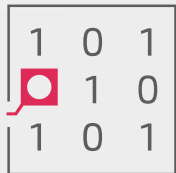


MODELLBASIERTE CODE-GENERIERUNG FÜR HETEROGENE FPGA-SOC SYSTEME

KREATIVITÄT FÜR KERNFUNKTIONALITÄT
EINSETZEN, ALLES ANDERE (SAUBER)
GENERIEREN LASSEN

Alexander Wirthmüller
aw@mpsitechnologies.com



MPSI
TECHNOLOGIES

FPGA-SoC Übersicht

Hardware und ihre Anwendungen

Produktlinien

Abgrenzung: CPU-seitig geeignet für Embedded Linux



- ab 2011: Zynq 7000 mit Dual 32-bit ARM CPU und SRAM-basiertem FPGA, intern verbunden über AXI
- ab 2016: Zynq UltraScale+ mit Quad 64-bit ARM CPU und weiteren integrierten Kernen wie Beschleunigern



- ab 2012: CycloneV mit Dual 32-bit ARM CPU
- ab 2016: Stratix 10 mit Quad 64-bit ARM CPU



- ab 2019: PolarFire SoC mit Quad 64-bit RISC-V CPU und Antifuse-basiertem FPGA

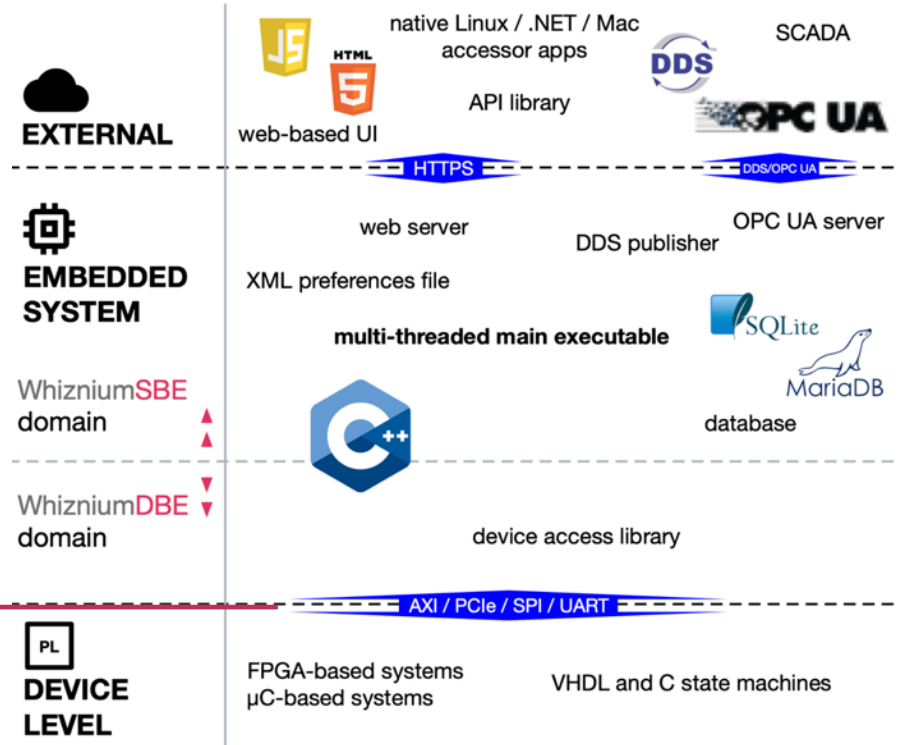
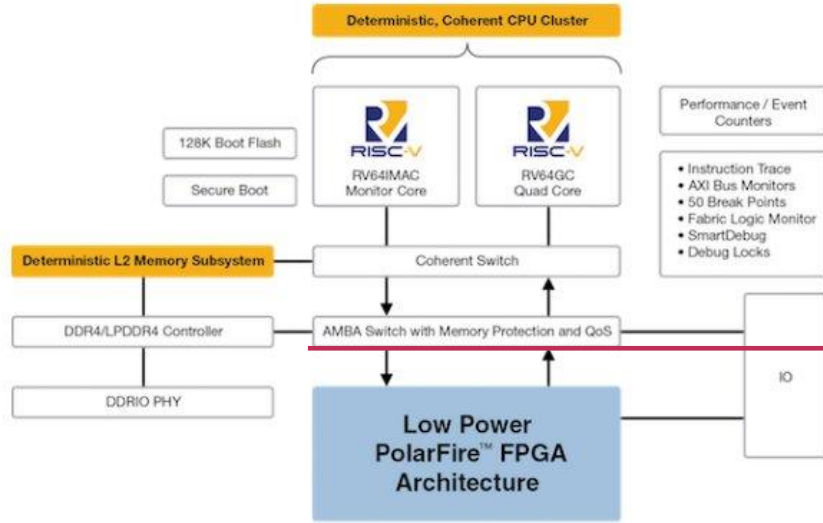
Typische Anwendungen

- Klassische Anwendungsgebiete von FPGA's bei denen zusätzlich high-level Steuerung erwünscht ist
- "Datenreduktion" bzw. Vorverarbeitung von Quellen mit hoher Bandbreite
- Kameras: Binning, Pixel-level Filter, Kompression, Ecken- und Objekterkennung
- A/D-Wandler: Spektralanalyse, DSP-Filter
- Taktgenaue Ansteuerung von Mixed-Signal ASIC's
- Hier nicht betrachtet: Data Center / Hardware-Beschleuniger Anwendungen

Der FPGA-SoC Software “Full-Stack”

Von VHDL über C++ bis HTTPS und XML

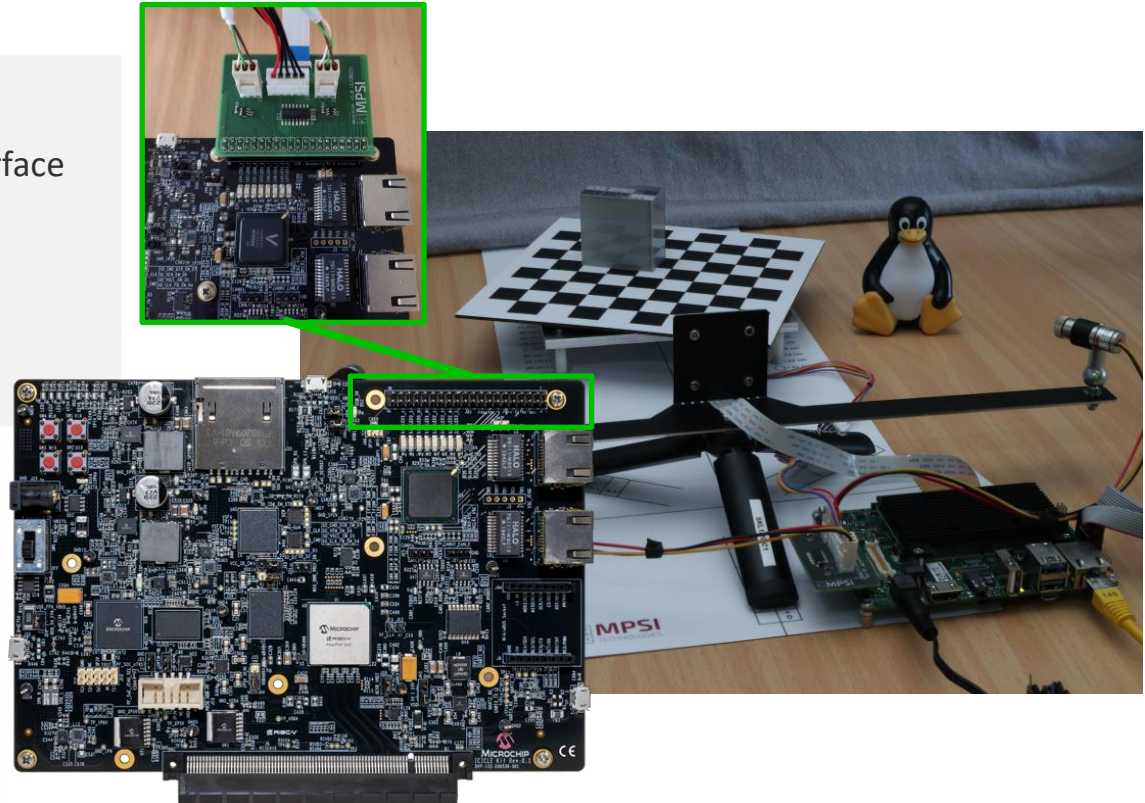
Microchip PolarFire SoC Block-Diagramm



Die Beispielanwendung: Hardware

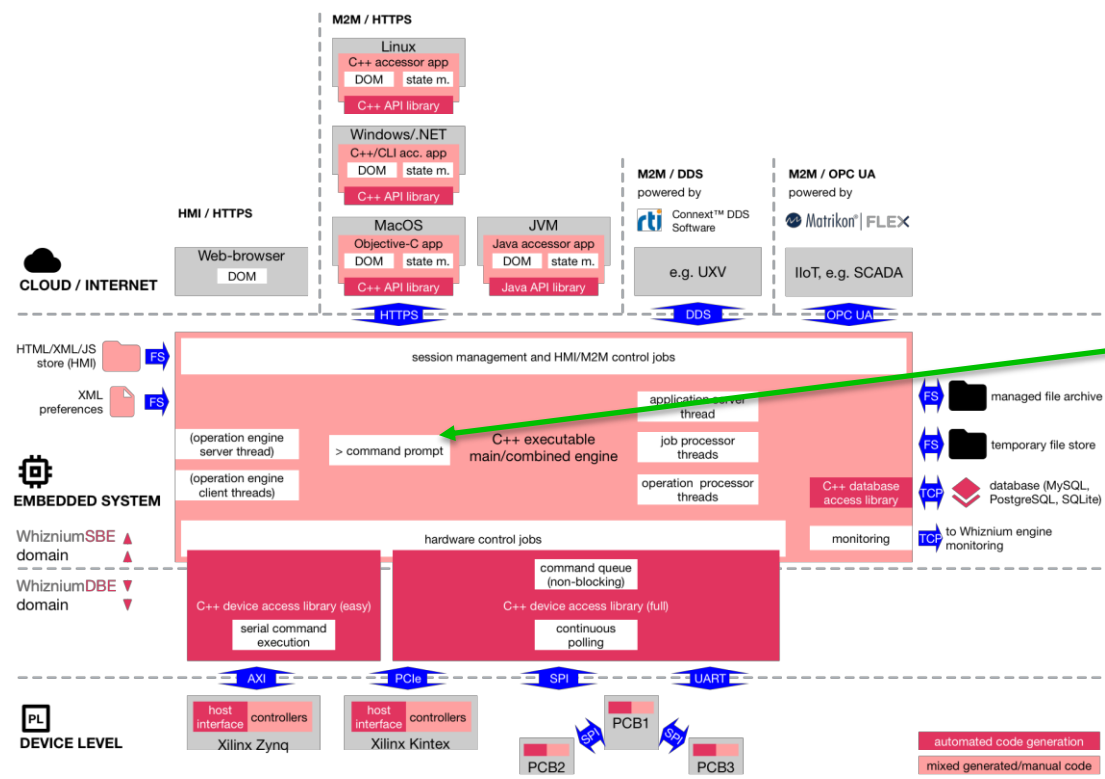
Ein kompakter 3D-Laserscanner

- Drehteller mit Schrittmotor-Antrieb
- 5 Megapixel Kamera mit Parallel-Interface
- Zwei modulierbare Linienlaser
- Microchip PolarFire SoC Icicle Kit mit Adapterplatine



Die Beispielanwendung: Software

Von Kamera-Rohdaten zur Punktwolke im Web-Browser



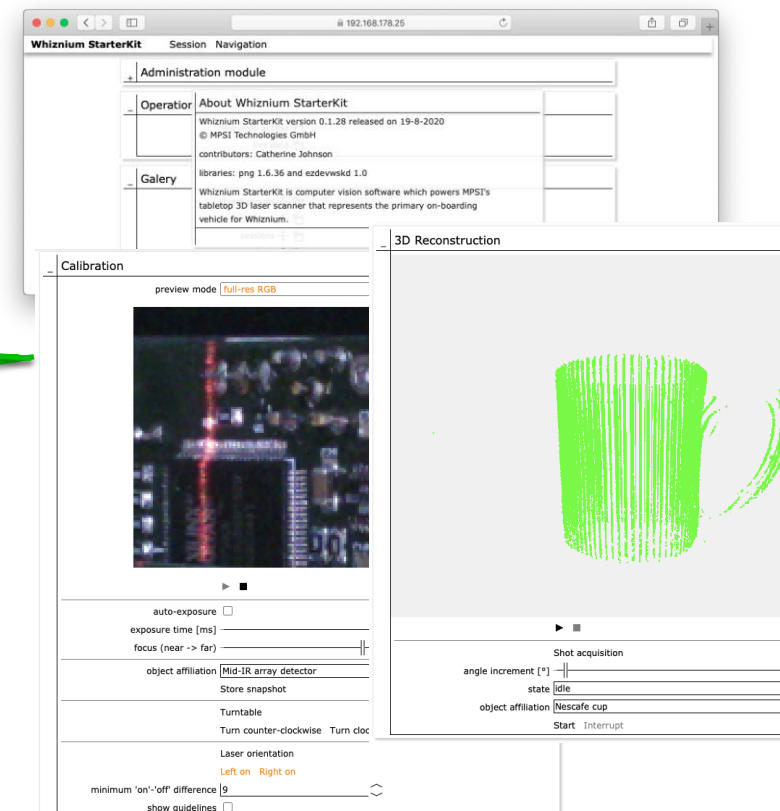
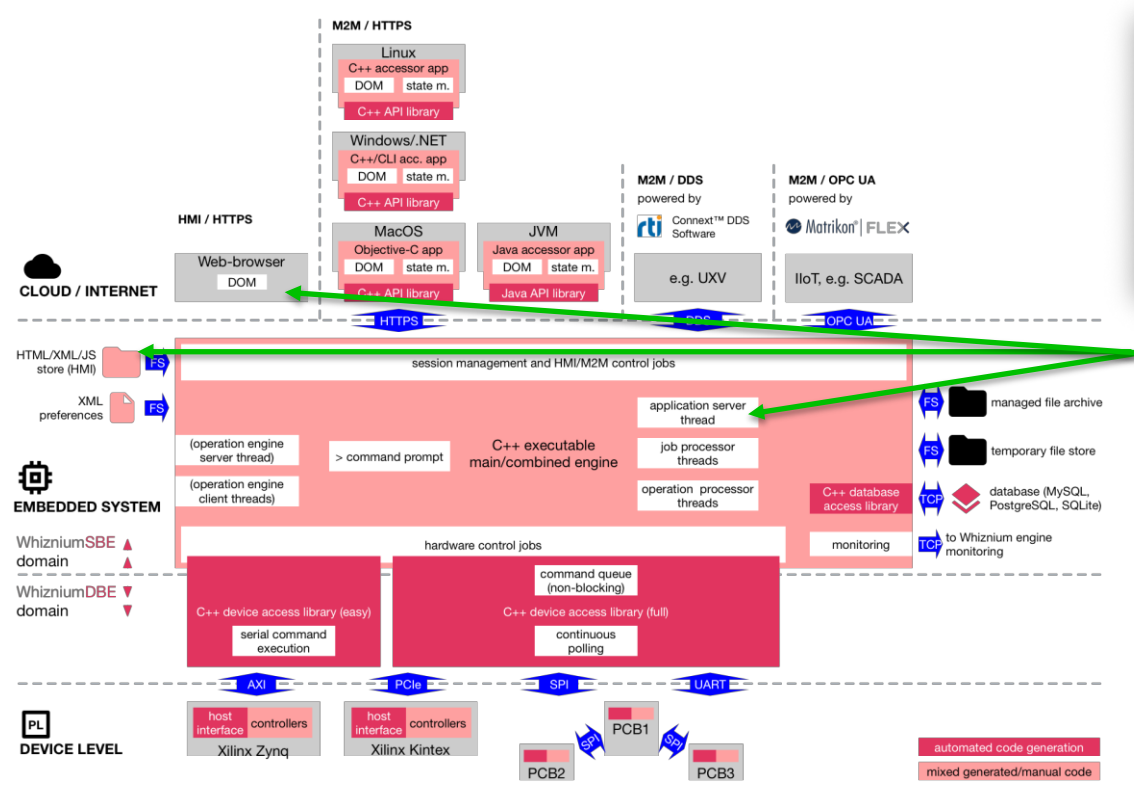
```
root@cicle-kit-es:~/whiznium/bin/wzskcmbd# ./wzskcmbd
Welcome to Whiznium StarterKit vl.0.5!
starting 4 job processor threads ... {19255, 19256, 19257, 19258} success
starting 1 operation processor threads ... {19259} success
starting application server ... success
starting OPC UA server ... success
Initialization complete.

Wzskcmbd >> showJobs
+ RootWzsk (1)
- JobWzskSrcV412/SRV (2)
- JobWzskSrcSysinfo/SRV (3)
- JobWzskSrcFpga/SRV (4)
+ JobWzskIprTrace/SRV (5)
- JobWzskActLaser/CLI (6)
- JobWzskSrcV412/CLI (7)
+ JobWzskIprCorner/SRV (8)
- JobWzskSrcV412/CLI (9)
+ JobWzskIprAngle/SRV (10)
- JobWzskIprCorner/CLI (11)
- JobWzskActServo/SRV (12)
- JobWzskActLaser/SRV (13)
+ JobWzskActExposure/SRV (14)
- JobWzskSrcV412/CLI (15)
+ JobWzskAcqPcloud/SRV (16)
- JobWzskIprTrace/CLI (17)
- JobWzskActServo/CLI (18)
+ JobWzskAcqPreview/SRV (19)
- JobWzskSrcV412/CLI (20)
+ JobWzskAcqFpgapvw/SRV (21)
- JobWzskSrcFpga/CLI (22)
+ JobWzskAcqFpgaf1g/SRV (23)
- JobWzskSrcFpga/CLI (24)
+ M2messWzsk (25)
- JobWzskSrcSysinfo/CLI (26)
- JobWzskIprTrace/CLI (27)
- JobWzskIprCorner/CLI (28)
- JobWzskActServo/CLI (29)
- JobWzskActExposure/CLI (30)
- JobWzskActLaser/CLI (31)
- JobWzskAcqPcloud/CLI (32)
- JobWzskAcqPreview/CLI (33)

Wzskcmbd >>
```

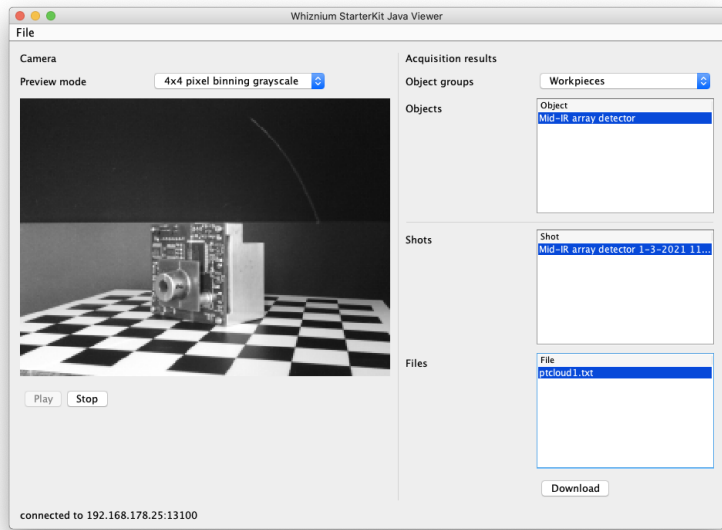
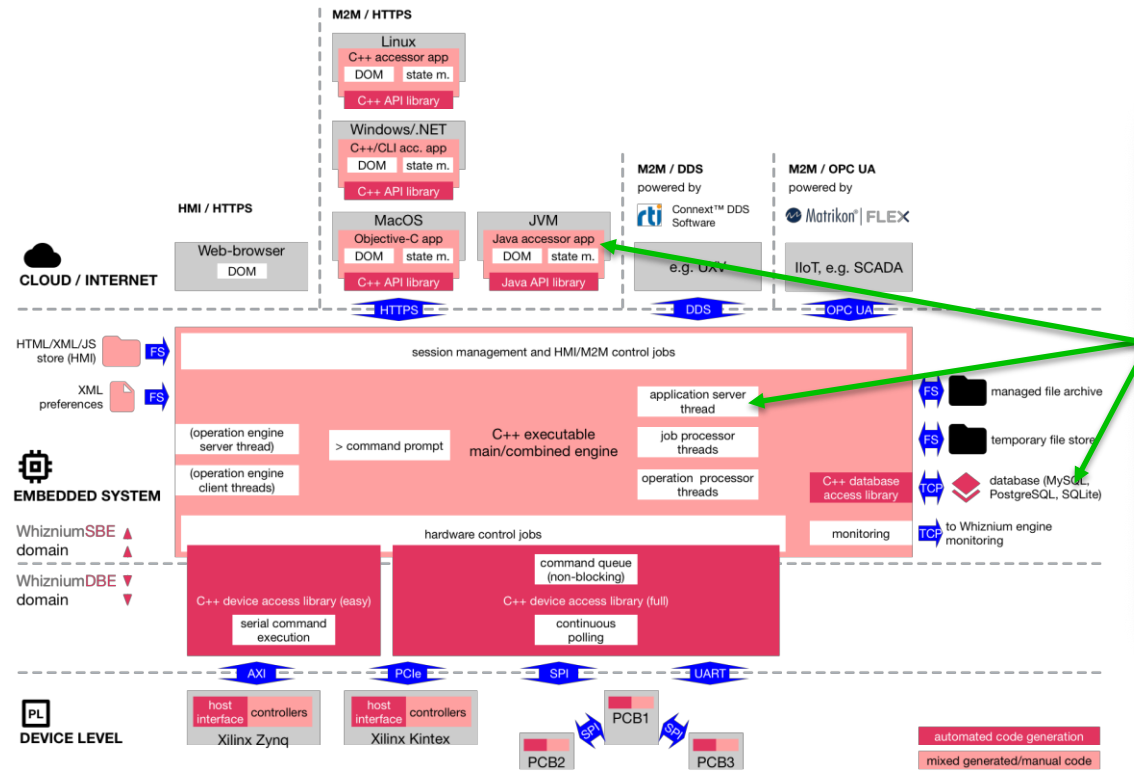
Die Beispielanwendung: Software

Von Kamera-Rohdaten zur Punktwolke im Web-Browser



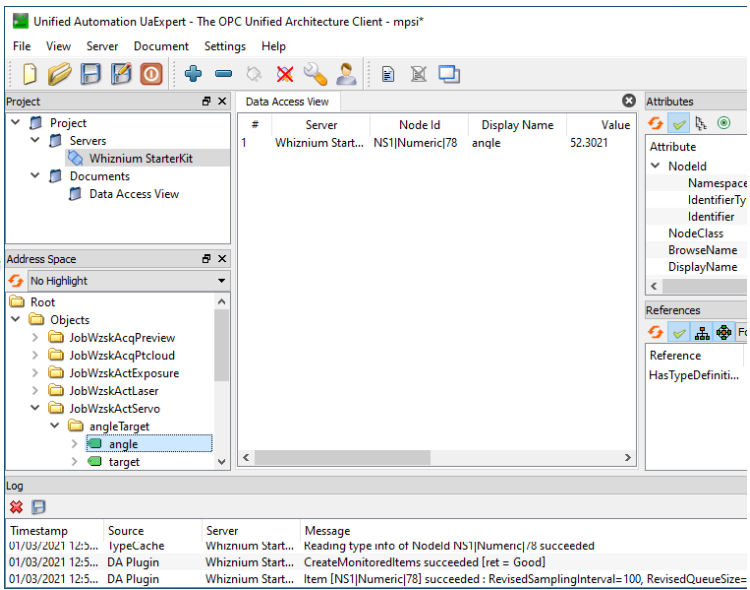
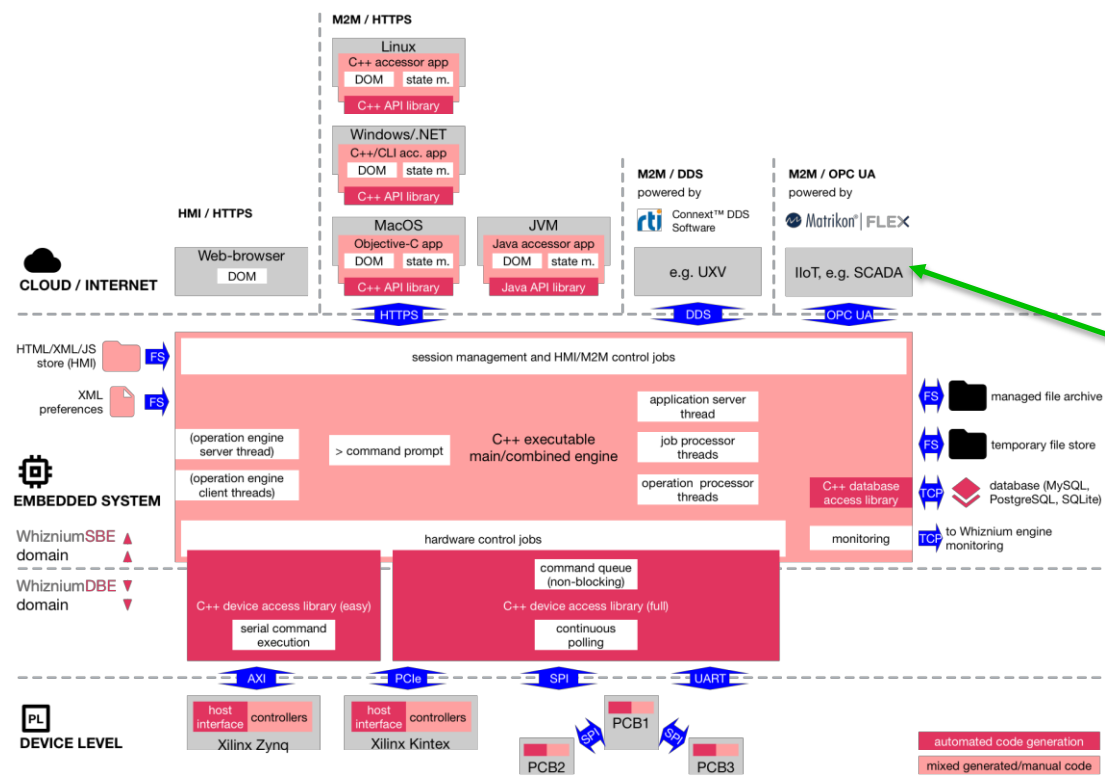
Die Beispielanwendung: Software

Von Kamera-Rohdaten zur Punktwolke im Web-Browser



Die Beispielanwendung: Software

Von Kamera-Rohdaten zur Punktwolke im Web-Browser



Modellbeschreibung der Embedded Software

WhizniumDBE für die FPGA-Ebene, WhizniumSBE für Linux-Anwendungssoftware

- Sukzessiver Aufbau des Modells in einer SQL Datenbank mittels Import (I)- und Generierungs (G)-Schritten
- Erst dann Schreiben von Quellcode-Bäumen
- 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)

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition, Befehlsdefinition, Feinstruktur

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

lexWdbeMdl v1.1.14							
ImelMUnit	srefSilRefWdbeMUnit	sref	Title	Easy	srefKToolch	Comment	
fpga	mpfs250t-fcvg484	iccl	Microchip PolarFire Soc Icicle kit	true	libero		
	ImelMModule.srefIxVBa	hsrefSupRefWdb	srefTplRefWdbeMModule	sref	Comment		
	wrp		mpfs_ip_AXI_v1_0	iccl_ip_AXI			
	top	iccl_ip_AXI	top_mchp_v1_0	top			
		ImelAMModuleF Val					
		fExtclk	125000				
		extresetNNotP	true				
		ImelAMModulePar.end					
		ImelMGeneric.s	Defval				
		fMclk	50000				
		ImelMGeneric.end					
	...						
	ectr	iccl_ip_AXI;top		step	stepper motor control (28BYJ-48 via ULN2003)		
	...						
	ImelMModule.end						
ImelMUnit.end							

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

lexWdbeCxx v1.1.9								
ImelMUnit.sref								
iccl								
ImelMModule.hsrefSup sref								
iccl_ip_AXI;top step								
ImelMController.								
^								
ImelMVector2.sreflxVBase sref srefsKOption								
tixlin VecVWskdlcclStepState filfed;notit								
ImelMVectoritem2.sref Title Comment								
idle								
move								
ImelMVectoritem2.end								
ImelMVector2.end								
ImelMCommand2.refNum sref sreflxVRetype srefivrRefWd srefRvrRef srefRerRe Comment								
...								
0 moveto void								
ImelAMCommandInvpar2.sref sreflxWdbeVPartype 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)								
ImelAMCommandInvpar2.end								
...								
ImelMCommand2.end								
ImelMController.end								
ImelMModule.end								
ImelMUnit.end								

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

lexWdbeFin v1.1.9													
ImelMUnit.sref													
iccl													
ImelMModule.hsrefSup sref													
iccl_ip_AXI;top step													
...													
ImelMProcess.sref clkSrefWdbe asrSrefWdbeMS Falling Syncrst Extip Comment													
op mclk reset false state(init) or (sta false main operation													
ImelMFsm.													
^													
ImelMFsmstate sref Extip Comment													
0 init false													
ImelAMFsm: Cond1 lp1 Cond2 lp2 Cond3 lp3 Cond4 lp4													
inv reqInvMoveto moveto													
inv reqInvSet set													
inv reqInvZero zero													
ready else													
ImelAMFsmstateStep.end													
0 ready false													
ImelAMFsm: Cond1 lp1 Cond2 lp2 Cond3 lp3 Cond4 lp4													
runB Tstep/=0 not targetNotSteady and rng steady													
runB Tstep/=0 targetNotSteady and not atTarget target													
ready Tstep/=0 else hold													
ImelAMFsmstateStep.end													
...													
ImelMFsmstate.end													
ImelMFsm.end													
ImelMProcess.end													
ImelMModule.end													
ImelMUnit.end													

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

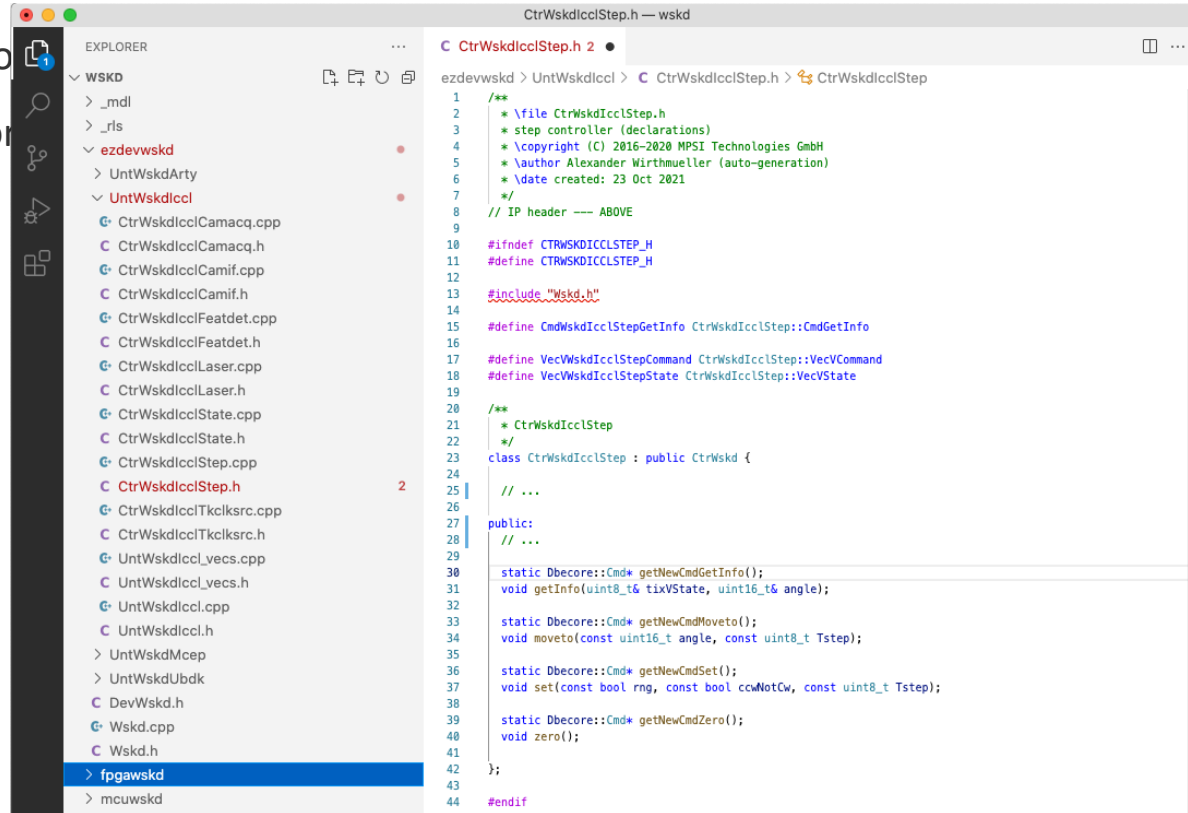
Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition, Befehlsdefinition, Feinstruktur
- Linux-seitig vordergründig: aufrufbarer Befehl

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition
- Linux-seitig vor



The screenshot shows a code editor with two panes. The left pane is the 'EXPLORER' view showing a project structure. The right pane shows the source code of 'CtrWskdIcclStep.h'.

EXPLORER View:

- Wskd
 - _mdl
 - _rls
 - ezdevwskd
 - UntWskdArty
 - UntWskdIccl
 - CtrWskdIcclCamacq.cpp
 - CtrWskdIcclCamacq.h
 - CtrWskdIcclCamif.cpp
 - CtrWskdIcclCamif.h
 - CtrWskdIcclFeatdet.cpp
 - CtrWskdIcclFeatdet.h
 - CtrWskdIcclLaser.cpp
 - CtrWskdIcclLaser.h
 - CtrWskdIcclState.cpp
 - CtrWskdIcclState.h
 - CtrWskdIcclStep.cpp
 - CtrWskdIcclStep.h**
 - CtrWskdIcclTkclsrc.cpp
 - CtrWskdIcclTkclsrc.h
 - UntWskdIccl_vecs.cpp
 - UntWskdIccl_vecs.h
 - UntWskdIccl.cpp
 - UntWskdIccl.h
 - UntWskdMcep
 - UntWskdUbdk
 - DevWskd.h
 - Wskd.cpp
 - Wskd.h
 - fpgawskd
 - mcuwskd

Source Code (CtrWskdIcclStep.h):

```
1  /**
2   * \file CtrWskdIcclStep.h
3   * step controller (declarations)
4   * \copyright (C) 2016-2020 MPSI Technologies GmbH
5   * \author Alexander Wirthmueller (auto-generation)
6   * \date created: 23 Oct 2021
7   */
8  // IP header --- ABOVE
9
10 #ifndef CTRWSKDICCLSTEP_H
11 #define CTRWSKDICCLSTEP_H
12
13 #include "Wskd.h"
14
15 #define CmdWskdIcclStepGetInfo CtrWskdIcclStep::CmdGetInfo
16
17 #define VecWskdIcclStepCommand CtrWskdIcclStep::VecVCommand
18 #define VecWskdIcclStepState CtrWskdIcclStep::VecVState
19
20 /**
21  * CtrWskdIcclStep
22  */
23 class CtrWskdIcclStep : public CtrWskd {
24 // ...
25 public:
26 // ...
27
28 static Dbcore::Cmd* getNewCmdGetInfo();
29 void getInfo(uint8_t& tixVState, uint16_t& angle);
30
31 static Dbcore::Cmd* getNewCmdMoveto();
32 void moveto(const uint16_t angle, const uint8_t Tstep);
33
34 static Dbcore::Cmd* getNewCmdSet();
35 void set(const bool rng, const bool ccwNotCw, const uint8_t Tstep);
36
37 static Dbcore::Cmd* getNewCmdZero();
38 void zero();
39
40 };
41
42 #endif
```


Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

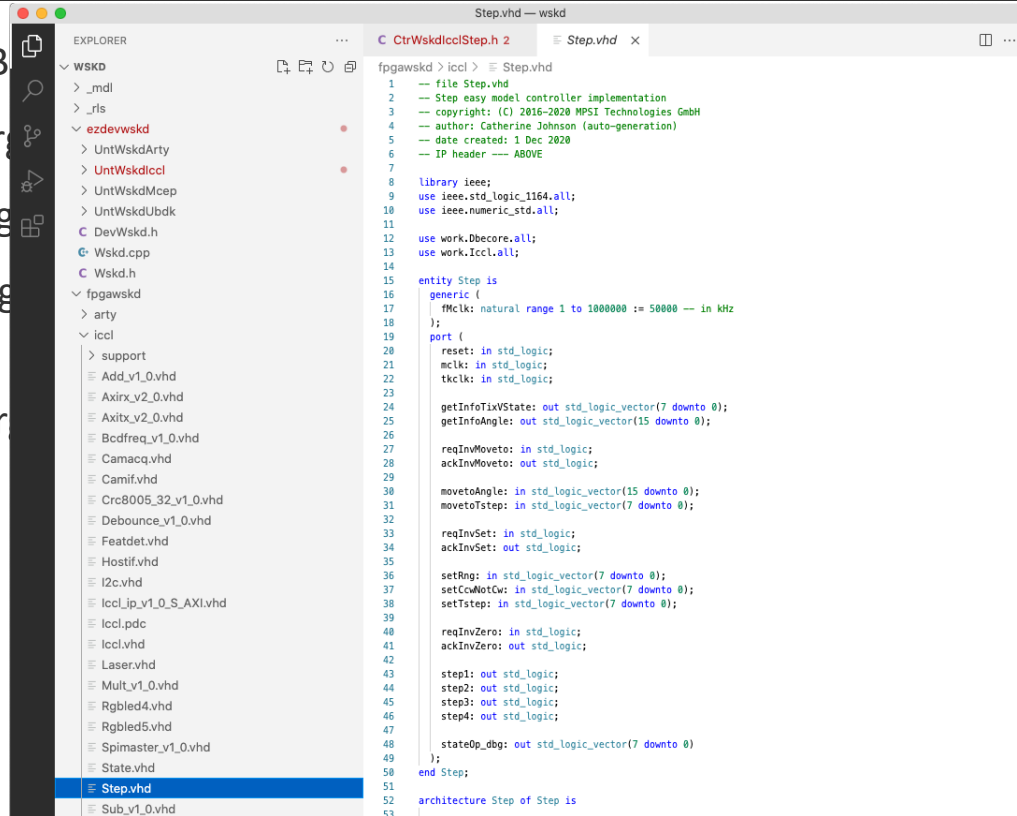
Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition, Befehlsdefinition, Feinstruktur
- Linux-seitig vordergründig: aufrufbarer Befehl
- Linux-seitig hintergründig: Umwandlung in Bytecode und Übergabe an Character Device Treiber (AXI)
- FPGA-seitig hintergründig: Empfang und Dekodierung des Bytecode in “host interface” Modul, CRC-Absicherung
- FPGA-seitig vordergründig: Handshake-Signale

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition, B
- Linux-seitig vorder
- Linux-seitig hinterg
- FPGA-seitig hinterg
- Absicherung
- FPGA-seitig vorder



```
Step.vhd -- wskd
C CtrlWskdIcclStep.h 2 Step.vhd x
fpgawskd > iccl > Step.vhd
1 -- file Step.vhd
2 -- Step easy model controller implementation
3 -- copyright: (C) 2016-2020 MPSI Technologies GmbH
4 -- author: Catherine Johnson (auto-generation)
5 -- date created: 1 Dec 2020
6 -- IP header --- ABOVE
7
8 library ieee;
9 use ieee.std_logic_1164.all;
10 use ieee.numeric_std.all;
11
12 use work.Dbecore.all;
13 use work.Iccl.all;
14
15 entity Step is
16 generic (
17     Mclk: natural range 1 to 1000000 := 50000 -- in kHz
18 );
19 port (
20     reset: in std_logic;
21     mclk: in std_logic;
22     tkclk: in std_logic;
23
24     getInfoTixVState: out std_logic_vector(7 downto 0);
25     getInfoAngle: out std_logic_vector(15 downto 0);
26
27     reqInvMoveto: in std_logic;
28     ackInvMoveto: out std_logic;
29
30     movetoAngle: in std_logic_vector(15 downto 0);
31     movetoTstep: in std_logic_vector(7 downto 0);
32
33     reqInvSet: in std_logic;
34     ackInvSet: out std_logic;
35
36     setRng: in std_logic_vector(7 downto 0);
37     setCwMotCw: in std_logic_vector(7 downto 0);
38     setTstep: in std_logic_vector(7 downto 0);
39
40     reqInvZero: in std_logic;
41     ackInvZero: out std_logic;
42
43     step1: out std_logic;
44     step2: out std_logic;
45     step3: out std_logic;
46     step4: out std_logic;
47
48     stateOp_dbg: out std_logic_vector(7 downto 0)
49 );
50 end Step;
51
52 architecture Step of Step is
53
```

ter Device Treiber (AXI)
nterface” Modul, CRC-

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

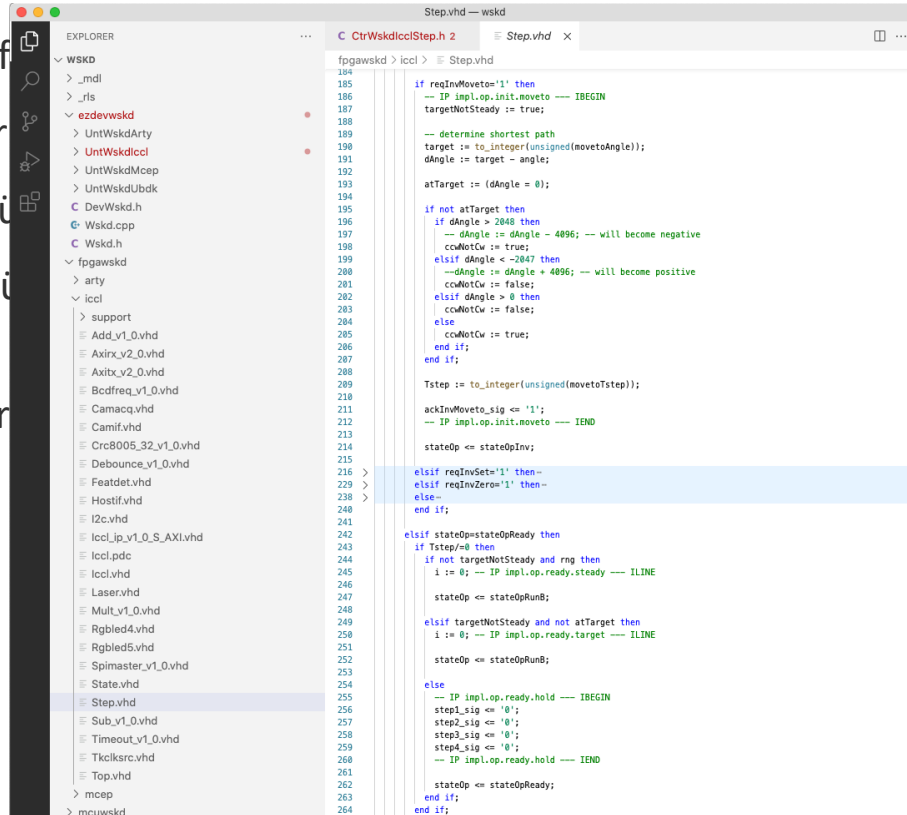
Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition, Befehlsdefinition, Feinstruktur
- Linux-seitig vordergründig: aufrufbarer Befehl
- Linux-seitig hintergründig: Umwandlung in Bytecode und Übergabe an Character Device Treiber (AXI)
- FPGA-seitig hintergründig: Empfang und Dekodierung des Bytecode in “host interface” Modul, CRC-Absicherung
- FPGA-seitig vordergründig: Handshake-Signale
- FPGA-seitig manuell zu erledigen: Implementierung des Zustandsautomaten

Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition, Befehl
- Linux-seitig vordergründig
- Linux-seitig hintergründig
- FPGA-seitig hintergründig
- Absicherung
- FPGA-seitig vordergründig
- FPGA-seitig manuell



```
Step.vhd -- wskd
C CtrlWskdIcclStep.h 2 Step.vhd
fpgawskd > iccl > Step.vhd
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264

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
      cwMotCw := true;
    elsif dAngle < -2047 then
      --dAngle := dAngle + 4096; -- will become positive
      ccwMotCw := false;
    elsif dAngle > 0 then
      cwMotCw := true;
    else
      ccwMotCw := false;
    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;
  end if;
```

Character Device Treiber (AXI)
"post interface" Modul, CRC-

ten

Deep Dive II: Kamera-Vorschaubilder unterwegs

Binning im FPGA, Verarbeitung in C++ Code sowie Weiterleitung ans UI

- Pixels werden mit ca. 25MHz über paralleles 8-bit Interface in FPGA im RAW8 Format gestreamed
- Vier Vorschau-Modi (2560 x 1280 zu 160 x 120 RGB vs. 2048 x 1536 zu 256 x 192 grayscale), manuelle Implementierung in Zustandsautomaten + 2/4kB Puffer
- "buffer transfers" neben "commands" die zweite generierbare Funktionalität über das "host interface"
- Abruf in separatem Thread, Eingliederung in den "job tree" mittels "call"
- Erzeugung von XML-Block mit Bilddaten, Base64-kodierte Übertragung
- Empfang mittels HTTPS/1.1 "long polling" in Web-Browser und Darstellung in HTML5 <canvas/>

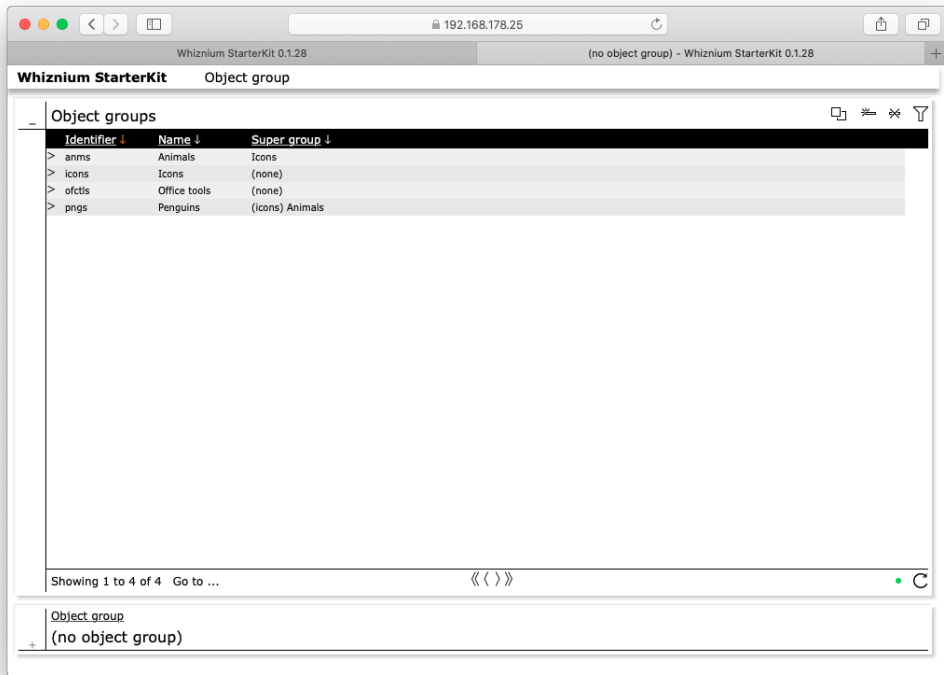
Deep Dive III: Metadaten und der “Lowering”-Prozess in Whiznium

Von SQL-Datenbankstruktur über Code und UI-Elemente zu XML-/JSON-Blöcken

- Alle WhizniumSBE-Anwendungen werden durch eine SQLite3-Datenbank unterstützt
- “Database structure”: Tabellen “Objektgruppe” (1:N) “Objekt” (1:N) “Aufnahme” und zugehöriges “Basic user interface”

Deep Dive III: Metadaten und der “Lowering”-Prozess in Whiznium

Von SQL-Datenbankstruktur über Code und UI-Elemente zu XML-/JSON-Blöcken

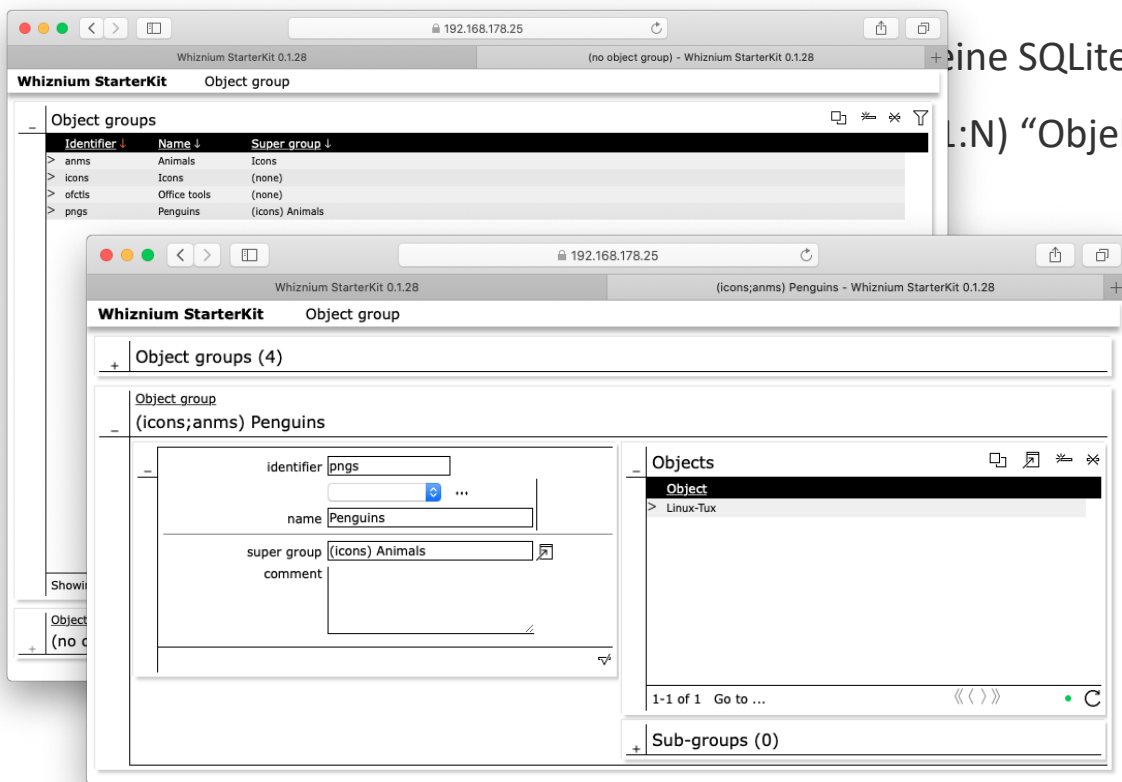


+ Eine SQLite3-Datenbank unterstützt

1:N “Objekt” (1:N) “Aufnahme” und zugehöriges

Deep Dive III: Metadaten und der “Lowering”-Prozess in Whiznium

Von SQL-Datenbankstruktur über Code und UI-Elemente zu XML-/JSON-Blöcken

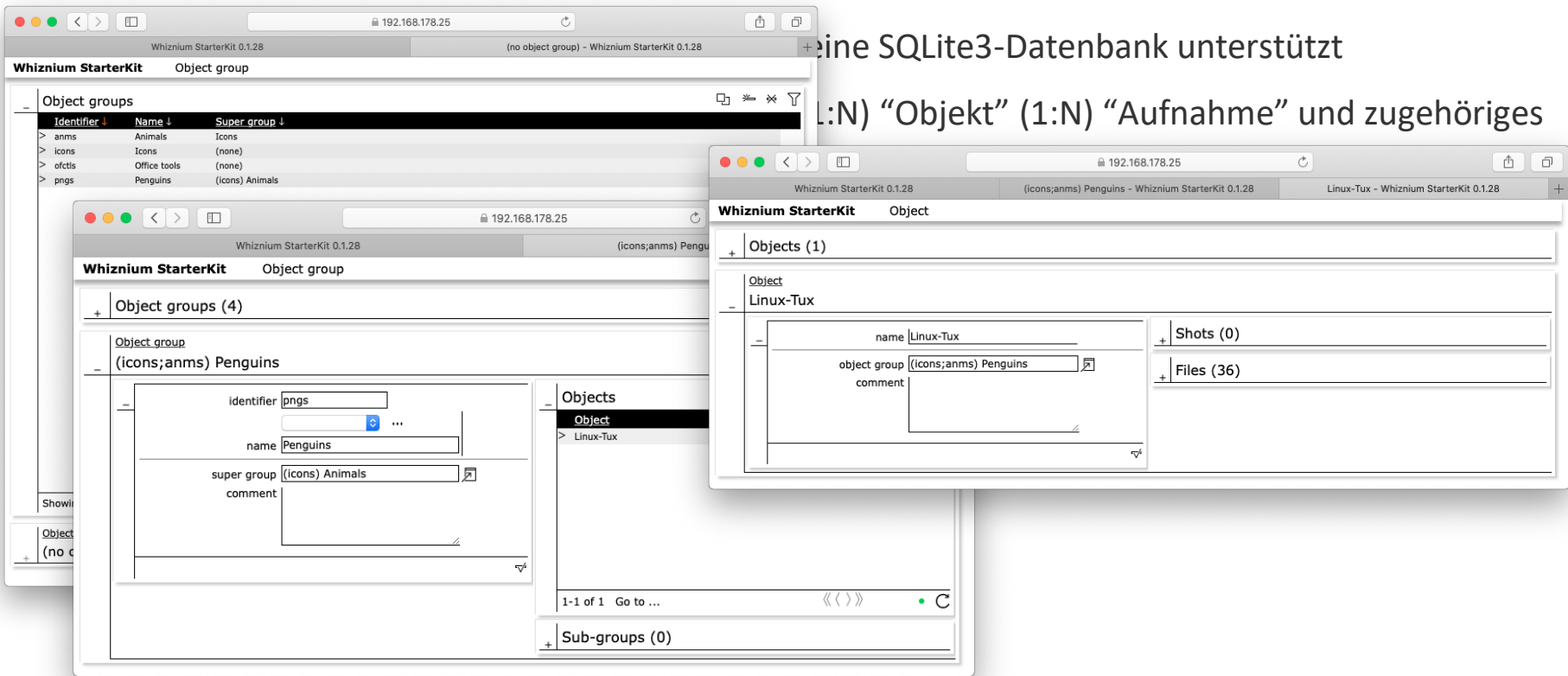


...eine SQLite3-Datenbank unterstützt

...1:N “Objekt” (1:N) “Aufnahme” und zugehöriges

Deep Dive III: Metadaten und der “Lowering”-Prozess in Whiznium

Von SQL-Datenbankstruktur über Code und UI-Elemente zu XML-/JSON-Blöcken



Deep Dive III: Metadaten und der “Lowering”-Prozess in Whiznium

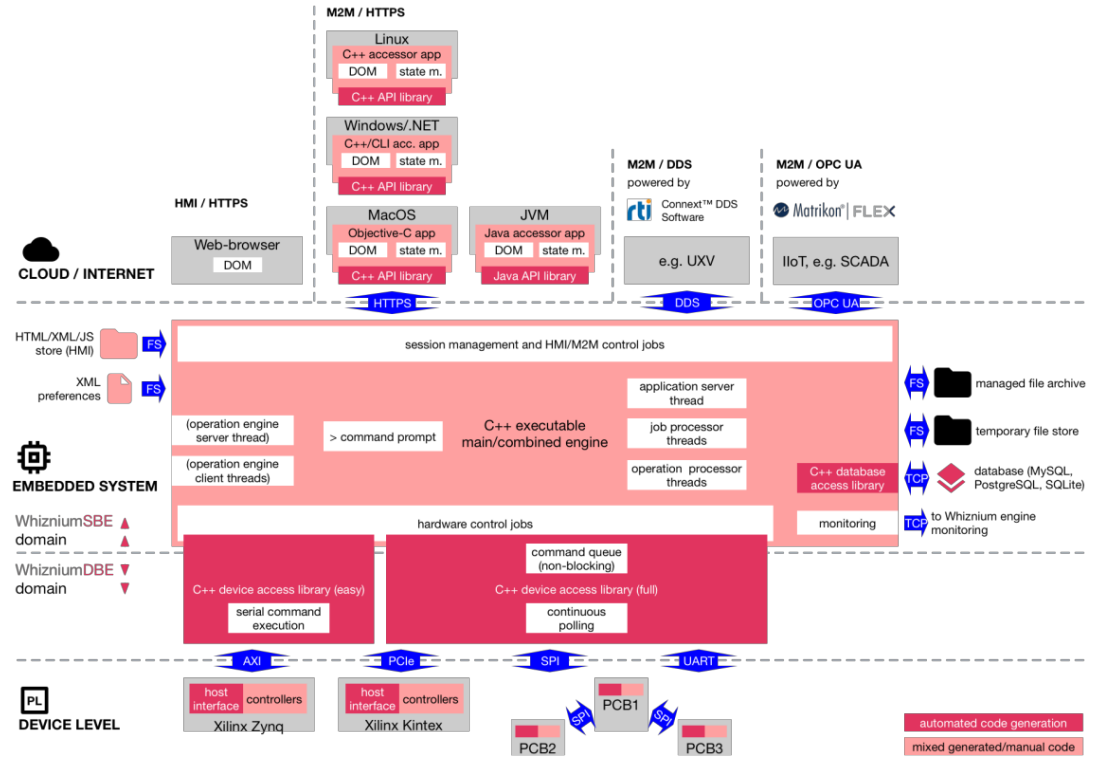
Von SQL-Datenbankstruktur über Code und UI-Elemente zu XML-/JSON-Blöcken

- Alle WhizniumSBE-Anwendungen werden durch eine SQLite3-Datenbank unterstützt
- “Database structure”: Tabellen “Objektgruppe” (1:N) “Objekt” (1:N) “Aufnahme” und zugehöriges “Basic user interface”
- Erster “lowering”-Schritt: mehrsprachige UI-Elemente, “queries”, “panels” und “controls”
- Zweiter “lowering”-Schritt: “blocks” und “dispatches” für Engine <-> App-Kommunikation
- Code-Generierung “database access library”
- Code-Generierung “Engine”-seitig: Klassen für “cards”, “panels” und “queries” welche dynamisch für WebUI-Sessions Objekte generieren und auf Benutzer-Eingaben reagieren
- Code-Generierung “App”-seitig: HTML und JavaScript

Schlussfolgerung Beispielanwendung

Modellieren lohnt sich

- Die schwer zu überschauende Menge der für FPGA-SoC's relevanten Software-Technologien wird durch eine einzelne kohärente Methode sauber abgedeckt



Whiznium Konzepte

Modularität, Transparenz und Wiederverwendbarkeit

- Whiznium ist Open Source; generierter Code unterliegt keinen Lizenz einschränkungen
- Whiznium generiert sauber strukturierte, gut leserliche Quellcode-Bäume, die “out-of-the-box” synthetisier- bzw. kompilierfähig sind
- Manuelle Ergänzungen werden über das Konzept der “insertion points” ermöglicht
- Bei Iterationen des Quellcodes (z.B. nach Erweiterung des Modells) werden die manuellen Ergänzungen weitergetragen
- Es werden wenige externe, gut bewährte, ausschließlich Open Source, Programmbibliotheken eingesetzt, Standards werden strikt eingehalten
- WhizniumDBE kennt parametrisierbare “module templates”. Neben zugehörigen VHDL-Dateien ist die Intervention in der WhizniumDBE Master-Datenbank mit spezifischem C++ -Code möglich
- WhizniumSBE kennt parametrisierbare “capability templates”. Die Intervention in der WhizniumSBE Master-Datenbank mit spezifischem C++-Code ist möglich

Whiznium Werkzeuge

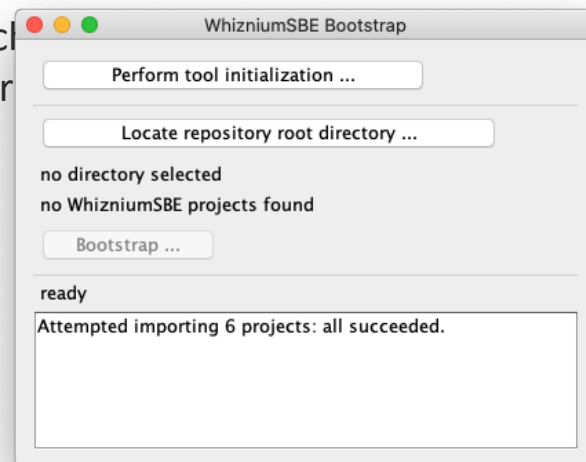
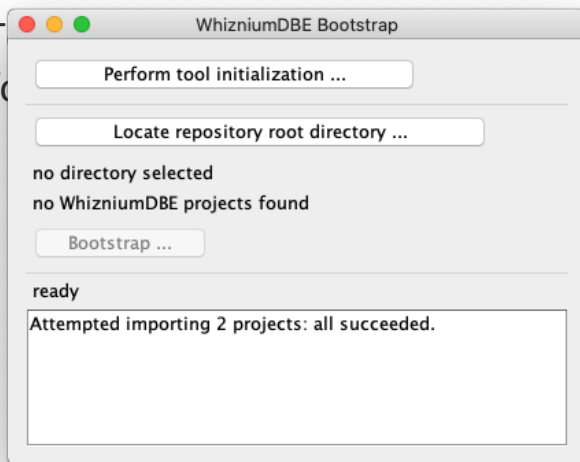
Einbindung in den Entwickler-Alltag

- WhizniumSBE und WhizniumDBE sind Linux-basierte "daemons" (und WhizniumSBE-Projekte), welche Modellinformationen und Quellcode-Bäume über HTTPS empfangen und verbreiten
- Die Java-Tools WhizniumDBE/SBE Bootstrap ermöglichen das Initialisieren von WhizniumDBE/SBE mit Projektinformationen aus einer lokalen Verzeichnisstruktur

Whiznium Werkzeuge

Einbindung in den Entwickler-Alltag

- WhizniumSBE und WhizniumDBE sind Linux-basierte "daemons" (und WhizniumSBE-Projekte), welche Modellinformationen und Quellcode-Bäume über HTTPS empfangen und verbreiten
- Die Java-Tools ermöglichen die Einbindung von WhizniumDBE/SBE mit Projektinformationen in den Entwicklungsprozess



Whiznium Werkzeuge

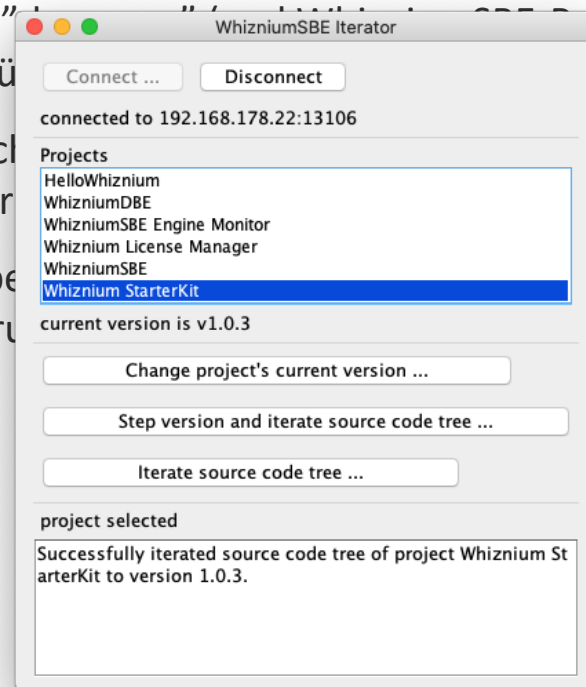
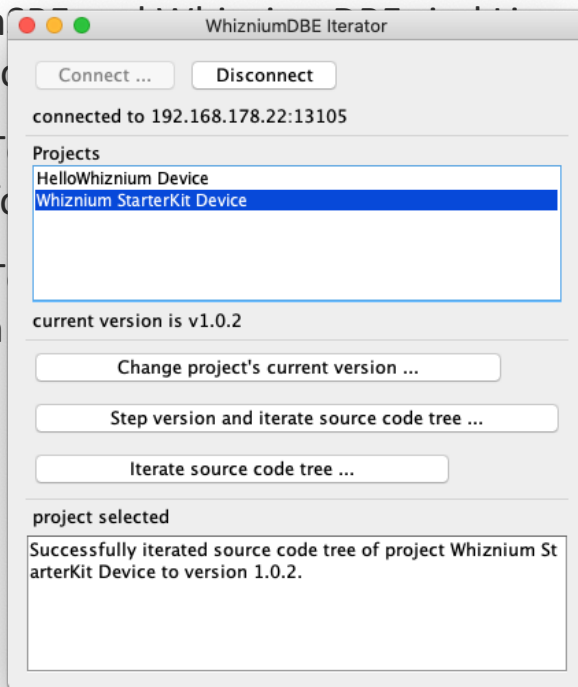
Einbindung in den Entwickler-Alltag

- WhizniumSBE und WhizniumDBE sind Linux-basierte "daemons" (und WhizniumSBE-Projekte), welche Modellinformationen und Quellcode-Bäume über HTTPS empfangen und verbreiten
- Die Java-Tools WhizniumDBE/SBE Bootstrap ermöglichen das Initialisieren von WhizniumDBE/SBE mit Projektinformationen aus einer lokalen Verzeichnisstruktur
- Die Java-Tools WhizniumDBE/SBE Iterator helfen dabei, lokale Quellcode-Bäume von der aktuellen Version in eine neue Version zu überführen. API-Aufrufe ersetzen Klicks

Whiznium Werkzeuge

Einbindung in den Entwickler-Alltag

- WhizniumDBE/SBE (Model-basierte "HelloWorld"-Projekte), welche Modelle in Java-Tree-Strukturen übersetzen
- Die Java-Tree-Struktur ermöglicht die Veranschaulichung der Verzeichnisse der Java-Tree-Struktur
- Die Java-Tree-Struktur hilft dabei, die API-Aufrufe zu verstehen



Whiznium Werkzeuge

Einbindung in den Entwickler-Alltag

- WhizniumSBE und WhizniumDBE sind Linux-basierte "daemons" (und WhizniumSBE-Projekte), welche Modellinformationen und Quellcode-Bäume über HTTPS empfangen und verbreiten
- Die Java-Tools WhizniumDBE/SBE Bootstrap ermöglichen das Initialisieren von WhizniumDBE/SBE mit Projektinformationen aus einer lokalen Verzeichnisstruktur
- Die Java-Tools WhizniumDBE/SBE Iterator helfen dabei, lokale Quellcode-Bäume von der aktuellen Version in eine neue Version zu überführen. API-Aufrufe ersetzen Klicks
- WhizniumDBE-Code wird mit den gängigen Tools Vivado, Quartus, Libero SoC und Simplicity Studio entwickelt
- WhizniumSBE-Code wird mit den gängigen Toolchains gcc/Clang (cross-)kompiliert und kann z.B. in VS Code (remote-)gedebugged werden
- Das Yocto-Projekt hilft, für jede FPGA-SoC Plattform ein passendes Embedded Linux zu bauen, auf dem WhizniumSBE-Projekte laufen

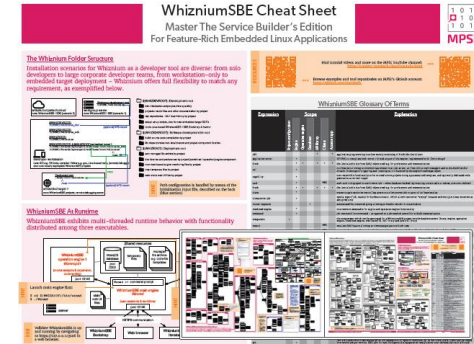
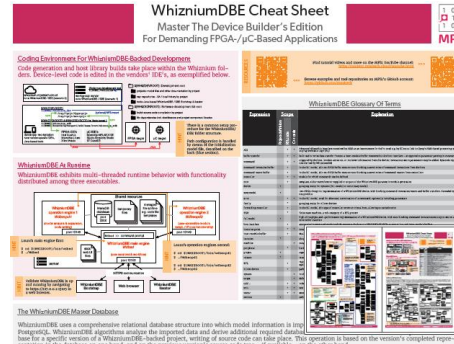
Online-Materialien zu Whiznium

“Getting Started” und mehr

- Die Whiznium-Tools sind auf GitHub frei verfügbar, mitsamt Installationsanweisungen

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

- Das Open Source StarterKit ist verfügbar für diverse Plattformen, die jeweiligen Vendor-Toolspezifischen Anweisungen finden sich ebenfalls auf [GitHub](#)
- The Whiznium Developer Experience auf [YouTube](#)
- Für fortgeschrittene Nutzer gibt es WhizniumSBE/DBE Cheatsheets, welche als Referenz beim Erstellen von Modelldateien dienen



Ausblick

Neue Funktionalität 2021/22

- Upgrade der automatisch generierten Web User Interfaces von “Vanilla JavaScript” zu Vue.js
- Weitere Starter Kit Varianten mit intel PSG CycloneV sowie Lattice CrossLinkNX
- Zusätzliches API-Binding für macOS/Swift
- Ausbau der “Dataflows and Algorithms” Funktionalität in WhizniumDBE
- Open Source “Governance” wanted!

Vielen Dank!

Fragen?

Alexander Wirthmüller

aw@mpsitechnologies.com

+49 (89) 4524 3826

www.mpsitech.com