

# Marek Funtowicz

Embedded Software / FPGA Firmware Engineer

## Personal Details

**Address:** Stanisława Ligonia 14/37  
44-280 Rydułtowy

**Phone:** +48 724 235 978

**E-mail:** ice.marek@yahoo.com

**Git:** github.com/IceDefcon

**Web:** icedefcon.github.io

## Education

**2014 – 2017 The University of Sheffield**  
Electronics and Communication Engineering  
**Master of Engineering**

**2013 – 2014 Open Study College**  
Pure Mathematics  
**A-Level**

**2012 – 2014 Central College Nottingham**  
Electrical and Electronic Engineering  
**Higher National Diploma**

## Skill

**FPGA:** Intel, Xilinx

**CPU:** x86, ARM, PowerPC

**RTL:** VHDL

**Code:** ASM, C, C++, Python,

**OS:** FreeRTOS, VxWorks,  
Linux, Autosar

**Com:** USB, UART, SPI, I2C,  
JTAG, GPIO, TCP, CAN

**Circuit:** Mentor graphics, Altium,  
Multisim,

**Test:** Oscilloscope, Multimeters  
Logic Analyzer

**Repo:** GIT, Bitbucket, Gerrit

**Matrix:** Matlab, Octave

**Web:** HTML, PHP, CSS,  
Java script

**DB:** MySQL

**Lang:** Polish (Native),  
English (Fluent)

## Personal Research and Development

### CPU & FPGA Computer Platform (C, C++, VHDL, Python)

- ❖ x86 → Powered by Ubuntu
- ❖ LNX → ARM Cortex-A8 powered by Debian
- ❖ FPGA → Intel Cyclone IV FPGA for parallel computation

### System specification:

- ❖ x86 → Graphical User Interface for the overall master control
- ❖ x86 → Network Client (Python GUI)
- ❖ LNX → Network Server (C++ Application)
- ❖ LNX → Char device for IO between Kernel and User space
- ❖ LNX → Block RAM device for chip configuration in FPGA
- ❖ LNX → SPI driver for bidirectional FPGA communication over DMA
- ❖ LNX → Bidirectional GPIO signaling for FPGA/Kernel ISR processing
- ❖ LNX → Work queues and kthread design techniques
- ❖ FPGA → Watchdog interface for Kernel synchronization
- ❖ FPGA → SPI interface for data serialization and parallelization
- ❖ FPGA → Parametrized I2C controller driven from LNX Kernel
- ❖ FPGA → Parametrized FIFO controller driven from LNX Kernel
- ❖ FPGA → Offload controller and packet switch for FIFO data flow
- ❖ FPGA → PWM controller for the motor throttle control
- ❖ FPGA → UART controller for the logs acquisition
- ❖ FPGA → PID controllers for feedback control :: Pending R&D
- ❖ FPGA → Gyroscope/Accelerometer devices for measurement acquisition

### Platform will be upgraded to Xilinx Zynq UltraScale+™ MPSoC

- ❖ System will be powered by the Ultra96-V2
- ❖ ARM Cortex-A53, ARM Cortex-R5 and a Mali-400 MP2 GPU
- ❖ More powerful FPGA chip fully integrated with CPU over DMA
- ❖ The SPI interface for CPU<->FPGA comms replaced with a DMA
- ❖ Adaptation of the the current LNX kernel module to Xilinx system
- ❖ 40-pin general-purpose I/O and low-speed peripheral interfaces
- ❖ 60-pin high-speed connector (handling higher data rates interfaces)

**Software Engineer****Prototyping of the new generation wireless charging systems in the automotive industry (ASM, C)**

- ❖ Read and analyze prototype PCB schematics
- ❖ Customer defect analysis, compute and test solutions
- ❖ Interrupt monitor to monitor latency of ISR
- ❖ Communication problems between the charger and the receiver using ASK and FSK protocols
- ❖ Adaptation of solutions from EPP protocol into newer MPP stack
- ❖ Process various number of ADC measurements: voltage, current and temperatures signals
- ❖ Integration with the Autosar OS (Davinci code generator)
- ❖ Integration with RTD from NXP (Tresos)
- ❖ Integration with Qi-Library from NXP
- ❖ Code reviews

**Software Engineer****Complex CPU & FPGA system to emulate the behavior of cellular network (C, C++, VHDL)**

- ❖ x86 HTML interface to control load binaries required for specific configuration of cellular network
- ❖ x86 Test Application to configure parameters of LTE or 5G cells
- ❖ Store and process cell parameters from Test Application by the PowerPC processor boards using instances of C++ classes
- ❖ Configure LTE or 5G cells in FPGA using cell parameters stored by the C++ classes
- ❖ Communication of PowerPC processor boards with FPGA over SRIO
- ❖ JIRA to process internal and customer issues
- ❖ Issues was varying from invalid cell configurations in Test Application with respect to LTE and 5G specifications
- ❖ Up to the low level PHY problems like: Thread crashes, PowerPC disassembly and investigation of instruction code to find the bug

**Embedded Systems Engineer****Design Firmware and RTL for Power Electronic Device using AURIX 32-bit Tri-Core Microcontroller and ARTIX 7 Xilinx FPGA (Soft Power Bridge)**

- ❖ Integration of the LWIP stack drivers with the current CPU stack
- ❖ Investigation of ASCLIN UART drivers for logging and communication
- ❖ Research Microblaze firmware in ARTIX 7 FPGA
- ❖ Investigate and analyze circuit schematics and PCB layout

**Junior Hardware Engineer****Design computer hardware for capturing, processing and streaming live video using Intel Altera FPGA technology and firmware control**

- ❖ Design FPGA hardware modules in VHDL
- ❖ Design QSYS networks for Nios II processing
- ❖ On-chip memory management for Nios II processing
- ❖ Test RTL modules with Modelsim
- ❖ Test and debug RTL modules with signal tap and live PCB
- ❖ Nios II firmware for communication with FPGA over Avalon bus
- ❖ Debug Nios II firmware on live targets
- ❖ Integrate BSP and Intel HAL drivers
- ❖ Investigate booting codes for embedded Intel Nios II processor
- ❖ FPGA Pin and Chip planning → IO Banks, Voltages, Transceivers
- ❖ Investigate and analyze circuit schematics and PCB layout
- ❖ Integrate FPGA hardware code and CPU firmware with the circuit schematic and PCB Layout

**Prototyping embedded graphical processing system using Xilinx FPGA technology and Linux firmware control**

- ❖ Develop booting sequence for Zynq Ultrascale MPSoC in QSPI → hand over FPGA control to Linux kernel via U-Boot
- ❖ Research and modify U-boot source → Configure MAC addressing
- ❖ Research XEN Hypervisor → Register XEN watchdog
- ❖ Develop UART communication for Zynq Ultrascale MPSoC → register UARTLITE driver in Linux to communicate with external chip for TMDS processing and HDCP control
- ❖ Research kernel → Version control
- ❖ Research device tree → Petalinux overlay
- ❖ Research root file system → Network configuration
- ❖ Customize Petalinux → Kernel, Root file system and device tree
- ❖ Debug kernel in OS awareness mode → Eclipse environment