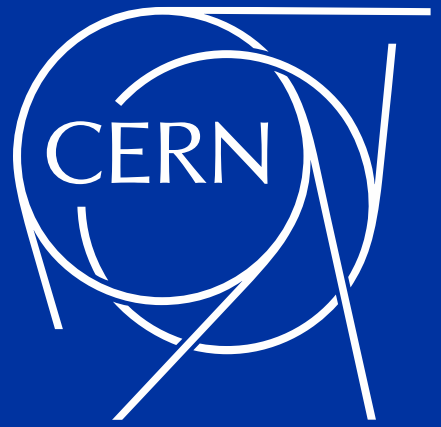
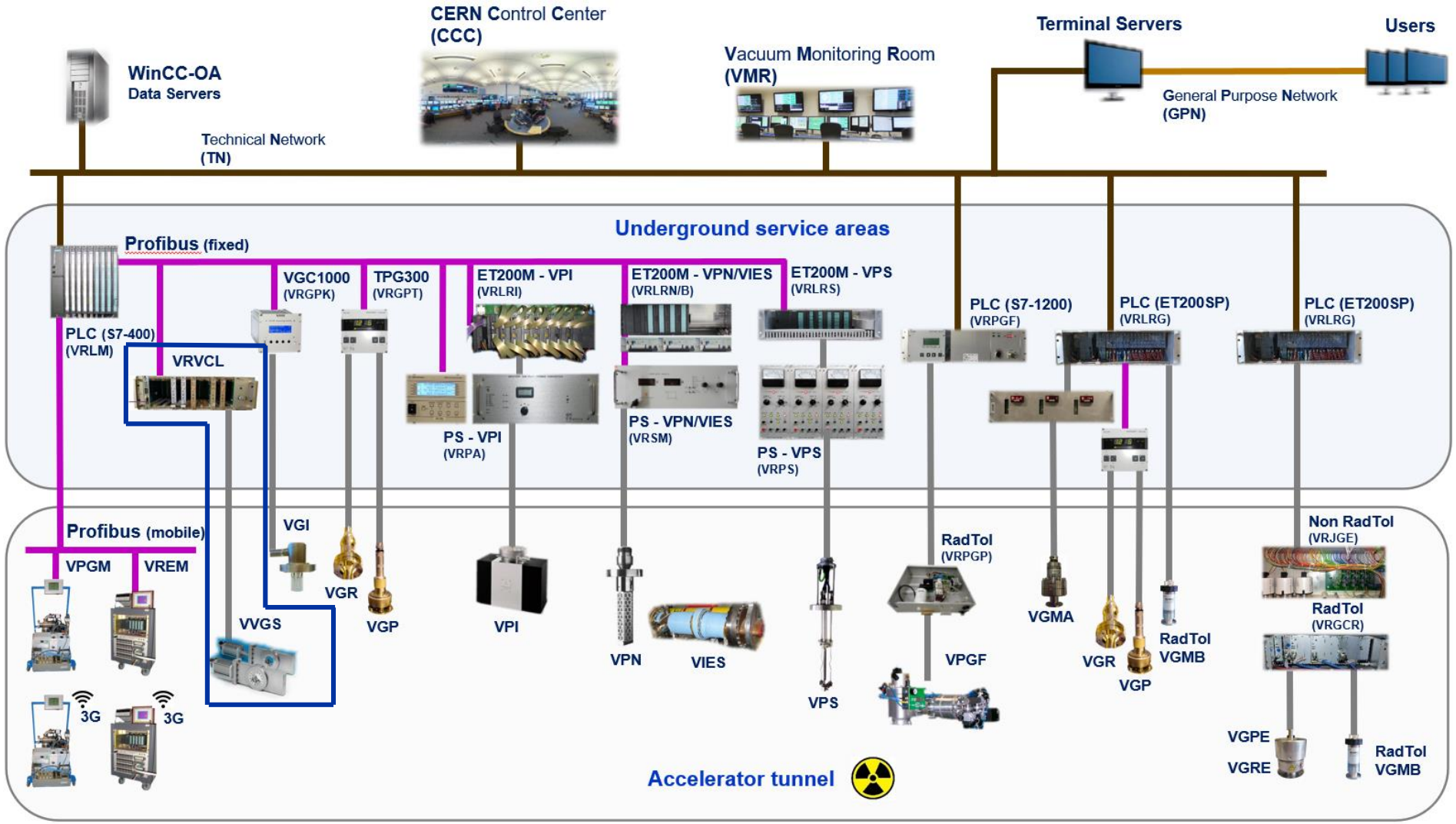


New Sector Valve Control Unit

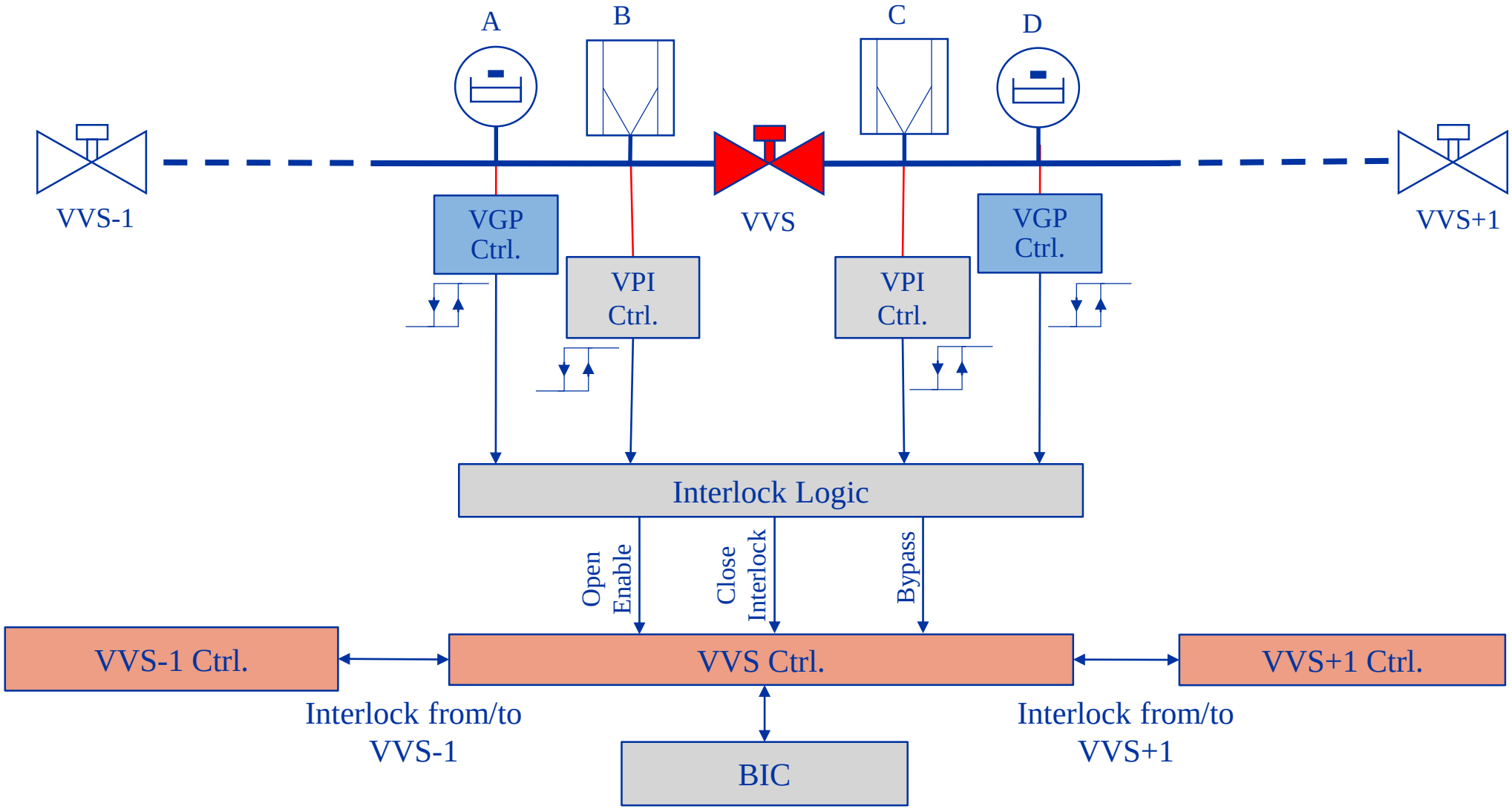
Sara M. G. Soares

6/12/2022

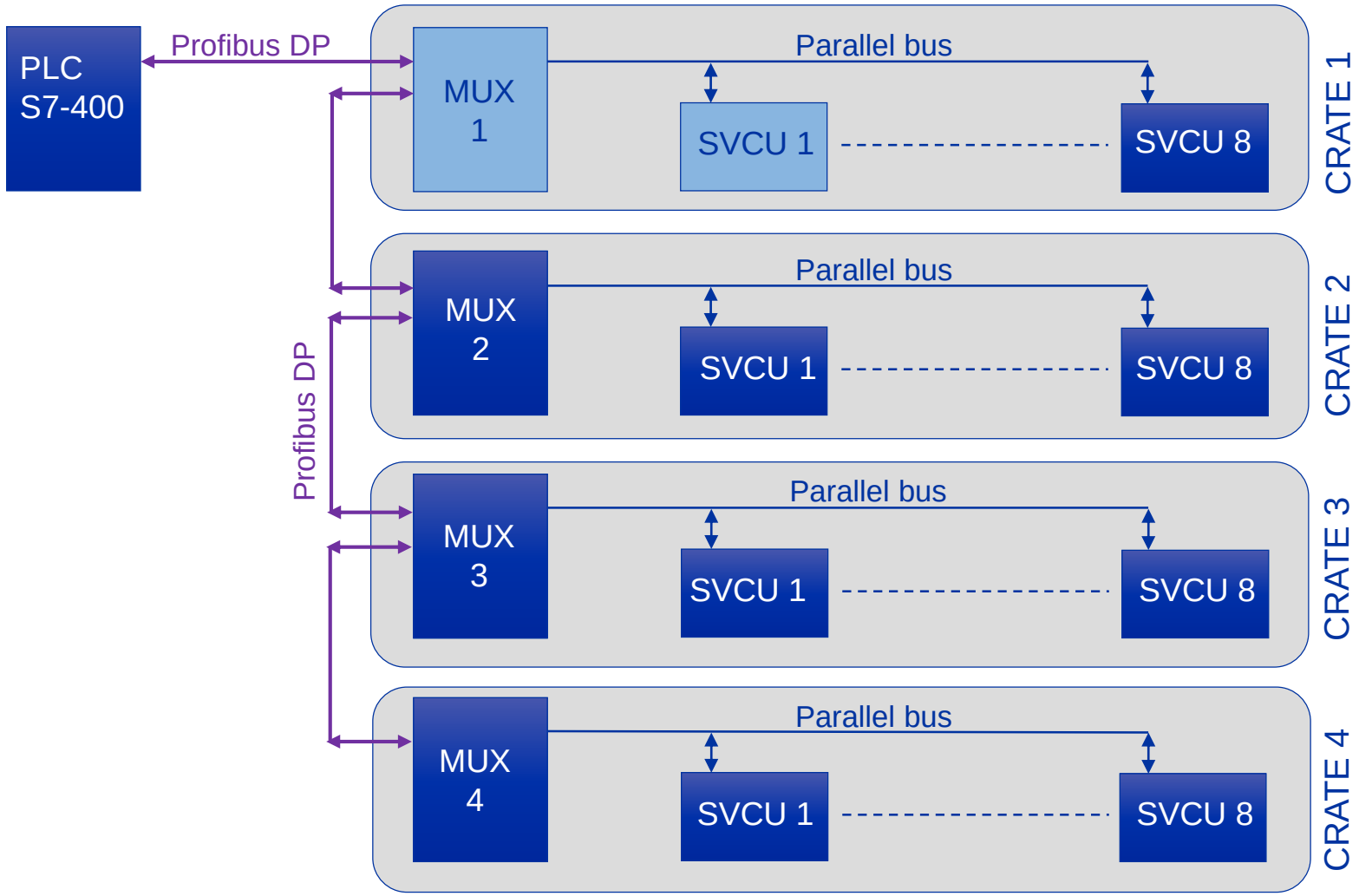
Vacuum Control System Architecture



VVS Interlock logic – LHC example



Sector Valve Controller Architecture



Programmable components

CPLD (**C**omplex **P**rogrammable **L**ogic **D**evice)

- Internally based on LUTs (Look-Up Tables)
- Contain embedded flash memory
- Smaller timing delays (compared to FPGAs)

FPGA (**F**ield **P**rogrammable **G**ate **A**rray)

- Internally based on CLBs (Configuration Logic Block)
- Flexible
- Timing delays depend on implementation method

- Programmable Logic Device family (PLD)
  - Programming a PLD consists of specifying how the logic gates will be interconnected upon power-up.
  - Inherent parallelism (fast and predictable reaction time)
  - No instructions are executed

Programmable components

Microcontroller (MCU)

- Computing system
- Designed for embedded application
- Digital I/Os
- DAC and ADC

- Programming a microcontroller consists of defining a list of instructions stored within its flash memory which are then executed sequentially after power-up.
- Relatively slow (compared to CPLD and FPGA)
- No parallelism

Sector Valve Controller

Profibus DP communication card (**MUX**)

Sector Valve Control card (**SVCU**)

- **Existing Version (in production)**
  - Based on microcontroller 18F6527 from Microchip
  - Profibus slave ASIC (**A**pplication **S**pecific **I**ntegrated **C**ircuits) VPC3+S from profichip

- **Existing Version (in production)**
  - Based on CPLD (**C**omplex **P**rogrammable **L**ogic **D**evice) MAX from Altera

New Design Motivations

- **SVCU:**
  - 20 years old design
  - CPLD not available anymore on the market
  - Quartus Altera software not compatible anymore
  - Logic described graphically and only with combinational logic
  - No Clock system
  - Obsolescence of other components
  - Only compatible with parallel backplane bus communication
  - No protection against surges of voltage/current
- **MUX:**
  - Only compatible with parallel backplane bus communication
  - Must integrate the functionalities of the new SVCU card
  - No protection against surges of voltage/current

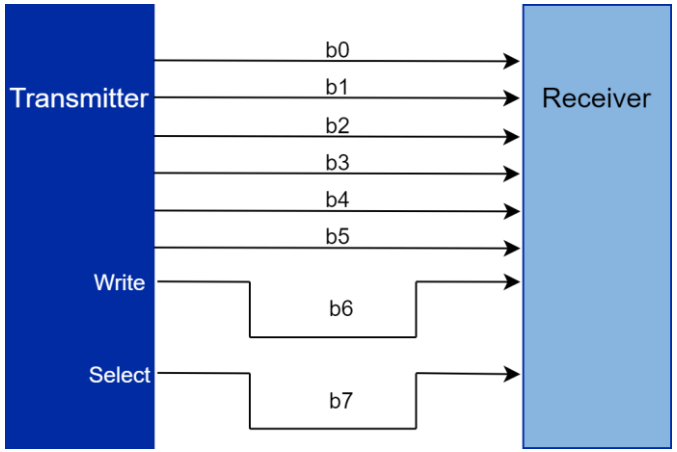
New Design Specifications

- Keep the existing crate and backplane
- Backward compatible with the current system for smooth deployment
- Implement SPI (**S**erial **P**eripheral **I**nterface) serial communication with differential lines (MUX + SVCU):
  - Aim to replace the current parallel communication
  - Improved reliability (less sensitive to glitches)
  - No limitation of data transmission (not limited by the number of lines of the parallel bus)
- Use FPGA and clock system for sequential logic and timing applications (SVCU)
  - Use of VHDL language for logic description
- Choose components with extended life on the market (MUX + SVCU)
- Improve I/Os overcurrent, overvoltage and ESD (Electrostatic Discharge) protection (MUX + SVCU)

Communication Protocols – Parallel Vs SPI

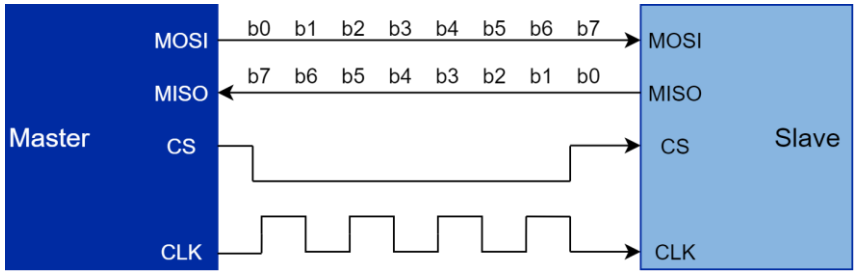
Parallel Communication

- **Length of message depends on number of wires**
  - Limited by the protocol and number of lines
- **Sensitive to noise/glitches**



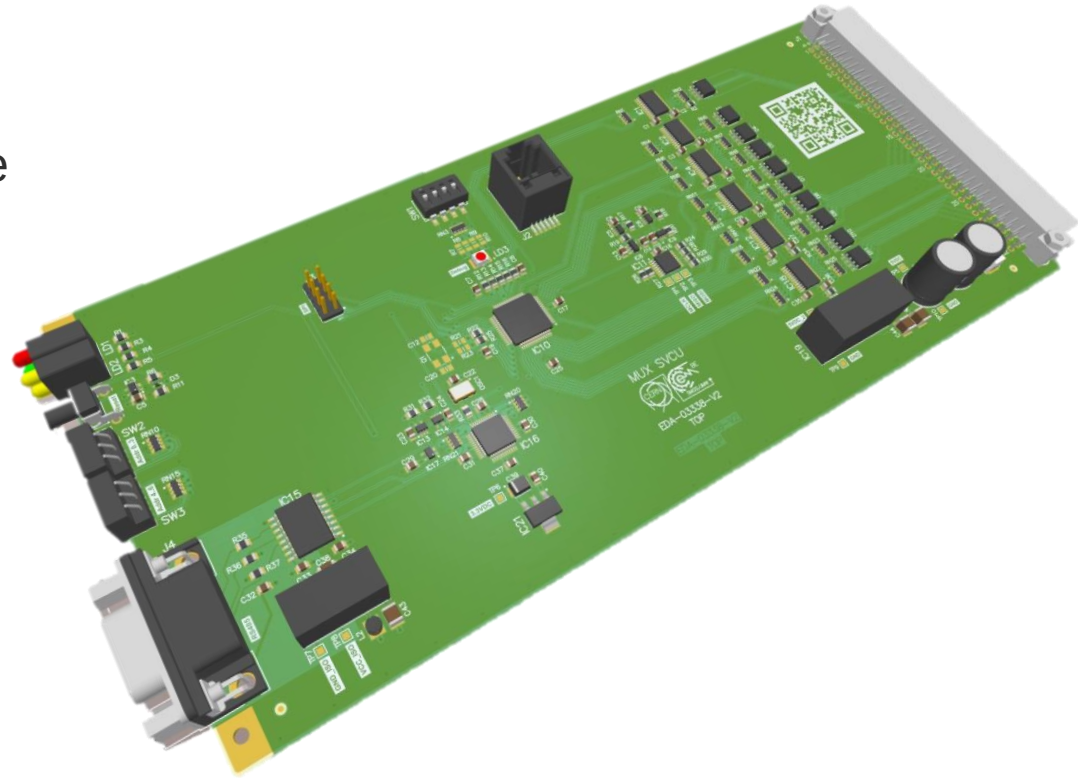
SPI communication

- **Arbitrary length of message**
  - Not limited by the number of lines
- **Faster transmission rate**
- **More robust against noise/glitches**



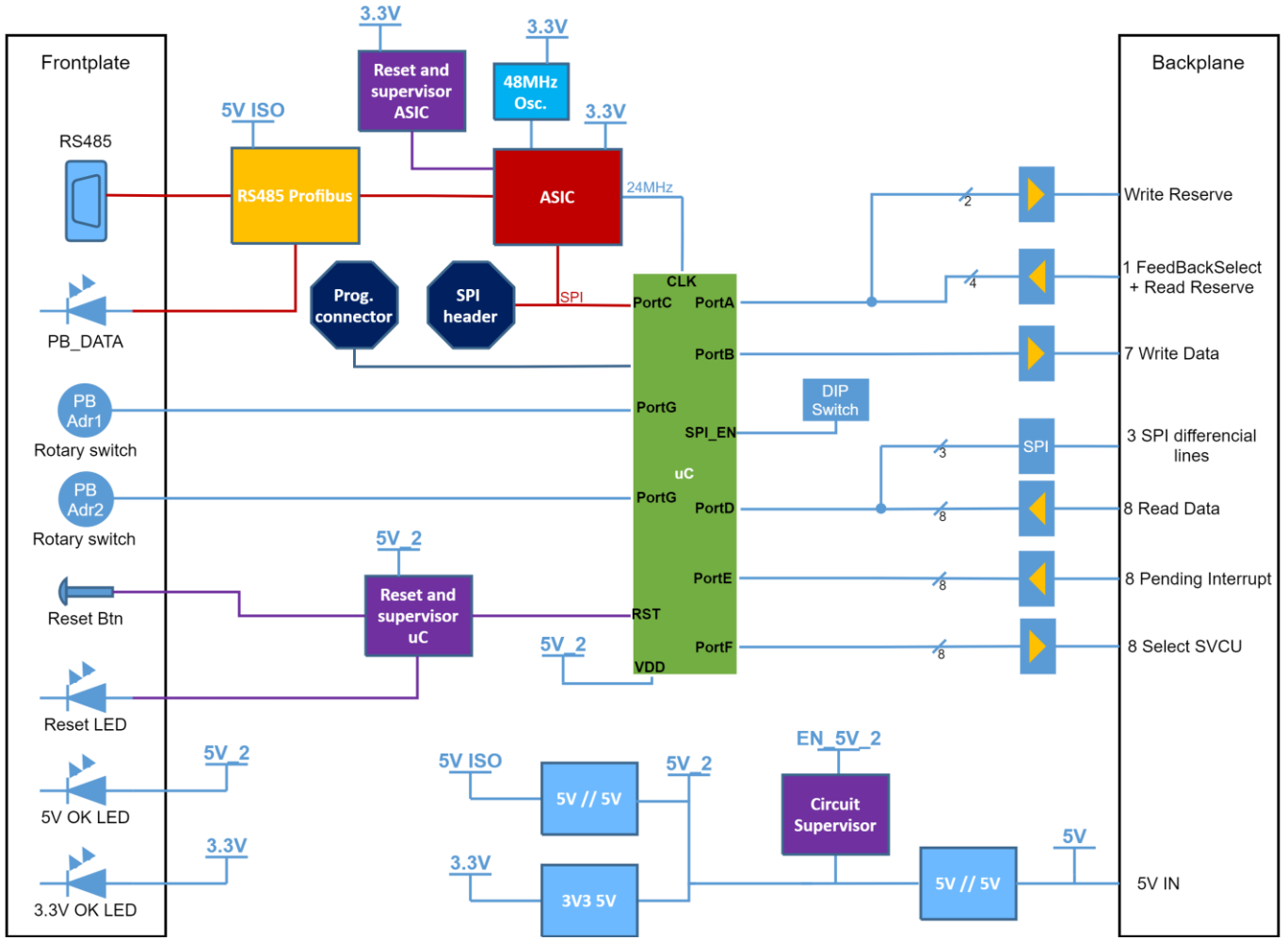
MUX card design

- Prototype of MUX card in production
  - New components
    - New buffers with output enable
    - New voltage supervisors
  - Improved protection against surges of current and voltage
    - TVS (Transient Voltage Suppression) diode arrays added to all backplane connection lines
  - SPI hardware added
    - Differential buffer with output enable



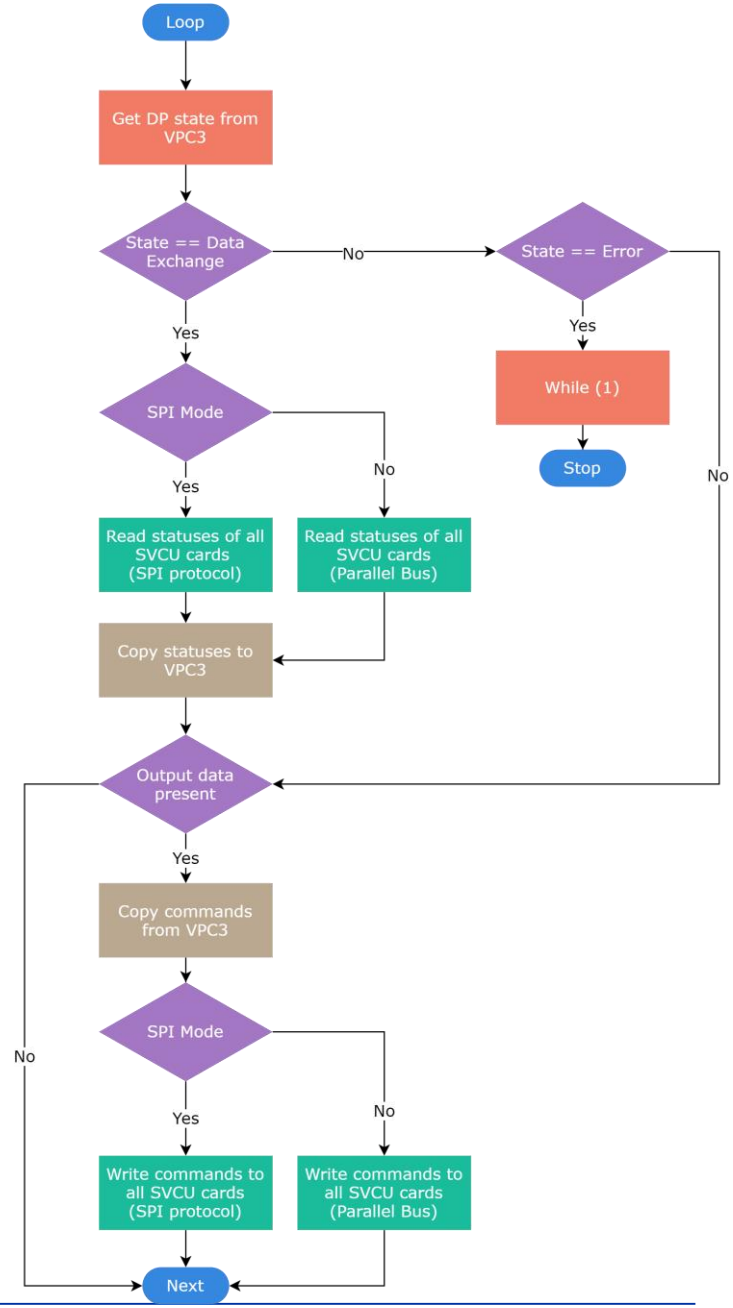
MUX card design - Hardware

- PIC18F6527 microcontroller
  - 8-bit microcontroller
  - SPI compatible
- Profibus slave ASIC VPC3+S
  - Provides 24 MHz clock to the microcontroller

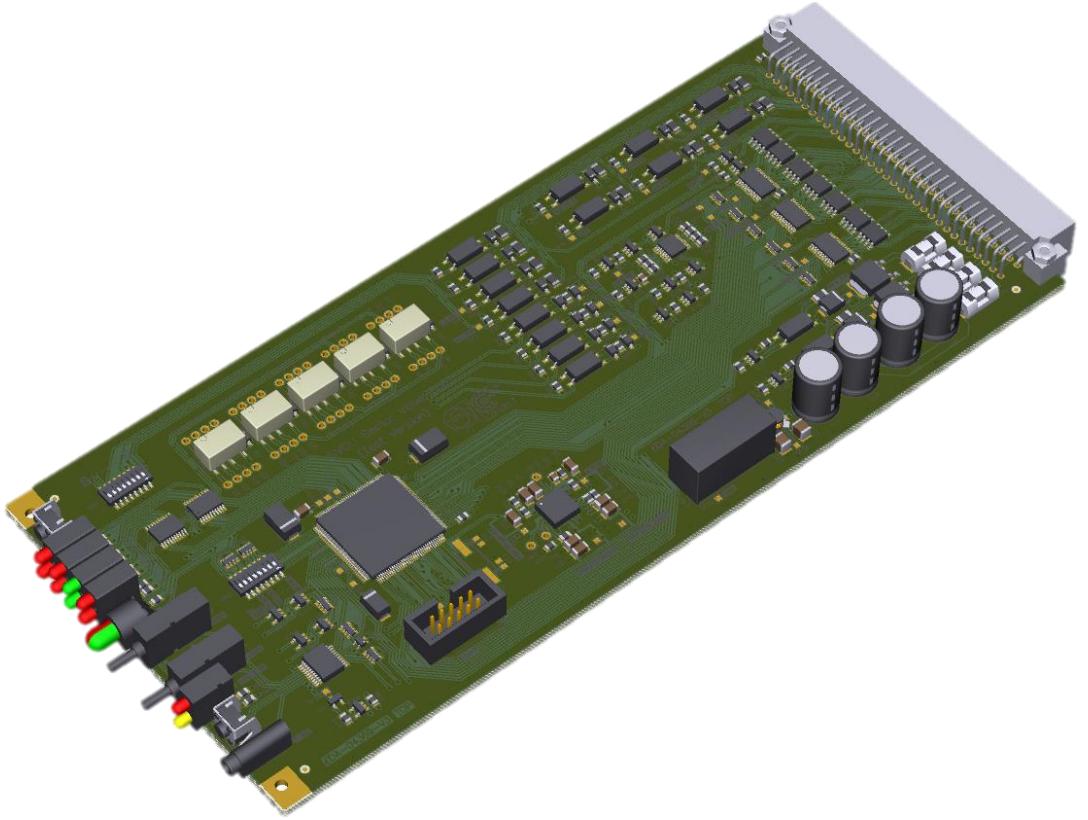


MUX card design - Firmware

- Programmed using C
- Program in a loop
- Checks state of the ASIC chip
- Depending on the Communication mode, it follows the SPI protocol or the Parallel bus protocol
- If program detects an error, it enters an infinite loop, and stops after the WatchDog timer is triggered



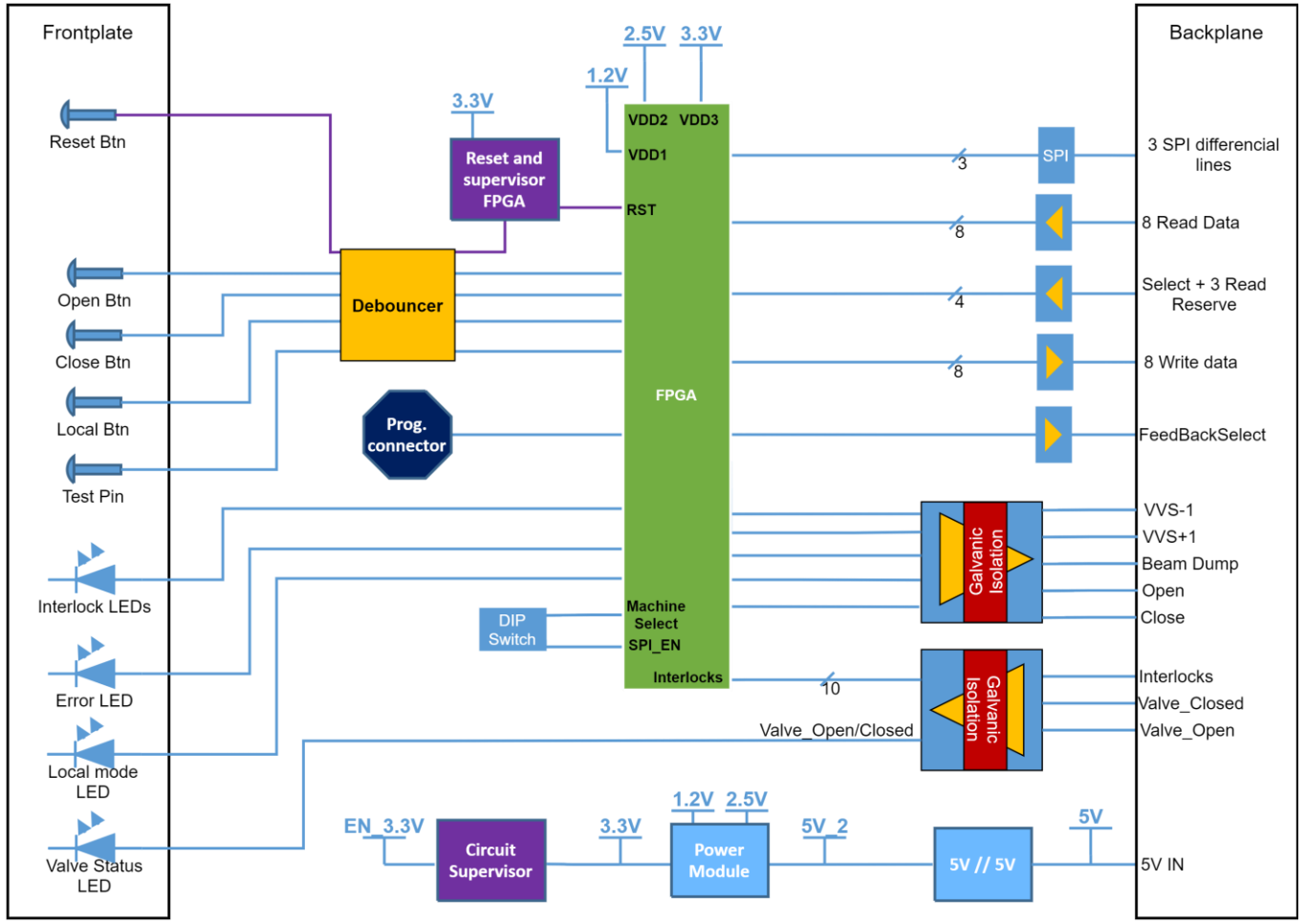
SVCU card design



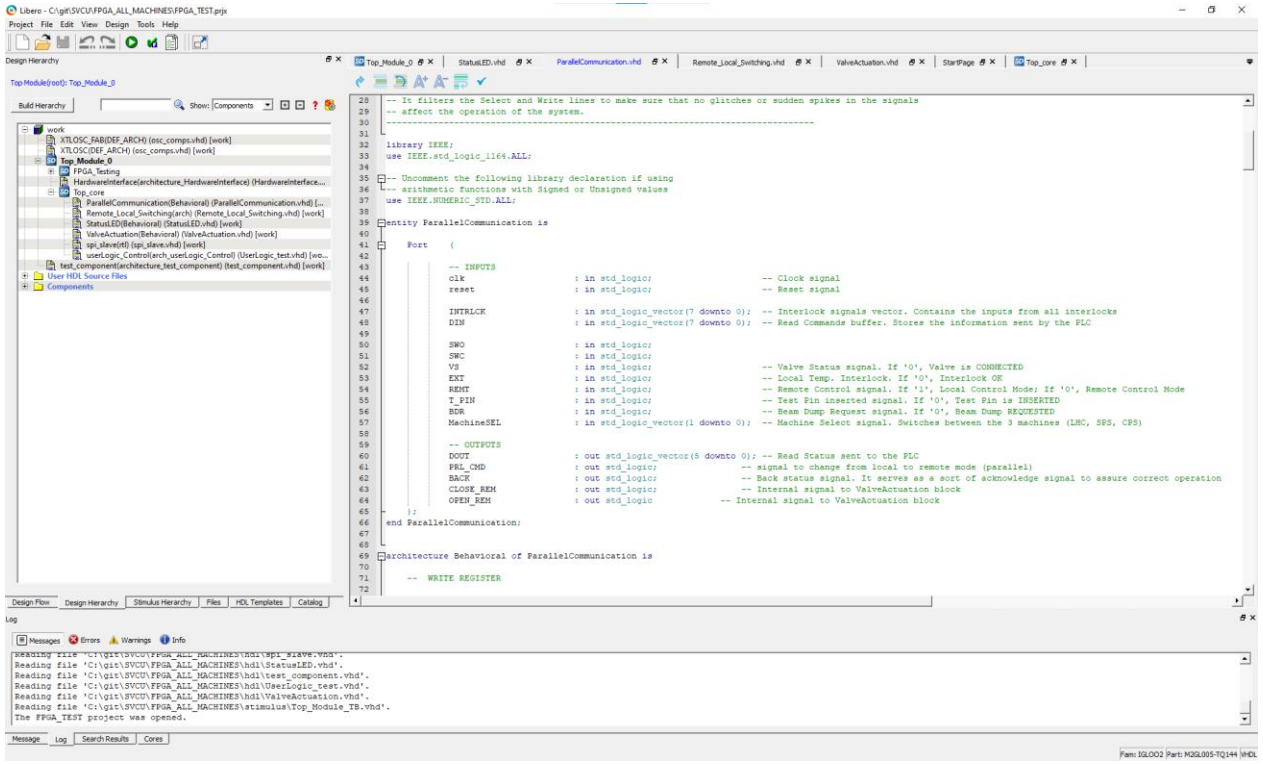
- Prototype in production
  - New components
    - New buffers with output enable
    - New voltage supervisors
  - SPI hardware added
    - Differential buffer with output enable
  - Added protection against surges of current and voltage
    - TVS diode arrays added to all backplane connection lines
  - Optimized circuit logic and behavior
    - Optimized logic for enabling of SPI/PRL components
    - Added pull-up/down resistors to fix signals at start-up to avoid errors

SVCU card design - Hardware

- IGLOO2 FPGA
  - On-chip 1 MHz oscillator
  - 60 dedicated I/O pins
  - 84 user I/O pins
- Galvanic isolation
  - Optocouplers for interlocks and valve actuation
  - Relays for Beam Dump and Valve status
- Dip Switch to control SPI or Parallel mode
- Dip Switch to select which Machine it operates in



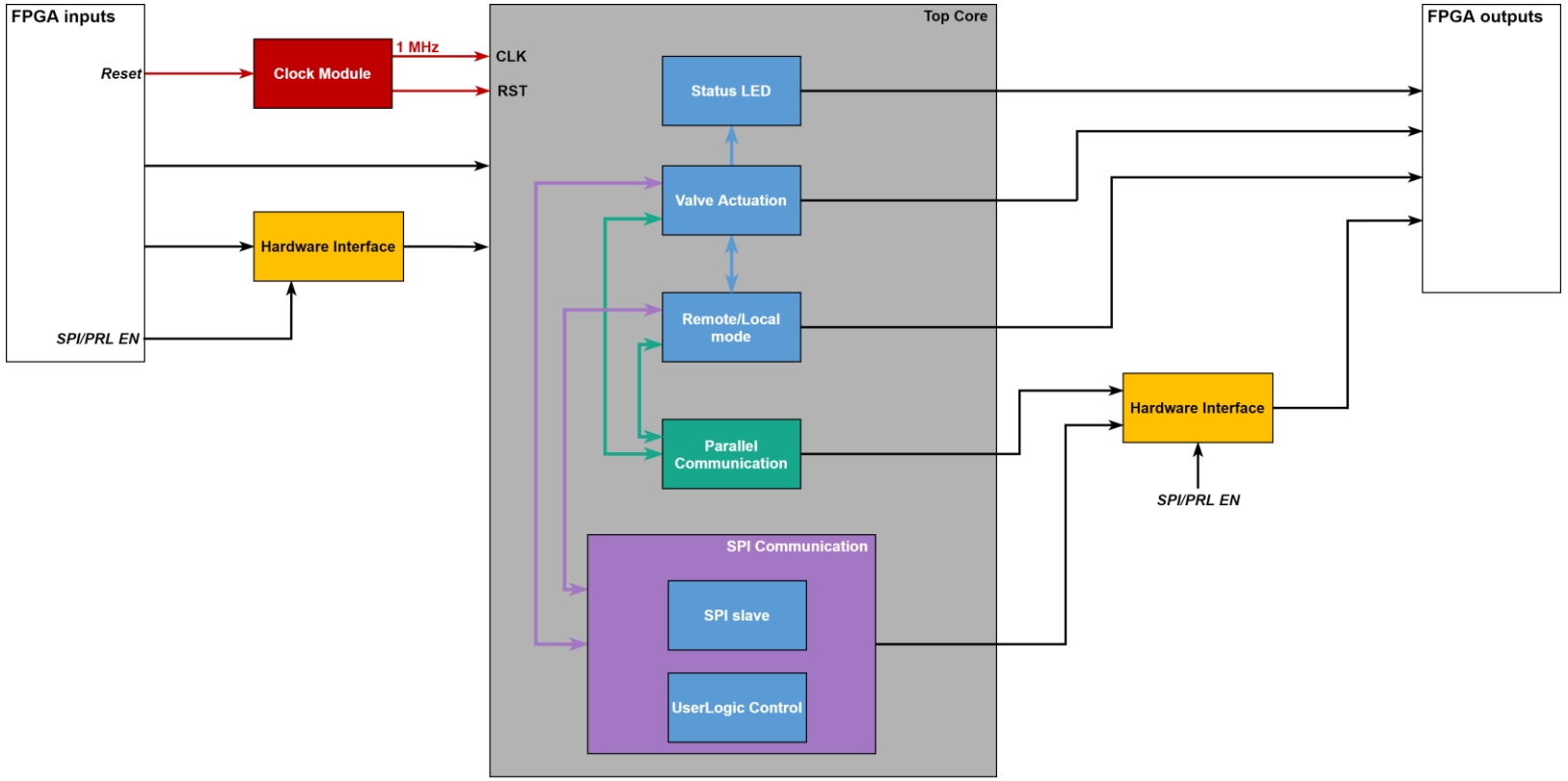
Gateware – VHDL for SVCU card



- Programmed using VHDL (**V**ery **H**igh-Speed Integrated Circuit **H**ardware **D**escription **L**anguage)
- Microsemi (Microchip) software used – Libero IDE (**I**ntegrated **D**evelopment **E**nvironment)
- Supports ModelSim (Simulation Tool)

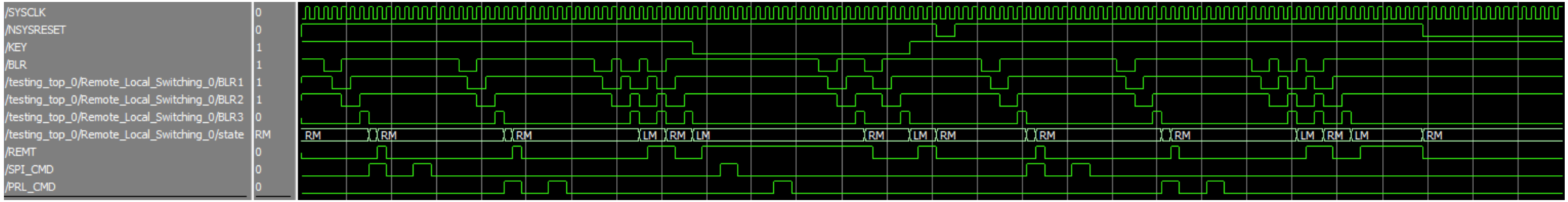
Gateware – VHDL functional blocks

- 5 Function blocks
- 1 MHz clock signal generated internally by the FPGA
- Hardware Interface block to translate signals for the different blocks:
  - Separation of SPI/PRL lines based on communication mode



Validation of the VHDL code and Prototypes

- VHDL code first simulated using ModelSim
- Pre-Synthesis simulation – Validates the logic of the written code
- Post-Synthesis simulation – Code is converted into a logic circuit and simulated. Validates the generated circuit



Validation of the VHDL code and Prototypes

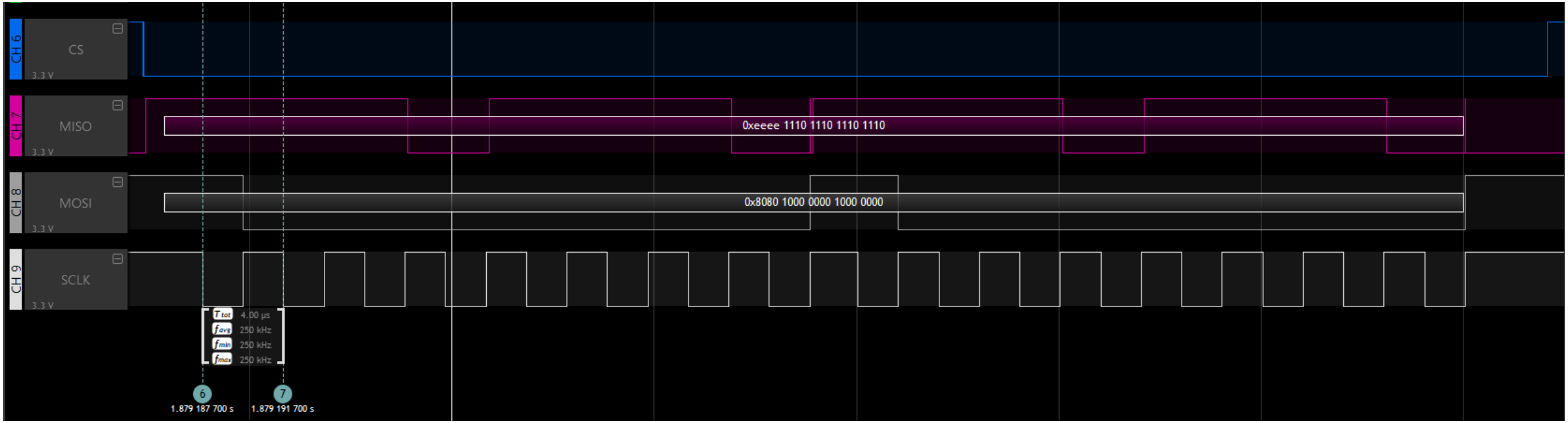
Laboratory testbench to test prototype cards

MUX card used to test Parallel Communication

Arduino used to test SPI communication

SPI Communication - Bus analyzer

- Use of digital bus analyzer to validate SPI messages



Validation of the VHDL code and Prototypes

Laboratory testbench to test the prototype cards

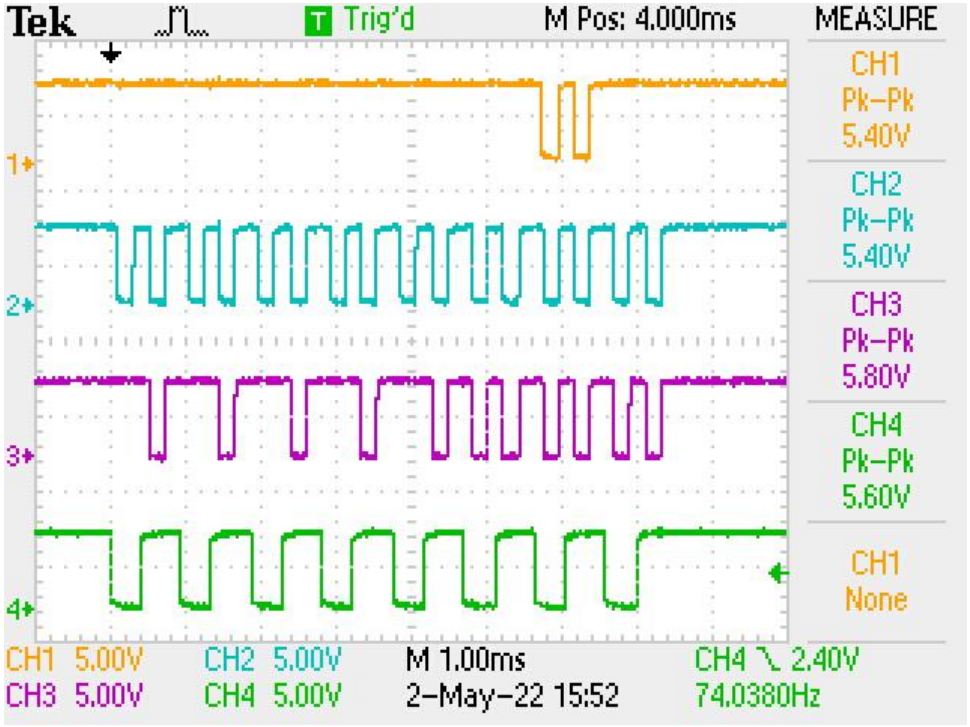
Parallel Communication

Select (card 7)

Back

Status

Page Select

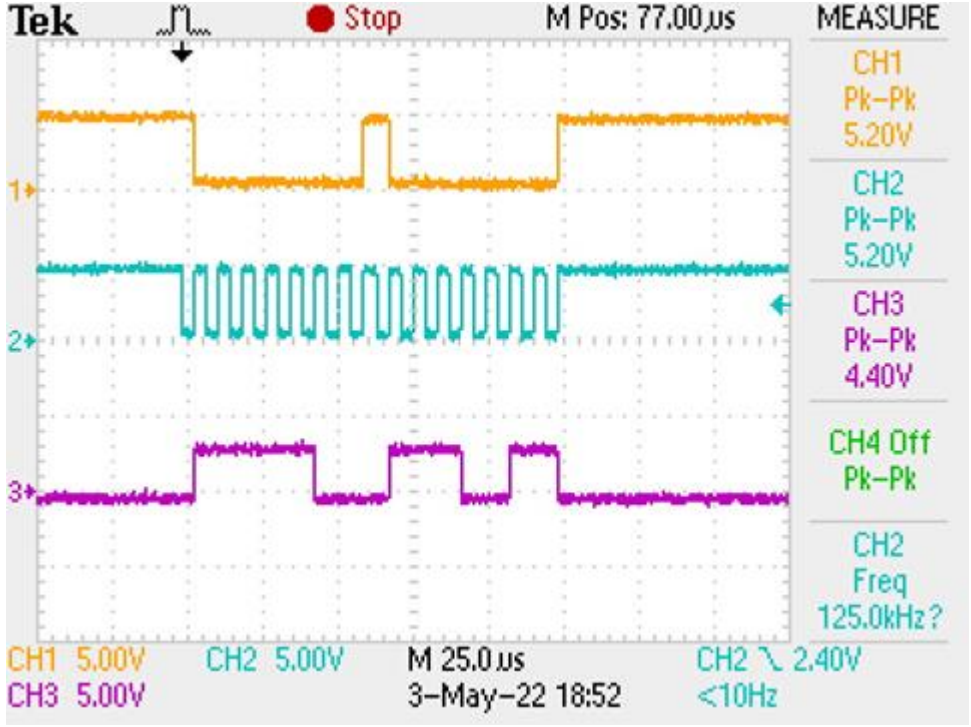


SPI communication

MOSI

Clock

MISO



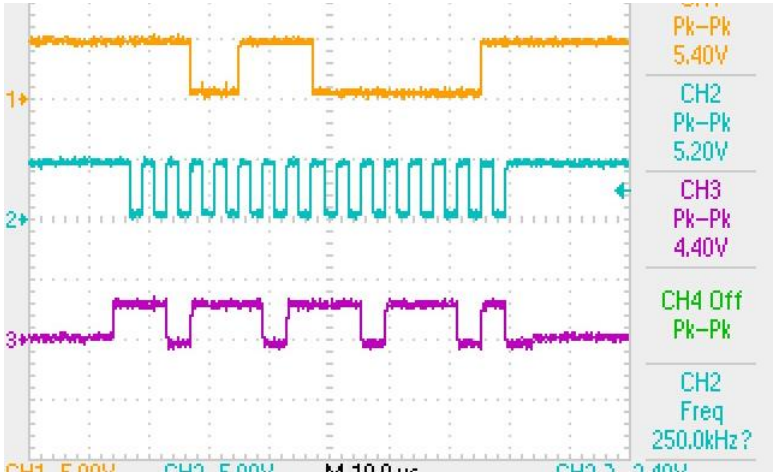
SPI transmission speed

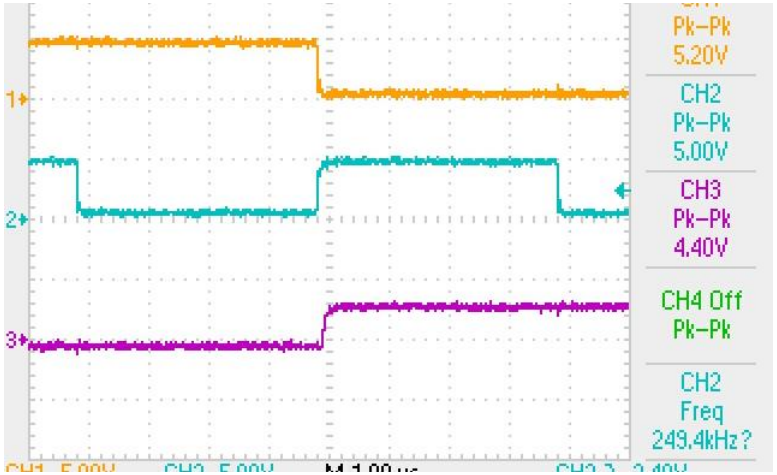
- Max SPI frequency:

$$\frac{SYSCLK}{2} = \frac{1\ MHz}{2} = 500\ kHz$$

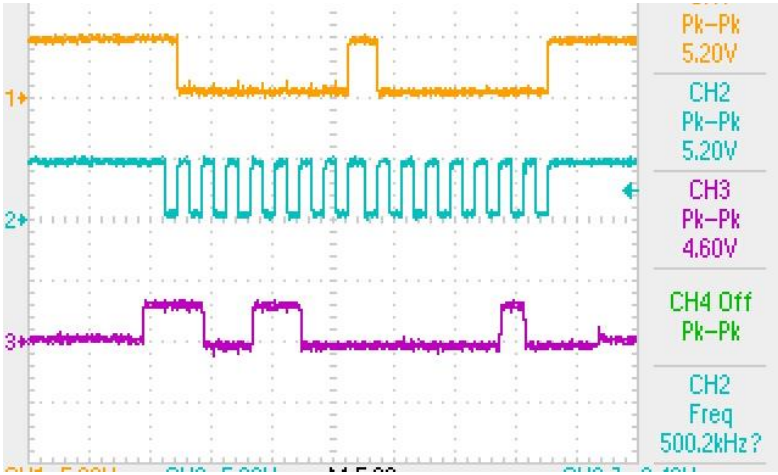
- Higher frequency = More distortion
- 3 Different frequencies tested:
  - 125 kHz
  - 250 kHz
  - 500 kHz – **Max frequency**

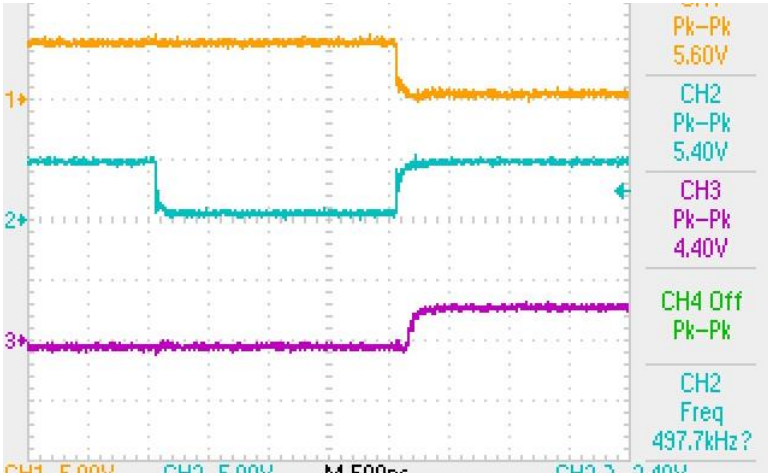
250 kHz





500 kHz

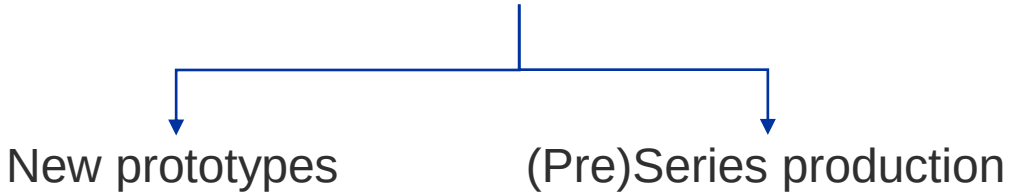




Future Work

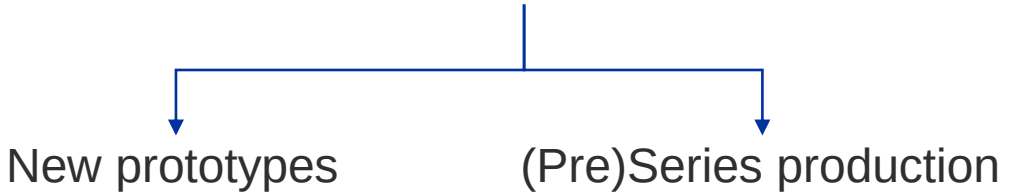
SVCU

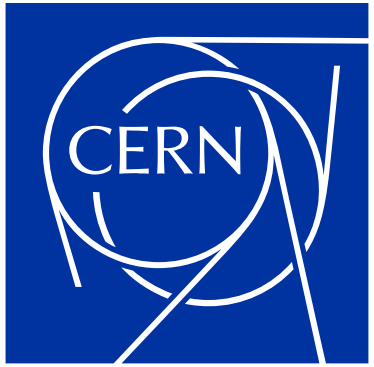
- **Produce prototypes**
- **Review VHDL code and rewrite it to fit new prototype**
- **Test and validate prototypes**



MUX

- **Produce prototypes**
- **Review C code and rewrite it to fit new prototype**
- **Test and validate prototypes**





Thank you!