MODELLBASIERTE CODE-GENERIERUNG FÜR HETEROGENE FPGA-SOC SYSTEME

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

Alexander Wirthmüller
aw@mpsitechnologies.com

FPGA-SoC Übersicht

Hardware und ihre Anwendungen

Produktlinien

Abgrenzung: CPU-seitig geeignet für Embedded Linux

XILINX ZYNQ

- 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

intel CYCLONE STRATIX

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

MICROCHIP Microsemi Product Portfolio RISC-V

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

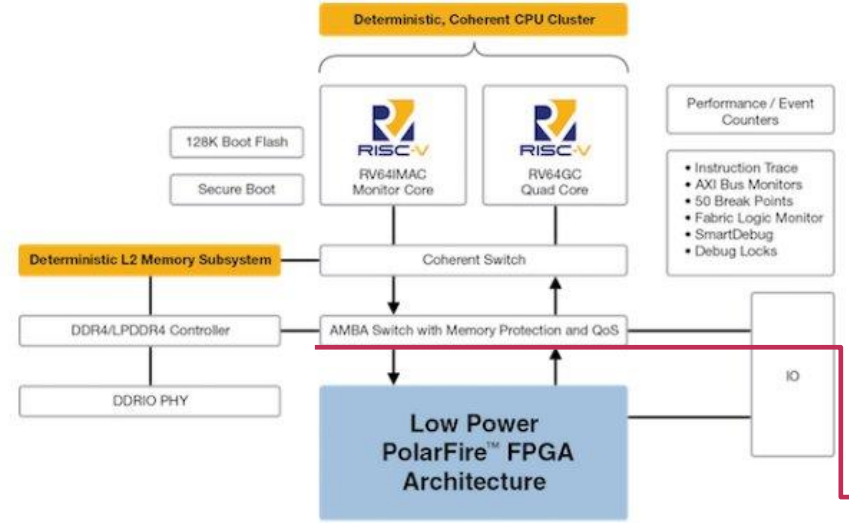
Typische Anwendungen

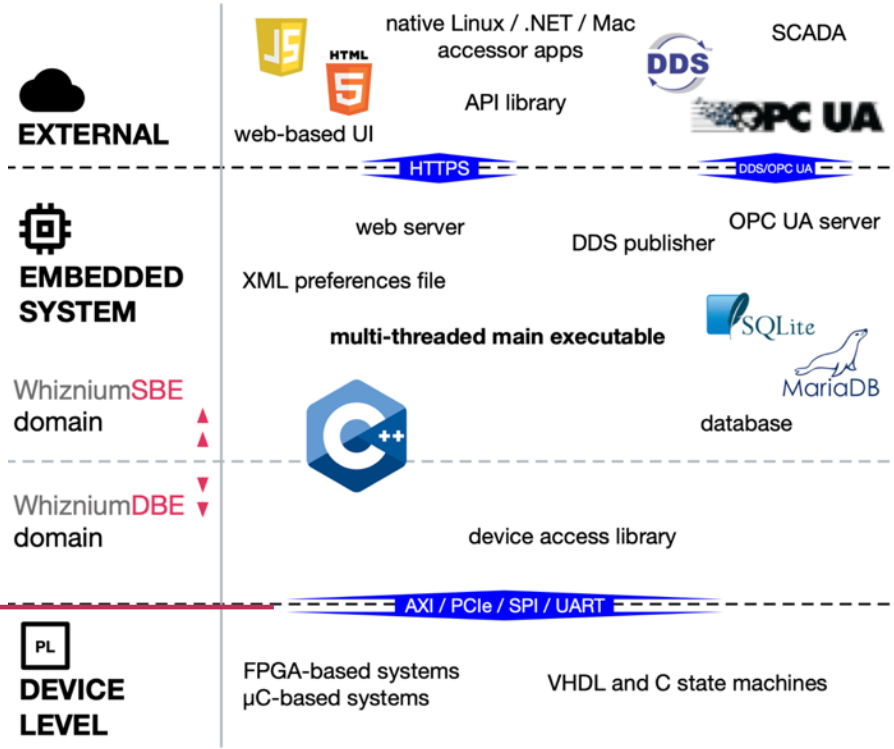
- Klassische Anwendungsgebiete von FPGA's bei denen zusätzlich high-level Steuerung erwünscht ist
- "Datenredutkion" bzw. Vorverarbitung von Quellen mit hoher Bandbreite
- Kameras: Binning, Pixel-level Filter, Kompression, Ecken- und Objekterkennung
- A/D-Wandler: Spektralanalyse, DSP-Filter
- Taktgenaue 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

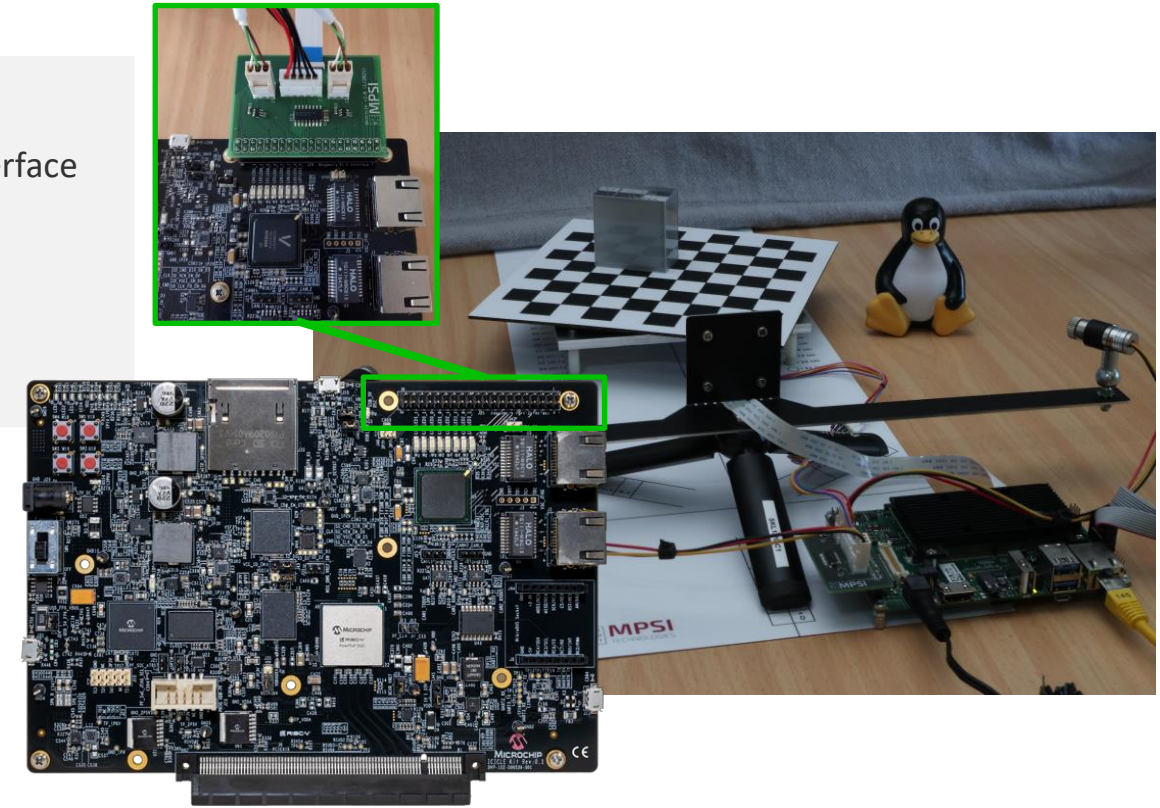




Die Beispielanwendung: Hardware

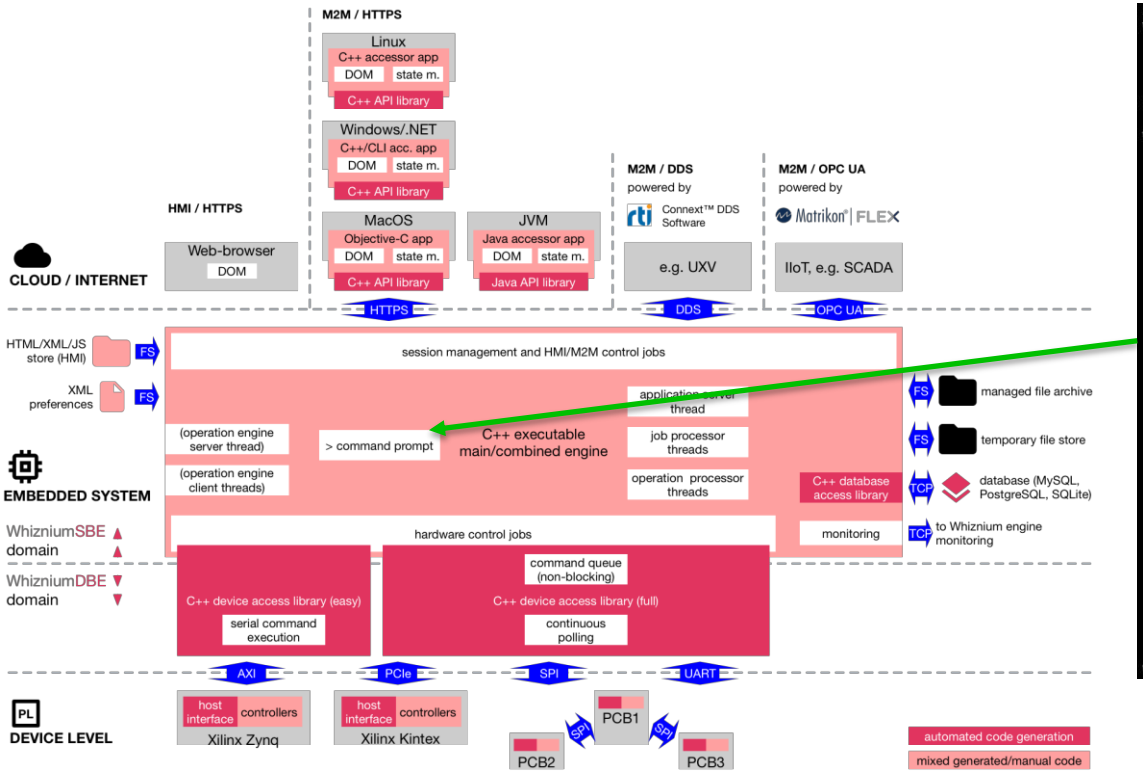
Ein kompakter 3D-Laserscanner

- 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 Punktewolke im Web-Browser

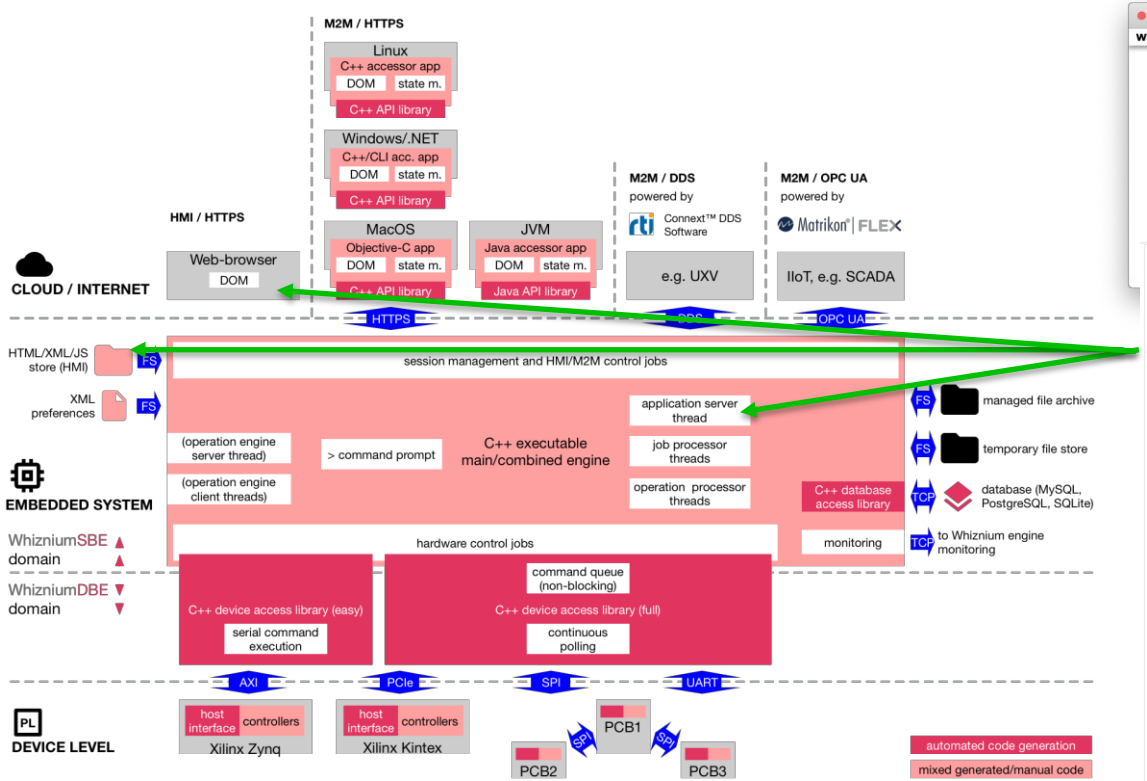


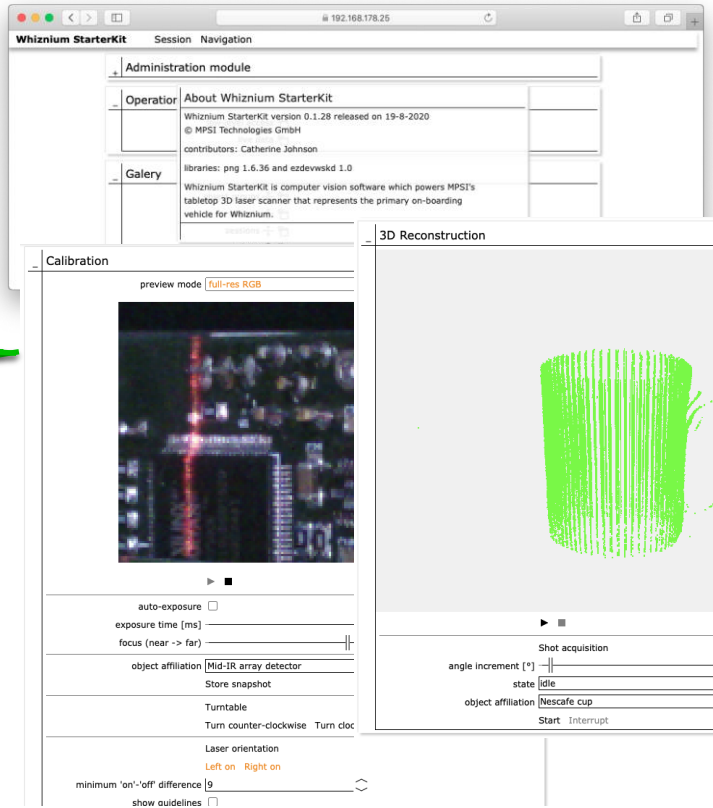

```
root@icicle-kit-es:~/whiznium/bin/wzskcmbd# ./Wzskcmbd
Welcome to Whiznium StarterKit v1.0.5!
        starting 4 job processor threads ... {19255, 19256, 19257, 19258} success
        starting 1 operation processor threads ... {19259} success
        starting application server ... success
        starting OPC UA server ... success
Initialization complete.

Wzskcmbd >> showJobs
        + RootWzsk (1)
                - JobWzskSrcV4l2/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)
                + JobWzskAcqPtcloud/SRV (16)
                        - JobWzskIprTrace/CLI (17)
                        - JobWzskActServo/CLI (18)
                + JobWzskAcqPreview/SRV (19)
                        - JobWzskSrcV412/CLI (20)
                + JobWzskAcqFpgapvw/SRV (21)
                        - JobWzskSrcFpga/CLI (22)
                + JobWzskAcqFpgaflg/SRV (23)
                        - JobWzskSrcFpga/CLI (24)
                + M2msessWzsk (25)
                        - JobWzskSrcSysinfo/CLI (26)
                        - JobWzskIprTrace/CLI (27)
                        - JobWzskIprCorner/CLI (28)
                        - JobWzskActServo/CLI (29)
                        - JobWzskActExposure/CLI (30)
                        - JobWzskActLaser/CLI (31)
                        - JobWzskAcqPtcloud/CLI (32)
                        - JobWzskAcqPreview/CLI (33)
Wzskcmbd >>
```

Die Beispielanwendung: Software

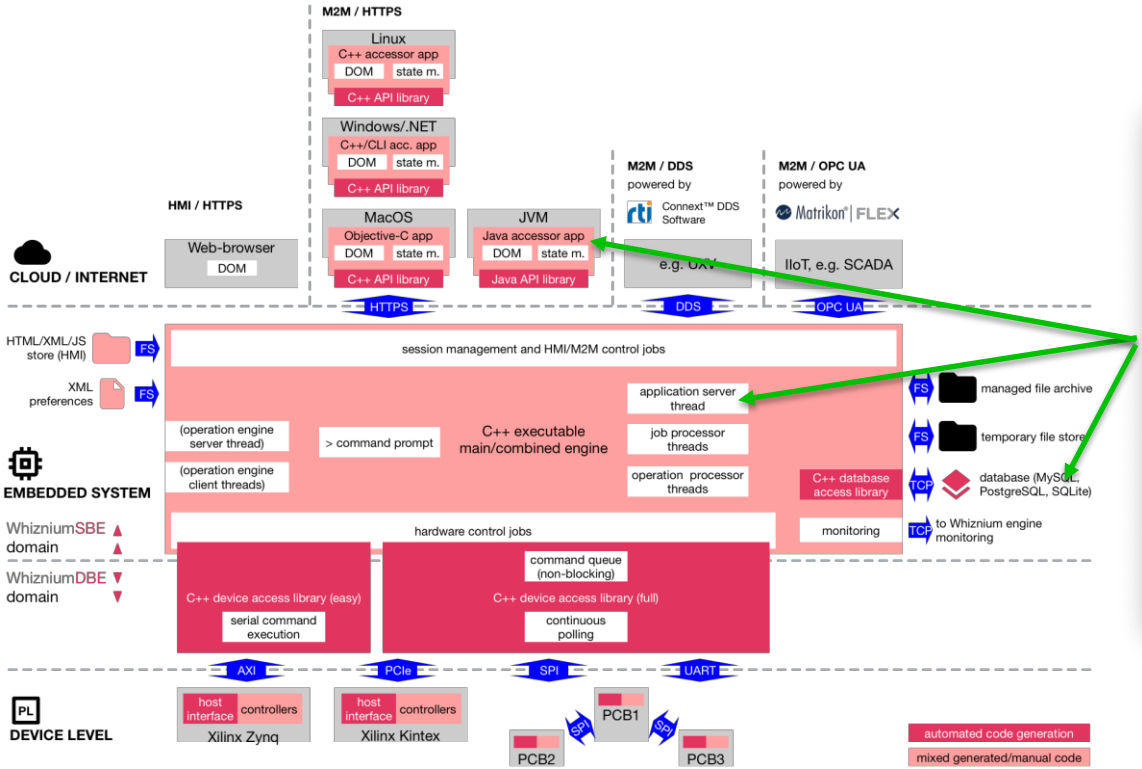
Von Kamera-Rohdaten zur Punktewolke im Web-Browser

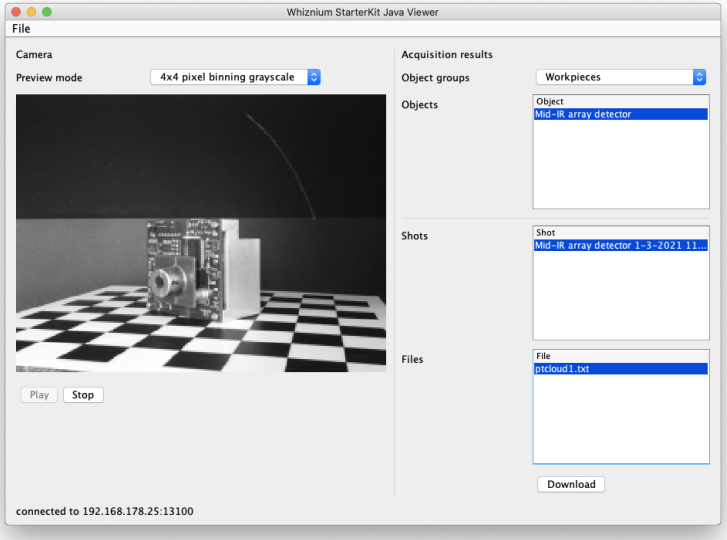




Die Beispielanwendung: Software

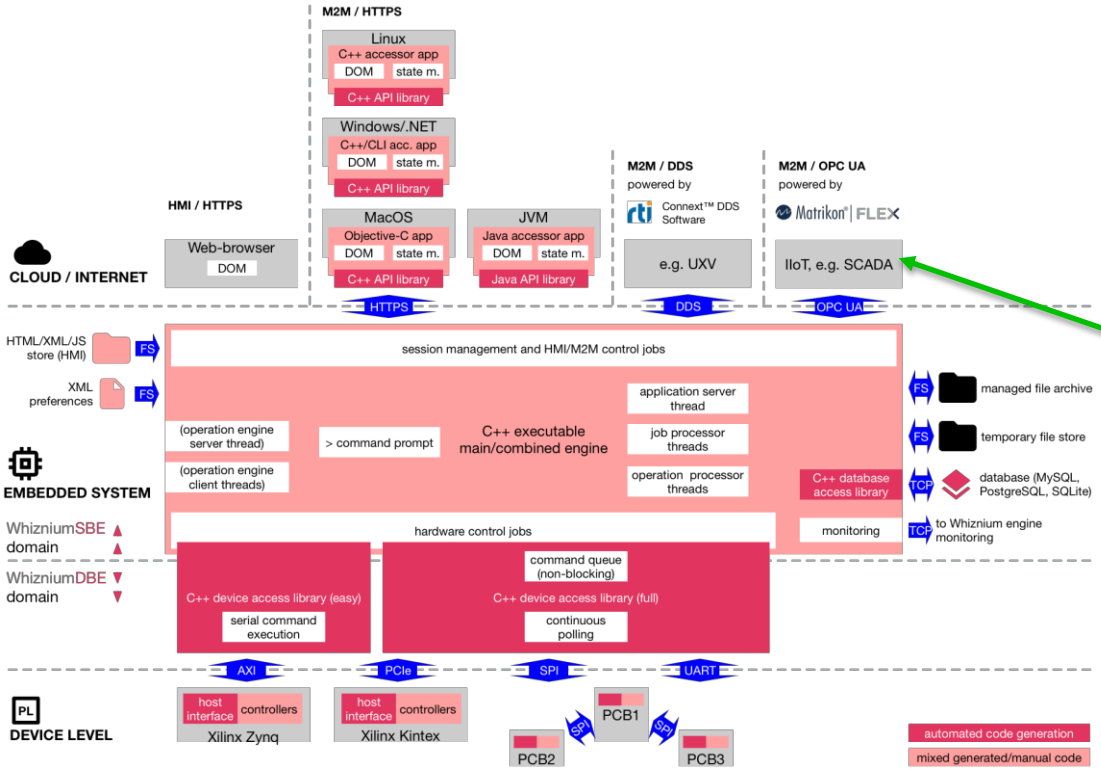
Von Kamera-Rohdaten zur Punktewolke im Web-Browser

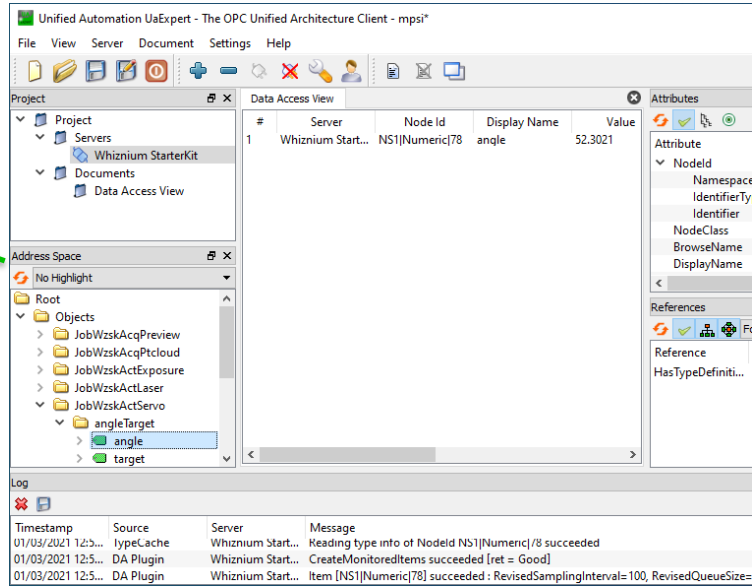




Die Beispielanwendung: Software

Von Kamera-Rohdaten zur Punktewolke 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

IexWdbeMdl v1.1.14								
ImeIMUnit	srefSilRefWdbeMUnit	sref	Title	Easy	srefKToolch	Comment		
fpga	**mpfs250t-fcvg484**	**iccl**	**Microchip PolarFire Soc Icicle kit**	**true**	**libero**			
	ImeIMModule.srefIxVBa	hsrefSupRefWdb	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								

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

Ansteuerung des Drehteller-Schrittmotors

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											

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

Ansteuerung des Drehteller-Schrittmotors

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													

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

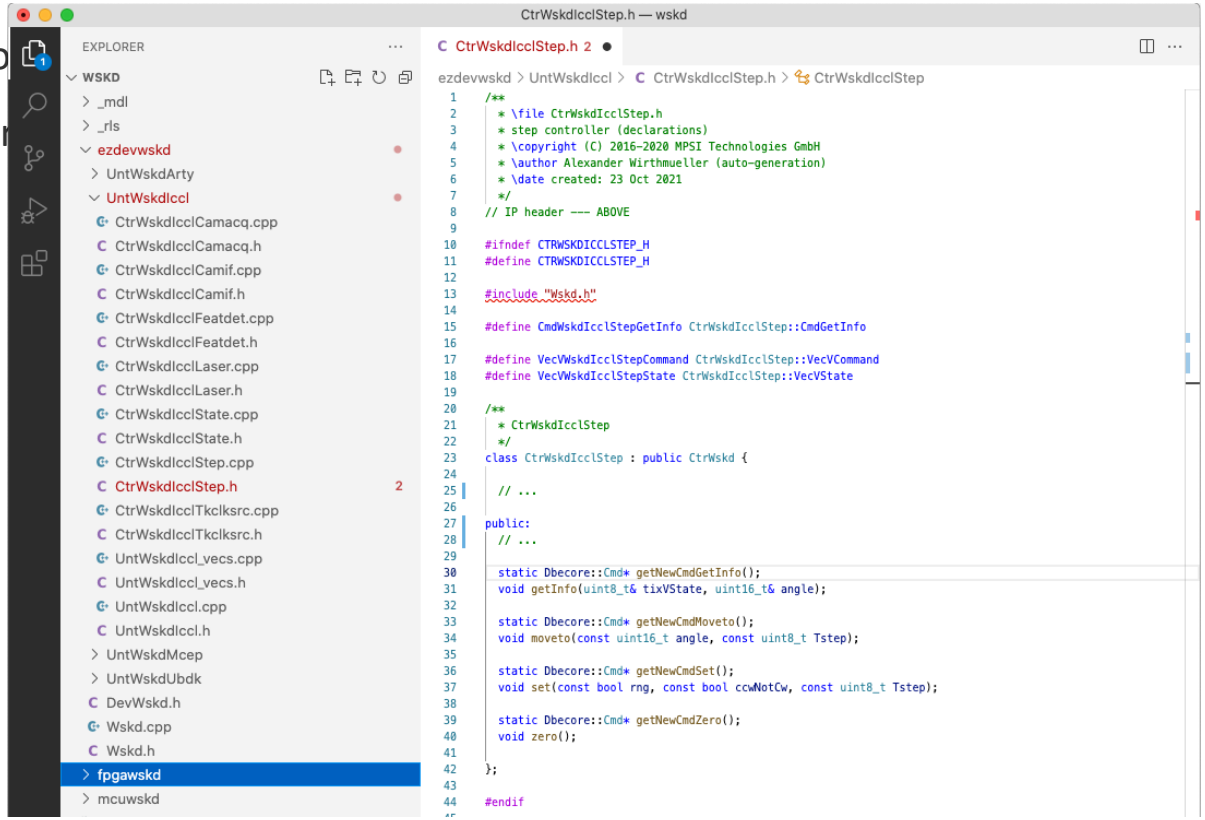
Ansteuerung des Drehteller-Schrittmotors

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

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

Ansteuerung des Drehteller-Schrittmotors

- Moduldefinitio
- Linux-seitig vor



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

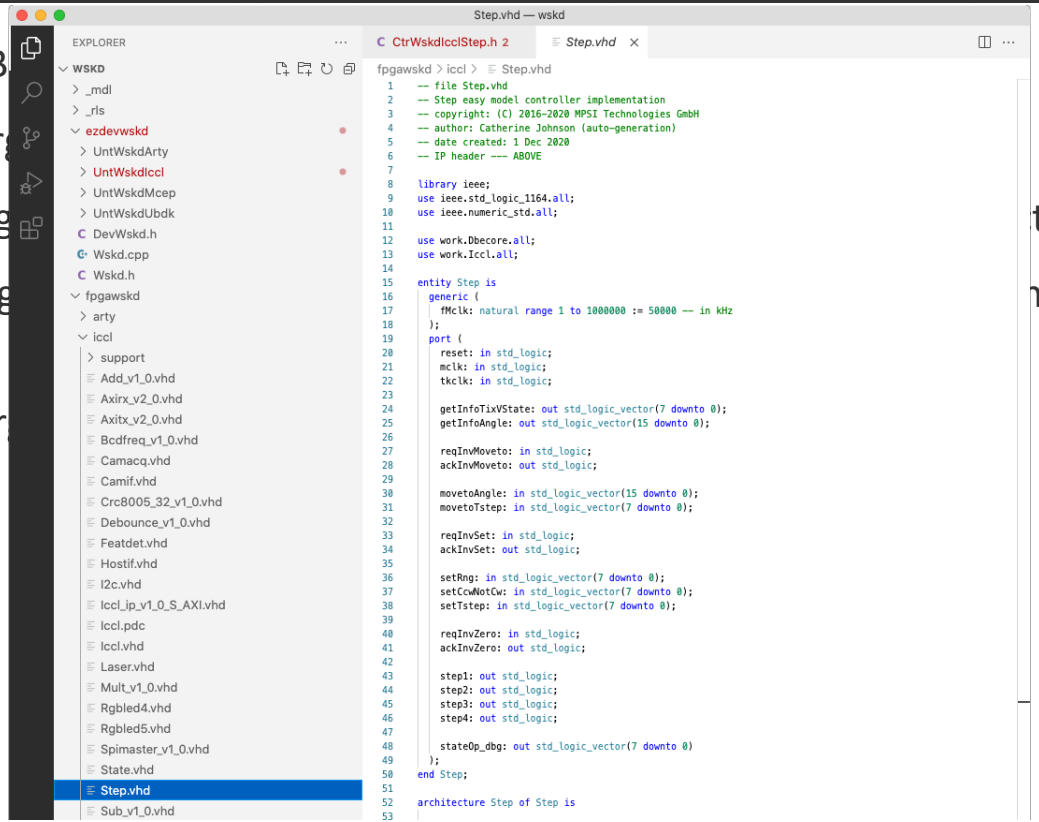
Deep Dive I: Vom C++ Befehl zum RTL Zustandsautomat

Ansteuerung des Drehteller-Schrittmotors

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

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

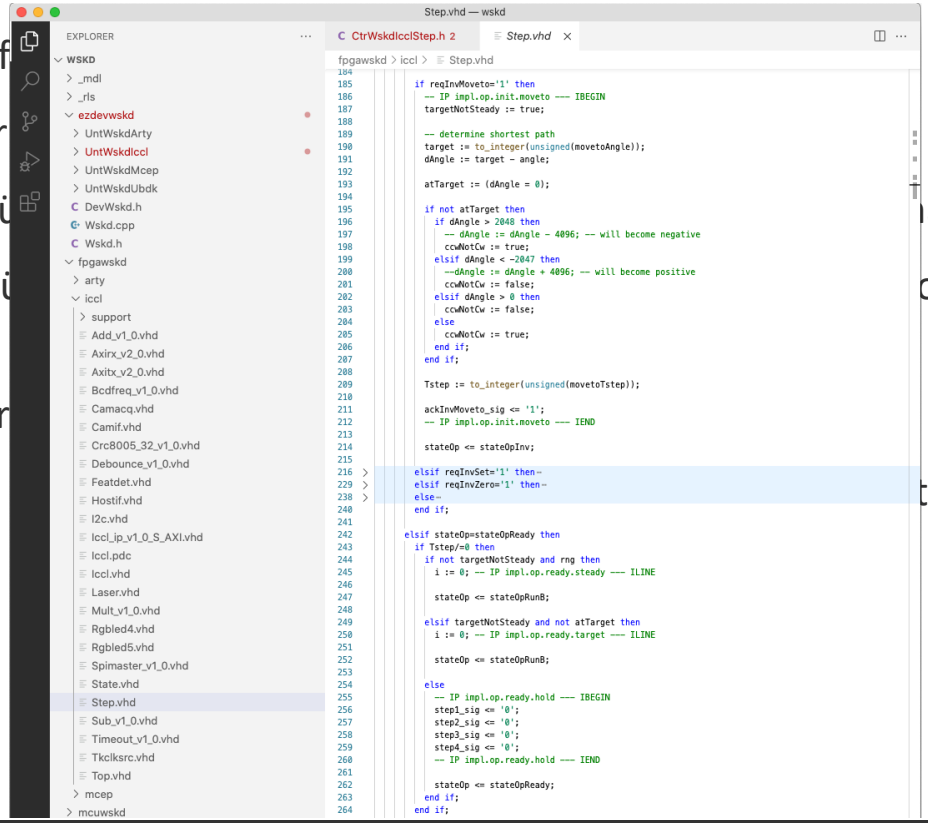
Ansteuerung des Drehteller-Schrittmotors

- Moduldefinition, Bef
- Linux-seitig vordergr
- Linux-seitig hintergrü
- FPGA-seitig hintergrü
- FPGA-seitig vordergr
- FPGA-seitig manuell

aracter Device Treiber (AXI)

ost interface" Modul, CRC-Absicherung

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/>

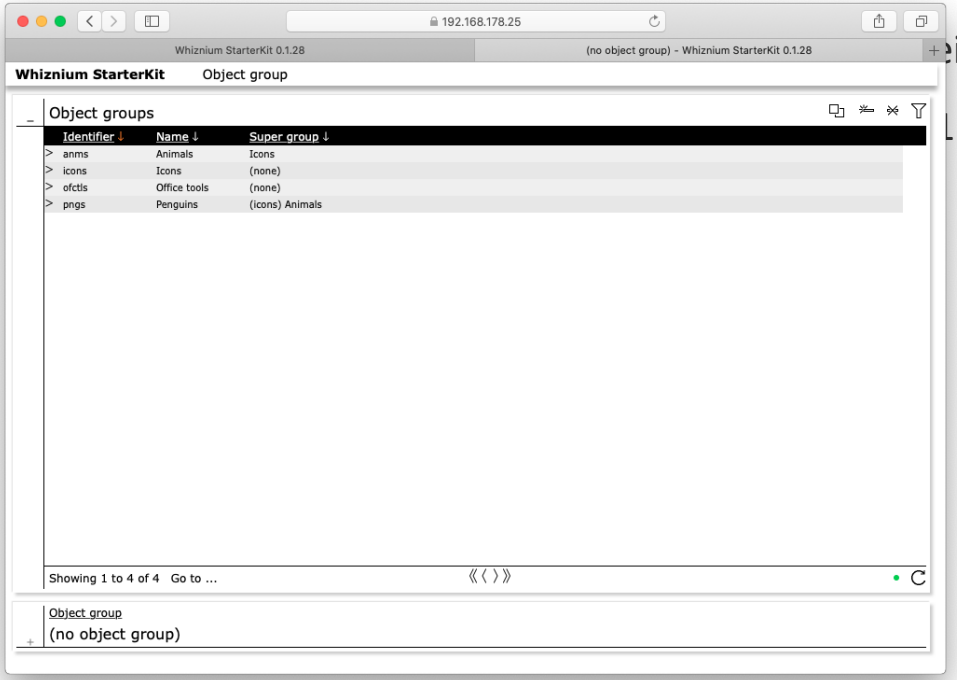
Deep Dive III: Metadaten und der “Lowering”-Prozess in Whiznium

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

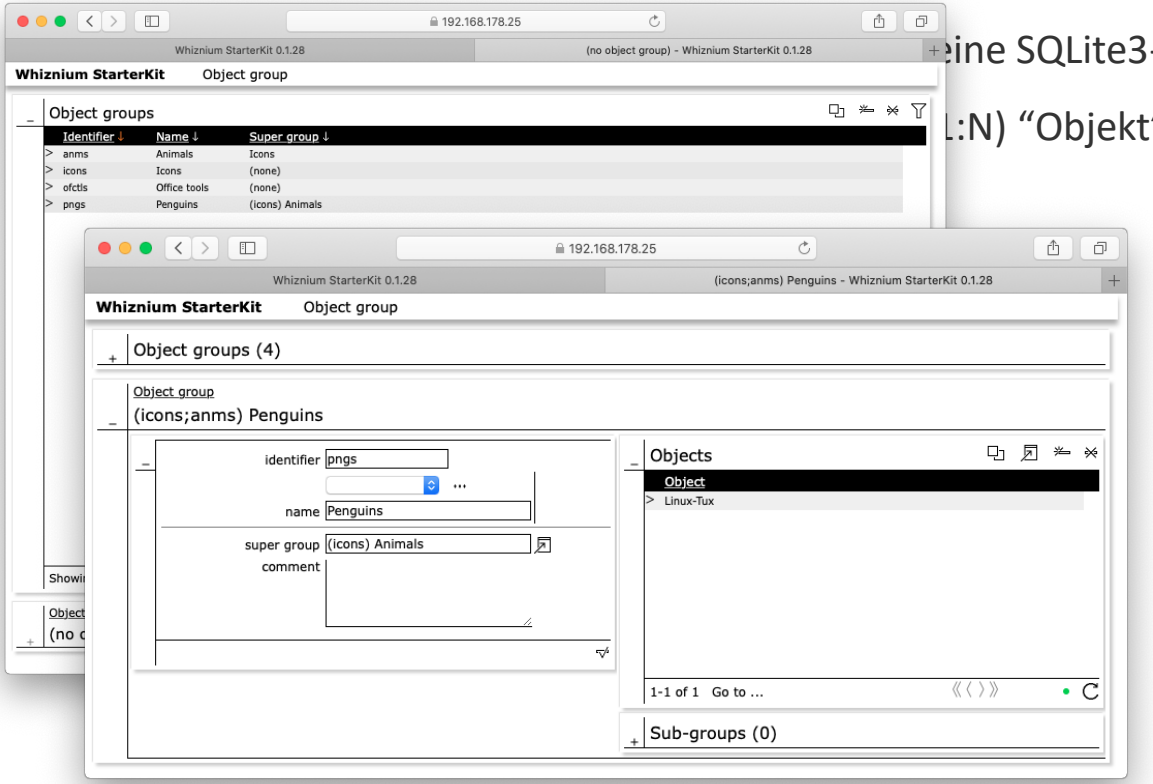


ine SQLite3-Datenbank unterstützt

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

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

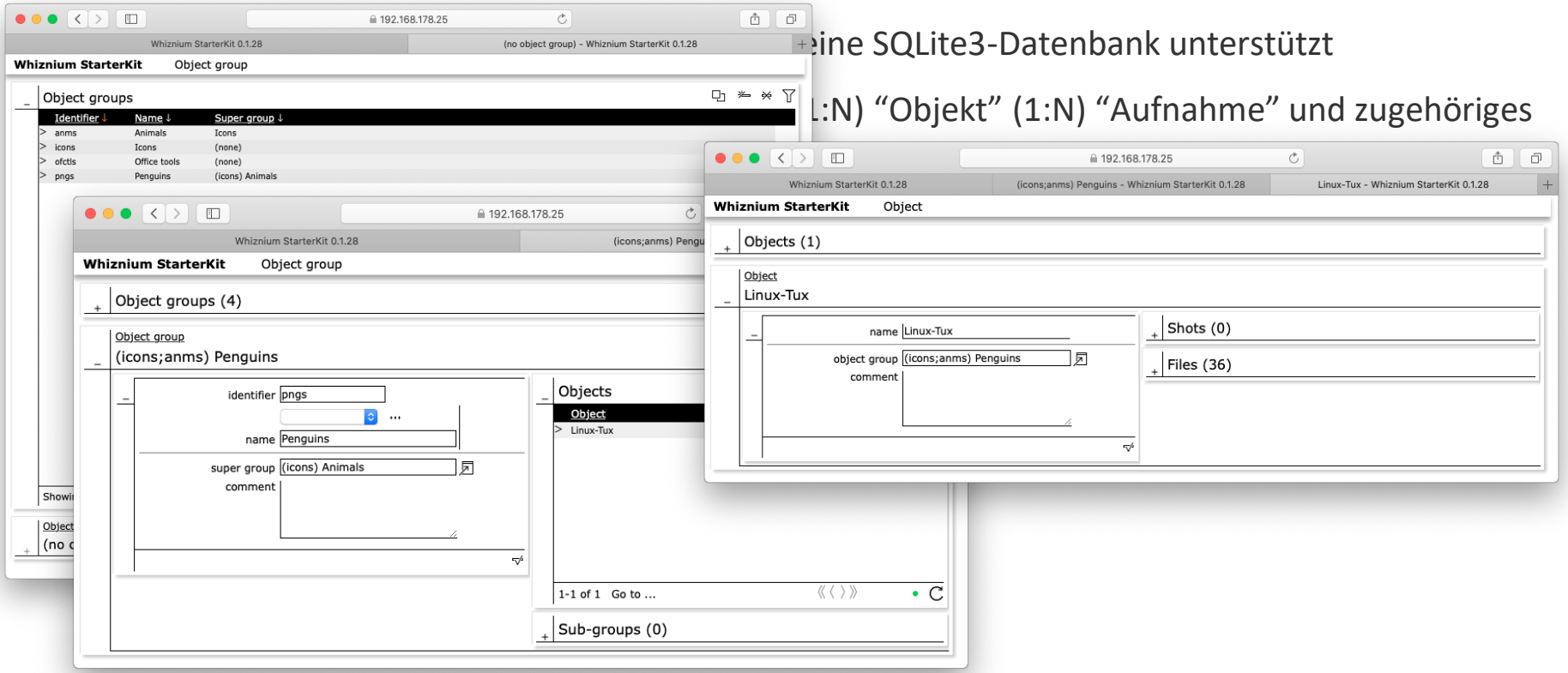
Von SQL-Datenbankstruktur über Code und UI-Elemente zu XML-/JSON-Blöcken



- ...ine SQLite3-Datenbank unterstützt
- ...: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

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

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

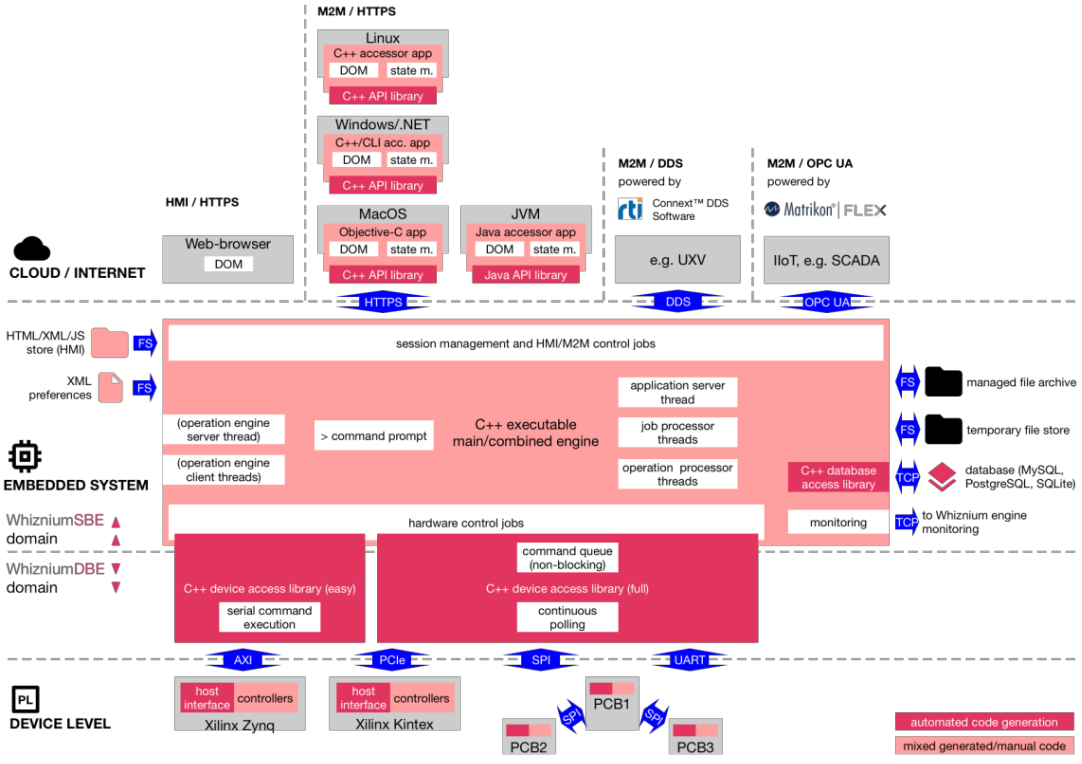
Von SQL-Datenbankstruktur über Code und UI-Elemente zu XML-/JSON-Blöcken

- 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 Lizenzeinschränkungen
- 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

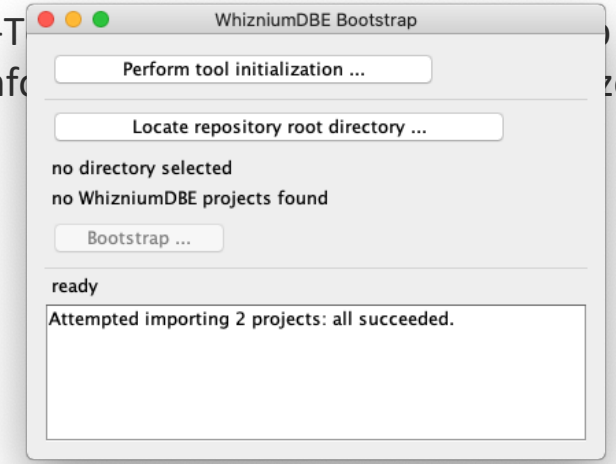
Einbindung in den Entwickler-Alltag

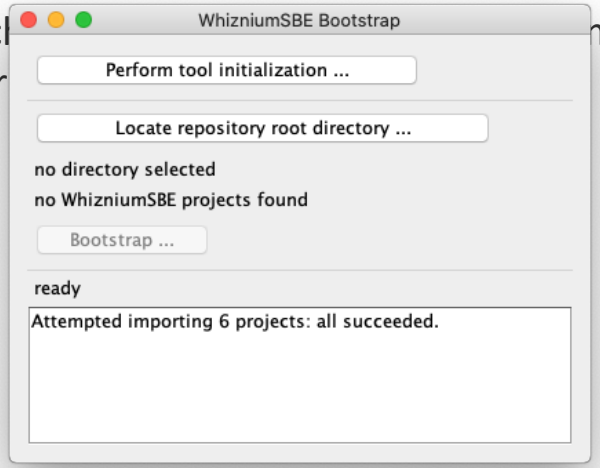
- WhizniumSBE und WhizniumDBE sind Linux-basierte ”daemons” (und WhizniumSBE-Projekte), welche Modellinformationen und Quellcode-Bäume über HTTPS empfangen und verbreiten
- Die Java-Tools WhizniumDBE/SBE Bootstrap ermöglichen das Initialisieren von WhizniumDBE/SBE mit Projektinformationen aus einer lokalen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





Whiznium Werkzeuge

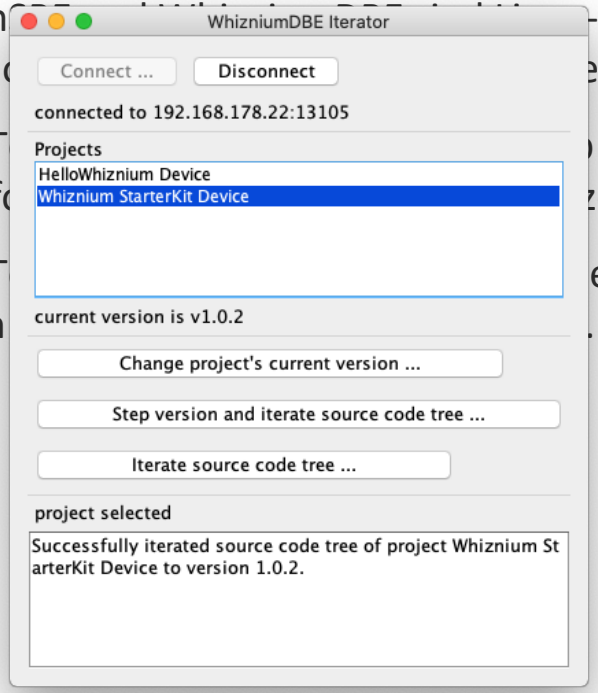
Einbindung in den Entwickler-Alltag

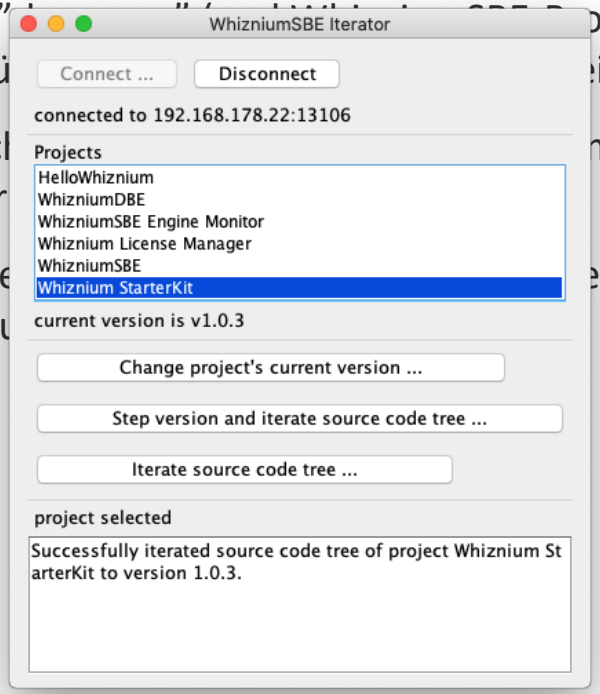
- WhizniumSBE und WhizniumDBE sind Linux-basierte ”daemons” (und WhizniumSBE-Projekte), welche Modellinformationen und Quellcode-Bäume über HTTPS empfangen und verbreiten
- Die Java-Tools WhizniumDBE/SBE Bootstrap ermöglichen das Initialisieren von WhizniumDBE/SBE mit Projektinformationen aus einer lokalen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

- Whiznium[illegible]-basierte "[illegible]" (und Whiznium SBE-Projekte), welche Mo[illegible]e-Bäume ü[illegible]iten
- Die Java-T[illegible] ermöglich[illegible]mDBE/SBE mit Projektinfo[illegible]zeichnisstr[illegible]
- Die Java-T[illegible]elfen dabe[illegible]r aktuellen Version in[illegible]. API-Aufru[illegible]





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

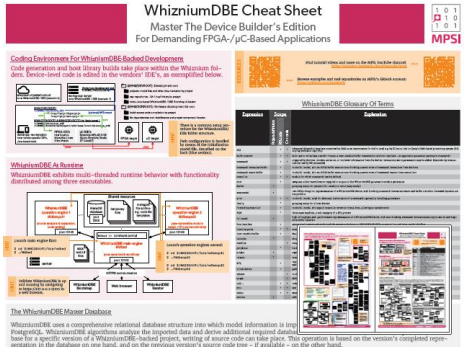
Online-Materialien zu Whiznium

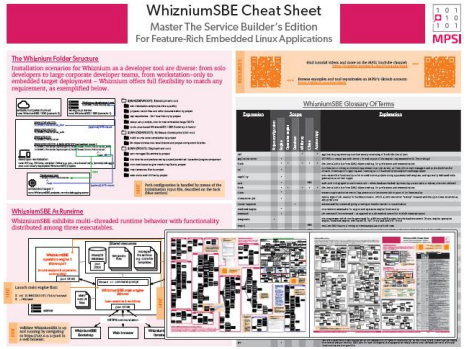
“Getting Started” und mehr

- 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-Tool-spezifischen Anweisungen finden sich ebenfalls auf GitHub
- The Whiznium Developer Experience auf YouTube
- Für fortgeschrittene Nutzer gibt es WhiznumSBE/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