 subsystem

The μG4-150 platform has an USB 3.0 interface (type B connector shown on the Fig. 8) allowing up to 5 Gbps bidirectional transfers on it. This interface is serviced by the Cypress FX3 SuperSpeed USB Controller [4] that is connected to the FPGA via 32 bit generic programmable interface (Cypress GPIFII) running at frequencies up to 100 MHz. It allows up to 400 MBps transfers from FPGA to USB 3.0 interface or vice versa (without protocol and controller software overhead).



Obrázek 8. A - USB 3.0 controller, B - USB 3.0 connector, C - Reset jumpers (set to reset controller from FPGA), D - Reset button, E - PMOD[0] (with 2.5 V on the left and GND on the right), F - PMOD[1] (with 2.5 V on the left and GND on the right), G - PMOD[2] (with 2.5 V on the left and GND on the right), H - Flash memory, I - I2C pinheader (from the left to the right: 2.5 V, SCL, SDA and GND) J - UART testing points (up: TX, down: RX), K - JTAG interface (from the left to the right: 2.5 V, TCK, TMS, TDI, TRST, TDO and GND).

Connection to the FPGA is evident from this UCF example (assuming IOSTANDARD is implicitly "LVCMOS25"):

```
NET "USB_DQ[0]"  LOC = W10;
NET "USB_DQ[1]"  LOC = W9;
NET "USB_DQ[2]"  LOC = AA10;
NET "USB_DQ[3]"  LOC = V13;
```

```
NET "USB_DQ[4]"  LOC = W13;
NET "USB_DQ[5]"  LOC = V11;
NET "USB_DQ[6]"  LOC = V7;
NET "USB_DQ[7]"  LOC = W11;
NET "USB_DQ[8]"  LOC = W8;
NET "USB_DQ[9]"  LOC = AA8;
NET "USB_DQ[10]" LOC = AB10;
NET "USB_DQ[11]" LOC = Y9;
NET "USB_DQ[12]" LOC = Y10;
NET "USB_DQ[13]" LOC = AB9;
NET "USB_DQ[14]" LOC = AB8;
NET "USB_DQ[15]" LOC = R15;
NET "USB_DQ[16]" LOC = W18;
NET "USB_DQ[17]" LOC = AB19;
NET "USB_DQ[18]" LOC = AA18;
NET "USB_DQ[19]" LOC = Y19;
NET "USB_DQ[20]" LOC = Y18;
NET "USB_DQ[21]" LOC = V18;
NET "USB_DQ[22]" LOC = T18;
NET "USB_DQ[23]" LOC = V19;
NET "USB_DQ[24]" LOC = Y17;
NET "USB_DQ[25]" LOC = W17;
NET "USB_DQ[26]" LOC = V17;
NET "USB_DQ[27]" LOC = AB18;
NET "USB_DQ[28]" LOC = T16;
NET "USB_DQ[29]" LOC = U17;
NET "USB_DQ[30]" LOC = R16;
NET "USB_DQ[31]" LOC = T17;

NET "USB_PCLK"    LOC = AB11;

NET "USB_CTL[0]"  LOC = AB14;
NET "USB_CTL[1]"  LOC = Y14;
NET "USB_CTL[2]"  LOC = W14;
NET "USB_CTL[3]"  LOC = T15;
NET "USB_CTL[4]"  LOC = W15;
NET "USB_CTL[5]"  LOC = AA14;
NET "USB_CTL[6]"  LOC = AB15;
NET "USB_CTL[7]"  LOC = AB7;
NET "USB_CTL[8]"  LOC = AB17;
NET "USB_CTL[9]"  LOC = Y15;
NET "USB_CTL[10]" LOC = Y16;
NET "USB_CTL[11]" LOC = AB16;
NET "USB_CTL[12]" LOC = AA16;
```

USB_DQ[*] is programmable data interface, USB_PCLK is (from the FPGA point of view) external clock used for communication on data interface and USB_CTL[*] is programmable control interface. More information can be found in the Cypress documentation. The controller can inform about different events using its interrupt. For resetting the controller, reset signal can be used. Follows UCF template (assuming IOSTANDARD is implicitly "LVCMOS25") for interrupt and reset signals (both connected to the FPGA):

```
NET "USB_INT_N"   LOC = AB21;
NET "USB_RESET_N" LOC = AA21;
```

But the controller reset can be set externally via platform jumpers shown on the Fig. 8. So it is possible to reset controller externally from the FPGA or using second push button on the platform.

There is also I2C flash memory from Microchip [5] available for booting purposes of the controller. The memory size is 1024 kb and its I2C interface is connected to the controller. On the other side it is connected to the platform pinheader (e.g. for external flash programming) and also to the FPGA. Special FPGA enable signal (USB3_I2C_EN) can be used for enabling or disabling access to this memory as memory itself has write protect signal connected firmly to the ground. Follows UCF template or I2C interface to the FPGA (assuming IOSTANDARD is implicitly "LVCMOS25"):

```
NET "USB_I2C_EN" LOC = T7;
NET "USB_SCL"    LOC = Y8;
NET "USB_SDA"    LOC = Y7;
```

For selecting location of the boot file for the controller, PMODs are bring out to the platform jumpers as shown on the Fig. 8. Different sources of boot file will be scanned depending on PMODs adjustment as it is shown in Table 4, where NC means not connected and 2.5 V/GND means connection to the power/ground jumper.

USB 3.0 controller boots from	PMOD[0]	PMOD[1]	PMOD[2]
Sync ADMux (16-bit)	NC	GND	GND
Async ADMux (16-bit)	NC	GND	2.5 V
USB boot	NC	2.5 V	2.5 V
Async SRAM (16-bit)	GND	NC	NC
I2C, On Failure, USB Boot is Enabled)	NC	2.5 V	NC
I2C only	2.5 V	NC	NC
SPI, On Failure, USB Boot is Enabled	GND	NC	2.5 V

Tabulka 4. USB 3.0 controller boot options depending on the PMODs configurations.

UART interface (but only RX and TX data signals) is connected directly to the FPGA and internal FPGA multiplexor can be used to switch this UART

to physical UART interface (see Serial port section). This UART has also its testing point on the platform shown on the Fig. 8. Here is an UCF example for UART:

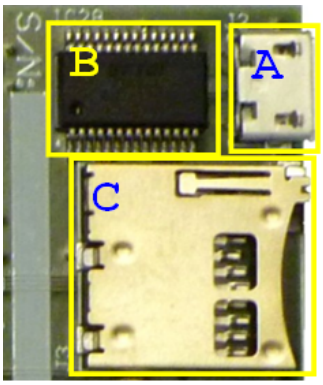
```
NET "USB_UART_TX" LOC = D5 | IOSTANDARD = LVCMOS15;
NET "USB_UART_RX" LOC = E4 | IOSTANDARD = LVCMOS15;
```

Other controller GPIOs, as is 45, 50 (I2S_CLK), 51 (I2S_SD), 52 (I2S_WS), 53 (UART_RTS), 54 (UART_CTS), 57 (I2S_MCLK), are all not connected to the FPGA or elsewhere. Finally, there is a separate full JTAG interface available to the controller. This interface is shown on the Fig. 8.

3.6 Serial port

Not only for debugging purposes μG4-150 platform consists also from the simple UART interface (again just RX and TX data signals) that can be accessed via micro (type B) USB compatible port shown on the Fig. 9. Translation from simple UART to USB interface is done by the FT232R USB UART IC chip from FTDI [2] also shown on the Fig. 9. FPGA port mapping and required voltage level for this UART is shown in this example:

```
NET "UART_TX" LOC = H8 | IOSTANDARD = LVCMOS15;
NET "UART_RX" LOC = J7 | IOSTANDARD = LVCMOS15;
```



Obrázek 9. A - USB connector, B - UART to USB convertor, C - microSD slot.

3.7 Micro Secure Digital port

μG4-150 platform is equipped with micro Secure Digital (SD) port shown on the Fig. 9. This port can be used for micro SD compatible memory cards (or

SDHC/SDXC) running at standard 3.3 V. Connection to the FPGA and required voltage level is evident from following UCF example:

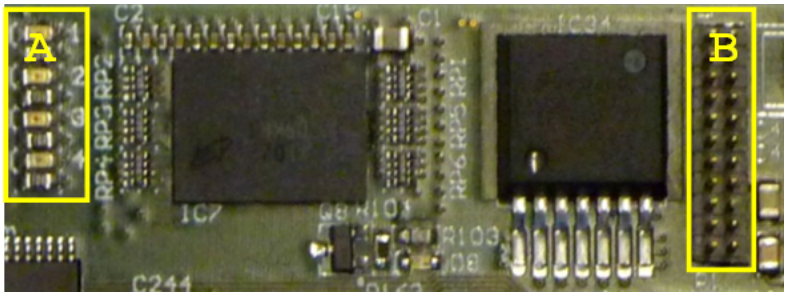
```
NET "SD_CS_N" LOC = D17 | IOSTANDARD = LVCMOS15;
NET "SD_CLK"  LOC = E16 | IOSTANDARD = LVCMOS15;
NET "SD_DI"   LOC = A18 | IOSTANDARD = LVCMOS15;
NET "SD_DO"   LOC = B18 | IOSTANDARD = LVCMOS15;
```

Signal SD_CS_N is equivalent with CD/DAT3, SD_CLK is equivalent with SCLK, SD_DI is equivalent with CMD, SD_DO is equivalent with DAT0 and last interface signals (i.e. DAT1 and DAT2) are not connected.

3.8 LED indicators

The platform is equipped with LED indicators. The green ones can be used for debugging or other signalization purposes as they are connected directly to the FPGA. These green LEDs are active in low voltage level (e.g. construction LED(3) <= '0'; in VHDL ensure shinning of fourth LED) and are shown on the Fig. 10. The UCF should look like (assuming IOSTANDARD is implicitly "LVCMOS25"):

```
NET "LED[0]" LOC = C5;
NET "LED[1]" LOC = A5;
NET "LED[2]" LOC = C17;
NET "LED[3]" LOC = A17;
```



Obrázek 10. A - Green LEDs (upper one: 0) and B - P1 pinheader.

The most of blue LEDs (not shown on the Fig. 10) are used to indicate proper function of platform power suppliers (i.e. one for 5V@5A, for two 3.3V@1A, for 2.5@3A, for 1.5V@3A, for 1.2V@4A and for 0.75V@3A supplier). The last blue LED (near the Push button on the Fig. 5) indicates that FPGA is booted (connected to the FPGA_DONE pin), otherwise the only one on-board red LED is shining.

3.9 General purpose inputs and outputs

There are sixteen general purpose input and output (GPIO) pins connected directly to the FPGA via P1 pinheader that is shown on the Fig. 10.

These GPIOs can be used to extend connectivity of the platform using, for example, PMODs. GPIOs connection is explained in Table 5.

5 V	5 V
1.5 V	1.5 V
GPIO[0]	GPIO[1]
GPIO[2]	GPIO[3]
GPIO[4]	GPIO[5]
GPIO[6]	GPIO[7]
GPIO[8]	GPIO[9]
GPIO[10]	GPIO[11]
GPIO[12]	GPIO[13]
GPIO[14]	GPIO[15]
GND	GND

Tabulka 5. GPIOs and other pins on the pinheader P1.

GPIOs are connected to the 1.5 V bank (shared with DDR3 memories), so UCF for them should look like:

```
NET "GPIO[0]" LOC = L15 | IOSTANDARD = LVCMOS15;
NET "GPIO[1]" LOC = W22 | IOSTANDARD = LVCMOS15;
NET "GPIO[2]" LOC = K18 | IOSTANDARD = LVCMOS15;
NET "GPIO[3]" LOC = W20 | IOSTANDARD = LVCMOS15;
NET "GPIO[4]" LOC = L17 | IOSTANDARD = LVCMOS15;
NET "GPIO[5]" LOC = V20 | IOSTANDARD = LVCMOS15;
NET "GPIO[6]" LOC = M17 | IOSTANDARD = LVCMOS15;
NET "GPIO[7]" LOC = U19 | IOSTANDARD = LVCMOS15;
NET "GPIO[8]" LOC = M16 | IOSTANDARD = LVCMOS15;
NET "GPIO[9]" LOC = R19 | IOSTANDARD = LVCMOS15;
NET "GPIO[10]" LOC = M18 | IOSTANDARD = LVCMOS15;
NET "GPIO[11]" LOC = P19 | IOSTANDARD = LVCMOS15;
NET "GPIO[12]" LOC = N16 | IOSTANDARD = LVCMOS15;
NET "GPIO[13]" LOC = P18 | IOSTANDARD = LVCMOS15;
NET "GPIO[14]" LOC = P17 | IOSTANDARD = LVCMOS15;
NET "GPIO[15]" LOC = P20 | IOSTANDARD = LVCMOS15;
```

4 Development support

It is recommended to use Xilinx EDK (Embedded Development Kit) and Xilinx SDK (Software Development Kit) tools for fast development and prototyping.

There are several FPGA IP cores available from Xilinx for use with the μG4-150 platform (e.g. SPI, IIC, UART, GPIO) and more IP cores are available from authors. Also the μG4-150 platform description files as well as design examples for several interfaces are available on request. It is also possible to get a special port of Linux OS for Xilinx MicroBlaze soft processor running in the FPGA of μG4-150 platform. There is also a bootloader that can be very useful for booting operating system from memory cards available on the platform. Please contact authors if you are interested in this software and firmware support or if you need any other hardware technical support.

Reference

1. Electronics, W.; Corp.: 3V 128M-bit Serial Flash Memory with dual and quad SPI (W25Q128BV), Datasheet.
2. (FTDI), F. T. D. I.; Ltd.: FT232R USB UART IC Datasheet.
3. Micrel; Inc.: KSZ9021RL/RN: Gigabit Ethernet Transceiver with RGMII Support, Datasheet.
4. Semiconductor, C.; Corp.: EZ-USB? FX3 SuperSpeed USB Controller, Datasheet.
5. Technology, M.; Inc.: 1024K I2C? CMOS Serial EEPROM: 24AA1025/24LC1025/24FC1025, Preliminary Datasheet.
6. Technology, M.; Inc.: 2Gb: x4, x8, x16 DDR3 SDRAM Features, Datasheet.
7. Xilinx; Inc.: LogiCORE IP Tri-Mode Ethernet MAC - Product Specification DS818.
8. Xilinx; Inc.: SP605 MultiBoot Design, XTP059.
9. Xilinx; Inc.: Spartan-6 FPGA Memory Controller - User Guide UG388.
10. Xilinx; Inc.: Spartan-6 Family Overview, Product Specification - Xilinx Datasheet DS160. 2011, version 2.0.
    URL `http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf`
11. Xilinx; Inc.: Spartan-6 FPGA Configuration - User Guide UG380. 2011, version 2.3.
    URL `http://www.xilinx.com/support/documentation/user_guides/ug380.pdf`
12. Xilinx; Inc.: Spartan-6 FPGA Data Sheet: DC and Switching Characteristics - Product Specification DS162. 2011, version 3.0.
    URL `http://www.xilinx.com/support/documentation/data_sheets/ds162.pdf`
13. Xilinx; Inc.: Spartan-6 FPGA Packaging and Pinouts - Product Specification UG385. 2011, version 2.2.
    URL `http://www.xilinx.com/support/documentation/user_guides/ug385.pdf`