FPGA Stellwerk und Eulynx

Jöhstadt, 08.09.2021

David Beyeler

Vision meets reality.

Supercomputing Systems AG
Technopark 1
CH-8005 Zürich

Phone +41 43 456 16 00
Fax +41 43 456 16 10
www.scs.ch

super computing systems

Wer ist SCS?

- SCS – Super Computing Systems – ist ein Entwicklungsdienstleister
- ~ 120 IngenieurInnen
- Gegründet 1993
- Arbeitsort: Technopark Zürich
- IPR-Regelung: IPR’s gehen in der Regel zum Kunden
- Aktiv in Industrie, Multimedia, Medizin, Energie, Block-Chain, Automotiv, Ticketing, ÖV, Safety, Security und vielem mehr

Inhalt

1. Vorstellung FPGA Stellwerk
   1. Ausgangslage
   2. Zielsetzung
   3. Lösungskonzept
2. Eulynx Erweiterung
   1. Anbindung der Eulynx Peripherie
   2. Y-Umschaltung für Testbetrieb neuer Technologien
3. Prototyp DRSS
   1. Konzept
   2. Implementation

1.1 Ausgangslage

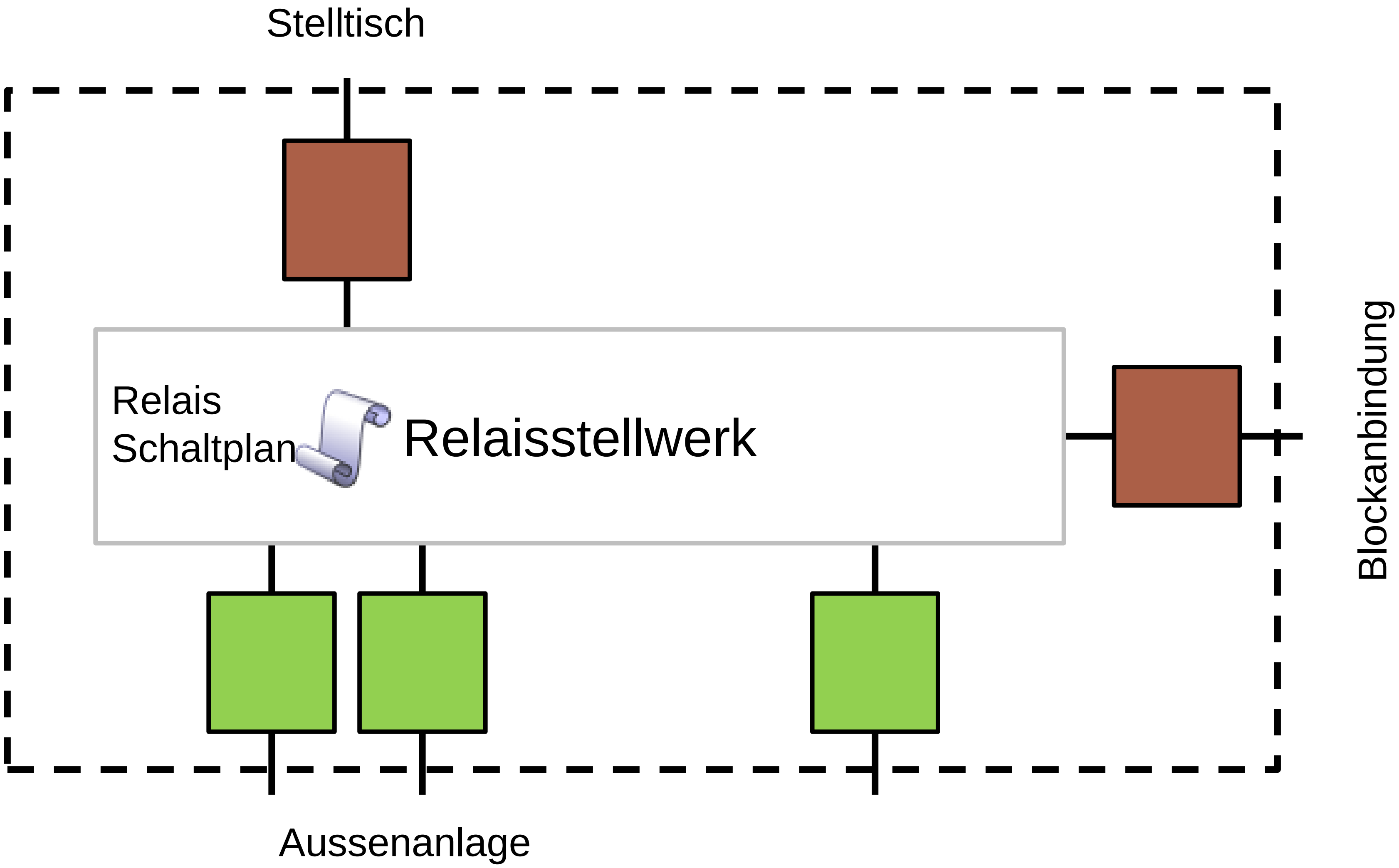
- DB hat einige Relais - Stellwerke vom Typ DrS2 in Betrieb
- Die Sicherheitstechnik hat sich bewährt
- Einige Stellwerke sind älter als 50 Jahre
- Komponenten sind heute teilweise teuer und schwer zu beschaffen
- Die Aussenanlagen können noch weitere Jahrzehnte genutzt werden.
- In Zukunft wird es ein Engpass bei Fahrdienstleitern geben

  → Fernsteuerung und Automatisierung notwendig
- Fernsteuerung/ Automatisierung von DrS2 Stellwerke ist heute aufwendig/ teuer
- Migration von DrS2 auf ETCS ist nicht möglich /sehr aufwendig

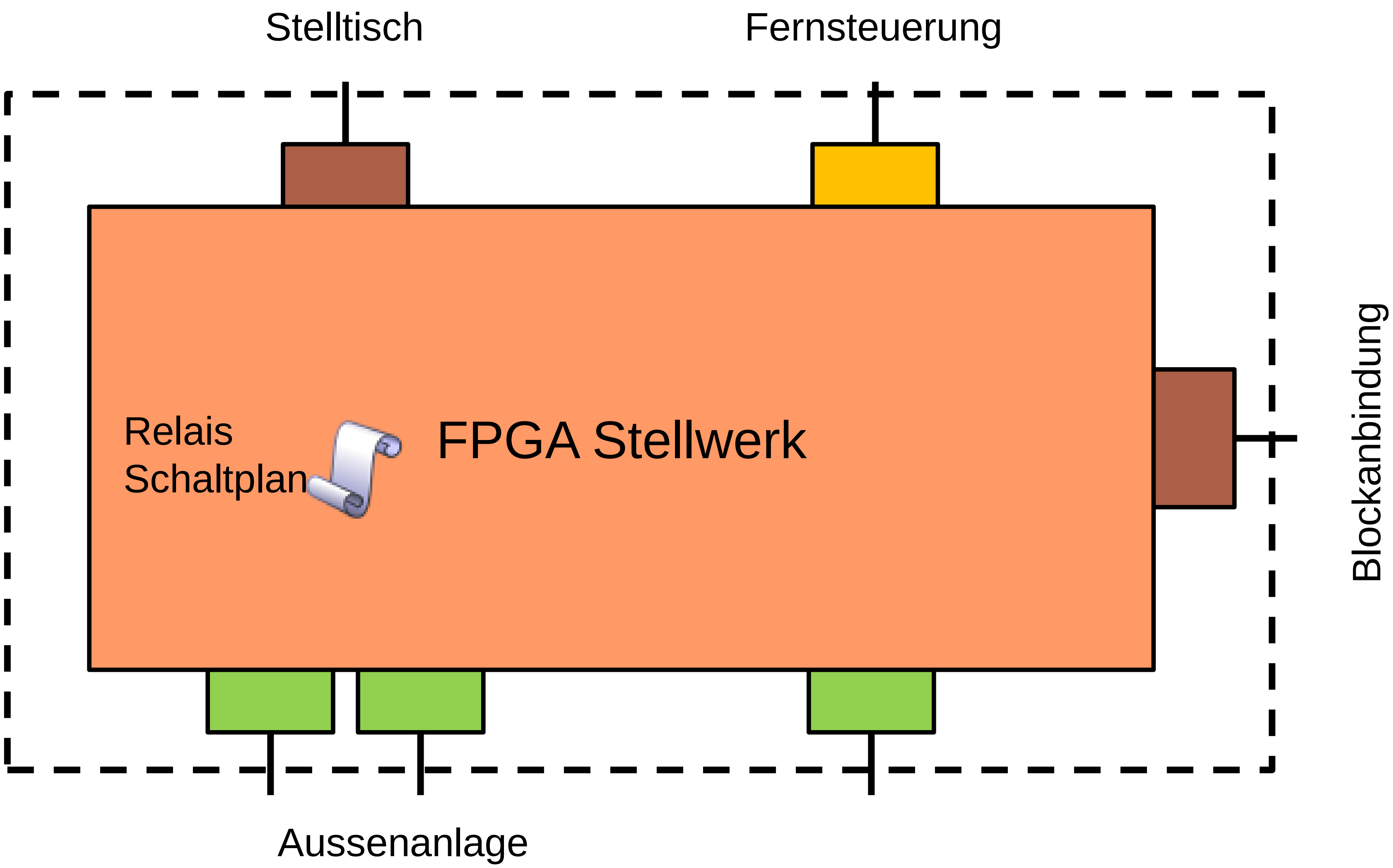
1.2 Zielsetzung FPGA Stellwerk

- Die bewährte Sicherheitstechnik von Relaisstellwerken soll beibehalten werden
- Die Relaistechnik soll durch eine neue zukunftsträchtige elektronische Technologie abgelöst werden
- Die bewährte Sicherheitstechnik soll dabei übernommen werden; die Code - Generierung soll möglichst vollautomatisch ab dem Relaisschaltplan erfolgen → **Vereinfachter Zulassungsprozess** (gegenüber Neubau)
- Die bestehenden Aussenanlagen sollen weiter genutzt werden
- Fernsteuerung/ Automatisierung sollen möglich sein
- Der Umbau soll während laufendem Betrieb möglich sein
- Migration auf ETCS möglich

1.2 Zielsetzung– Ausgangslage Relais Stellwerk



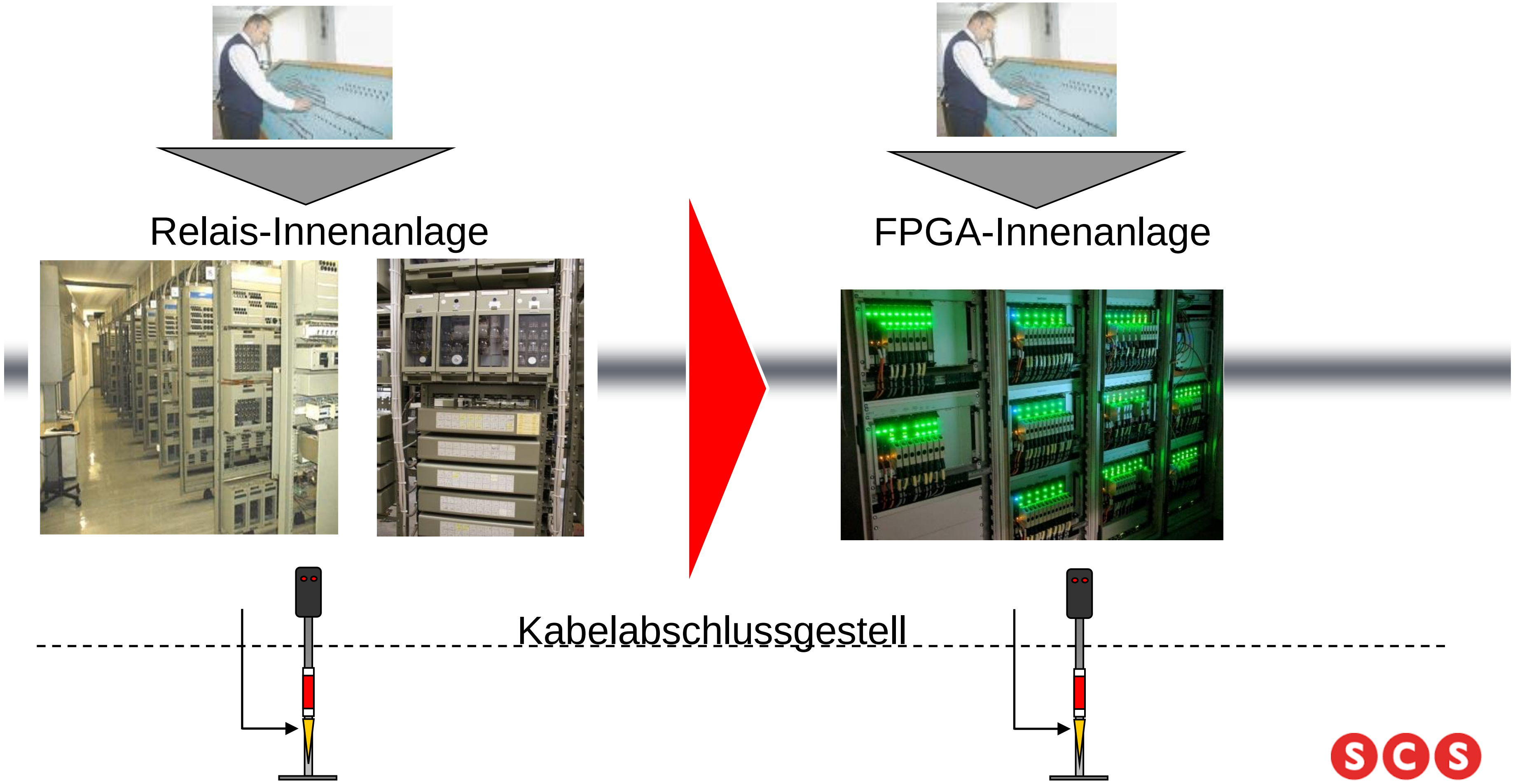
1.2 Zielsetzung– FPGA Stellwerk



1.2 Zielsetzung –
Übergang vom DR S2 zum FPGA-Stellwerk

vor der Teilerneuerung

nach der Teilerneuerung



Relais-Innenanlage

FPGA-Innenanlage

Kabelabschlussgestell

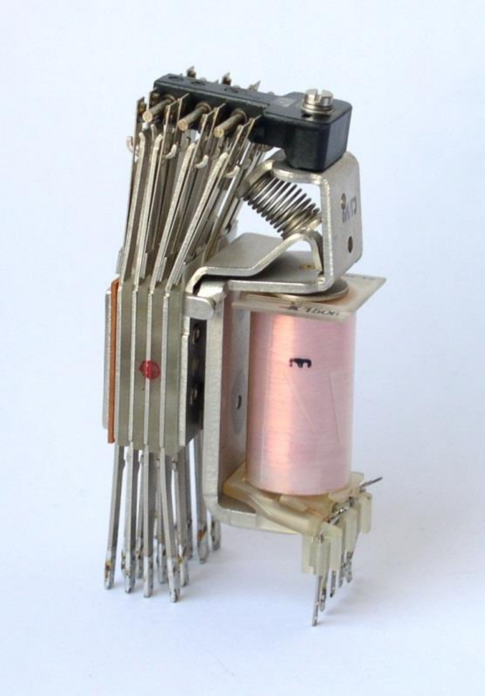
super computing systems

1.3 Lösungskonzept

Relay to Chip – wie funktioniert das?

Die Sicherheit des Relaisstellwerkes basiert auf zwei Grundelementen

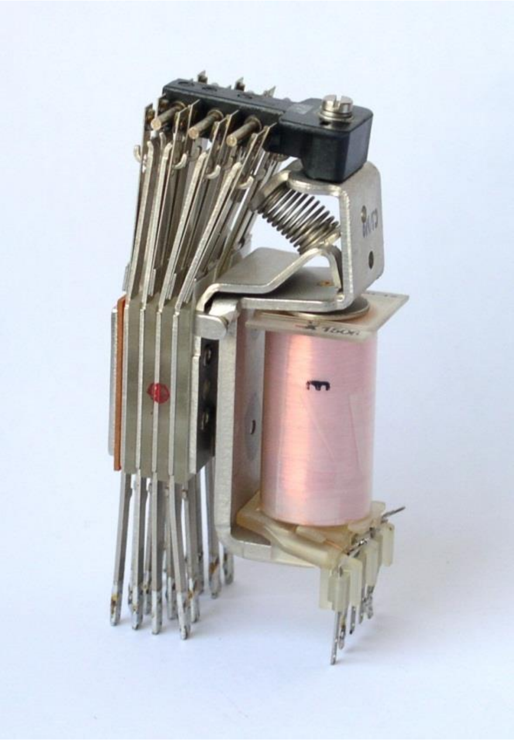
- Signalrelais
  - Unverlierbare Eigenschaften
  - Zwangsführung
- Relaisschaltungstechnik
  - Funktionale Verschaltung
  - Schaltschritte
  - Hemmnisse



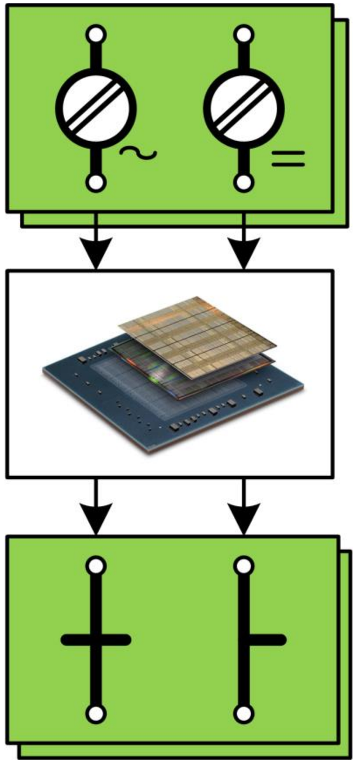
https://de.wikipedia.org/wiki/Signalrelais#/media/Datei:Siemens_K50_Relais.jpg

1.3 Lösungskonzept

Relay to Chip – wie funktioniert das?



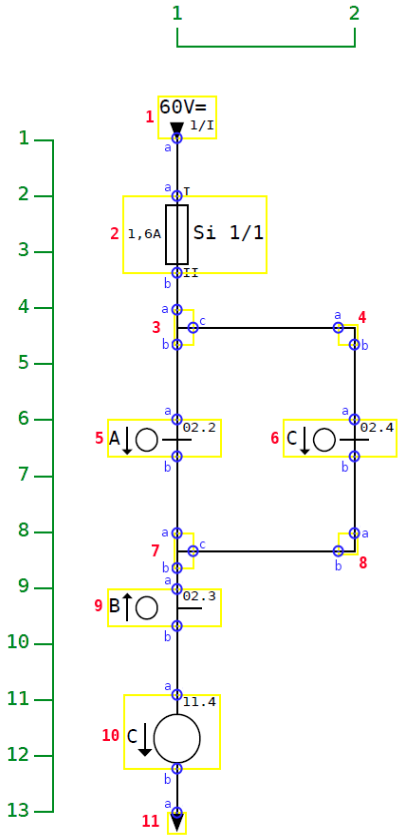
https://de.wikipedia.org/wiki/Signalrelais#/media/Datei:Siemens_K50_Relais.jpg



- Signalrelais
  - Unverlierbare Eigenschaften
  - Zwangsführung
- FPGA Plattform
  - Redundante Strukturen
  - Funktionelle Nachbildung

1.3 Lösungskonzept

Relay to Chip – wie funktioniert das?



1:1

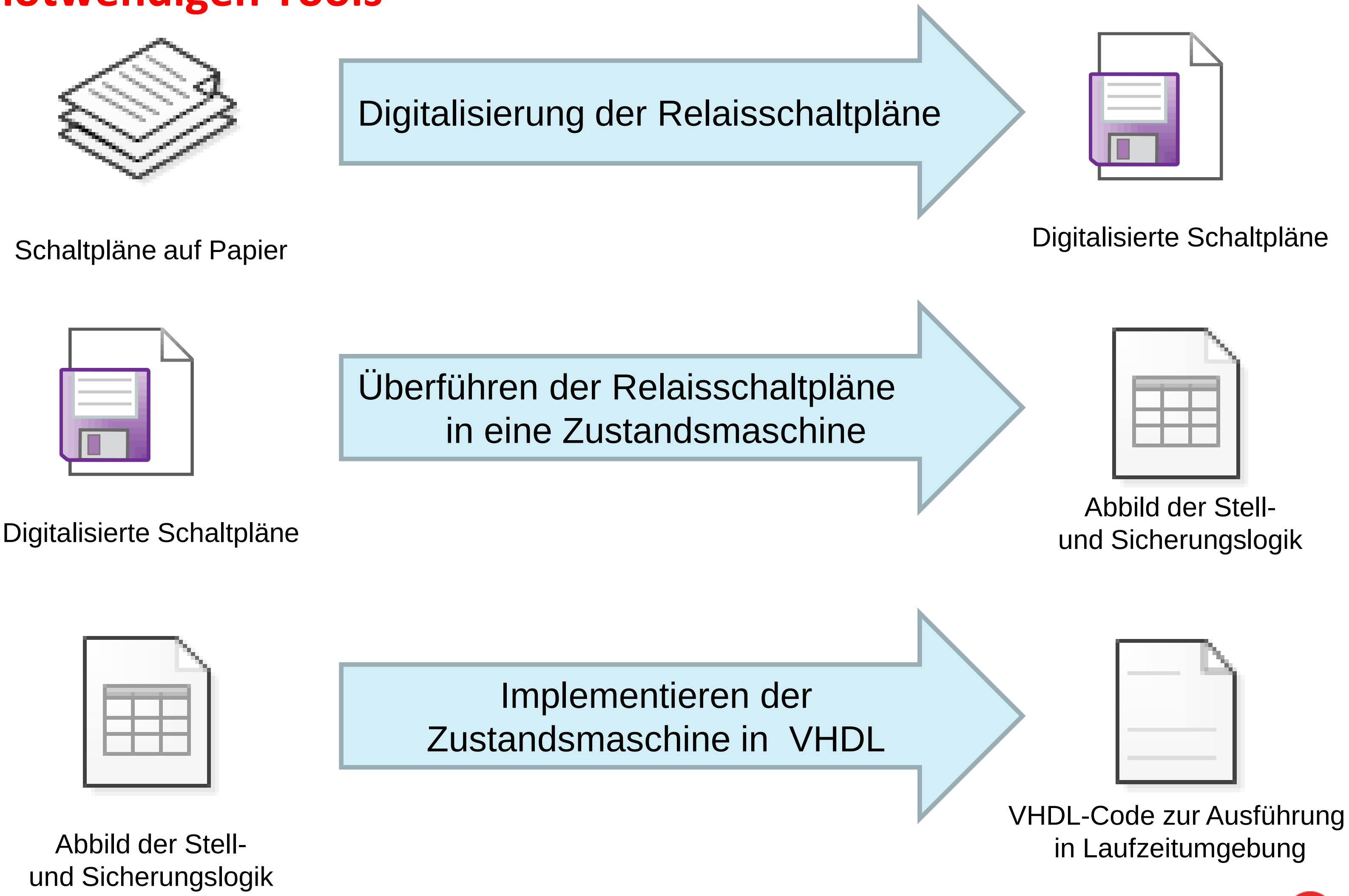
```
-- Copyright (c) 2011 Supercomputing Systems AG, all rights reserved.

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.stw_pkg.all;

entity fahrstrassen_signalrelais is
    generic (
        num_efs : positive
        );
    port (
        -- Eingaenge
        signals_einzelfahrstrassen         : in signals_einzelfahrstrasse_signalrelais_vector(num_efs-1 downto 0);
        weichenueberwacher_gfs_dauernd     : in std_logic_vector(max_weichenueberwacher-1 downto 0);  -- Falls nicht verwendet: setze konstant '1'
        weichenueberwacher_gfs_einmalig    : in std_logic_vector(max_weichenueberwacher-1 downto 0);  -- Falls nicht verwendet: setze konstant '1'
        start_ziel_gleisfreimelder_gfs     : in std_logic;  -- Gleisfreimelder des Startabschnitts bei Einfahrt, bzw. Zielabschnitts bei Ausfahrt
        fst_ruecknahme_gruppentaste        : in std_logic;  -- Falls nicht verwendet: setze konstant '0'
        tastenpruefrelais_1                : in std_logic;  -- Falls nicht verwendet: setze konstant '0'
        tastenpruefrelais_2                : in std_logic;  -- Falls nicht verwendet: setze konstant '0'
        taste_gesamtfahrstrasse            : in std_logic;
        schluesselsperren_gfs              : in std_logic_vector(max_schluesselsperren-1 downto 0);
        aufloeserelais                     : in std_logic;
        blockpruefer_1_wirkwicklung        : in std_logic;  --(Falls nicht verwendet setze konstant '0')
        blockpruefer_2_rueckstellwicklung : in std_logic;  -- (Falls nicht verwendet setze konstant '1')
        block_gleisfreimeldehilfsrelais    : in std_logic;  -- (Falls nicht verwendet setze konstant '1')
        fst_festleger_wirkwicklung         : in std_logic_vector(num_efs-1 downto 0);

        -- Zustaende aktuell
        efs_signalrelais_p : in std_logic_vector(num_efs-1 downto 0);
        gfs_signalrelais_p : in std_logic;

        -- Zustaende zukünftig
        efs_signalrelais_n : out std_logic_vector(num_efs-1 downto 0);
        gfs_signalrelais_n : out std_logic
        );

end fahrstrassen_signalrelais;

architecture relais_logik of fahrstrassen_signalrelais is
begin

    fst_signalrelais : process(aufloeserelais, block_gleisfreimeldehilfsrelais,
                               blockpruefer_1_wirkwicklung,
                               blockpruefer_2_rueckstellwicklung,
                               efs_signalrelais_p, fst_festleger_wirkwicklung,
                               fst_ruecknahme_gruppentaste, gfs_signalrelais_p,
                               schluesselsperren_gfs,
                               signals_einzelfahrstrassen,
                               start_ziel_gleisfreimelder_gfs,
                               taste_gesamtfahrstrasse, tastenpruefrelais_1,
                               tastenpruefrelais_2,
                               weichenueberwacher_gfs_dauernd,
                               weichenueberwacher_gfs_einmalig)
        variable a, b, c, d, e, f, g, m, n, o, u            : std_logic_vector(num_efs-1 downto 0);
        variable h, i, j, k, l, p, q, r, s, t, v, w, x, y, z : std_logic;
    begin
        for index in num_efs-1 downto 0 loop
            a(index) := signals_einzelfahrstrassen(index).taste_einzelfahrstrasse;
            b(index) := signals_einzelfahrstrassen(index).start_ziel_gleisfreimelder_efs;
            c(index) := '0';
            if signals_einzelfahrstrassen(index).weichenueberwacher_einmalig = (max_weichenueberwacher-1 downto 0 => '1') then
                c(index) := '1';
            end if;
            d(index) := '0';
            if signals_einzelfahrstrassen(index).rot2ueberwacher = (max_rot2ueberwacher-1 downto 0 => '1') then
```

- Relaisschaltungstechnik
  - Funktionale Verschaltung
  - Schaltschritte
  - Hemmnisse

- VHDL Code
  - Funktionale Verschaltung
  - Schaltschritte
  - Hemmnisse

1.3 Lösungskonzept –

Skizze der notwendigen Tools



Schaltpläne auf Papier

Digitalisierung der Relaisschaltpläne

Digitalisierte Schaltpläne

Digitalisierte Schaltpläne

Überführen der Relaisschaltpläne in eine Zustandsmaschine

Abbild der Stell- und Sicherungslogik

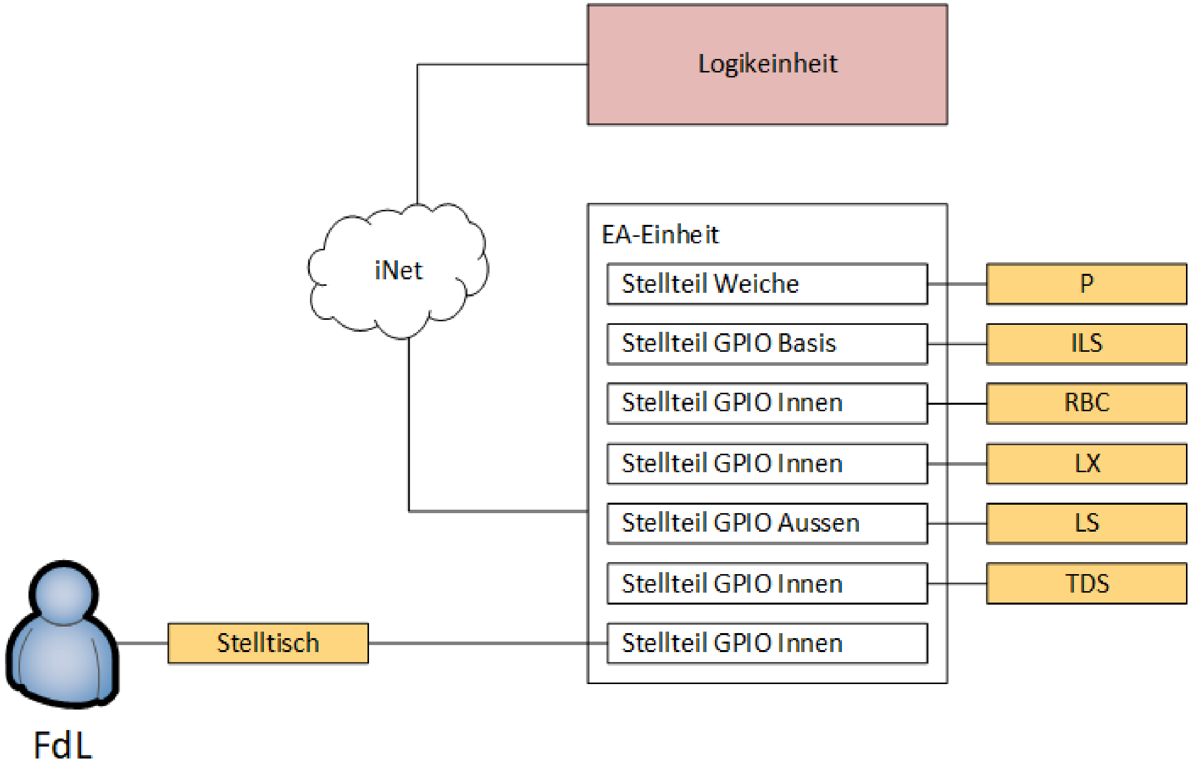
Abbild der Stell- und Sicherungslogik

Implementieren der Zustandsmaschine in VHDL

VHDL-Code zur Ausführung in Laufzeitumgebung

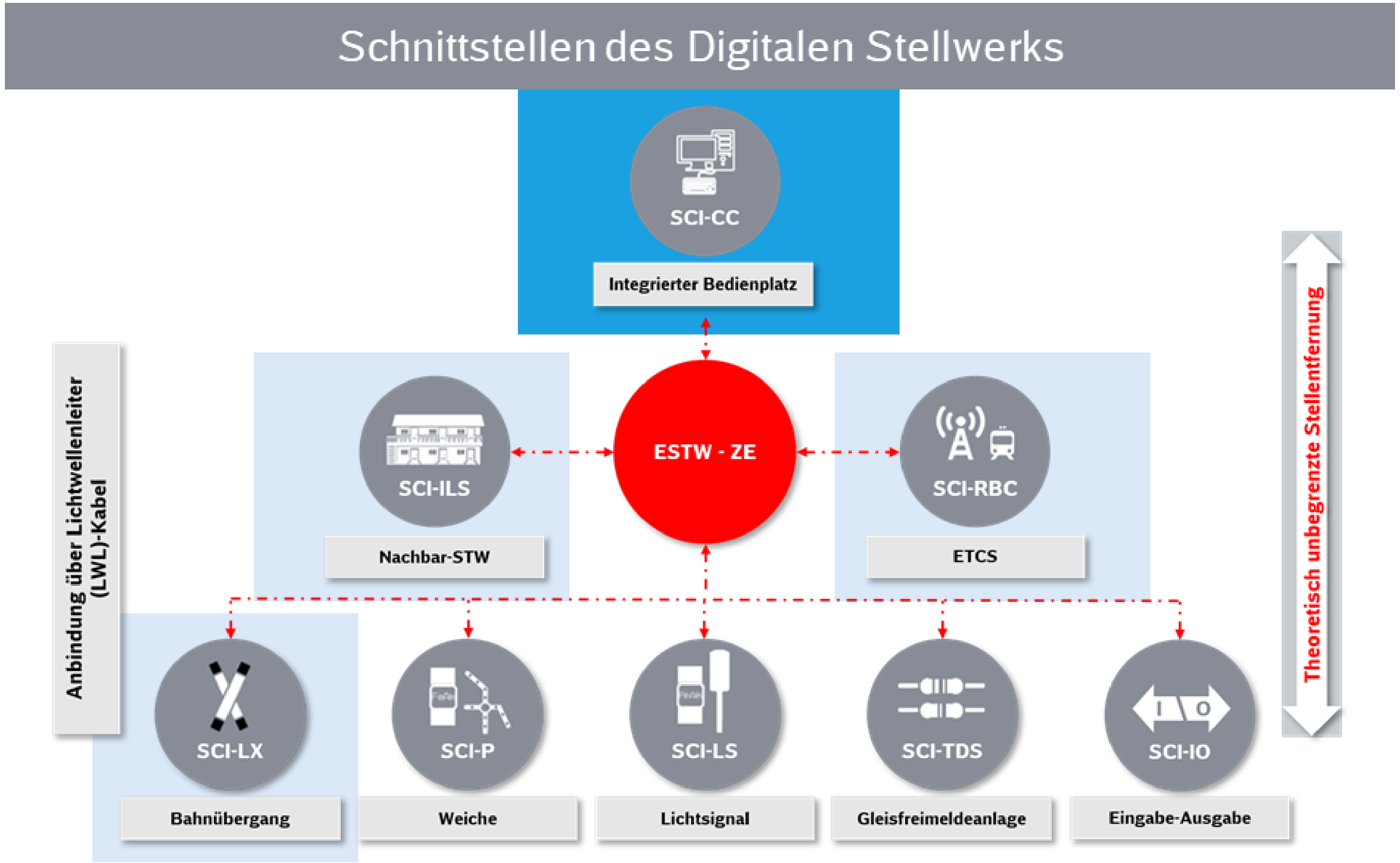
1.3 Lösungskonzept –
FPGA Stellwerk und Schnittstellen

- Stelltisch wird vor Ort durch FdL bedient
- Stell- und Sicherungslogik wird auf LE ausgeführt
- Anbindung von Innen- und Aussenanlage durch Stellteile
- Anbindung von Block, BÜSA, … durch Stellteile
- Stellteile basieren auf Baukasten:
  - Schalter
  - Spannungsdetektion
  - Strommessung
  - Kommunikationsanbindung



2.1 Eulynx Erweiterung –
Ausgangslage

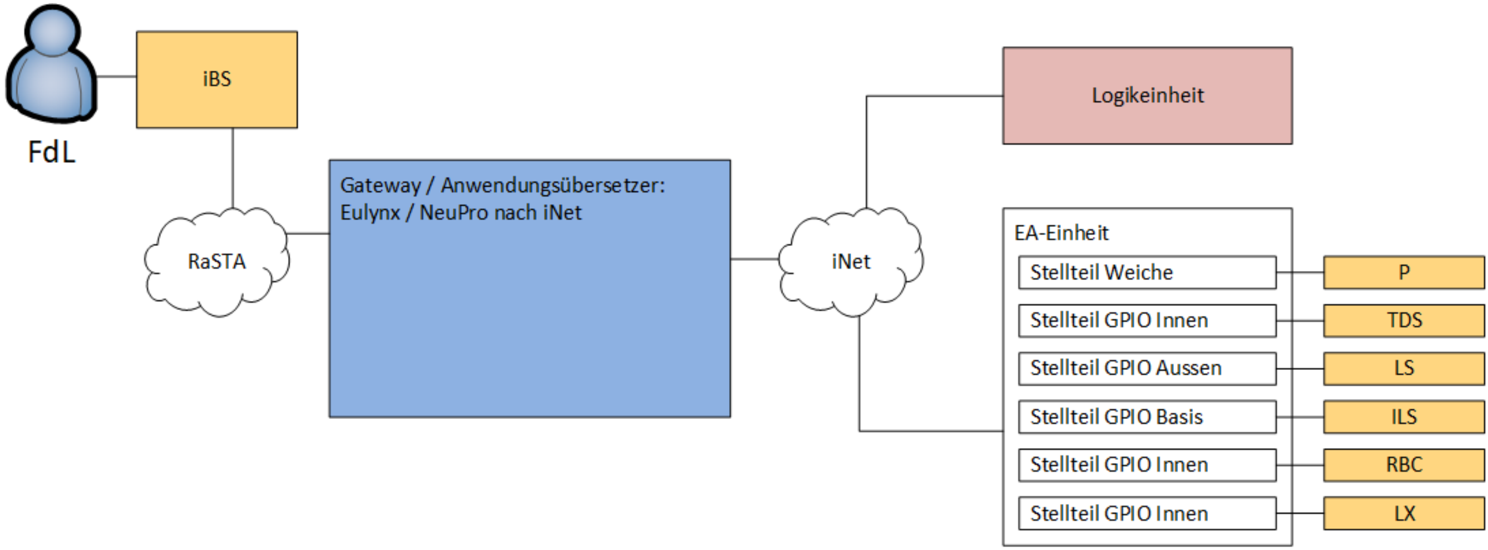
- FPGA-Stellwerk ersetzt Innenanlage von Relais Stellwerken
- Eulynx definiert Schnittstellen von DSTW (nicht abschliessend):
  - Anbindung Block -> SCI-ILS
  - Anbindung BÜSA -> SCI-LX
  - Anbindung ETCS -> SCI-RBC
  - Integrierter Bedienplatz -> SCI-CC



2.1.1 Eulynx Erweiterung –

Anbindung der Eulynx Peripherie – Bedienschnittstelle

- Kommandos von Standard-Bedienschnittstelle über RaSTA/Eulynx an Gateway/Anwendungsübersetzer danach an FPGA Stellwerk
- Stell- und Sicherungslogik wird auf Logikeinheit implementiert
- Ansteuerung der Innen- und Aussenanlage wird durch FPGA Stellwerk ausgeführt



FdL

iBS

RaSTA

Gateway / Anwendungsübersetzer:
Eulynx / NeuPro nach iNet

iNet

Logikeinheit

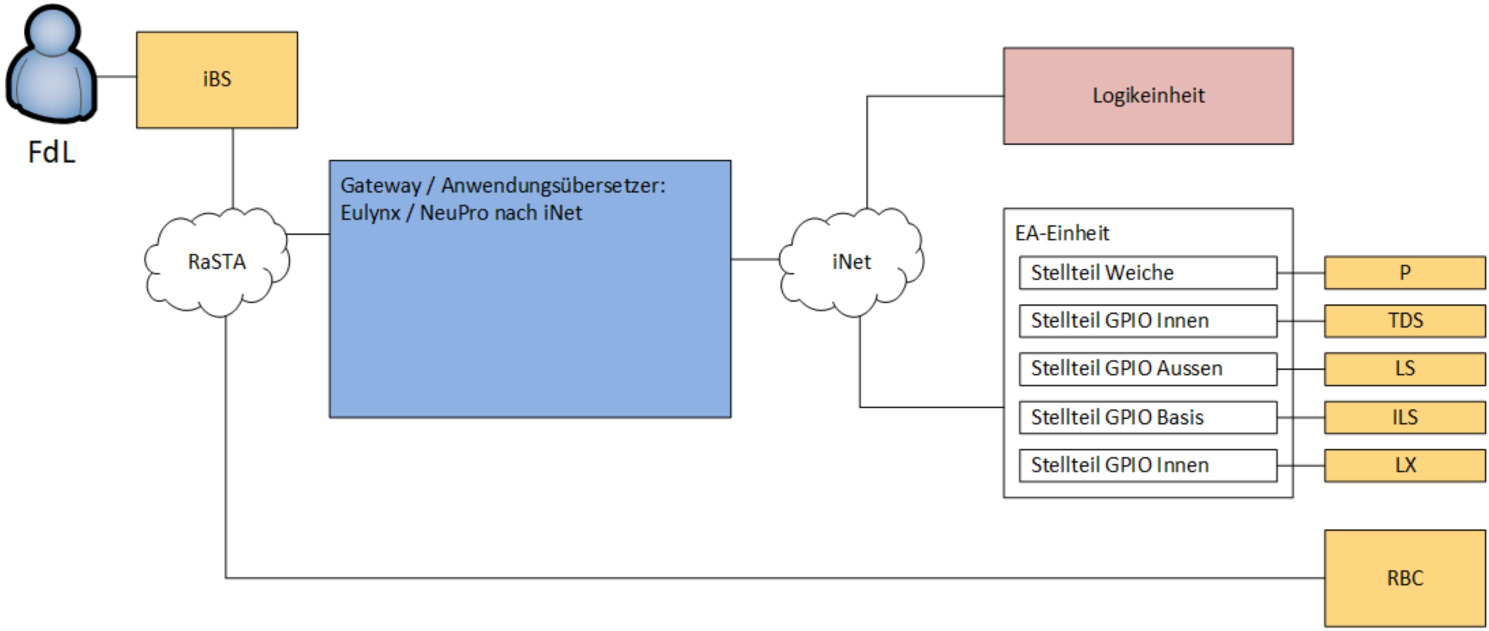
EA-Einheit

Stellteil Weiche	P
Stellteil GPIO Innen	TDS
Stellteil GPIO Aussen	LS
Stellteil GPIO Basis	ILS
Stellteil GPIO Innen	RBC
Stellteil GPIO Innen	LX

2.1.2 Eulynx Erweiterung –

Anbindung der Eulynx Peripherie – ETCS

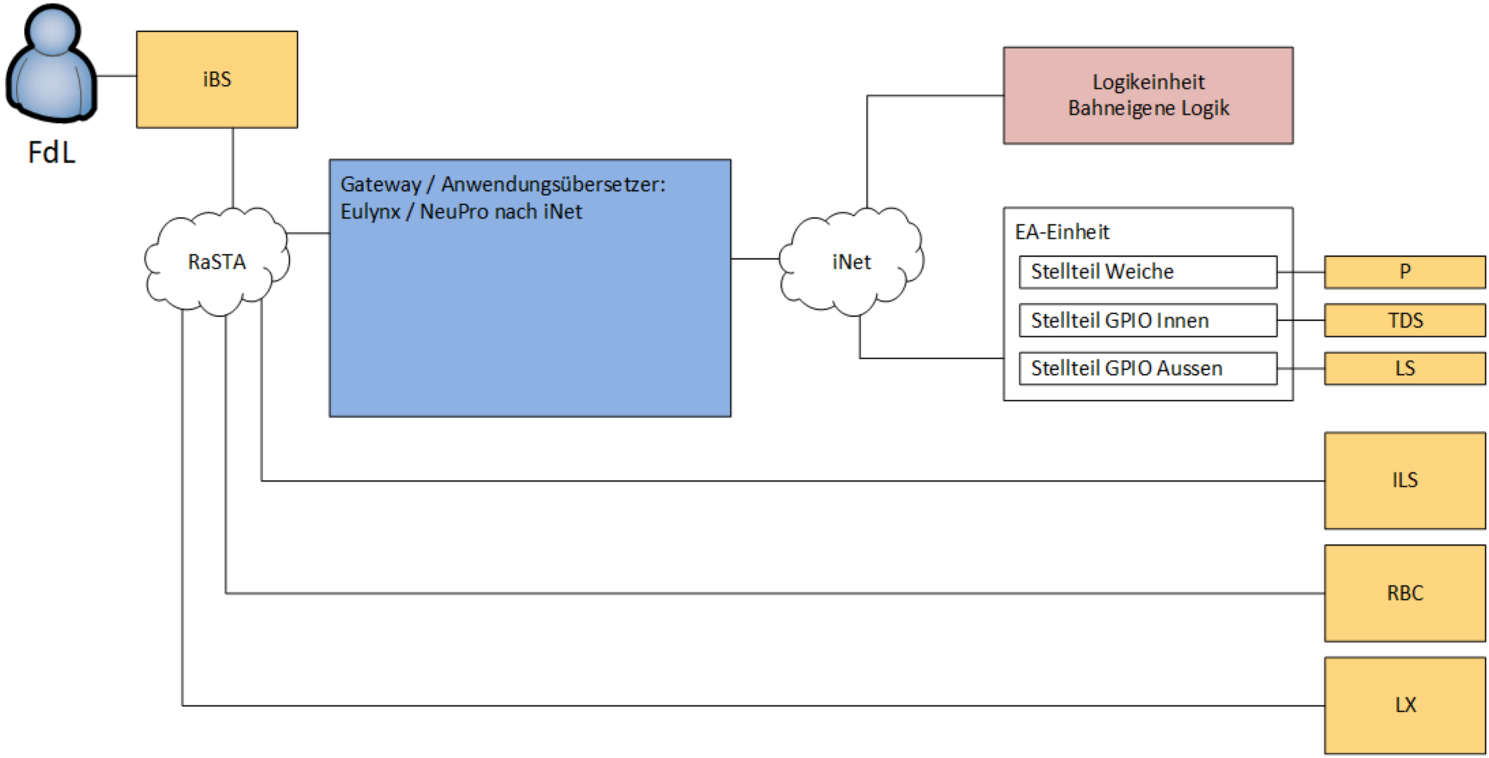
- Stell- und Sicherungslogik wird auf Logikeinheit implementiert
- Ansteuerung der Innen- und Aussenanlage wird durch FPGA Stellwerk ausgeführt
- ETCS Kommandos werden über Gateway/Anwendungsübersetzer mit RaSTA/Eulynx an RBC gesendet



2.1.3 Eulynx Erweiterung –

Unterschiedliche Begrifflichkeiten

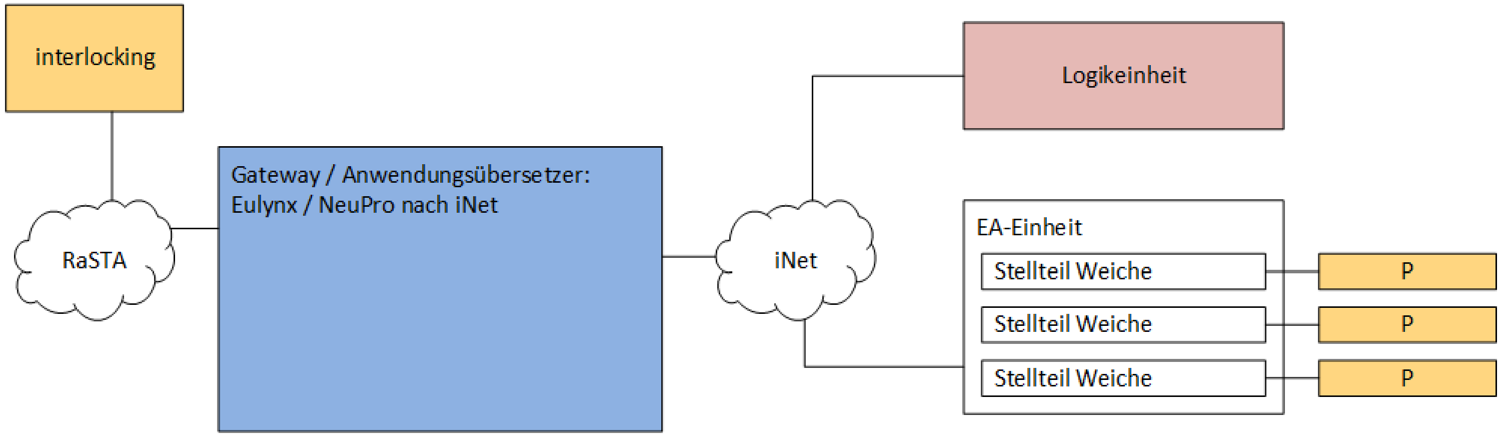
- Die Logische Eulynx Information wird vom Gateway in «Alt-Technik» (z.B. DrS 2-Kontaktinformation) **übersetzt**, das stellt insbesondere beim Block eine Herausforderung dar.
- Eigene Sicherungslogik auf die bestehenden DrS 2 Baugruppen aufsetzen



2.2.1 Eulynx Erweiterung –

Object Controller (exp. Subsystem Point)

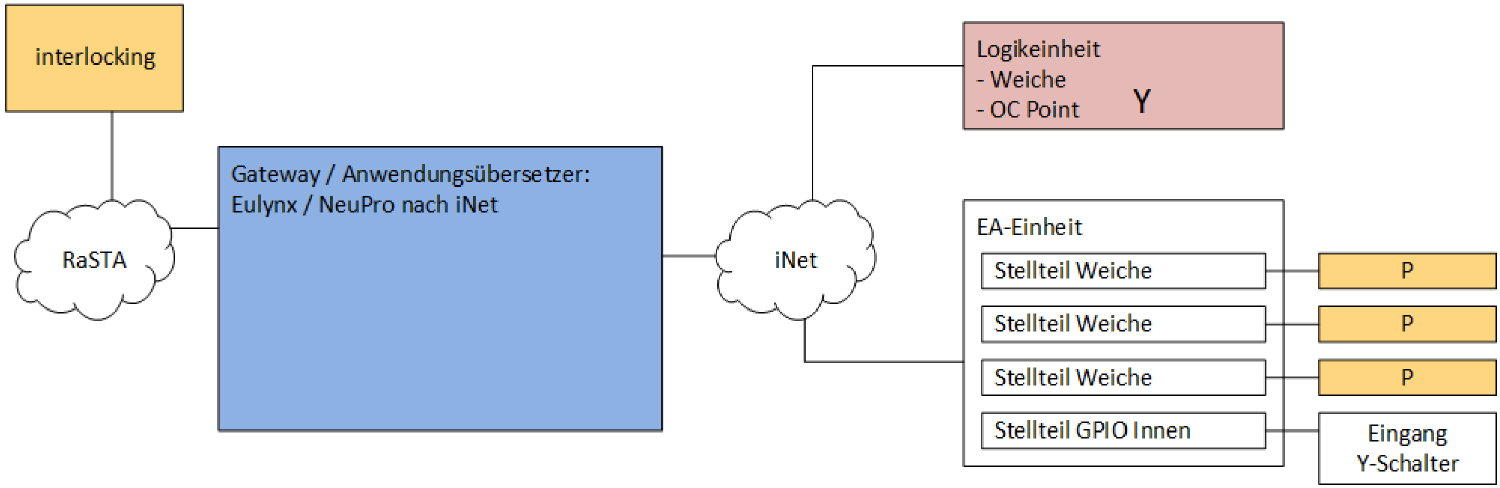
- Stell- und Sicherungslogik wird «DSTW» ausgeführt
- FPGA Stellwerk ist Object Controller abgesetzten Weichen
- Weichen-Kommandos über RaSTA/Eulynx an Gateway/Anwendungsübersetzer danach an FPGA Stellwerk
- Ansteuerung der Weiche durch FPGA Stellwerk ausgeführt



2.2.2 Eulynx Erweiterung –

Y-Umschaltung zum testen neuer Technologien

- Während Testnächten soll z.B. Weiche von Ansteuerung FPGA-Stw. auf DSTW umgeschaltet werden
- FPGA Stellwerk ist Object Controller abgesetzten Weichen
- Weichen-Kommandos über RaSTA/Eulynx an Gateway/Anwendungsübersetzer danach an FPGA Stellwerk
- Ansteuerung der Weiche durch FPGA Stellwerk ausgeführt



interlocking

RaSTA

Gateway / Anwendungsübersetzer:
Eulynx / NeuPro nach iNet

iNet

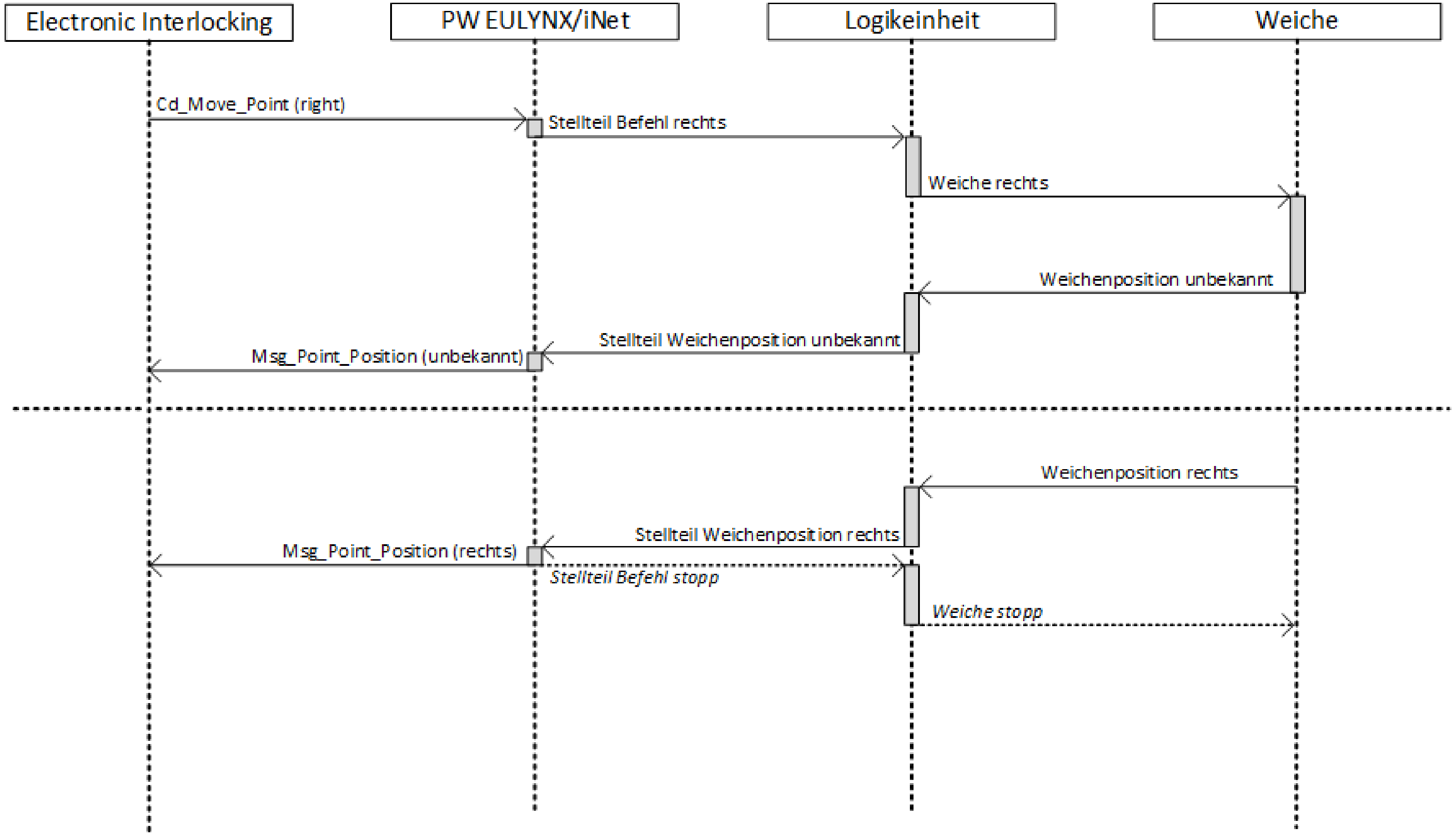
Logikeinheit
- Weiche
- OC Point

Y

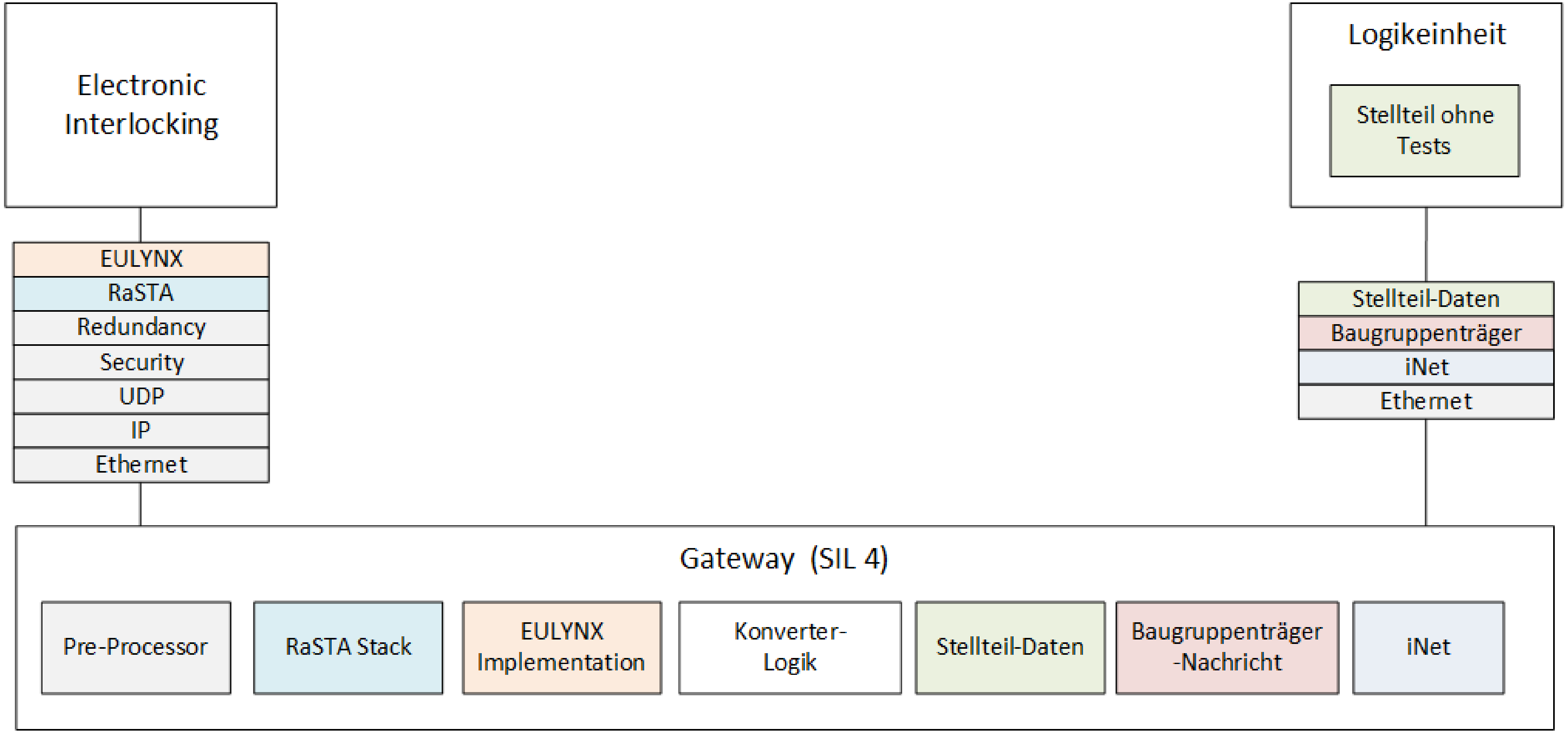
EA-Einheit

Stellteil Weiche	P
Stellteil Weiche	P
Stellteil Weiche	P
Stellteil GPIO Innen	Eingang Y-Schalter

3.1 DRSS – Prototyp Implementation OC-Point

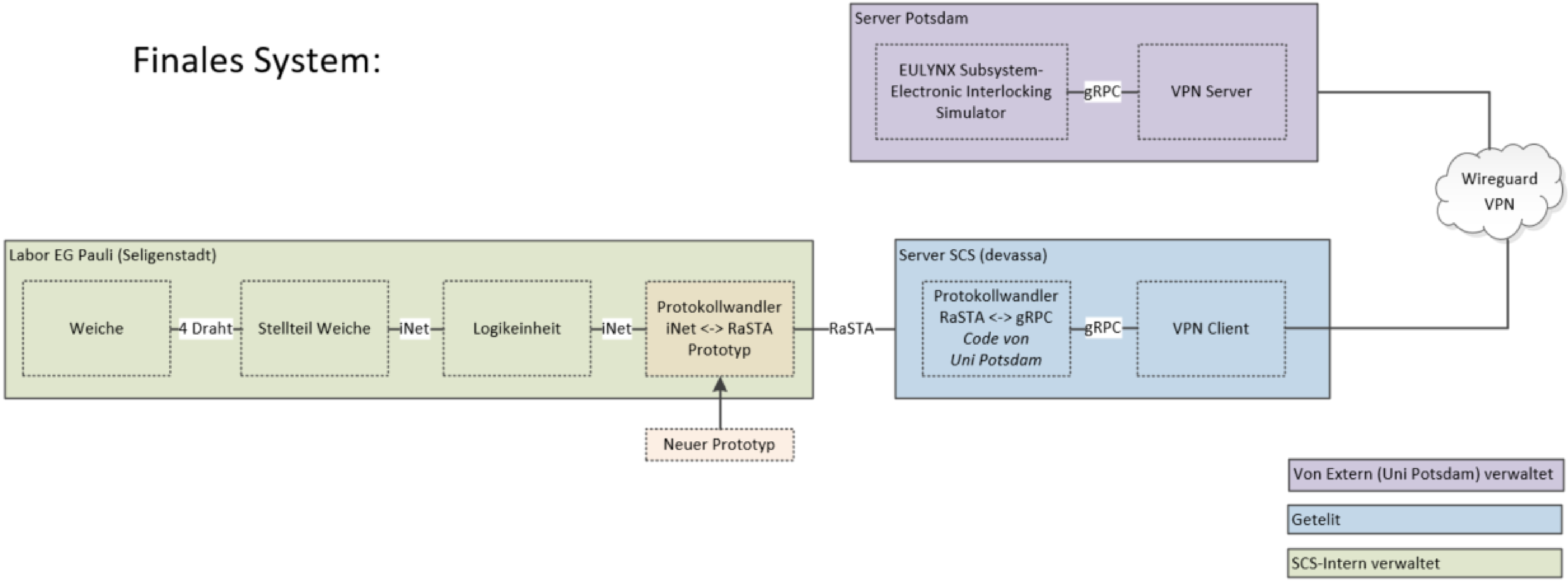


3.1 DRSS – Prototyp Implementation OC-Point

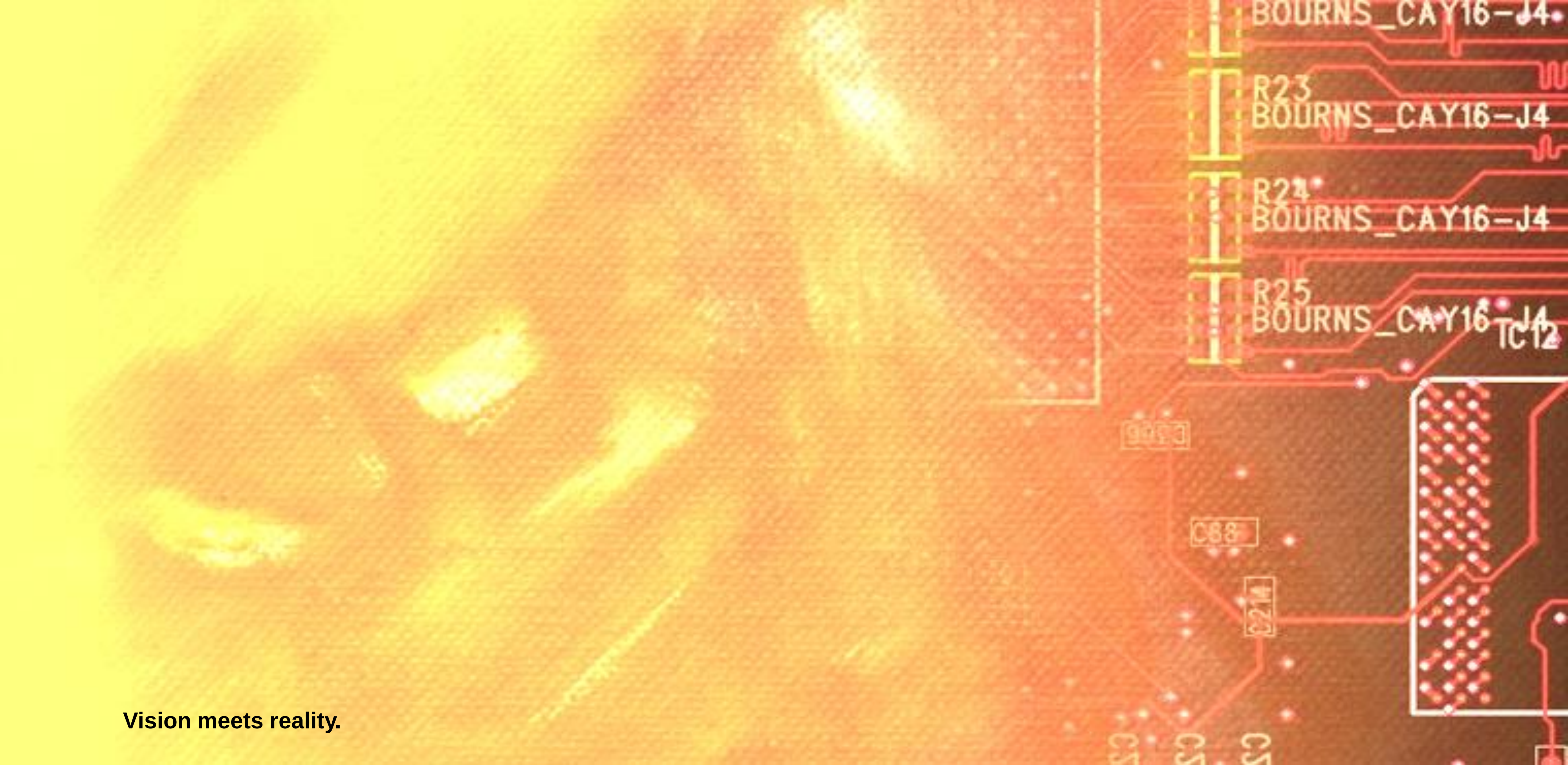


3.2 DRSS – Prototyp Konzept

Finales System:



Fragen?



Vision meets reality.

Supercomputing Systems AG
Technopark 1
CH-8005 Zürich

Phone +41 43 456 16 00
Fax +41 43 456 16 10
www.scs.ch

SCS
super computing systems