Ein Computer-Vision Projekt - zehn Plattformen

Modellbasiertes Design erleichtert die Arbeit
(ein Stück weit)

MPSI TECHNOLOGIES

Alexander Wirthmüller
aw@mpsitechnologies.com

Hallo

Über mich

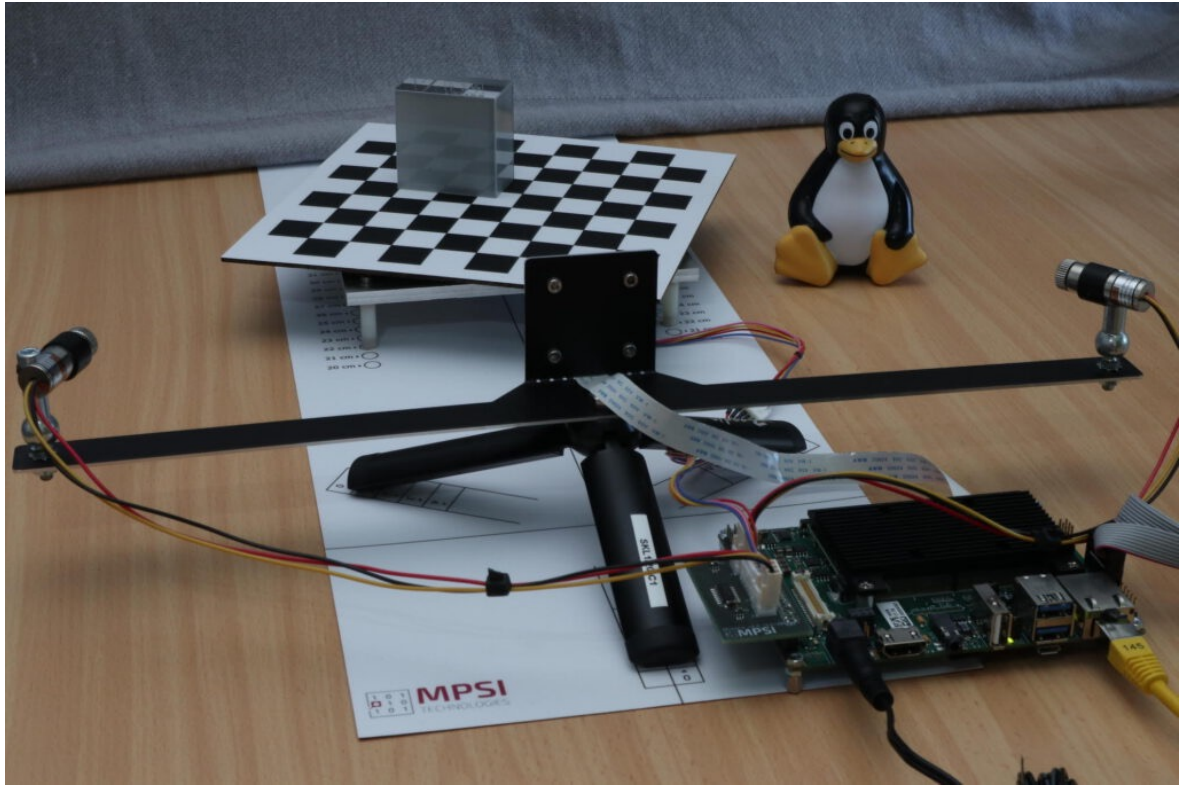
- In München
- Diplom in Elektro- und Informationstechnik

- Gründer der MPSI Technologies GmbH
- Embedded Software Entwicklung besser (bequemer, schneller, …) machen mit modellbasierter Code-Generierung

- Senior Staff Engineer bei Symeo / indie Semiconductor (Industrial Radar)

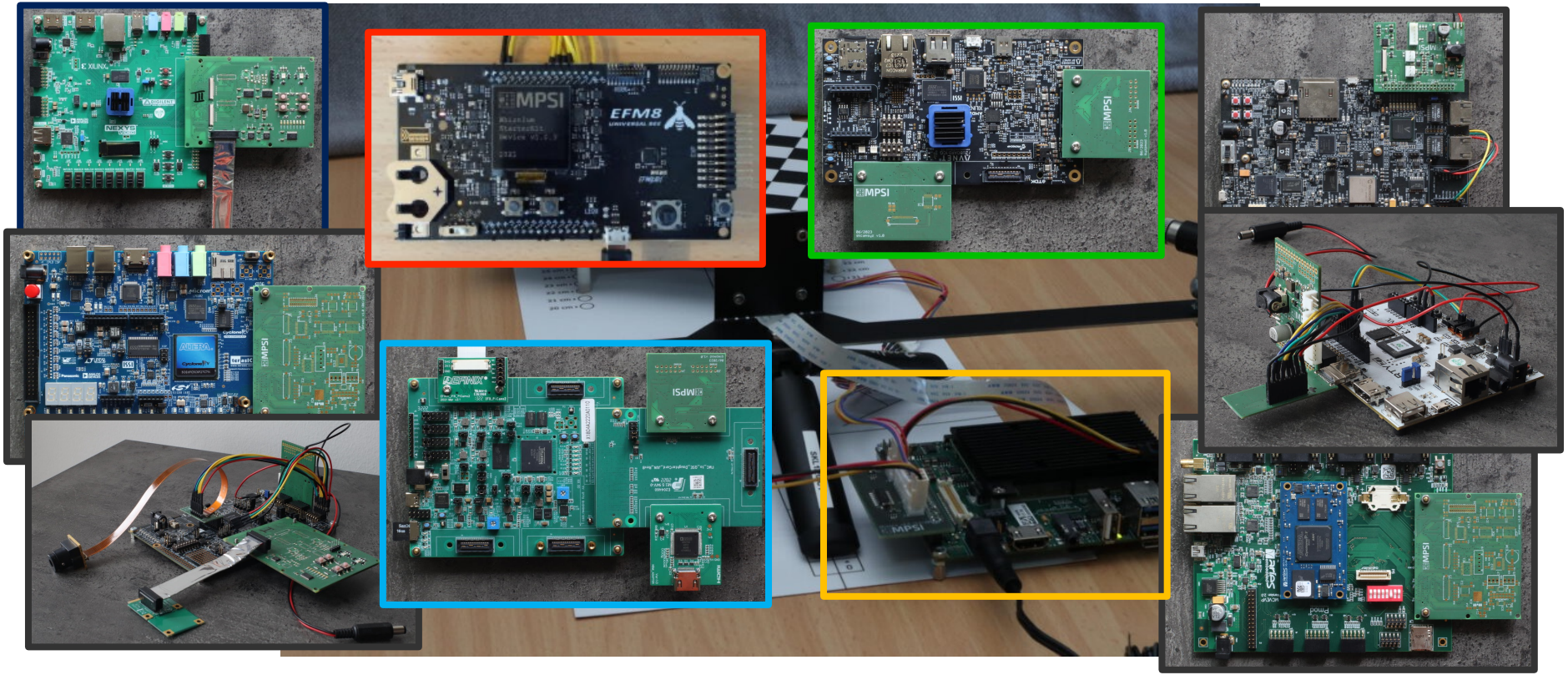
Ein Computer-Vision Projekt

Unzählige Möglichkeiten es umzusetzen



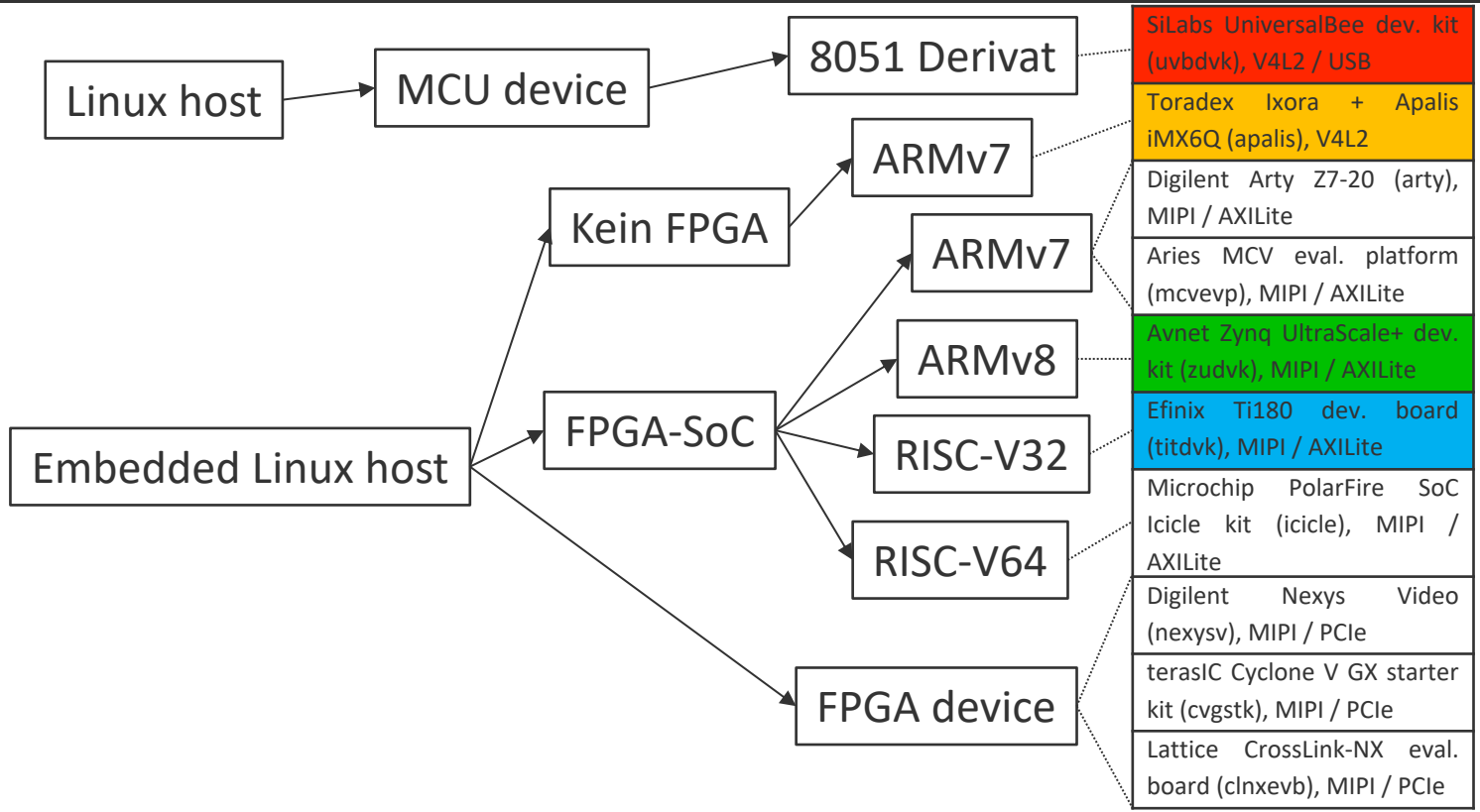
Ein Computer-Vision Projekt

Unzählige Möglichkeiten es umzusetzen



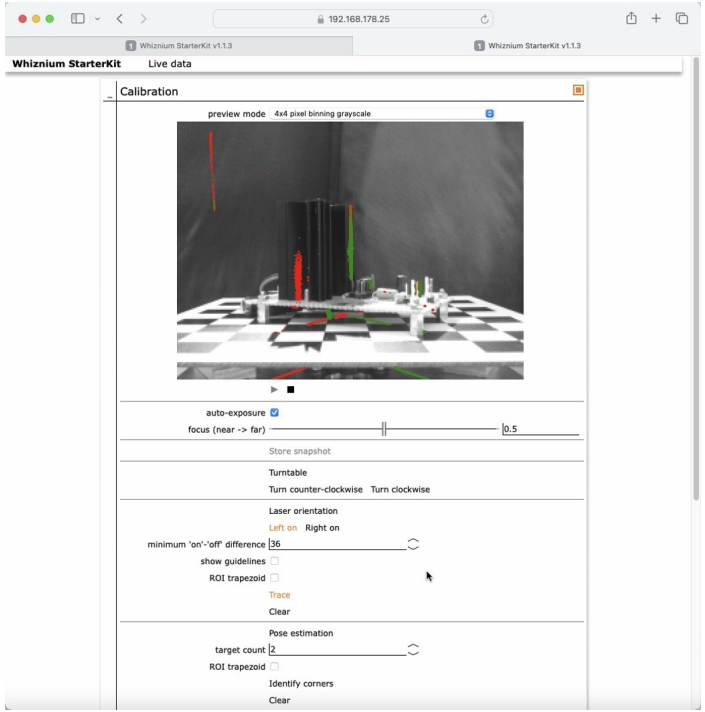
Ein Computer-Vision Projekt

Übersicht



Ein Computer-Vision Projekt

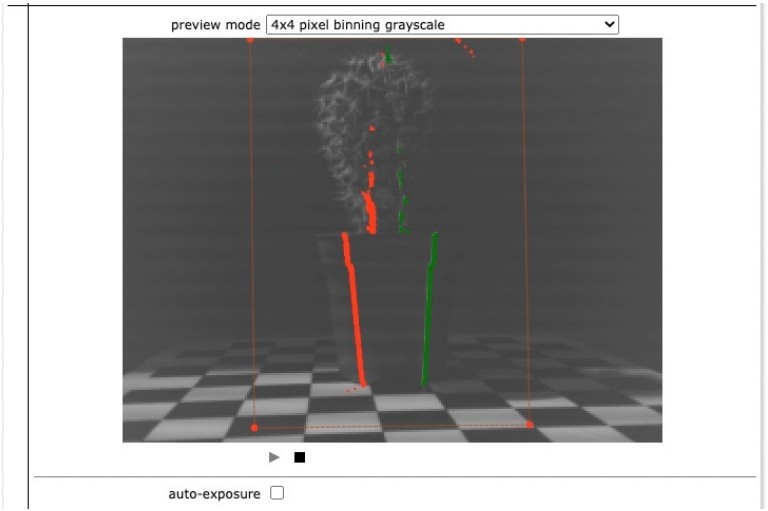
Live klassisch



Vollständiges Video: https://content.mpsitech.cloud/ese2023/wzsk_Dec2023_classic.mp4

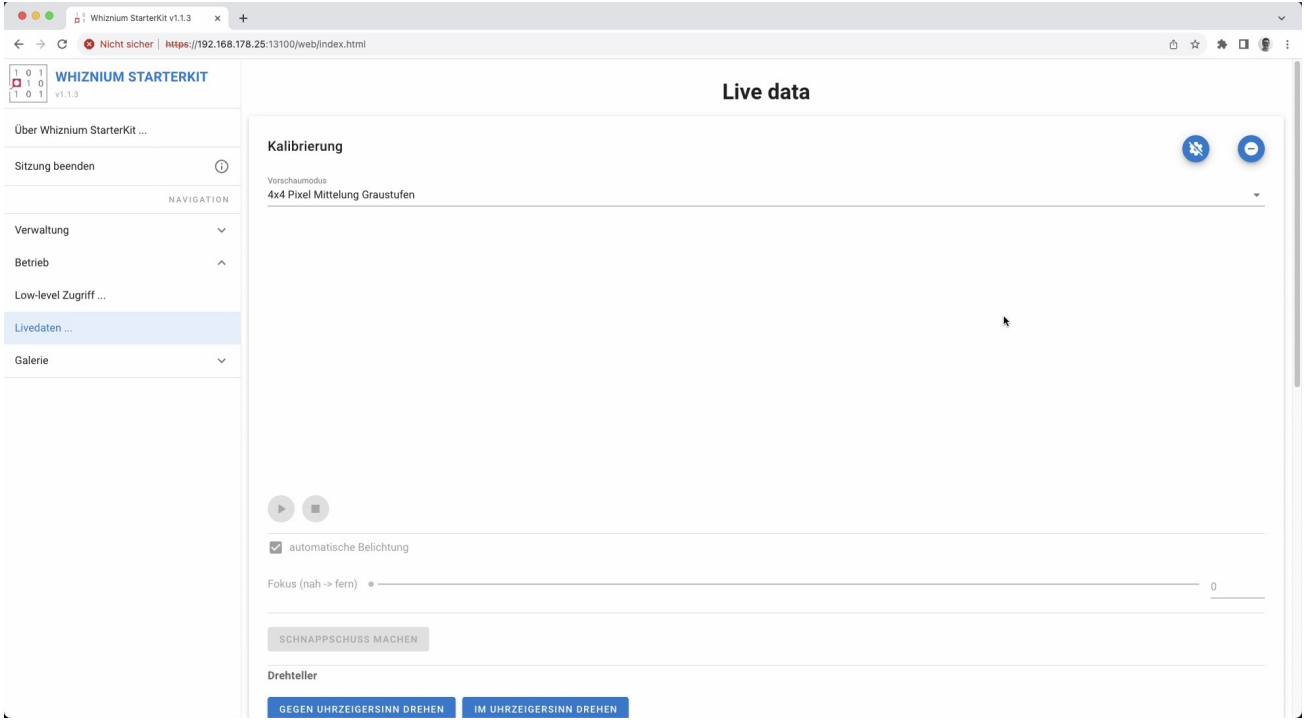
Ein Computer-Vision Projekt

Live klassisch



Ein Computer-Vision Projekt

Live modern



Vollständiges Video: https://content.mpsitech.cloud/ese2023/wzsk_Dec2023_modern.mp4

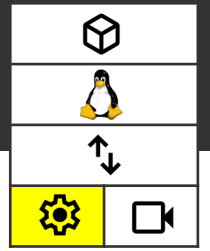
Suche nach Gemeinsamkeiten

Aspekte

Build und Deployment 	Automatisierung / Skript-barkeit Hardware Multiplexing	
Linux Host 	Hardware-Abstraktion Lauffähigkeit auf verschiedenen Architekturen	
Host - Device Schnittstelle 	Bytecode-Transfer Abstraktion von der physischen Schnittstelle	
Low-speed Hardware 	Standard-Schnittstellen	Standard-Schnittstellen Algorithmen-Implementierung Kamera 

3x SPI

MCU, FPGA und Linux Implementierungen



- MCU: Register beschreiben (optional: auf Interrupt warten)
- FPGA: req/ack *Handshake* mit SPI Modul
- Linux: spidev ioctl()

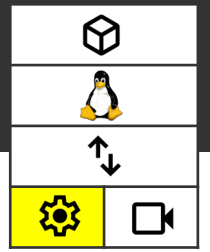
```
case stateOpSetLeft:
    // IP op.setLeft --- IBEGIN
    if (flags.ackLaserToMutexLock == 1) {
        CS = 0;

        SPI0DAT = 0x00 | ((shrdatHostif.invbuf[IXINVBUF_setL] & 0x0
        while (SPI0CN0_SPIF == 0) {};

        dummy = SPI0DAT;
        SPI0CN0_SPIF = 0;

        SPI0DAT = ((shrdatHostif.invbuf[IXINVBUF_setL+1] & 0x3F) <<
        while (SPI0CN0_SPIF == 0) {};

        dummy = SPI0DAT;
        SPI0CN0_SPIF = 0;

        CS = 1;
```

```
elsif stateOp=stateOpIdle then
    if reqInvSet='1' then
        -- IP impl.op.idle.prepL --- IBEGIN
        rNotL := false;

        txbuf(0) := "0000" & setL(9 downto 6);
        txbuf(1) := setL(5 downto 0) & "00";

        spilen <= std_logic_vector(to_unsigned(sizeTxbuf, 17));

        bytecnt := 0;
        -- IP impl.op.idle.prepL --- IEND

        stateOp <= stateOpSetC;
    end if;

elsif stateOp=stateOpSetA then
    if dneSpi='1' then
        reqSpi <= '0'; -- IP impl.op.setA.spioff --- ILINE
```

```
shrdat.txbufSpi[0] = 0x00 | (0x0F & (i >> 6));
shrdat.txbufSpi[1] = 0xFC & (i << 2);

res = ioctl(shrdat.fdSpi, SPI_IOC_MESSAGE(1), &shrdat.xferSpi);
if (res < 0) throw WzskException(WzskException::SPIDEV, {{"msg","error setting left la
```

Das FPGA-Modell für die MCU-Implementierung nutzen?

Geht ...

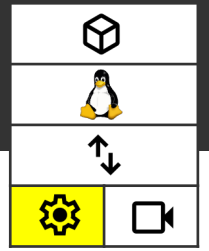


- *Request/acknowledge Handshake* mit *signals* (VHDL) wird zu *main event loop* mit #define maskenbasierten Aufrufen (C)

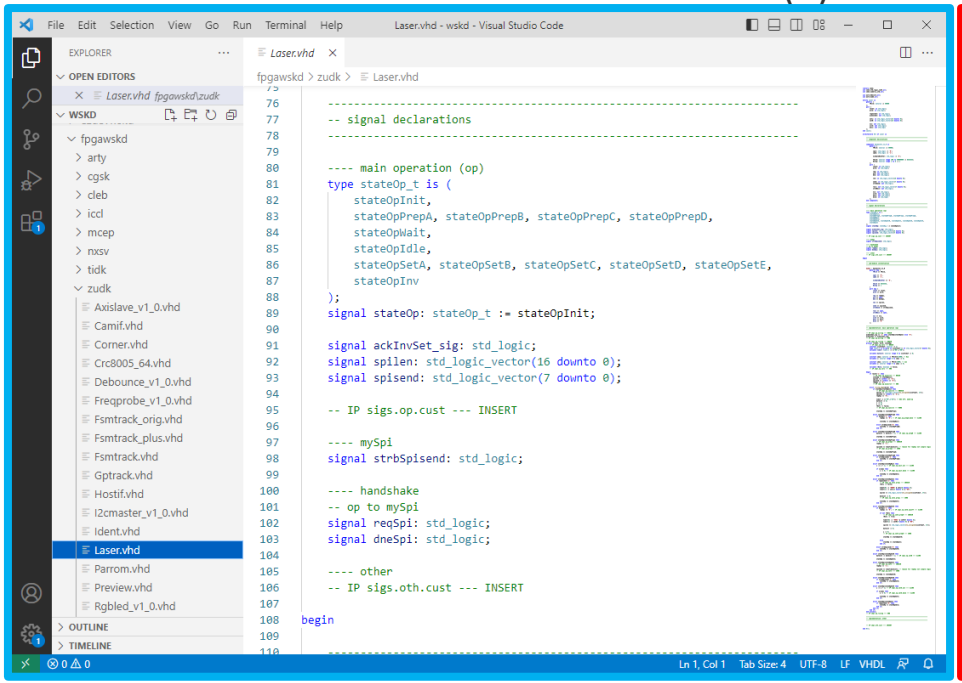
BOOST
WhizniumDBE

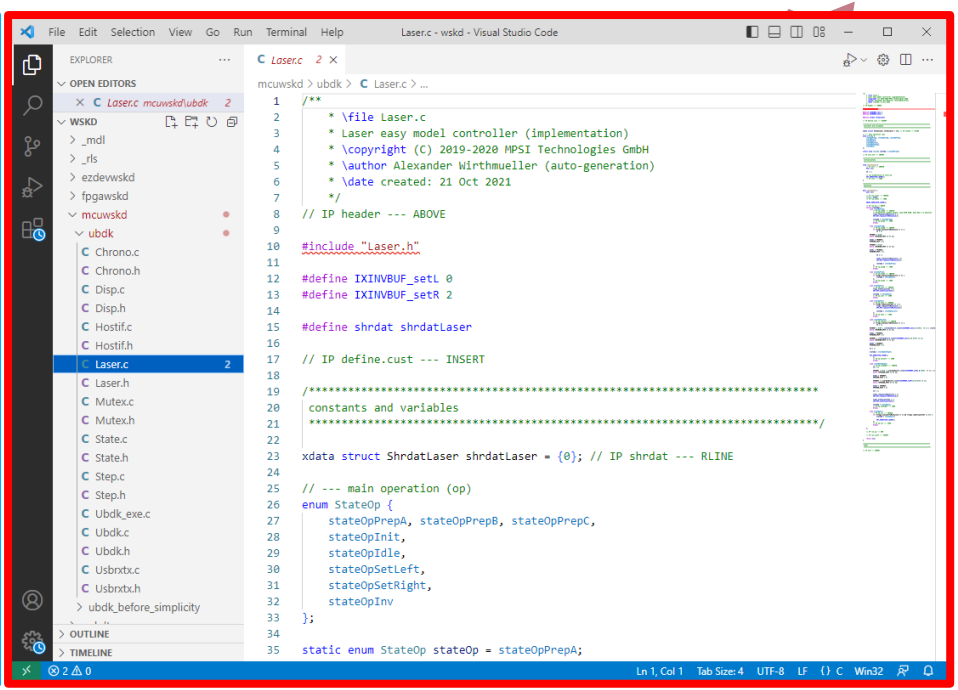
Das FPGA-Modell für die MCU-Implementierung nutzen?

Geht ...



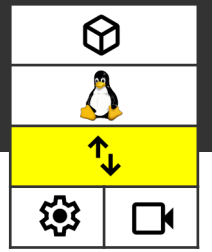
- *Request/acknowledge Handshake* mit *signals* (VHDL) wird zu *main event loop* mit #define maskenbasierten Aufrufen (C)





Host-Device Kommunikation

PHY-unabhängig

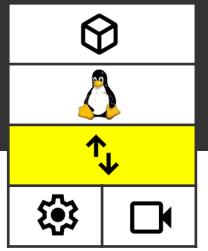


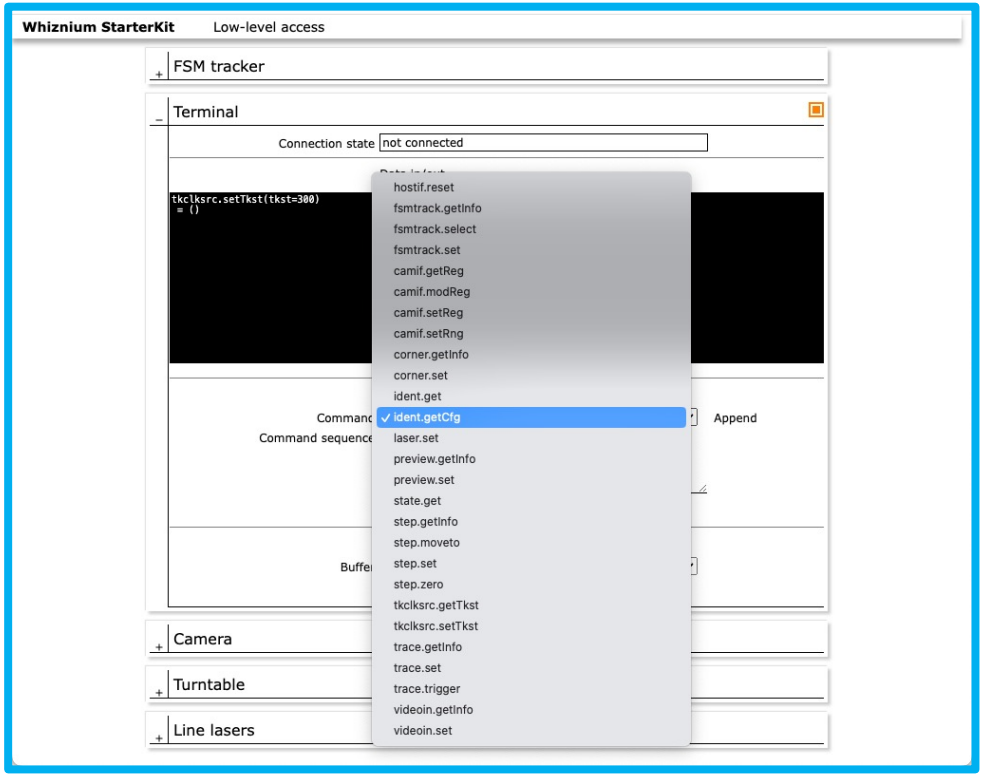
- Unterteilung von Device-Funktionalität in Module
- Zuweisung von Befehlssatz (sowie *buffer transfers*) zu Modulen
- C++ sowie C/VHDL Code-Generierung für *glue code* (UART, SPI, AXILite, PCIe)

BOOST
WhiznumDBE

Host-Device Kommunikation

PHY-unabhängig





zu Modulen

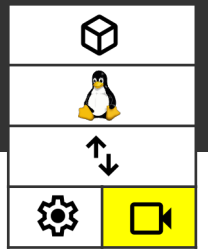
(UART, SPI, AXILite, PCIe)

BOOST WhizniumDBE

```
root@buildroot:~/whiznium/bin/wskdterm# ./Wskdterm tidk /dev/dbeaxilite
Welcome to the interactive terminal for Wskd version 1.1.14, unit UntWskdTidk!
UntWskdTidk [disconnected] >> connect
UntWskdTidk >> showCmds
    hostif.reset()
    fsmtrack.getInfo()
    fsmtrack.select(tixVSource={hostifOp,previewFrmbufB},staTixVTrigger={void,pwup,sw4_sig},stoTixVTrigger={void,pwup,s
    fsmtrack.set(rng={false,true},TCapt=[uint32])
    camif.getReg(addr=[uint16])
    camif.modReg(addr=[uint16],mask=[uint8],val=[uint8])
    camif.setReg(addr=[uint16],val=[uint8])
    camif.setRng(rng={false,true})
    corner.getInfo()
    corner.set(rng={false,true},linNotLog={false,true},thd=[uint8])
    ident.get()
    laser.set(l=[uint16],r=[uint16])
    preview.getInfo()
    preview.set(rng={false,true},grayNotRgb={false,true},rawNotBin={false,true},altNotCst={false,true},rawX0=[uint16],r
    state.get()
    step.getInfo()
    step.moveto(angle=[uint16],Tstep=[uint8])
    step.set(rng={false,true},ccwNotCw={false,true},Tstep=[uint8])
    step.zero()
    tkclksrc.getTkst()
    tkclksrc.setTkst(tkst=[uint32])
    trace.getInfo()
    trace.set(rng={false,true},deltaNotAbs={false,true},lvlFirst=[uint8],lvlSecond=[uint8])
    trace.trigger()
    videoin.getInfo()
    videoin.set(rng={false,true})
```

Kamera

MIPI CSI-2



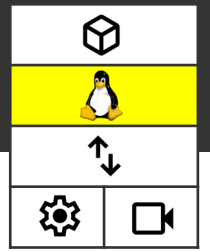
- Linux: V4L2 (“Video for Linux 2”) erledigt die Hardware-Abstraktion vorbildlich
- FPGA’s: alle Hersteller bieten MIPI CSI-2 Blöcke an, die man (relativ) standardisiert einbinden kann

Hersteller-Flexibel bleiben in FPGA’s

- Grafischen *design entry* vermeiden so weit es geht, stattdessen: RTL Code
- Hersteller-spezifische IP nur für Spezial-Features (z.B. Gigabit-Transceiver) nutzen
- Keine Spezial-Features: SPI, UART, Speicher, ...
- Open Source Wrapper für Standard-High-Speed-Interfaces (PCI Express, DDR SDRAM, ...) nutzen

Hardware-Abstraktion auf der Linux Host-Seite

Kapselung der Funktionalität



- Konzept des *Job Tree* (Hierarchische Runtime-Struktur von C++ Objekten)

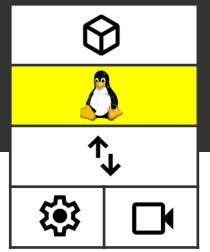
```
root@apalis-imx6:~/whiznium/bin/wzskcmbd# uname -a
Linux apalis-imx6 4.14.117-3.0.2+ge43e3a26e1b7 #1 SMP Sun Sep 6 13:39:42 UTC 2020 armv7l GNU/Linux

root@apalis-imx6:~/whiznium/bin/wzskcmbd# ./run.sh
Welcome to Whiznium StarterKit v1.1.3!
    starting 4 job processor threads ... {2145, 2147, 2151, 2152} success
    starting 1 operation processor threads ... {2153} success
    starting application server ... success
    starting DDS publisher ... success
    starting OPC UA server ... success
Initialization complete.

Wzskcmbd >> showJobs
    + RootWzsk (1)
        - JobWzskSrcZudvk/SRV (2)
        - JobWzskSrcV4l2/SRV (3)
        - JobWzskSrcUvbdvk/SRV (4)
        - JobWzskSrcTitdvk/SRV (5)
        - JobWzskSrcSysinfo/SRV (6)
        - JobWzskSrcPwmonuart/SRV (7)
        - JobWzskSrcNexysv/SRV (8)
        - JobWzskSrcMcvevp/SRV (9)
        - JobWzskSrcIcicle/SRV (10)
        - JobWzskSrcCvgxstk/SRV (11)
        - JobWzskSrcClnxevb/SRV (12)
        - JobWzskSrcArty/SRV (13)
        + JobWzskIprTrace/SRV (14)
            - JobWzskActLaser/CLI (15)
            - JobWzskSrcV4l2/CLI (16)
        + JobWzskIprCorner/SRV (17)
            - JobWzskSrcV4l2/CLI (18)
        + JobWzskIprAngle/SRV (19)
            - JobWzskIprCorner/CLI (20)
        - JobWzskActServo/SRV (21)
        - JobWzskActLaser/SRV (22)
        + JobWzskActExposure/SRV (23)
            - JobWzskSrcV4l2/CLI (24)
        + JobWzskAcqPtcloud/SRV (25)
            - JobWzskIprTrace/CLI (26)
            - JobWzskActServo/CLI (27)
        + JobWzskAcqPreview/SRV (28)
            - JobWzskSrcV4l2/CLI (29)
        - JobWzskAcqFpgapvw/SRV (30)
        - JobWzskAcqFpgaflg/SRV (31)
        ...
```

```
root@buildroot:~/whiznium/bin/wzskcmbd# uname -a
Linux buildroot 5.10.0 #4 SMP Thu Aug 24 05:42:20 CEST 2023 riscv32 GNU/Linux

root@buildroot:~/whiznium/bin/wzskcmbd# ./Wzskcmbd
Welcome to Whiznium StarterKit v1.1.3!
    starting 4 job processor threads ... {227, 228, 230, 232} success
    starting 1 operation processor threads ... {233} success
    starting application server ... success
Initialization complete.

Wzskcmbd >> showJobs
    + RootWzsk (1)
        - JobWzskSrcZudvk/SRV (2)
        - JobWzskSrcV4l2/SRV (3)
        - JobWzskSrcUvbdvk/SRV (4)
        - JobWzskSrcTitdvk/SRV (5)
        - JobWzskSrcSysinfo/SRV (6)
        - JobWzskSrcPwmonuart/SRV (7)
        - JobWzskSrcNexysv/SRV (8)
        - JobWzskSrcMcvevp/SRV (9)
        - JobWzskSrcIcicle/SRV (10)
        - JobWzskSrcCvgxstk/SRV (11)
        - JobWzskSrcClnxevb/SRV (12)
        - JobWzskSrcArty/SRV (13)
        + JobWzskIprTrace/SRV (14)
            - JobWzskActLaser/CLI (15)
            - JobWzskAcqFpgaflg/CLI (16)
        + JobWzskIprCorner/SRV (17)
            - JobWzskAcqFpgaflg/CLI (18)
        + JobWzskIprAngle/SRV (19)
            - JobWzskIprCorner/CLI (20)
        + JobWzskActServo/SRV (21)
            - JobWzskSrcTitdvk/CLI (22)
        + JobWzskActLaser/SRV (23)
            - JobWzskSrcTitdvk/CLI (24)
        + JobWzskActExposure/SRV (25)
            - JobWzskSrcTitdvk/CLI (26)
        + JobWzskAcqPtcloud/SRV (27)
            - JobWzskIprTrace/CLI (28)
            - JobWzskActServo/CLI (29)
        + JobWzskAcqPreview/SRV (30)
            - JobWzskAcqFpgapvw/CLI (31)
        + JobWzskAcqFpgapvw/SRV (32)
            - JobWzskSrcTitdvk/CLI (33)
        + JobWzskAcqFpgaflg/SRV (34)
            - JobWzskSrcTitdvk/CLI (35)
```

BOOST WhizniumSBE

Hardware-Abstraktion auf der Linux Host-Seite

Kapselung der Funktionalität



- Konzept des *Job Tree* (Hierarchische Runtime-Struktur von C++ Objekten)

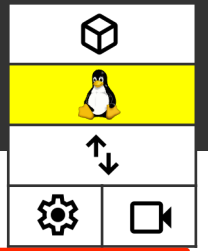
```
root@apalis-imx6:~/whiznium/bin/wzskcmbd# uname -a
Linux apalis-imx6 4.14.117-3.0.2+ge43e3a26e1b7 #1 SMP Sun Sep 6 13:39:42 UTC 2020 armv7l GNU/Linux

root@apalis-imx6:~/whiznium/bin/wzskcmbd# ./run.sh
Welcome to Whiznium StarterKit v1.1.3!
    starting 4 job processor threads ... {2145, 2147, 2151, 2152} success
    starting 1 operation processor threads ... {2153} success
    starting application server ... success
    starting DDS publisher ... success
    starting OPC UA server ... success
Initialization complete.

Wzskcmbd >> showJobs
    + RootWzsk (1)
        - JobWzskSrcZudvk/SRV (2)
        - JobWzskSrcV4l2/SRV (3)
        - JobWzskSrcUvbdvk/SRV (4)
        - JobWzskSrcTitdvk/SRV (5)
        - JobWzskSrcSysinfo/SRV (6)
        - JobWzskSrcPwmonuart/SRV (7)
        - JobWzskSrcNexysv/SRV (8)
        - JobWzskSrcMcvevp/SRV (9)
        - JobWzskSrcIcicle/SRV (10)
        - JobWzskSrcCvgxstk/SRV (11)
        - JobWzskSrcClnxevb/SRV (12)
        - JobWzskSrcArty/SRV (13)
        + JobWzskIprTrace/SRV (14)
            - JobWzskActLaser/CLI (15)
            - JobWzskSrcV4l2/CLI (16)
        + JobWzskIprCorner/SRV (17)
            - JobWzskSrcV4l2/CLI (18)
        + JobWzskIprAngle/SRV (19)
            - JobWzskIprCorner/CLI (20)
        - JobWzskActServo/SRV (21)
        - JobWzskActLaser/SRV (22)
        + JobWzskActExposure/SRV (23)
            - JobWzskSrcV4l2/CLI (24)
        + JobWzskAcqPtcloud/SRV (25)
            - JobWzskIprTrace/CLI (26)
            - JobWzskActServo/CLI (27)
        + JobWzskAcqPreview/SRV (28)
            - JobWzskSrcV4l2/CLI (29)
        - JobWzskAcqFpgapvw/SRV (30)
        - JobWzskAcqFpgaflg/SRV (31)
        ...
```

```
root@ZUBoard:~# uname -a
Linux ZUBoard 5.15.36-xilinx-v2022.2 #1 SMP Mon Oct 3 07:50:07 UTC 2022 aarch64 GNU/Linux

root@ZUBoard:~/whiznium/bin/wzskcmbd# ./Wzskcmbd
Welcome to Whiznium StarterKit v1.1.3!
    starting 2 job processor threads ... {631, 632} success
    starting 1 operation processor threads ... {633} success
    starting application server ... success
Initialization complete.

Wzskcmbd >> showJobs
    + RootWzsk (1)
        - JobWzskSrcZudvk/SRV (2)
        - JobWzskSrcV4l2/SRV (3)
        - JobWzskSrcUvbdvk/SRV (4)
        - JobWzskSrcTitdvk/SRV (5)
        - JobWzskSrcSysinfo/SRV (6)
        - JobWzskSrcPwmonuart/SRV (7)
        - JobWzskSrcNexysv/SRV (8)
        - JobWzskSrcMcvevp/SRV (9)
        - JobWzskSrcIcicle/SRV (10)
        - JobWzskSrcCvgxstk/SRV (11)
        - JobWzskSrcClnxevb/SRV (12)
        - JobWzskSrcArty/SRV (13)
        + JobWzskIprTrace/SRV (14)
            - JobWzskActLaser/CLI (15)
            - JobWzskAcqFpgaflg/CLI (16)
        + JobWzskIprCorner/SRV (17)
            - JobWzskAcqFpgaflg/CLI (18)
        + JobWzskIprAngle/SRV (19)
            - JobWzskIprCorner/CLI (20)
        - JobWzskActServo/SRV (21)
            - JobWzskSrcZudvk/CLI (22)
        - JobWzskActLaser/SRV (23)
            - JobWzskSrcZudvk/CLI (24)
        - JobWzskActExposure/SRV (25)
            - JobWzskSrcZudvk/CLI (26)
        + JobWzskAcqPtcloud/SRV (27)
            - JobWzskIprTrace/CLI (28)
            - JobWzskActServo/CLI (29)
        + JobWzskAcqPreview/SRV (30)
            - JobWzskAcqFpgapvw/CLI (31)
        - JobWzskAcqFpgapvw/SRV (32)
            - JobWzskSrcZudvk/CLI (33)
        - JobWzskAcqFpgaflg/SRV (34)
            - JobWzskSrcZudvk/CLI (35)
```

BOOST WhizniumSBE

Hardware-Abstraktion auf der Linux Host-Seite

Kapselung der Funktionalität



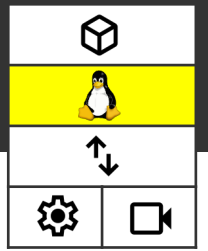
- Konzept des *Job Tree* (Hierarchische Runtime-Struktur von C++ Objekten)

```
root@apalis-imx6:~/whiznium/bin/wzskcmbd# uname -a
Linux apalis-imx6 4.14.117-3.0.2+ge43e3a26e1b7 #1 SMP Sun Sep 6 13:39:42 UTC 2020 armv7l GNU/Linux

root@apalis-imx6:~/whiznium/bin/wzskcmbd# ./run.sh
Welcome to Whiznium StarterKit v1.1.3!
    starting 4 job processor threads ... {2145, 2147, 2151, 2152} success
    starting 1 operation processor threads ... {2153} success
    starting application server ... success
    starting DDS publisher ... success
    starting OPC UA server ... success
Initialization complete.

Wzskcmbd >> showJobs
    + RootWzsk (1)
        - JobWzskSrcZudvk/SRV (2)
        - JobWzskSrcV4l2/SRV (3)
        - JobWzskSrcUvbdvk/SRV (4)
        - JobWzskSrcTitdvk/SRV (5)
        - JobWzskSrcSysinfo/SRV (6)
        - JobWzskSrcPwmonuart/SRV (7)
        - JobWzskSrcNexysv/SRV (8)
        - JobWzskSrcMcvevp/SRV (9)
        - JobWzskSrcIcicle/SRV (10)
        - JobWzskSrcCvgxstk/SRV (11)
        - JobWzskSrcClnxevb/SRV (12)
        - JobWzskSrcArty/SRV (13)
        + JobWzskIprTrace/SRV (14)
            - JobWzskActLaser/CLI (15)
            - JobWzskSrcV4l2/CLI (16)
        + JobWzskIprCorner/SRV (17)
            - JobWzskSrcV4l2/CLI (18)
        + JobWzskIprAngle/SRV (19)
            - JobWzskIprCorner/CLI (20)
        - JobWzskActServo/SRV (21)
        - JobWzskActLaser/SRV (22)
        + JobWzskActExposure/SRV (23)
            - JobWzskSrcV4l2/CLI (24)
        + JobWzskAcqPtcloud/SRV (25)
            - JobWzskIprTrace/CLI (26)
            - JobWzskActServo/CLI (27)
        + JobWzskAcqPreview/SRV (28)
            - JobWzskSrcV4l2/CLI (29)
        - JobWzskAcqFpgapvw/SRV (30)
        - JobWzskAcqFpgaflg/SRV (31)
        ...
```

```
root@ZUBoard:~# uname -a
mpsitech@chip:~$ uname -a
Linux chip 6.2.0-37-generic #38~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Nov  2 18:01:13 UTC 2 x86_64 x86_64 x86_64 GNU/Linux

mpsitech@chip:~/whiznium/bin/wzskcmbd$ ./Wzskcmbd
Welcome to Whiznium StarterKit v1.1.3!
    starting 4 job processor threads ... {7094, 7095, 7096, 7097} success
    starting 1 operation processor threads ... {7098} success
    starting application server ... success
Initialization complete.

Wzskcmbd >> showJobs
    + RootWzsk (1)
        - JobWzskSrcZudvk/SRV (2)
        - JobWzskSrcV4l2/SRV (3)
        - JobWzskSrcUvbdvk/SRV (4)
        - JobWzskSrcTitdvk/SRV (5)
        - JobWzskSrcSysinfo/SRV (6)
        - JobWzskSrcPwmonuart/SRV (7)
        - JobWzskSrcNexysv/SRV (8)
        - JobWzskSrcMcvevp/SRV (9)
        - JobWzskSrcIcicle/SRV (10)
        - JobWzskSrcCvgxstk/SRV (11)
        - JobWzskSrcClnxevb/SRV (12)
        - JobWzskSrcArty/SRV (13)
        + JobWzskIprTrace/SRV (14)
            - JobWzskActLaser/CLI (15)
            - JobWzskSrcV4l2/CLI (16)
        + JobWzskIprCorner/SRV (17)
            - JobWzskSrcV4l2/CLI (18)
        + JobWzskIprAngle/SRV (19)
            - JobWzskIprCorner/CLI (20)
        + JobWzskActServo/SRV (21)
            - JobWzskSrcUvbdvk/CLI (22)
        + JobWzskActLaser/SRV (23)
            - JobWzskSrcUvbdvk/CLI (24)
        + JobWzskActExposure/SRV (25)
            - JobWzskSrcV4l2/CLI (26)
        + JobWzskAcqPtcloud/SRV (27)
            - JobWzskIprTrace/CLI (28)
            - JobWzskActServo/CLI (29)
        + JobWzskAcqPreview/SRV (30)
            - JobWzskSrcV4l2/CLI (31)
        - JobWzskAcqFpgapvw/SRV (32)
        - JobWzskAcqFpgaflg/SRV (33)
```

Hardware-Abstraktion auf der Linux Host-Seite

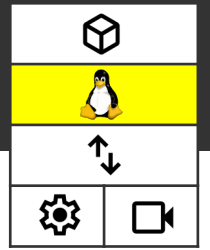
Nutzung verschiedener CPU *Extensions*



- Beispiel Vorschaubild: 8x8 RGB / 4x4 Graustufen *Binning*
- NEON (ARM) vs. SSE2 (intel) vs. universell - gelöst mit #ifdef’s

Hardware-Abstraktion auf der Linux Host-Seite

Nutzung verschiedener CPU *Extensions*



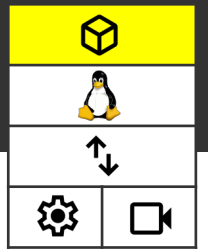
```arm
#ifdef __arm__
    // hard-wired to 4x4 super-pixels
    const uint64_t zero64 = 0;

    uint16x8_t data;
    uint32x4_t dataAcc2;
    uint64x2_t dataAcc4;

    uint64x2_t acc;
    uint16_t acc16;

    unsigned int ldix, stix;

    for (unsigned int i = 0; i < xchg->stgwzskframegeo.hGrrd; i += 4) {
        for (unsigned int j = 0; j < xchg->stgwzskframegeo.wGrrd; j += 8) {
            ldix = i * xchg->stgwzskframegeo.wGrrd + j;

            acc = vld1q_dup_u64(&zero64);

            for (unsigned int k = 0; k < 4; k++) {
                data = vld1q_u16(&(grrd16[ldix]));
                dataAcc2 = vpaddlq_u16(data);
                dataAcc4 = vpaddlq_u32(dataAcc2);

                acc = vaddq_u64(acc, dataAcc4);

                ldix += xchg->stgwzskframegeo.wGrrd;
            };

            stix = i/4 * xchg->stgwzskframegeo.wGrrd/4 + j/4;

            acc16 = vgetq_lane_u16(vreinterpretq_u16_u64(acc), 0);
            pvwgrrd16[stix] = acc16 >> 4;

            stix++;

            acc16 = vgetq_lane_u16(vreinterpretq_u16_u64(acc), 4);
            pvwgrrd16[stix] = acc16 >> 4;
        };
    };
```

```x86_64
#elif __x86_64__
    // hard-wired to 2x2 super-pixels
    uint16_t* buf = NULL;

    __m128i dataeven, dataodd, datashift;

    unsigned int ldix, stix;

    posix_memalign((void**) &buf, 16, xchg->stgwzskframegeo.wGrrd * siz

    for (unsigned int i = 0; i < xchg->stgwzskframegeo.hGrrd; i += 2) {
        for (unsigned int j = 0; j < xchg->stgwzskframegeo.wGrrd; j +=
            ldix = i * xchg->stgwzskframegeo.wGrrd + j;
            dataeven = _mm_load_si128((const __m128i*) &(grrd16[ldix]))

            ldix += xchg->stgwzskframegeo.wGrrd;
            dataodd = _mm_load_si128((const __m128i*) &(grrd16[ldix]));

            dataeven = _mm_add_epi16(dataeven, dataodd);
            datashift = _mm_slli_si128(dataeven, 2);

            dataeven = _mm_add_epi16(dataeven, datashift);
            dataeven = _mm_srli_epi16(dataeven, 2);

            stix = i/2 * xchg->stgwzskframegeo.wGrrd/2 + j/2;
            _mm_store_si128 ((__m128i*) buf, dataeven);
            pvwgrrd16[stix++] = buf[1];
            pvwgrrd16[stix++] = buf[3];
            pvwgrrd16[stix++] = buf[5];
            pvwgrrd16[stix] = buf[7];
            //pvwgrrd16[stix++] = _mm_extract_epi16(dataeven, 1);
            //pvwgrrd16[stix++] = _mm_extract_epi16(dataeven, 3);
            //pvwgrrd16[stix++] = _mm_extract_epi16(dataeven, 5);
            //pvwgrrd16[stix] = _mm_extract_epi16(dataeven, 7);
        };
    };

    delete[] buf;
```

```else
#else
    // hard-wired to 2x2 super-pixels
    unsigned int ldix, stix;

    uint32_t acc;

    for (unsigned int i = 0; i < xchg->stgwzskframegeo.hGrrd; i += 2) {
        for (unsigned int j = 0; j < xchg->stgwzskframegeo.wGrrd; j += 2) {
            ldix = i * xchg->stgwzskframegeo.wGrrd + j;

            acc = 0;

            for (unsigned int k = 0; k < 2; k++) {
                for (unsigned int l = 0; l < 2; l++) acc += grrd16[ldix+l];

                ldix += xchg->stgwzskframegeo.wGrrd;
            };

            stix = i/2 * xchg->stgwzskframegeo.wGrrd/2 + j/2;

            pvwgrrd16[stix] = acc >> 2;
        };
    };
```

Build-Automatisierung

Jenkins oder “was kann man in Skripte packen”

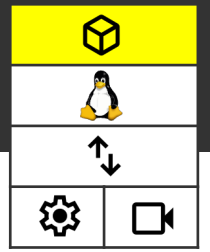


- Verwendung der Hersteller-IDE’s bei Beginn mit neuer Plattform unvermeidlich (aber auch effizient: Live-Debugging, etc.)
- Bei Funkionsaktualisierungen (oder Iteration des modellbasiert generierten Codes): Skriptbarkeit gefragt

- Subjektive Einschätzung Skriptbarkeit
  - FPGA-Hersteller Tools: .tcl getrieben - mittel
  - MCU Code: von IDE erstellte Makefiles nutzbar - mittel
  - Yocto und Buildroot für Linux-Distribution - gut
  - C++ Code für Linux Daemon - sehr gut

Test-Automatisierung

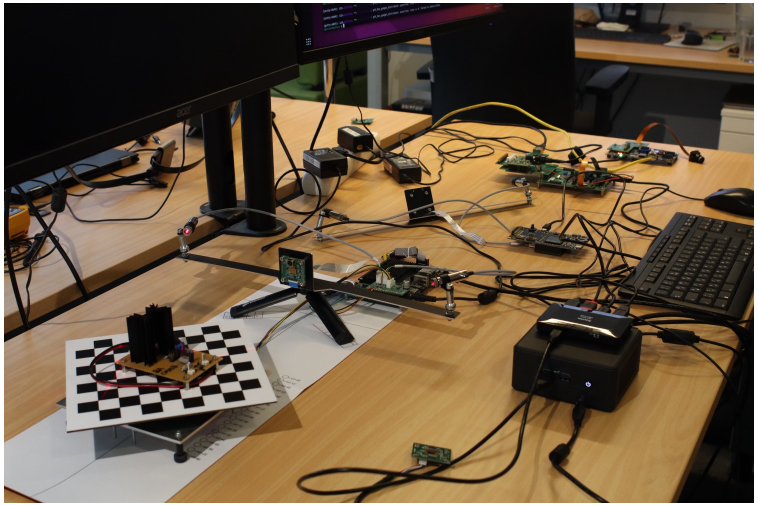
Ideen und logistische Herausforderungen



- API-getriebenes Auslösen der UI-Funktionen (automatisch generierte C++ API Bibliothek WhizniumSBE Projekten)

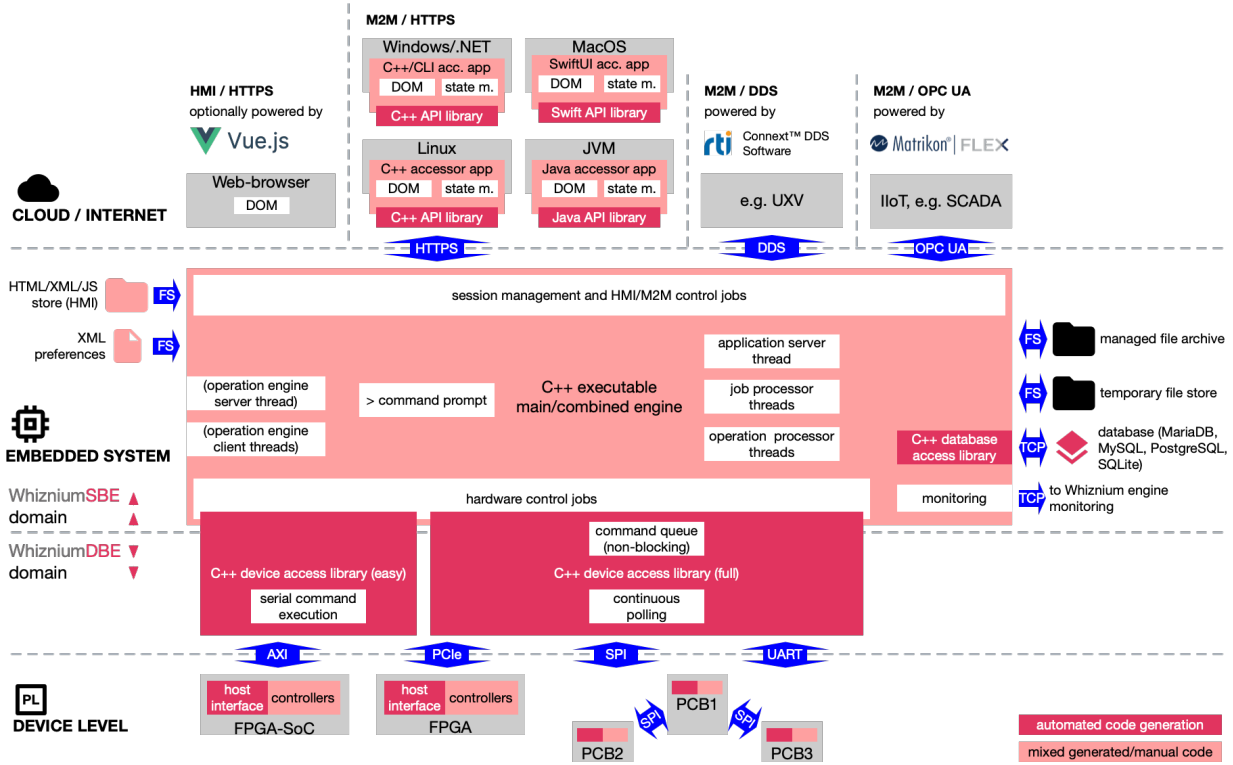
BOOST WhizniumSBE

- Loggen der Reaktion, Aufzeichnen von Referenzbildern
- Problem: Multiplexing; Low-level könnte mit FPGA gehen, MIPI Multiplexing nur aktiv / schwierig



Modellbasiertes Design mit Whiznium

Umfang



Modellbasiertes Design mit Whiznium

Funktionsweise

- Schrittweiser Modellaufbau in SQL Datenbank mit Import (I)- und Generierungs (G)-Schritten
- Erst dann automatisches Schreiben des Quellcodes
- Text-basierte Modelldateien (”diff”-bar)

WhizniumDBE (Device Builder’s Edition)

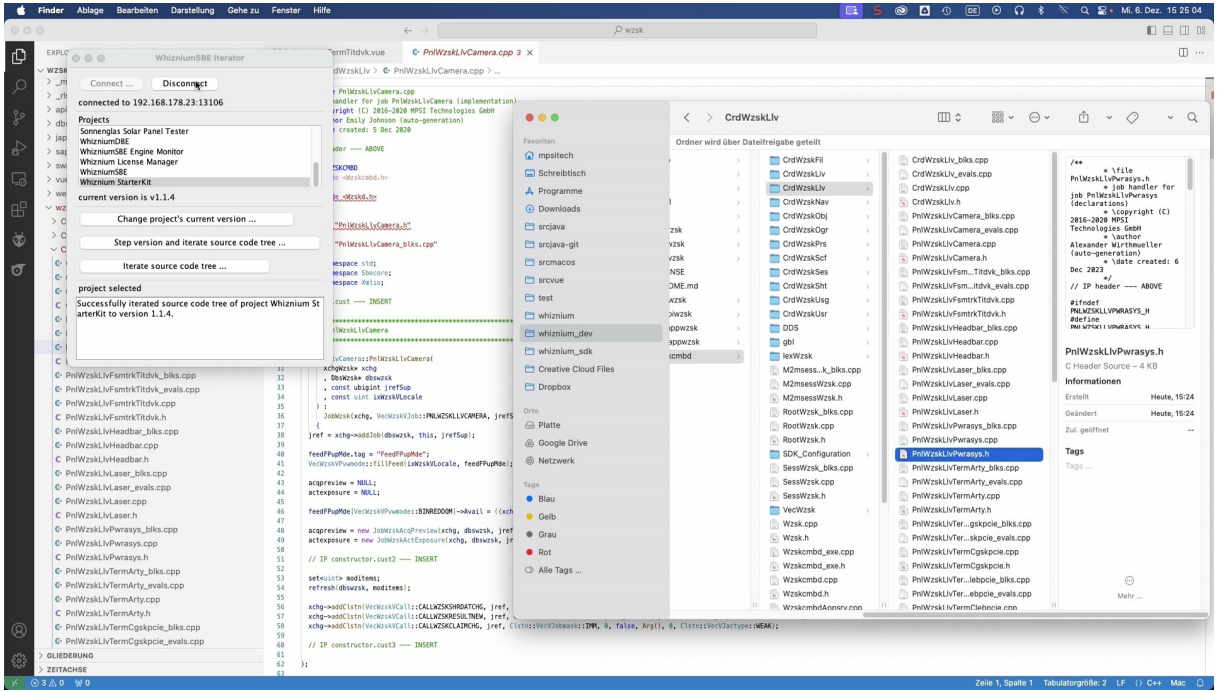
- Modular structure (I)
- Command set and buffer transfers (I)
- Data flows and algorithms (I)
- Fine structure (G)
- Custom fine structure (I)
- Finalization (G)

WhizniumSBE (Service Builder’s Edition)

- Deployment information (I)
- Global features (I)
- Database structure (I)
- Basic user interface structure (I)
- Import/export structure (I)
- Operation pack structure (I)
- Custom jobs (I)
- User interface (G)
- Custom user interface features (I)
- Job tree (G)
- Custom job tree features (I)
- Finalization (G)

Modellbasiertes Design mit Whiznium

Einbau in den Programmieralltag



Vollständiges Video: https://content.mpsitech.cloud/ese2023/WhizniumSBE_step.mp4

Zusammenfassung

- *Hardware Abstraction* geschieht – klar – in der Software
- Es lohnt sich, auf dem hardware-nächsten Niveau anzusetzen (feingliedrige Funktionalität)
- Ansatz, Code für alle Plattformen in einem Projekt beisammen zu halten
- Universelle bzw. bereits portierte Programmiersprachen (C, C++, ...) und Betriebssysteme (Linux, ...) erledigen den Rest

- Modellbasiertes Software-Design hilft an mehreren Stellen
  - *State Machines* nutzbar sowohl für MCU oder FPGA
  - PHY-agnostische Host-Device Kommunikation
  - Verwaltung komplexer *Job Trees* unter Einbeziehung aller Plattformen
  - uvm ...

Referenzen

Open Source Projekte und Dokumentation

- Konferenzbeiträge zu Whiznium

  https://www.mpsitech.com/documentation/presentations

- Aktuelle Laser Scanner Projektdokumentation auf GitHub

  https://mpsitech.github.io/Laser-Scanner-By-Platform

- Whizinum Dokumentation (Open Source Tool für modellbasierte Code-Generierung über den “embedded full-stack”)

  https://github.com/mpsitech/The-Whiznium-Documentation

- Whiznium Starter Kit Quellcode (Linux daemon)

  https://github.com/mpsitech/wzsk-Whiznium-StarterKit

- Whiznium Starter Kit Device Quellcde (RTL/C für alle Plattformen)

  https://github.com/mpsitech/wskd-Whiznium-StarterKit-Device

Vielen Dank!

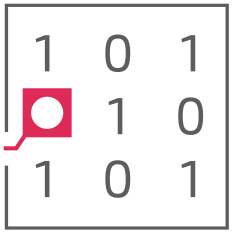
Fragen?

- https://www.linkedin.com/in/wirthmua
- https://github.com/mpsitech

Alexander Wirthmüller
Founder & Director

Phone: +49 (89) 4524 3826
Mobile: +49 (175) 918 5480
E-Mail: aw@mpsitech.com

MPSI Technologies GmbH
Agnes-Pockels-Bogen 1
80992 Munich
Germany
www.mpsitech.com



MPSI
TECHNOLOGIES