MAINTAINING PLATFORM FLEXIBILITY USING A MODEL-BASED SOFTWARE DESIGN APPROACH

MPSI TECHNOLOGIES

Alexander Wirthmüller
aw@mpsitechnologies.com

Introduction

About me

- Based in Munich
- Diploma in Electrical Engineering
- R&D Engineer at Mynaric (FPGA-based error-correction algorithms for free-space optical laser communications)
- Founder and Director at MPSI Technologies
- MPSI Technologies: make Embedded Software development more fun by replacing repetitive tasks by model-based source code generation

Making a case for platform flexibility

Silicon device landscape

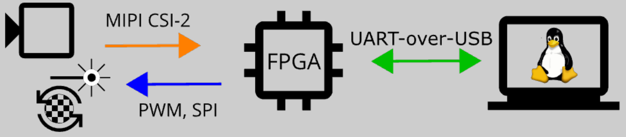
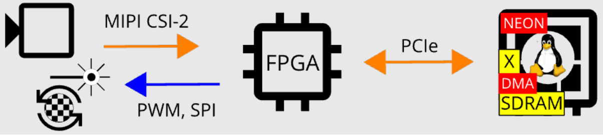
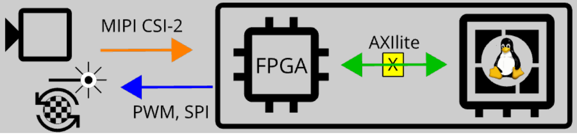
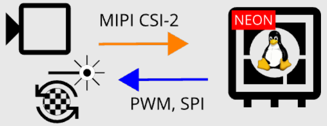
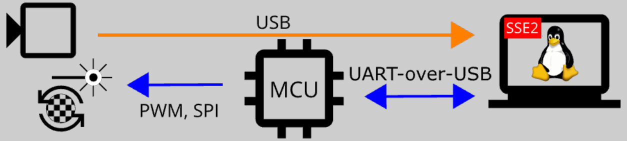
- Increasing number of contenders
- Specific strengths, can be:
  - Low static / dynamic power consumption
  - “Extra” features such as DSP blocks or high-performance or PHY-specific I/O’s
  - The right size / attractive price point
- Competition for FPGA-typical functionality from CPU’s featuring vector extensions / SIMD

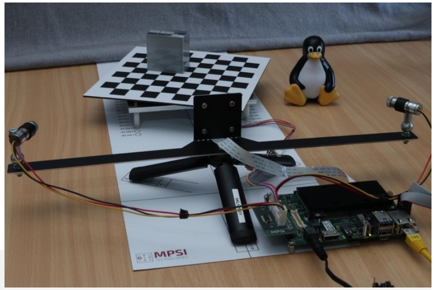
Application landscape

- Requirements are not written in stone, architectures need to adapt, e.g.:
  - 100Mbit/s vs. 1Gbit/s Ethernet
  - 1 megapixel vs. 5 megapixel camera modules
  - Single-channel vs. multi-channel DSP
- Skillset of available staff can influence FPGA vs. CPU decision making

Demo project: Tabletop 3D laser scanner

Hardware variants | Key software functionality

Vendor / device	Configuration
LATTICE SEMICONDUCTOR CrossLink-NX (v1)	
LATTICE SEMICONDUCTOR CrossLink-NX (v2)	 MIPI CSI-2 PWM, SPI FPGA PCIe NEON X DMA SDRAM
XILINX Zynq MICROCHIP PF SoC Microsemi Product Portfolio	
NXP i.MX6	
SILICON LABS UniversalBee	

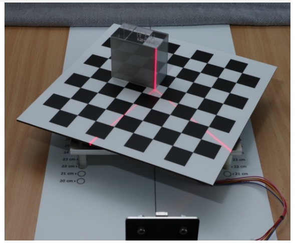


Features

- Turntable with stepper motor
- Tripod with camera/laser holder
- IMX335 MIPI CSI-2 camera (5MP) max. data rate 150MB/s @30fps
- Two adjustable red line lasers

< Bandwidth

- low < 500 kB/s
- medium < 50MB/s
- high > 50MB/s

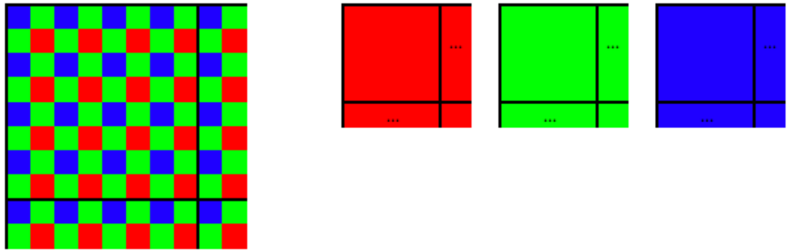


Demo project: Tabletop 3D laser scanner

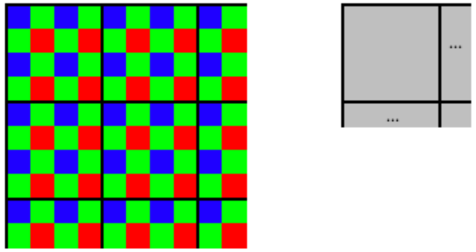
Hardware variants | Key software functionality

- Preview image acquisition

2560 x 1920 -> 160 x 120 (color)



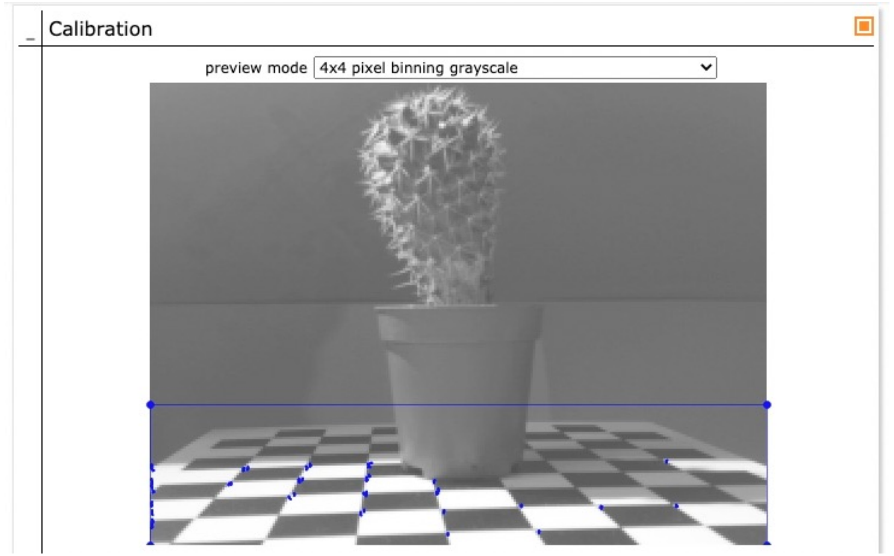
2048 x 1536 -> 512 x 384 (grayscale)



Demo project: Tabletop 3D laser scanner

Hardware variants | Key software functionality

- Preview image acquisition
- Checkerboard corner detection for orientation

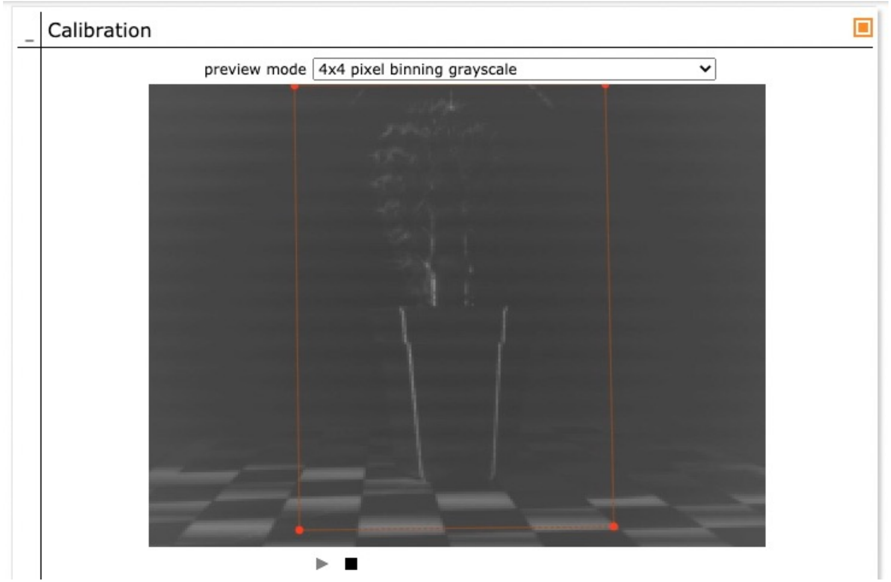


preview mode 4x4 pixel binning grayscale

Demo project: Tabletop 3D laser scanner

Hardware variants | Key software functionality

- Preview image acquisition
- Checkerboard corner detection for orientation
- On/off identification of line laser traces in frames



Demo project: Tabletop 3D laser scanner

Hardware variants | Key software functionality

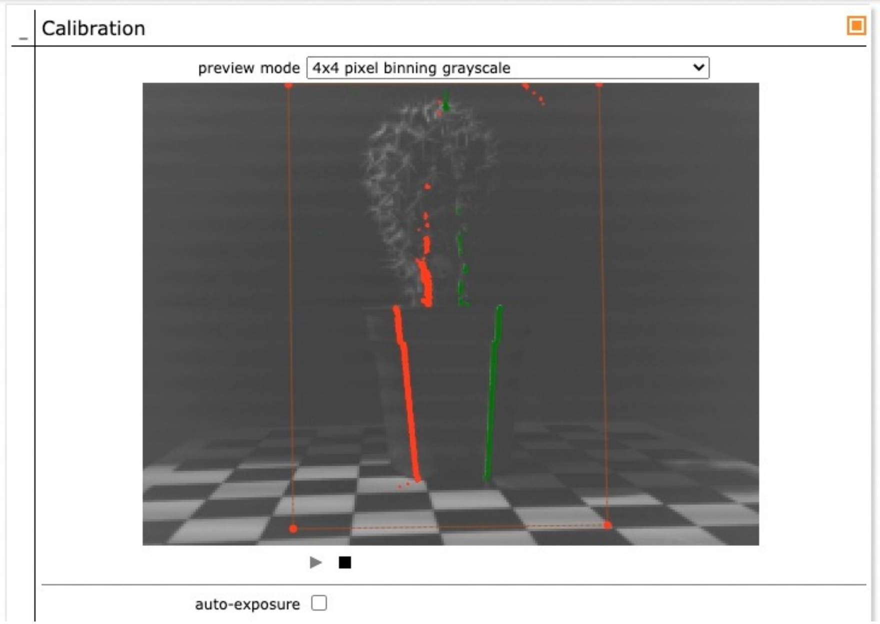
- Preview image acquisition
- Checkerboard corner detection for orientation
- On/off identification of line laser traces in frames

each algorithm can be performed

→ either on the Linux host or on the FPGA, ←

with varying load on the interconnect

→ yet each one host/FPGA source code tree ←



preview mode: 4x4 pixel binning grayscale

auto-exposure ☐

Spotlight on FPGA vendor IP 1/3

Dual-port RAM | MIPI CSI-2 PHY | Pipelined algorithm

- Most use cases can be covered using few parameters
  - Size
  - PortA/B widths 8/16/32/64 bits
  - PortA/B read- vs. write-only
  - Output buffer yes/no

```
Dpram (
  [vendor,]
  size [kB],
  width{A/B}{8/16/32/64},
  {rd/wr}only{A/B},
  buf{A/B}
)
```

```vhdl
entity Dpram_v1_0_size58kB is
  port (
    clkA: in std_logic;

    enA: in std_logic;
    weA: in std_logic;

    aA: in std_logic_vector(15 downto 0);
    drdA: out std_logic_vector(7 downto 0);
    dwrA: in std_logic_vector(7 downto 0);

    clkB: in std_logic;

    enB: in std_logic;

    aB: in std_logic_vector(13 downto 0);
    drdB: out std_logic_vector(31 downto 0)
  );
end Dpram_v1_0_size58kB;
```

- Route to vendor-independence
  - VHDL wrapper with standardized port names
  - Give instructions for configuring IP wizards

Spotlight on FPGA vendor IP 2/3

Dual-port RAM | MIPI CSI-2 PHY | Pipelined algorithm

- Silicon capabilities vary significantly
  - Lattice: integrated PHY/Decoder IP
  - Microchip: IOD IP, PLL and MIPI RX Decoder IP
  - Xilinx: SelectIO (from UltraScale+ native MIPI), “MIPI CSI-2 Receiver Subsystem“ IP to AXI Stream

```
Mipirx (
  [vendor,]
  fDDR [MHz],
  nLane{1,2,4},
  res{8,10,12,14}
)
```

- VHDL wrapper with standardized AXI Stream output

Spotlight on FPGA vendor IP 3/3

Dual-port RAM | MIPI CSI-2 PHY | Pipelined algorithm

- Example: Harris corner detection algorithm, matrix formula

$$M = \begin{bmatrix} \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}^2 & \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_{i,j=-2}^{2} \frac{\partial I}{\partial y}^2 \end{bmatrix} \qquad R = \det(M) - k\ \mathrm{trace}(M)^2$$

- Five-wide (14-/24-long) pipeline: signed multiplications and sums/differences
- Generic VHDL possible but limited control (latency, resource usage) for multiplications and three-input sums
- Optimized manual implementation requiring custom wait cycles
  - E.g. Xilinx: DSP48 macro and higher-level wizards “Adder/Subtracter“, “Multiplier“

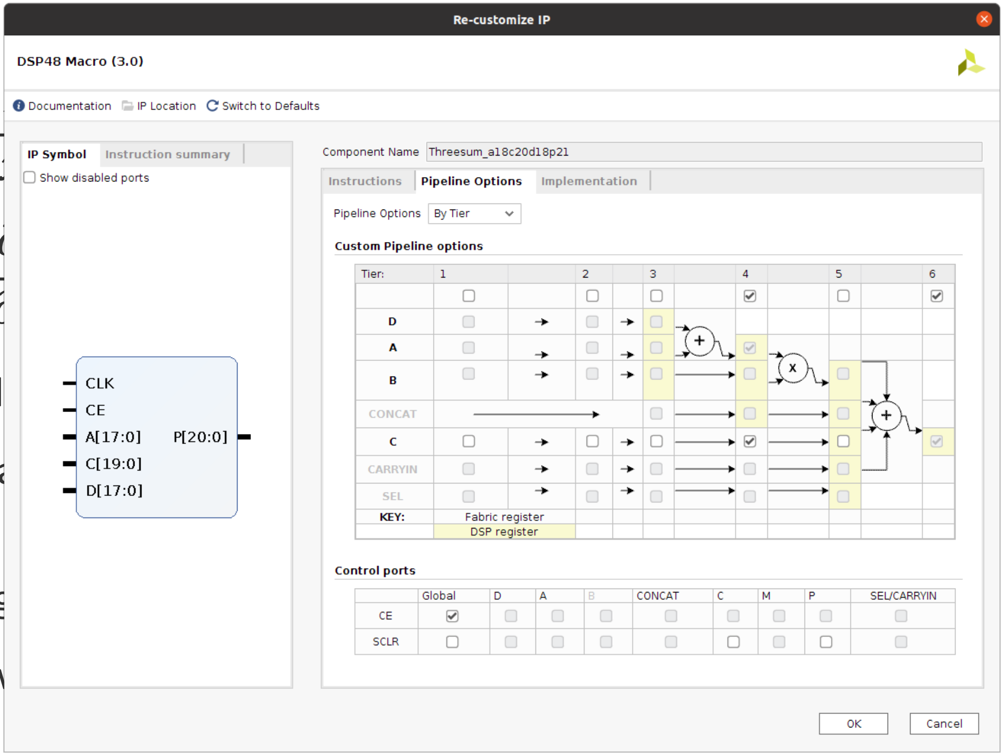
Spotlight on FPGA vendor IP 3/3

Dual-port RAM | MIPI CSI-2 PHY | Pipelined algorithm

- Example: Harris corner detection algorithm,

$$M = \begin{bmatrix} \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}^{2} & \sum_{i,j=-2}^{2} \\ \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_{i,j=-2}^{2} \end{bmatrix}$$

- Five-wide (14-/24-long) pipeline: signed mul
- Generic VHDL possible but limited control (l input sums
- Optimized manual implementation requirin
  - E.g. Xilinx: DSP48 macro and higher-level w


Re-customize IP

DSP48 Macro (3.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Instruction summary

☐ Show disabled ports

Component Name: Threesum_a18c20d18p21

Instructions | Pipeline Options | Implementation

Pipeline Options: By Tier

Custom Pipeline options

Tier:	1	2	3	4	5	6
	☐	☐	☐	✓	☐	✓
D	☐	☐	☐			
A	☐	☐	☐	✓		
B	☐	☐	☐	☐	☐	
CONCAT			☐	☐	☐	
C	☐	☐	☐	✓	☐	✓
CARRYIN	☐	☐	☐	☐	☐	
SEL	☐	☐	☐	☐	☐	
KEY:	Fabric register					
	DSP register					

Control ports

	Global	D	A	B	CONCAT	C	M	P	SEL/CARRYIN
CE	✓	☐	☐	☐	☐	☐	☐	☐	☐
SCLR	☐	☐	☐	☐	☐	☐	☐	☐	☐

OK | Cancel

Spotlight on FPGA vendor IP 3/3

Dual-port RAM | MIPI CSI-2 PHY | Pipelined algorithm

- Example: Harris corner detection algorithm, matrix formula

$$M = \begin{bmatrix} \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}^{2} & \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_{i,j=-2}^{2} \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_{i,j=-2}^{2} \frac{\partial I}{\partial y}^{2} \end{bmatrix} \qquad R$$

- Five-wide (14-/24-long) pipeline: signed multiplications and s
- Generic VHDL possible but limited control (latency, resource input sums
- Optimized manual implementation requiring custom wait cy
  - E.g. Xilinx: DSP48 macro and higher-level wizards “Adder/S

```
------------------------------------------------------------------------
-- implementation: Harris score pipeline forward operation (fwd)
------------------------------------------------------------------------

process (reset, mclk, stateFwd)

begin
  if reset='1' then
    -- ...

  elsif rising_edge(mclk) then
    if stateFwd=stateFwdRun then
      if ceScore='1' then
        xsqr3p1 <= xsqr(71 downto 54);
        xsqr3p2 <= xsqr3p1;

        xsqr4p1 <= xsqr(89 downto 72);
        xsqr4p2 <= xsqr4p1;

        colsumX4p1 <= colsumX(104 downto 84);
        colsumX4p2 <= colsumX4p1;
        colsumX4p3 <= colsumX4p2;
        colsumX4p4 <= colsumX4p3;


        -- ...


        diffI_IIp1 <= diffI_II;
      end if;
    end if;
  end if;
end process;
```

Spotlight on the host-FPGA interconnect

Layer model | Hardware abstraction | Command invocation | Buffer transfer

	Linux host	**FPGA**
application layer	C++ data processing	RTL algorithms, state machines, etc.
application layer handoff	target-specific C++ API library	target module RTL handshake
protocol layer	encode/decode C++ code	host interface RTL module
hardware abstraction layer	device driver	soft IP
physical layer	silicon IP and copper wires standard-compliant	

Spotlight on the host-FPGA interconnect

Layer model | Hardware abstraction | Command invocation | Buffer transfer

	Complexity	Net bandwidth	Support	Conditions
UART	2 wire	400 kB/s	i.MX6 (all 32/64bit SoC's)	4 Mbps on-PCB routing
UART over USB	2 wire	417 kB/s	FTDI	x64 host USB2.0 hi-speed, FT232R
SPI	3 wire	4.8 MB/s	OMAP3xxx (all 32/64bit SoC's)	40 MHz on-PCB routing
AXIlite	(on-chip)	50 MB/s	Zynq (all FPGA-SoC's)	32 bit words, 100 MHz clock
PCIe	3 diff. pair, 4 wire	250 MB/s	CrosslinkNX (all mid-range FPGA's)	one lane PCIe 1.x, 2.5 Gbps
AXI4	(on-chip)	776 MB/s	PolarFire SoC (all FPGA-SoC's)	64 bit x 256 bursts, 100 MHz clock

Spotlight on the host-FPGA interconnect

Layer model | Hardware abstraction | Command invocation | Buffer transfer

- Linux host
  - Character device driver (`open()`, `ioctl()`, `read()`, `write()`, `close()`)
  - Easily applicable for UART, SPI, AXIlite
  - User I/O API for PCIe and AXI4 with DMA (works with interrupts and callbacks)

- FPGA design
  - Generic UART, SPI, AXIlite modules for basic rx/tx(<number of words>)
  - PCIe IP by four major vendors free but not Open Source

Spotlight on the host-FPGA interconnect

Layer model | Hardware abstraction | Command invocation | Buffer transfer

- RTL module examples

```vhdl
entity Uartrx_v1_1 is
  generic(
    fMclk: natural range 1 to 1000000;

    fSclk: natural range 100 to 50000000
  );
  port(
    reset: in std_logic;

    mclk: in std_logic;

    req: in std_logic;
    ack: out std_logic;
    dne: out std_logic;

    len: in std_logic_vector(16 downto 0);

    d: out std_logic_vector(7 downto 0);
    strbD: out std_logic;

    rxd: in std_logic;

    burst: in std_logic
  );
end Uartrx_v1_1;
```

```vhdl
entity Spislave_v1_0 is
  generic (
    cpol: std_logic := '0';
    cpha: std_logic := '0';

    nssByteNotXfer: std_logic := '0';
    misoPrecphaNotCpha: std_logic := '0'
  );
  port (
    reset: in std_logic;

    mclk: in std_logic;

    req: in std_logic;
    ack: out std_logic;
    dne: out std_logic;

    len: in std_logic_vector(16 downto 0);

    send: in std_logic_vector(7 downto 0);
    strbSend: out std_logic;

    recv: out std_logic_vector(7 downto 0);
    strbRecv: out std_logic;

    nss: in std_logic;
    sclk: in std_logic;
    mosi: in std_logic;
    miso: inout std_logic
  );
end Spislave_v1_0;
```

```vhdl
process (extresetn, extclk)
  -- ...

begin
  if extresetn='0' then
    stateOp <= stateOpInit;
    -- ...

  elsif rising_edge(extclk) then
    if stateOp=stateOpInit then
      -- ...
      stateOp <= stateOpIdle;

    elsif stateOp=stateOpIdle the
      if (axi_bvalid='1' and loc_
        if rdyRx='1'then
          enRx <= '1';
        end if;

        stateOp <= stateOpWrrdyA;

      elsif (axi_bvalid='1' and l
        if rdyTx='1'then
          enTx <= '1';
        end if;

        stateOp <= stateOpRdrdyA;
      end if;

    -- ...
    end if;
  end if;
end process;
```

```vhdl
entity Axirx_v2_0 is
  port(
    reset: in std_logic;

    mclk: in std_logic;

    req: in std_logic;
    ack: out std_logic;
    dne: out std_logic;

    len: in std_logic_vector(21 downto 0); -- in words, max. 2^22-1

    d: out std_logic_vector(31 downto 0);
    strbD: out std_logic;

    rdyRx: out std_logic;
    enRx: in std_logic;

    rx: in std_logic_vector(31 downto 0);
    strbRx: in std_logic
  );
end Axirx_v2_0;
```

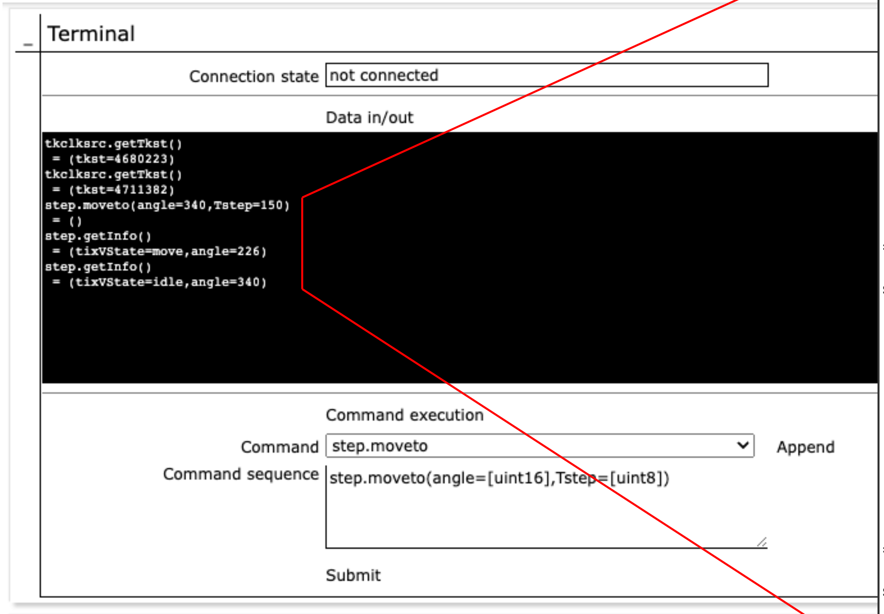
AXIlite

Spotlight on the host-FPGA interconnect

Layer model | Hardware abstraction | Command invocation | Buffer transfer

- Host: C++ API library forms byte code and initiates transfers guarded by CRC

	Linux host	FPGA
application layer	C++ data processing	RTL algorithms, state machines, etc.
application layer handoff	target-specific C++ API library	target module RTL handshake
protocol layer	encode/decode C++ code	host interface RTL module
hardware abstraction layer	device driver	soft IP
physical layer	silicon IP and copper wires standard-compliant	



Terminal

Connection state: not connected

Data in/out

```
tkclksrc.getTkst()
 = (tkst=4680223)
tkclksrc.getTkst()
 = (tkst=4711382)
step.moveto(angle=340,Tstep=150)
 = ()
step.getInfo()
 = (tixVState=move,angle=226)
step.getInfo()
 = (tixVState=idle,angle=340)
```

Command execution

Command: step.moveto | Append

Command sequence: step.moveto(angle=[uint16],Tstep=[uint8])

Submit

step.moveto(angle=340,Tstep=150)

```
   tx 0x02 06 01 0005 78FA
        1.  2.  3.   4.    5.
```
1. hostifToCmdinv
2. controller:step
3. command:moveto
4. length:5
5. CRC

```
   tx 0x0154 96 7B65
        1.   2.   3.
```
1. angle:340(uint16)
2. Tstep:150(uint8)
3. CRC

```
   rx 0xAAAA
        ACK
```
= ()

step.getInfo()

```
   tx 0x01 06 00 0005 F865
        1.  2.  3.   4.    5.
```
1. cmdretToHostif
2. controller:step
3. command:getInfo
4. length:5
5. CRC

```
   rx 0x00 00E2 A8E6
        1.   2.   3.
```
1. state:moving
2. angle:226(uint16)
3. CRC

=(tixVState=move,angle=226)

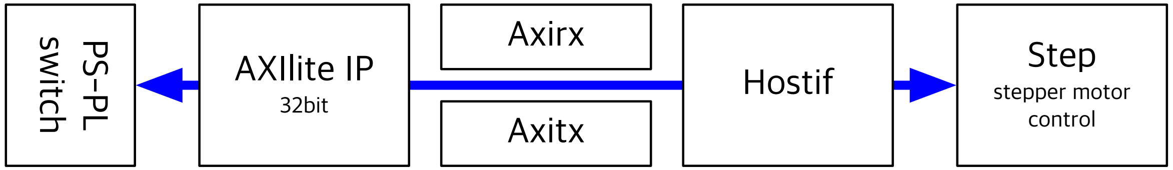
step.getInfo()

```
   tx 0x01 06 00 0005 F865
   rx 0x00 0154 AD52
```
=(tixVState=idle,angle=340)

Spotlight on the host-FPGA interconnect

Layer model | Hardware abstraction | Command invocation | Buffer transfer

- FPGA: ”host interface” module decodes the byte string and triggers a handshake with the ”step” target module

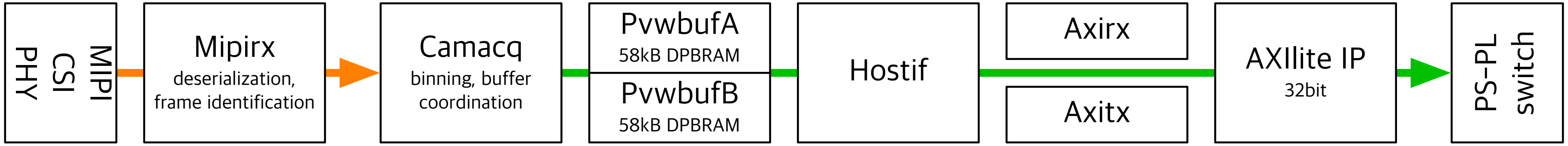


```
entity Step is
  generic (
    fMclk: natural range 1 to 1000000 := 50000 -- in kHz
  );
  port (
    reset: in std_logic;
    mclk: in std_logic;
    tkclk: in std_logic;

    ...

    reqInvMoveto: in std_logic;
    ackInvMoveto: out std_logic;

    movetoAngle: in std_logic_vector(15 downto 0);
    movetoTstep: in std_logic_vector(7 downto 0);

    ...

    nslp: out std_logic;
    m0: inout std_logic;
    dir: out std_logic;
    step0: out std_logic
  );
end Step;
```

Spotlight on the host-FPGA interconnect

Layer model | Hardware abstraction | Command invocation | Buffer transfer

- FPGA: reduce 2560x1920 YUV images @30fps (150MB/s)
  to 160x120 RGB images (1.73MB/s), then provide to host in A/B buffer



- Host: poll the buffer status, initiate buffer transfer and display

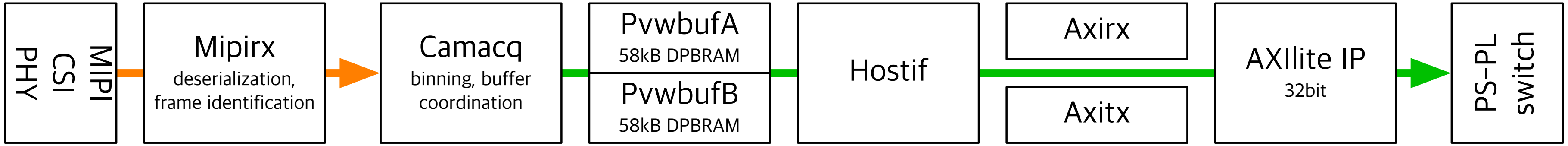
```
while (true) {
  if (shrdat.cancelPvw) break;

  shrdat.mPvw.lock("JobWzskAcqFpgapvw", "runPvw[2]");

  srv->srcarty->camacq_getPvwinfo(tixVPvwbufstate, tkst);

  if ((tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::ABUF) || (tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::BBUF)) {
    if (tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::ABUF) srv->srcarty->shrdat.hw.readPvwabufFromCamacq(sizeBuf, buf, datalen);
    else if (tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::BBUF) srv->srcarty->shrdat.hw.readPvwbbufFromCamacq(sizeBuf, buf, datalen);

    shrdat.mPvw.unlock("JobWzskAcqFpgapvw", "runPvw[2]");

  } else {
    shrdat.mPvw.unlock("JobWzskAcqFpgapvw", "runPvw[3]");

    nanosleep(&deltat, NULL);
  };
};
```

	Linux host	**FPGA**
application layer	C++ data processing	RTL algorithms, tate machines, etc.
application layer handoff	target-specific C++ API library	target module RTL handshake
protocol layer	encode/decode C++ code	host interface RTL module
hardware abstraction layer	device driver	soft IP
physical layer	silicon IP and copper wires standard-compliant	

Spotlight on the host-FPGA interconnect

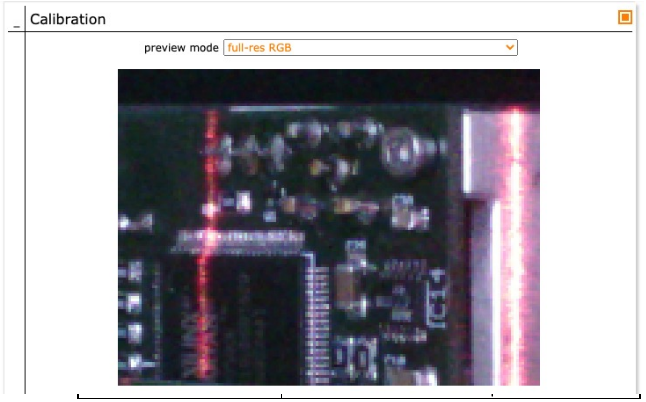
Layer model | Hardware abstraction | Command invocation | Buffer transfer

- FPGA: reduce 2560x1920 YUV images @30fps (150MB/s)
  to 160x120 RGB images (1.73MB/s), then provide to host in A/B buffer



- Host: poll the buffer status, initiate buffer transfer and display

```
while (true) {
  if (shrdat.cancelPvw) break;

  shrdat.mPvw.lock("JobWzskAcqFpgapvw", "runPvw[2]");

  srv->srcarty->camacq_getPvwinfo(tixVPvwbufstate, tkst);

  if ((tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::ABUF) || (tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::BBUF)) {
    if (tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::ABUF) srv->srcarty->shrdat.hw.readPvwabufFromCamacq(sizeBuf, buf, datalen);
    else if (tixVPvwbufstate == VecVWskdArtyCamacqPvwbufstate::BBUF) srv->srcarty->shrdat.hw.readPvwbbufFromCamacq(sizeBuf, buf, datalen);

    shrdat.mPvw.unlock("JobWzskAcqFpgapvw", "runPvw[2]");

  } else {
    shrdat.mPvw.unlock("JobWzskAcqFpgapvw", "runPvw[3]");

    nanosleep(&deltat, NULL);
  };
};
```



preview mode full-res RGB

Spotlight on host software

Job tree | Vector extensions

- Use fine granularity for C++ classes, and well-defined interfaces
- Example: user session (preview image) Lattice FPGA vs. Xilinx FPGA-SoC vs. NXP i.MX6

```
+ RootWzsk
  + SessWzsk
    + CrdWzskLlv
      + PnlWzskLlvCamera
        + JobWzskAcqPreview/S
          + JobWzskAcqFpgapvw/S
            - JobWzskSrcClnxevb/S
```

```
+ RootWzsk
  + SessWzsk
    + CrdWzskLlv
      + PnlWzskLlvCamera
        + JobWzskAcqPreview/S
          + JobWzskAcqFpgapvw/S
            - JobWzskSrcArty/S
```

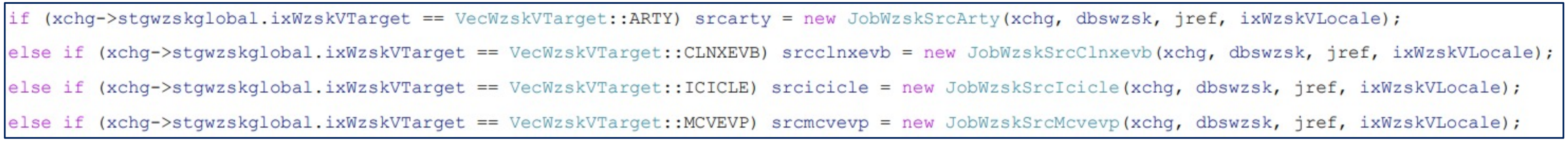
```
+ RootWzsk
  + SessWzsk
    + CrdWzskLlv
      + PnlWzskLlvCamera
        + JobWzskAcqPreview/S
          - JobWzskSrcV4l2/S
```

- Web UI *jobs* in blue, communicate over HTTP(S) using JSON/XML
- Preview acquisition *job* in green, reacts on new frame available and passes it on to web UI job
- FPGA preview *job* in orange, runs thread polling FPGA preview buffer status and transfers data
- Source *jobs* in red, interact with FPGA (UART-over-USB vs. AXIlite) vs. with camera using V4L2 API

Spotlight on host software

Job tree | Vector extensions

- Decision on which *jobs* to instantiate using global flag



```
if (xchg->stgwzskglobal.ixWzskVTarget == VecWzskVTarget::ARTY) srcarty = new JobWzskSrcArty(xchg, dbswzsk, jref, ixWzskVLocale);
else if (xchg->stgwzskglobal.ixWzskVTarget == VecWzskVTarget::CLNXEVB) srcclnxevb = new JobWzskSrcClnxevb(xchg, dbswzsk, jref, ixWzskVLocale);
else if (xchg->stgwzskglobal.ixWzskVTarget == VecWzskVTarget::ICICLE) srcicicle = new JobWzskSrcIcicle(xchg, dbswzsk, jref, ixWzskVLocale);
else if (xchg->stgwzskglobal.ixWzskVTarget == VecWzskVTarget::MCVEVP) srcmcvevp = new JobWzskSrcMcvevp(xchg, dbswzsk, jref, ixWzskVLocale);
```

Spotlight on host software

Job tree | Vector extensions

- Use `#ifdef` guards to determine architecture

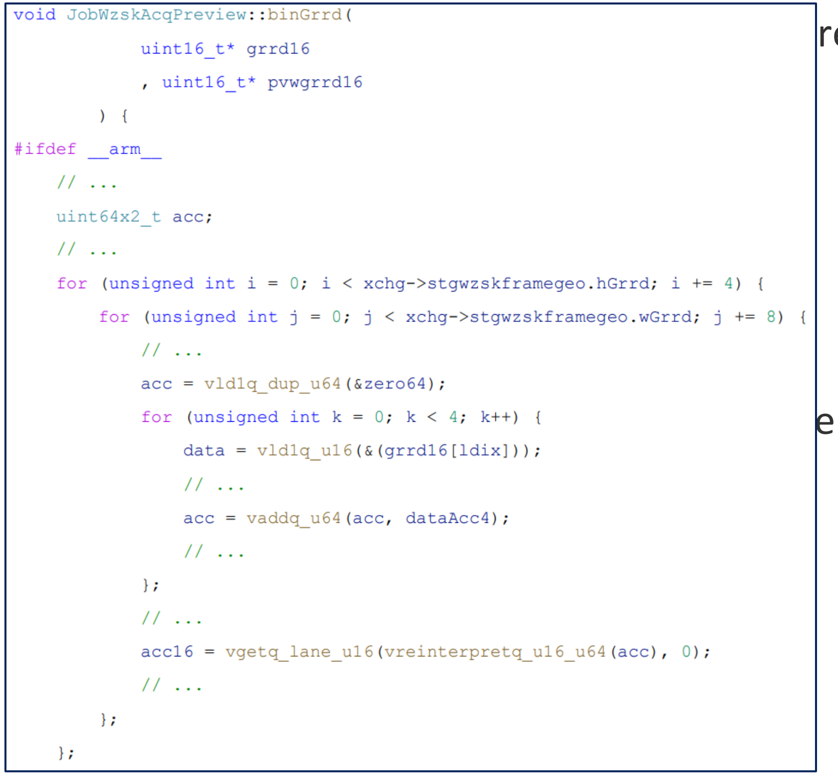
```
#ifdef __arm__
    #include <arm_neon.h>
#elif __x86_64__
    #include <emmintrin.h>
#endif
```

- Grayscale binning for ARM -> intel x64 -> others

Spotlight on host software

Job tree | Vector extensions

- re
- ers



```
void JobWzskAcqPreview::binGrrd(
            uint16_t* grrd16
            , uint16_t* pvwgrrd16
        ) {
#ifdef __arm__
    // ...
    uint64x2_t acc;
    // ...
    for (unsigned int i = 0; i < xchg->stgwzskframegeo.hGrrd; i += 4) {
        for (unsigned int j = 0; j < xchg->stgwzskframegeo.wGrrd; j += 8) {
            // ...
            acc = vld1q_dup_u64(&zero64);
            for (unsigned int k = 0; k < 4; k++) {
                data = vld1q_u16(&(grrd16[ldix]));
                // ...
                acc = vaddq_u64(acc, dataAcc4);
                // ...
            };
            // ...
            acc16 = vgetq_lane_u16(vreinterpretq_u16_u64(acc), 0);
            // ...
        };
    };
```

Spotlight on host software

Job tree | Vector extensions

- re
-

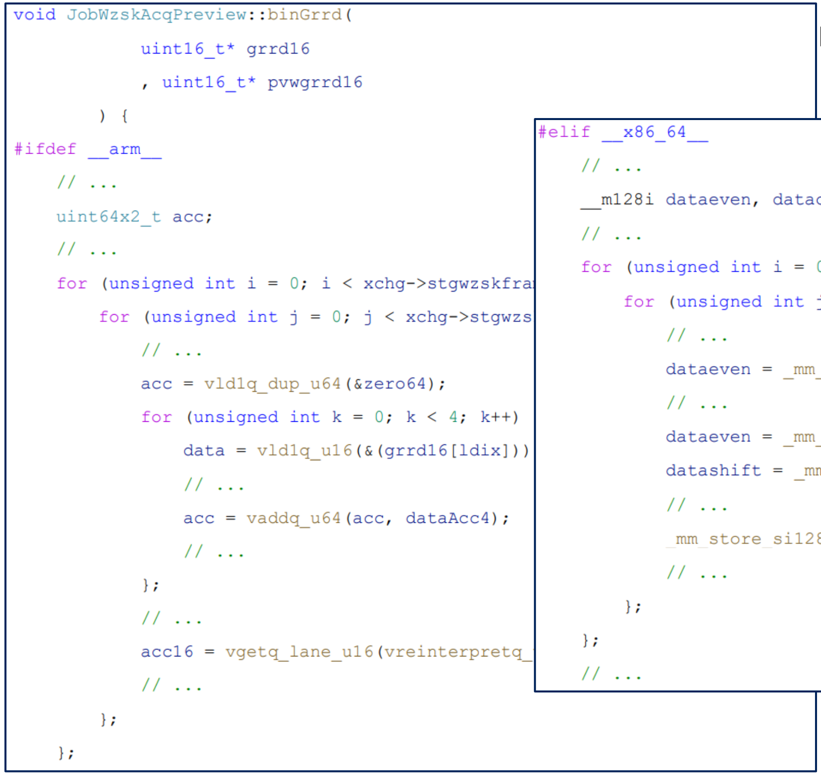
```
void JobWzskAcqPreview::binGrrd(
            uint16_t* grrd16
            , uint16_t* pvwgrrd16
        ) {
#ifdef __arm__
    // ...
    uint64x2_t acc;
    // ...
    for (unsigned int i = 0; i < xchg->stgwzskfra
        for (unsigned int j = 0; j < xchg->stgwzs
            // ...
            acc = vld1q_dup_u64(&zero64);
            for (unsigned int k = 0; k < 4; k++)
                data = vld1q_u16(&(grrd16[ldix]))
                // ...
                acc = vaddq_u64(acc, dataAcc4);
                // ...
            };
            // ...
            acc16 = vgetq_lane_u16(vreinterpretq_
            // ...
        };
    };
```

```
#elif __x86_64__
    // ...
    __m128i dataeven, dataodd, datashift;
    // ...
    for (unsigned int i = 0; i < xchg->stgwzskframegeo.hGrrd; i += 2) {
        for (unsigned int j = 0; j < xchg->stgwzskframegeo.wGrrd; j += 8) {
            // ...
            dataeven = _mm_load_si128((const __m128i*) &(grrd16[ldix]));
            // ...
            dataeven = _mm_add_epi16(dataeven, dataodd);
            datashift = _mm_slli_si128(dataeven, 2);
            // ...
            _mm_store_si128 ((__m128i*) buf, dataeven);
            // ...
        };
    };
    // ...
```

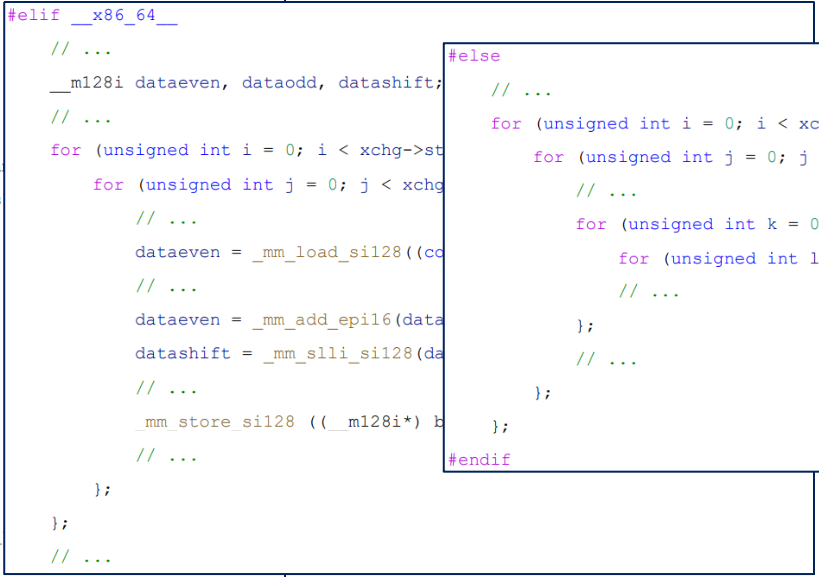
Spotlight on host software

Job tree | Vector extensions

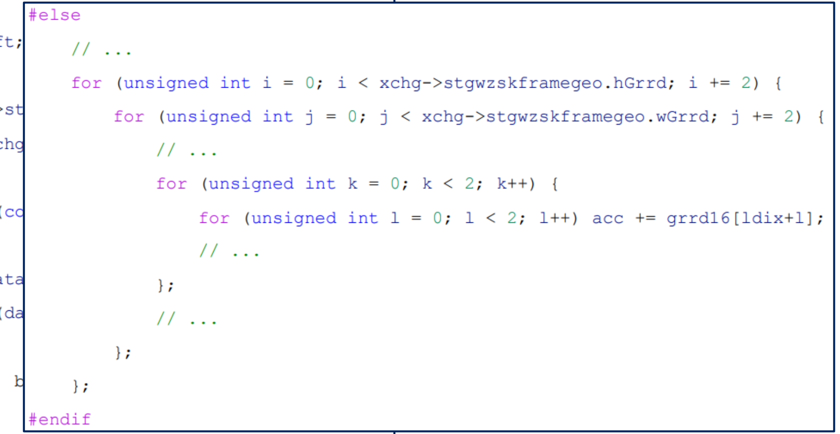
- re



```
void JobWzskAcqPreview::binGrrd(
            uint16_t* grrd16
            , uint16_t* pvwgrrd16
        ) {
#ifdef __arm__
    // ...
    uint64x2_t acc;
    // ...
    for (unsigned int i = 0; i < xchg->stgwzskfra
        for (unsigned int j = 0; j < xchg->stgwzs
            // ...
            acc = vld1q_dup_u64(&zero64);
            for (unsigned int k = 0; k < 4; k++)
                data = vld1q_u16(&(grrd16[ldix]))
                // ...
                acc = vaddq_u64(acc, dataAcc4);
                // ...
            };
            // ...
            acc16 = vgetq_lane_u16(vreinterpretq_
            // ...
        };
    };
```



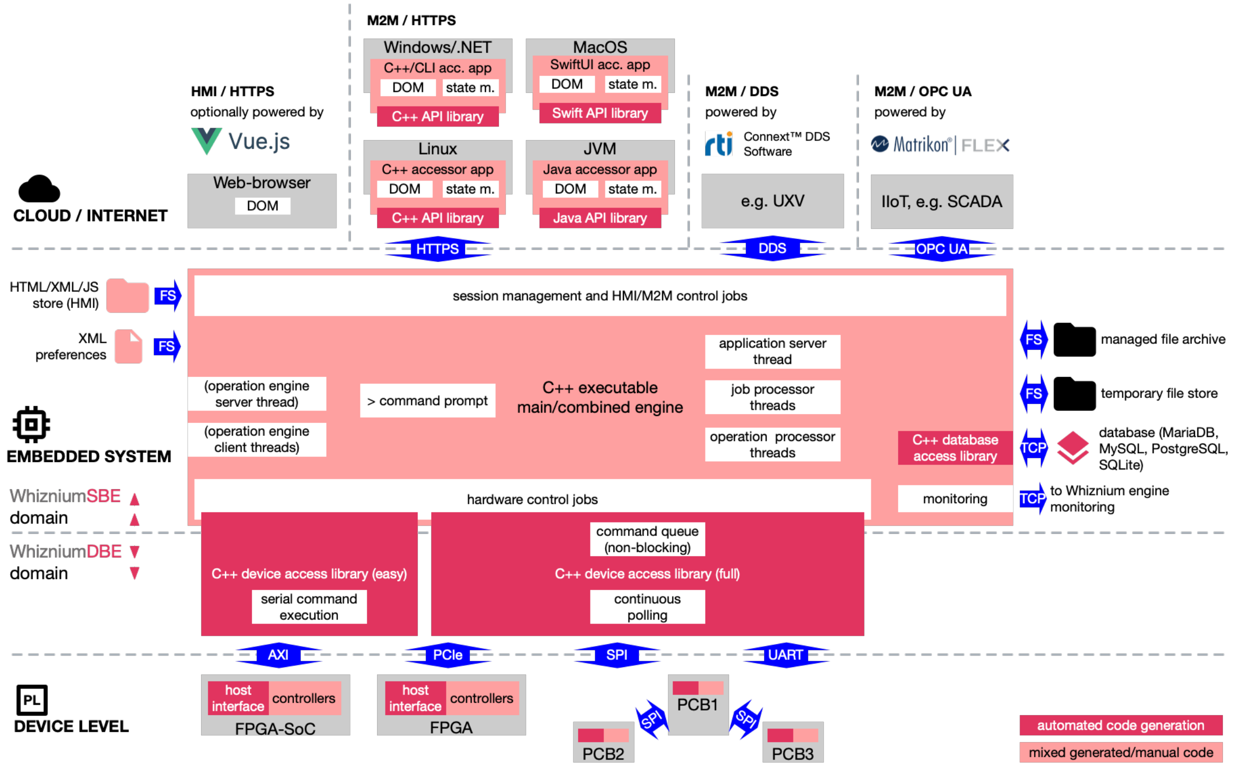
```
#elif __x86_64__
    // ...
    __m128i dataeven, dataodd, datashift;
    // ...
    for (unsigned int i = 0; i < xchg->st
        for (unsigned int j = 0; j < xchg
            // ...
            dataeven = _mm_load_si128((co
            // ...
            dataeven = _mm_add_epi16(data
            datashift = _mm_slli_si128(da
            // ...
            _mm_store_si128 ((__m128i*) b
            // ...
        };
    };
    // ...
```



```
#else
    // ...
    for (unsigned int i = 0; i < xchg->stgwzskframegeo.hGrrd; i += 2) {
        for (unsigned int j = 0; j < xchg->stgwzskframegeo.wGrrd; j += 2) {
            // ...
            for (unsigned int k = 0; k < 2; k++) {
                for (unsigned int l = 0; l < 2; l++) acc += grrd16[ldix+l];
                // ...
            };
            // ...
        };
    };
#endif
```

Model-based software design with Whiznium

Overview | Composition | Example



- Coverage of the "Embedded Full Stack"
- WhizniumDBE ("Device Builder's Edition") for FPGA / MCU level and its host access libraries (primary languages: C, VHDL)
- WhizniumSBE ("Service Builder's Edition") for Embedded Linux and "outside world" levels (primary languages: C++, HTML)

Model-based software design with Whiznium

Overview | Composition | Example

- Successive model composition within an SQL database using import (I) and generation (G) steps
- Output of source code trees only thereafter
- Text-based model files (”diffable”)

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)

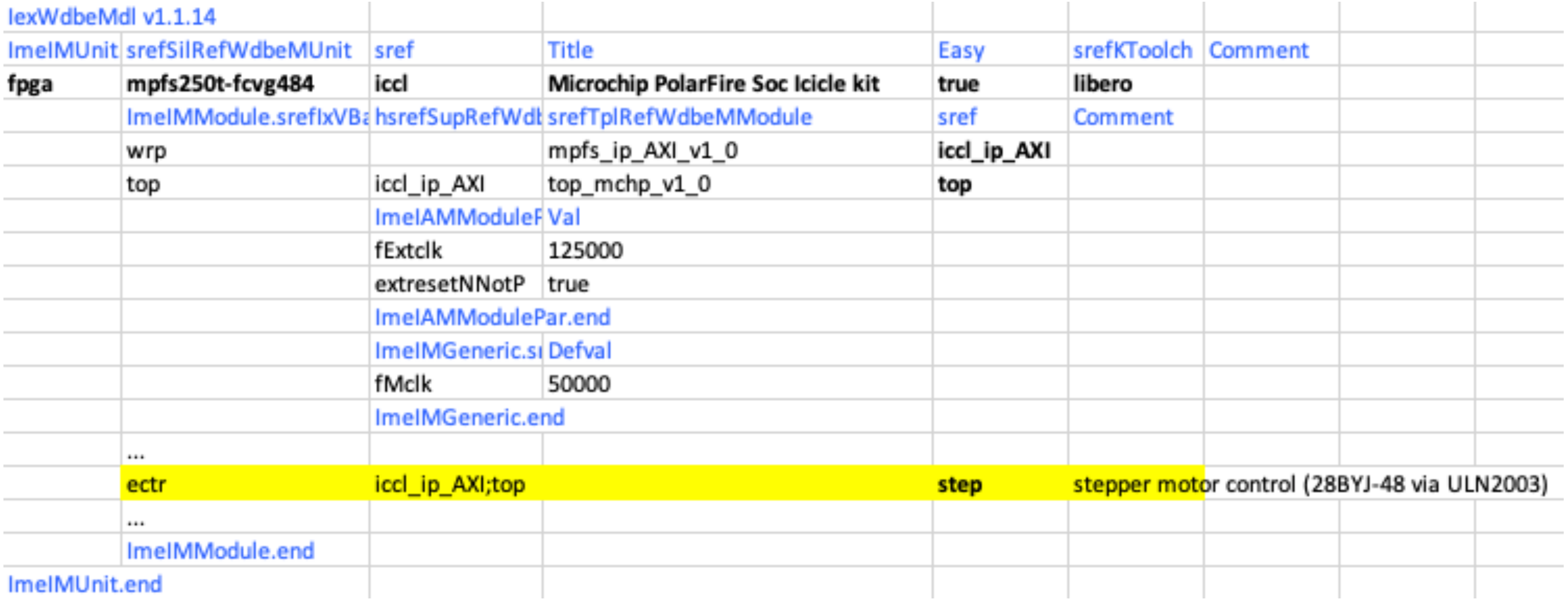
Model-based software design with Whiznium

Overview | Composition | Example

- Module definition, command definition, fine structure

Model-based software design with Whiznium

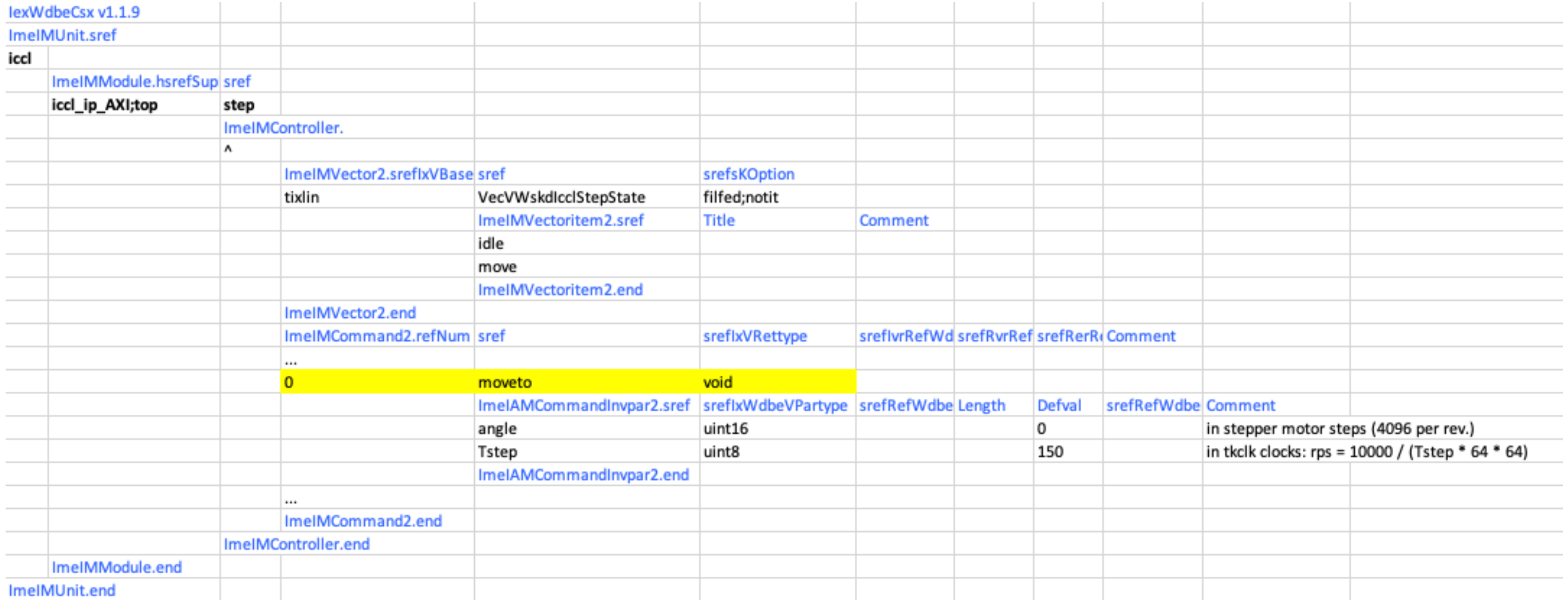
Overview | Composition | Example



IexWdbeMdl v1.1.14						
ImeIMUnit	srefSilRefWdbeMUnit	sref	Title	Easy	srefKToolch	Comment
fpga	**mpfs250t-fcvg484**	**iccl**	**Microchip PolarFire Soc Icicle kit**	**true**	**libero**	
	ImeIMModule.srefIxVBa	hsrefSupRefWdl	srefTplRefWdbeMModule	sref	Comment	
	wrp		mpfs_ip_AXI_v1_0	**iccl_ip_AXI**		
	top	iccl_ip_AXI	top_mchp_v1_0	**top**		
		ImeIAMModuleP	Val			
		fExtclk	125000			
		extresetNNotP	true			
		ImeIAMModulePar.end				
		ImeIMGeneric.s	Defval			
		fMclk	50000			
		ImeIMGeneric.end				
	...					
	ectr	iccl_ip_AXI;top		**step**	stepper motor control (28BYJ-48 via ULN2003)	
	...					
	ImeIMModule.end					
ImeIMUnit.end						

Model-based software design with Whiznium

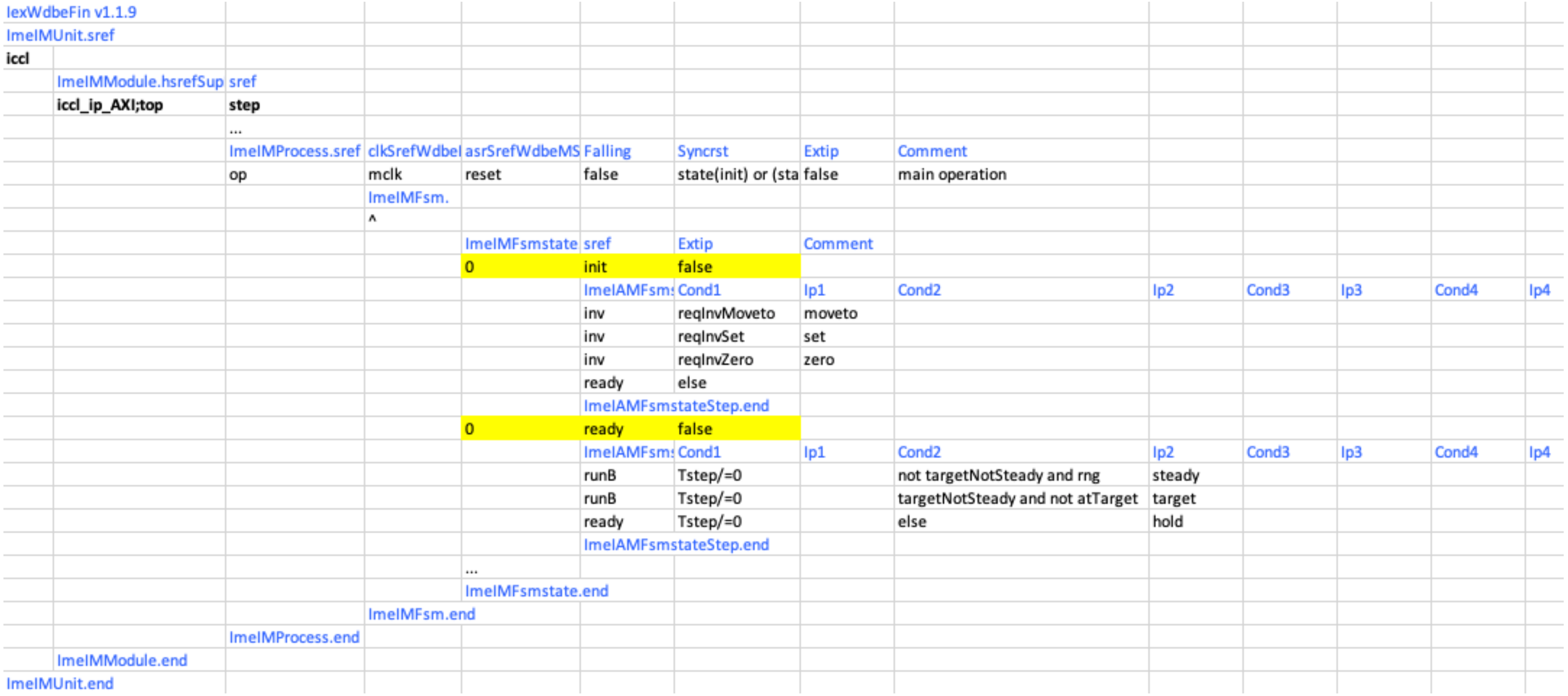
Overview | Composition | Example



IexWdbeCsx v1.1.9										
ImeIMUnit.sref										
iccl										
	ImeIMModule.hsrefSup	sref								
	iccl_ip_AXI;top	step								
		ImeIMController.								
		^								
			ImeIMVector2.srefIxVBase	sref	srefsKOption					
			tixlin	VecVWskdIcclStepState	filfed;notit					
				ImeIMVectoritem2.sref	Title	Comment				
				idle						
				move						
				ImeIMVectoritem2.end						
			ImeIMVector2.end							
			ImeIMCommand2.refNum	sref	srefIxVRettype	srefIvrRefWd	srefRvrRef	srefRerR	Comment	
			...							
			0	moveto	void					
				ImeIAMCommandInvpar2.sref	srefIxWdbeVPartype	srefRefWdbe	Length	Defval	srefRefWdbe	Comment
				angle	uint16			0		in stepper motor steps (4096 per rev.)
				Tstep	uint8			150		in tkclk clocks: rps = 10000 / (Tstep * 64 * 64)
				ImeIAMCommandInvpar2.end						
			...							
			ImeIMCommand2.end							
		ImeIMController.end								
	ImeIMModule.end									
ImeIMUnit.end										

Model-based software design with Whiznium

Overview | Composition | Example



IexWdbeFin v1.1.9													
ImeIMUnit.sref													
iccl													
	ImeIMModule.hsrefSup	sref											
	iccl_ip_AXI;top	step											
		...											
		ImeIMProcess.sref	clkSrefWdbel	asrSrefWdbeMS	Falling	Syncrst	Extip	Comment					
		op	mclk	reset	false	state(init) or (sta	false	main operation					
			ImeIMFsm.										
			^										
				ImeIMFsmstate	sref	Extip	Comment						
				0	init	false							
					ImeIAMFsm:	Cond1	Ip1	Cond2	Ip2	Cond3	Ip3	Cond4	Ip4
					inv	reqInvMoveto	moveto						
					inv	reqInvSet	set						
					inv	reqInvZero	zero						
					ready	else							
					ImeIAMFsmstateStep.end								
				0	ready	false							
					ImeIAMFsm:	Cond1	Ip1	Cond2	Ip2	Cond3	Ip3	Cond4	Ip4
					runB	Tstep/=0		not targetNotSteady and rng	steady				
					runB	Tstep/=0		targetNotSteady and not atTarget	target				
					ready	Tstep/=0		else	hold				
					ImeIAMFsmstateStep.end								
				...									
				ImeIMFsmstate.end									
			ImeIMFsm.end										
		ImeIMProcess.end											
	ImeIMModule.end												
ImeIMUnit.end													

Model-based software design with Whiznium

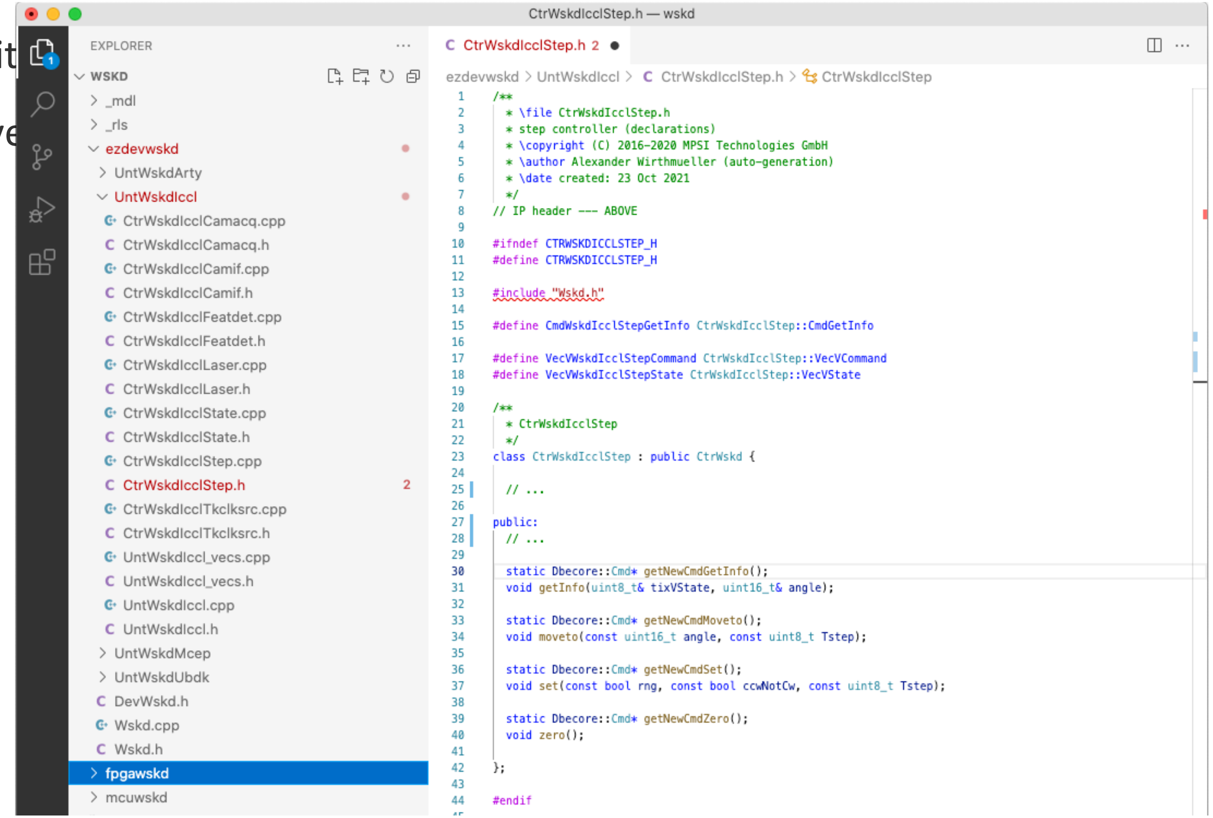
Overview | Composition | Example

- Module definition, command definition, fine structure
- Linux side developer-facing: executable API method

Model-based software design with Whiznium

Overview | Composition | Example

- Module definit
- Linux side deve



```
/**
  * \file CtrWskdIcclStep.h
  * step controller (declarations)
  * \copyright (C) 2016-2020 MPSI Technologies GmbH
  * \author Alexander Wirthmueller (auto-generation)
  * \date created: 23 Oct 2021
  */
// IP header --- ABOVE

#ifndef CTRWSKDICCLSTEP_H
#define CTRWSKDICCLSTEP_H

#include "Wskd.h"

#define CmdWskdIcclStepGetInfo CtrWskdIcclStep::CmdGetInfo

#define VecVWskdIcclStepCommand CtrWskdIcclStep::VecVCommand
#define VecVWskdIcclStepState CtrWskdIcclStep::VecVState

/**
  * CtrWskdIcclStep
  */
class CtrWskdIcclStep : public CtrWskd {

  // ...

public:
  // ...

  static Dbecore::Cmd* getNewCmdGetInfo();
  void getInfo(uint8_t& tixVState, uint16_t& angle);

  static Dbecore::Cmd* getNewCmdMoveto();
  void moveto(const uint16_t angle, const uint8_t Tstep);

  static Dbecore::Cmd* getNewCmdSet();
  void set(const bool rng, const bool ccwNotCw, const uint8_t Tstep);

  static Dbecore::Cmd* getNewCmdZero();
  void zero();

};

#endif
```

Model-based software design with Whiznium

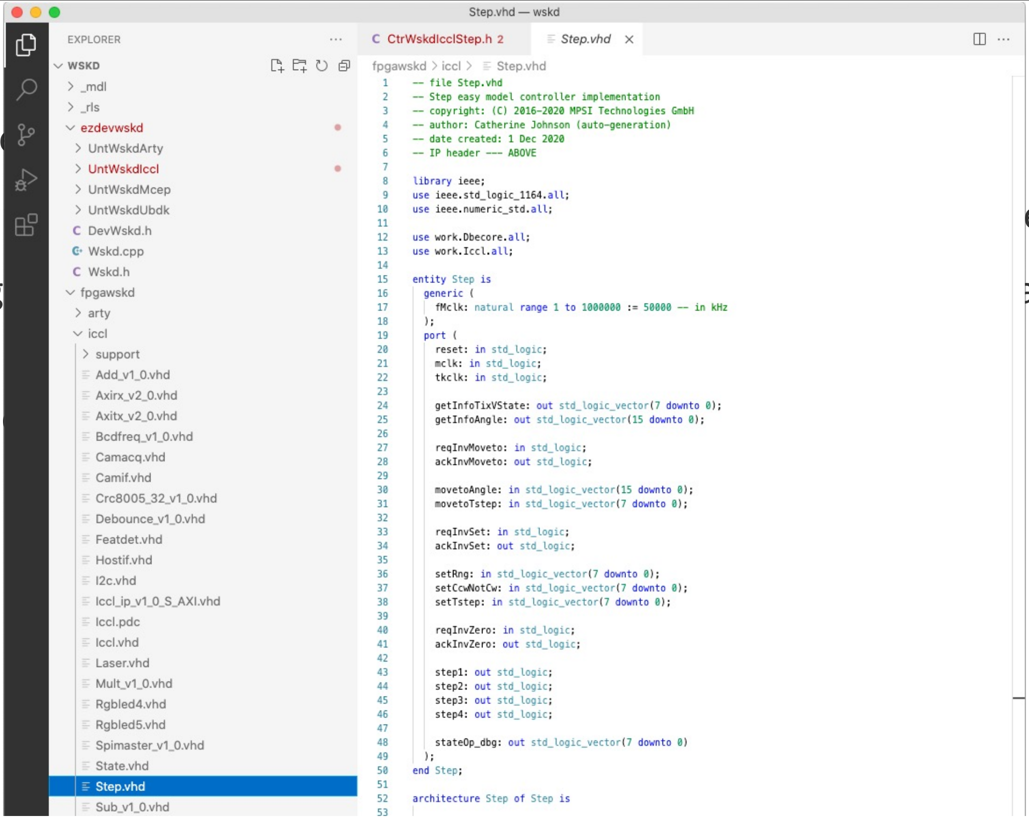
Overview | Composition | Example

- Module definition, command definition, fine structure
- Linux side developer-facing: executable API method
- Linux side in background: translation into byte code and invocation of character device driver (AXI)
- FPGA side in background: reception and decoding of byte code in “host interface” module, CRC evaluation
- FPGA side developer-facing: handshake signals

Model-based software design with Whiznium

Overview | Composition | Example

- Module definition,
- Linux side develope
- Linux side in backg er device driver (AXI)
- FPGA side in backg ace” module, CRC evaluation
- FPGA side develop



```
-- file Step.vhd
-- Step easy model controller implementation
-- copyright: (C) 2016-2020 MPSI Technologies GmbH
-- author: Catherine Johnson (auto-generation)
-- date created: 1 Dec 2020
-- IP header --- ABOVE

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

use work.Dbecore.all;
use work.Iccl.all;

entity Step is
  generic (
    fMclk: natural range 1 to 1000000 := 50000 -- in kHz
  );
  port (
    reset: in std_logic;
    mclk: in std_logic;
    tkclk: in std_logic;

    getInfoTixVState: out std_logic_vector(7 downto 0);
    getInfoAngle: out std_logic_vector(15 downto 0);

    reqInvMoveto: in std_logic;
    ackInvMoveto: out std_logic;

    movetoAngle: in std_logic_vector(15 downto 0);
    movetoTstep: in std_logic_vector(7 downto 0);

    reqInvSet: in std_logic;
    ackInvSet: out std_logic;

    setRng: in std_logic_vector(7 downto 0);
    setCcwNotCw: in std_logic_vector(7 downto 0);
    setTstep: in std_logic_vector(7 downto 0);

    reqInvZero: in std_logic;
    ackInvZero: out std_logic;

    step1: out std_logic;
    step2: out std_logic;
    step3: out std_logic;
    step4: out std_logic;

    stateOp_dbg: out std_logic_vector(7 downto 0)
  );
end Step;

architecture Step of Step is

```

Model-based software design with Whiznium

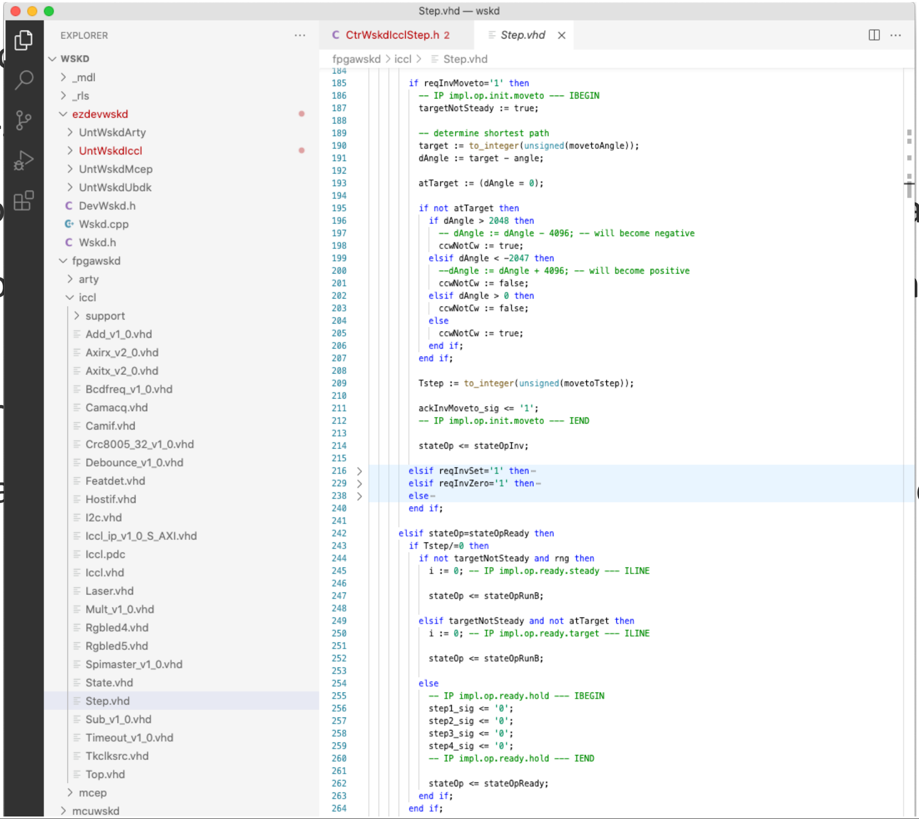
Overview | Composition | Example

- Module definition, command definition, fine structure
- Linux side developer-facing: executable API method
- Linux side in background: translation into byte code and invocation of character device driver (AXI)
- FPGA side in background: reception and decoding of byte code in “host interface” module, CRC evaluation
- FPGA side developer-facing: handshake signals
- FPGA side left for manual implementation: finite state machine reacting to command invocation

Model-based software design with Whiznium

Overview | Composition | Example

- Module definition, c
- Linux side developer
- Linux side in backgro   aracter device driver (AXI)
- FPGA side in backgro   nterface” module, CRC evaluation
- FPGA side developer
- FPGA side left for ma   o command invocation


Step.vhd — wskd

EXPLORER

WSKD
- _mdl
- _rls
- ezdevwskd
  - UntWskdArty
  - UntWskdIccl
  - UntWskdMcep
  - UntWskdUbdk
  - DevWskd.h
  - Wskd.cpp
  - Wskd.h
- fpgawskd
  - arty
  - iccl
    - support
    - Add_v1_0.vhd
    - Axirx_v2_0.vhd
    - Axitx_v2_0.vhd
    - Bcdfreq_v1_0.vhd
    - Camacq.vhd
    - Camif.vhd
    - Crc8005_32_v1_0.vhd
    - Debounce_v1_0.vhd
    - Featdet.vhd
    - Hostif.vhd
    - I2c.vhd
    - Iccl_ip_v1_0_S_AXI.vhd
    - Iccl.pdc
    - Iccl.vhd
    - Laser.vhd
    - Mult_v1_0.vhd
    - Rgbled4.vhd
    - Rgbled5.vhd
    - Spimaster_v1_0.vhd
    - State.vhd
    - Step.vhd
    - Sub_v1_0.vhd
    - Timeout_v1_0.vhd
    - Tkclksrc.vhd
    - Top.vhd
  - mcep
- mcuwskd

CtrWskdIcclStep.h 2 | Step.vhd

fpgawskd > iccl > Step.vhd

```
if reqInvMoveto='1' then
  -- IP impl.op.init.moveto --- IBEGIN
  targetNotSteady := true;

  -- determine shortest path
  target := to_integer(unsigned(movetoAngle));
  dAngle := target - angle;

  atTarget := (dAngle = 0);

  if not atTarget then
    if dAngle > 2048 then
      -- dAngle := dAngle - 4096; -- will become negative
      ccwNotCw := true;
    elsif dAngle < -2047 then
      --dAngle := dAngle + 4096; -- will become positive
      ccwNotCw := false;
    elsif dAngle > 0 then
      ccwNotCw := false;
    else
      ccwNotCw := true;
    end if;
  end if;

  Tstep := to_integer(unsigned(movetoTstep));

  ackInvMoveto_sig <= '1';
  -- IP impl.op.init.moveto --- IEND

  stateOp <= stateOpInv;

elsif reqInvSet='1' then⋯
elsif reqInvZero='1' then⋯
else⋯
end if;

elsif stateOp=stateOpReady then
  if Tstep/=0 then
    if not targetNotSteady and rng then
      i := 0; -- IP impl.op.ready.steady --- ILINE

      stateOp <= stateOpRunB;

    elsif targetNotSteady and not atTarget then
      i := 0; -- IP impl.op.ready.target --- ILINE

      stateOp <= stateOpRunB;

    else
      -- IP impl.op.ready.hold --- IBEGIN
      step1_sig <= '0';
      step2_sig <= '0';
      step3_sig <= '0';
      step4_sig <= '0';
      -- IP impl.op.ready.hold --- IEND

      stateOp <= stateOpReady;
    end if;
  end if;
```

Whiznium concepts

Modularity, transparency and re-usability

- Whiznium is Open Source; the generated code is subject to no license restrictions
- Whiznium generates well-organized, human-readable source code trees which can be synthesized / compiled "out-of-the-box"
- Manual modifications are enabled through the concept of "insertion points"
- Upon source code iteration (e.g. following model extension) manual modifications are carried over to the next version
- Generated code relies on few, well-proven external libraries, all of which are Open Source. Standards are strictly followed
- WhizniumDBE features parametrized "module templates". Besides corresponding VHDL files, template-specific intervention in the WhizniumDBE master database through C++ code is possible
- WhizniumSBE features parametrized "capability templates". Also here, template-specific intervention in the WhizniumSBE master database through C++ code is possible

Whiznium tools

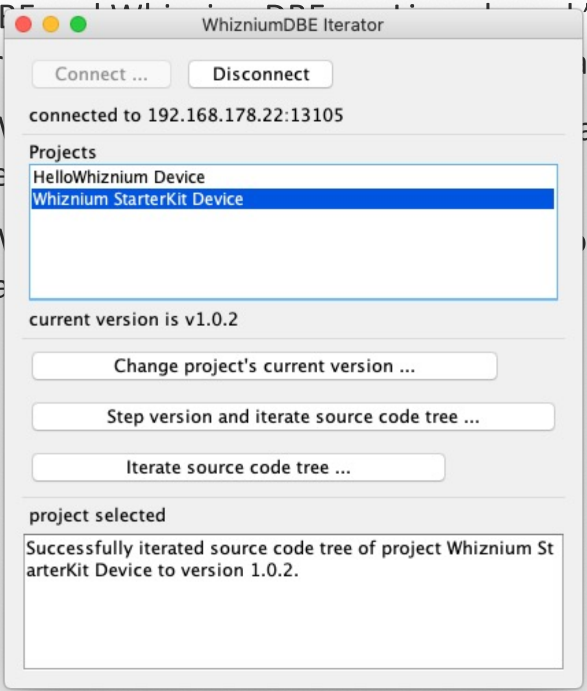
Incorporation into existing developer workflows

- WhizniumSBE and WhizniumDBE are Linux-based ”daemons” (and [fun fact] WhizniumSBE projects), which receive model information and send source code trees via HTTPS
- Java tools WhizniumDBE/SBE Bootstrap offer initialization of WhizniumDBE/SBE with project information stored in a local folder structure
- Java tools WhizniumDBE/SBE Iterator help transform local source code trees from the current version to the next. Here, API calls replace manual UI clicks

Whiznium tools

Incorporation into existing developer workflows

- WhizniumS ”daemons” , which receive model infor HTTPS
- Java tools W alization of ation stored in a local folde
- Java tools W rm local so n to the next. Here, API ca



WhizniumDBE Iterator

connected to 192.168.178.22:13105

Projects

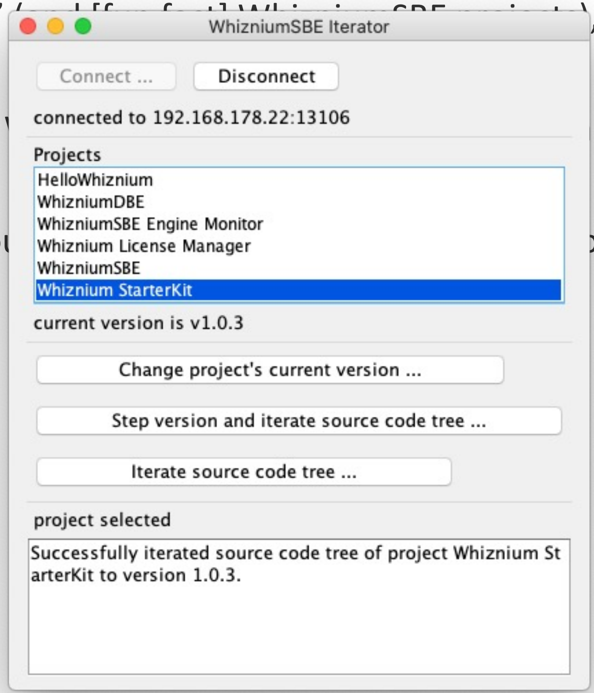
- HelloWhiznium Device
- Whiznium StarterKit Device

current version is v1.0.2

Change project's current version ...
Step version and iterate source code tree ...
Iterate source code tree ...

project selected

Successfully iterated source code tree of project Whiznium StarterKit Device to version 1.0.2.



WhizniumSBE Iterator

connected to 192.168.178.22:13106

Projects

- HelloWhiznium
- WhizniumDBE
- WhizniumSBE Engine Monitor
- Whiznium License Manager
- WhizniumSBE
- Whiznium StarterKit

current version is v1.0.3

Change project's current version ...
Step version and iterate source code tree ...
Iterate source code tree ...

project selected

Successfully iterated source code tree of project Whiznium StarterKit to version 1.0.3.

Whiznium tools

Incorporation into existing developer workflows

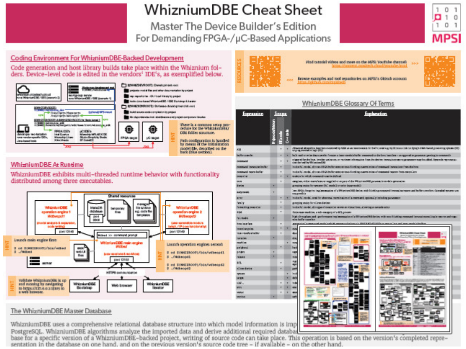
- WhizniumSBE and WhizniumDBE are Linux-based ”daemons” (and [fun fact] WhizniumSBE projects), which receive model information and send source code trees via HTTPS
- Java tools WhizniumDBE/SBE Bootstrap offer initialization of WhizniumDBE/SBE with project information stored in a local folder structure
- Java tools WhizniumDBE/SBE Iterator help transform local source code trees from the current version to the next. Here, API calls replace manual UI clicks
- WhizniumDBE code can be developed using the vendor-provided tools, e.g. Vivado, Quartus, Libero SoC or Simplicity Studio
- WhizniumSBE code can be (cross-)compiled using the industry-standard tool chains gcc/Clang. (Remote-)Debugging can be done using e.g. VS Code
- The Yocto project helps building custom Embedded Linux distributions for each FPGA-SoC platform. WhizniumSBE projects run on those distributions

Whiznium resources

- Both Whiznium tools are available free of charge on GitHub, including installation instructions

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

- The Open Source StarterKit ist available for various hardware platforms, with vendor-specific instructions also available on GitHub
- “The Whiznium Developer Experience” on YouTube is an ongoing Webinar series on Whiznium
- For advanced users WhizniumSBE/DBE cheat sheets are available which serve as reference for writing model files



WhizniumDBE Cheat Sheet
Master The Device Builder’s Edition
For Demanding FPGA-/µC-Based Applications

MPSI

Coding Environment For WhizniumDBE-Backed Development

Code generation and host library builds take place within the Whiznium folders. Device-level code is edited in the vendors' IDE's, as exemplified below.

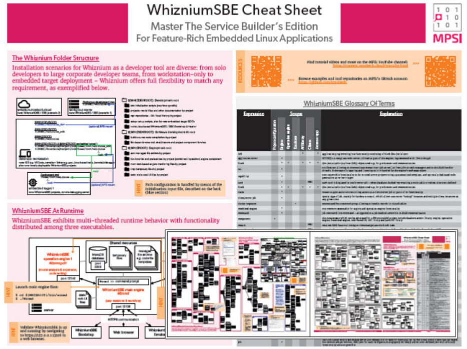
WhizniumDBE At Runtime

WhizniumDBE exhibits multi-threaded runtime behavior with functionality distributed among three executables.

WhizniumDBE Glossary Of Terms

The WhizniumDBE Master Database

[illegible]



WhizniumSBE Cheat Sheet
Master The Service Builder’s Edition
For Feature-Rich Embedded Linux Applications

MPSI

The Whiznium Folder Structure

Installation scenarios for Whiznium as a developer tool are diverse: from solo developers to large corporate developer teams, from workstation-only to embedded target deployment – Whiznium offers full flexibility to match any requirement, as exemplified below.

WhizniumSBE At Runtime

WhizniumSBE exhibits multi-threaded runtime behavior with functionality distributed among three executables.

WhizniumSBE Glossary Of Terms

[illegible]

Conclusion

- Avoid vendor lock-in where possible
  - Limit use of block diagrams
  - Use generic code for simple things (UART / SPI / AXIlite, math)
  - Write wrappers around vendor-specific silicon (memory / high-speed transceivers)

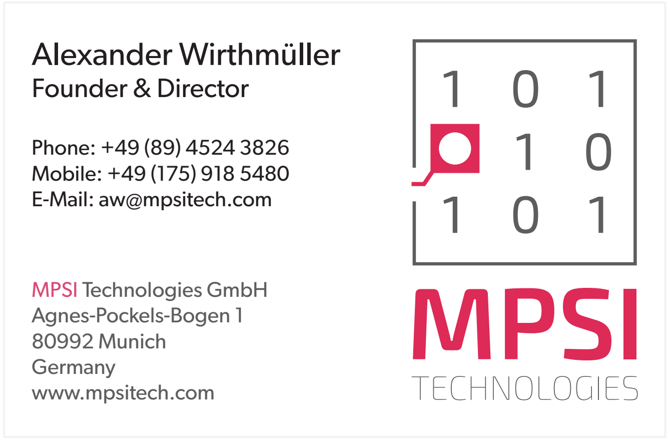
- Model-based source code generation helps further
  - Can abstract away hardware at the crucial host-FPGA interconnect, “single source of truth” maintains host-FPGA integrity
  - WhizniumDBE comes with a set of above mentioned wrappers
  - WhizniumDBE maintains a coarse-to-fine project model in a database and is user-extensible (by means of C++ code, e.g. for frequently used IP)

Thank You!

Questions?

Also, feel free to connect.

- 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