Vendor-agnostic probing of FPGA designs
For efficient run-time debugging

MPSI TECHNOLOGIES

Alexander Wirthmüller
aw@mpsitechnologies.com

Introduction

About me

- Based in Munich
- Diploma in Electrical Engineering

- Founder and Director at MPSI Technologies
- MPSI Technologies: make Embedded Software development more pleasant – by replacing repetitive tasks with model-based source code generation

- Senior Staff Engineer with Symeo / indie Semiconductor (industrial radar)

FPGA Design Probing

Scope | IDE-based vs. generic | Platforms

- For CPU code, iterations (compile + link) are relatively fast:
  Use debugger iff problem encountered (unexpected behavior, segmentation fault, …)
- Unit and integration tests are mandatory in quality projects
- Test-driven development also exists (but is the exception)

- FPGA designs, are costly (time-wise) to iterate, also harder to grasp for the human brain:
  Logic test-benches and simulation are part of every design
- Complexity starts with integration [of designs from other vendors]
- Full design simulation and randomized testing still possible (and done)
- “Outside-world” triggers and concurrency often hold surprises → case for design probing

FPGA Design Probing

Scope | IDE-based vs. generic | Platforms

Vendor offerings

- Altera: Signal Tap Logic Analyzer, Signal Probe + System Console
- AMD: Virtual I/O + Vivado
- Efinix: Virtual I/O + Efinity / GTKwave
- Lattice: Reveal + Diamond
  Microchip: SmartDebug + Windows UI

→ all require proprietary IP blocks and/or software, most use the JTAG interface as debug port

This work

- General-purpose tracking of 15-bit wide signals with (time-domain) compression
- FSM state tracking + statistics with compression
- Read-out to CPU and .vcd output for visualization in GtkWave

Key enablers

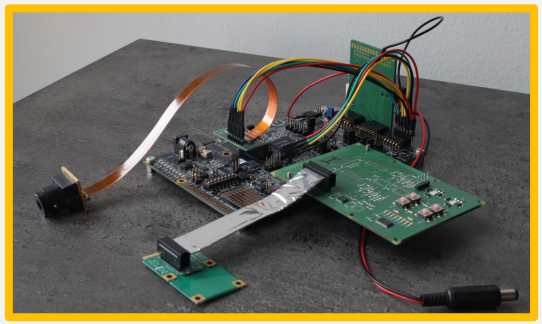
- PHY-agnostic CPU (host) interface with bytecode protocol
- Vendor-agnostic instantiation of memory primitives
- Powered by Open Source **WhiziumDBE**

FPGA Design Probing

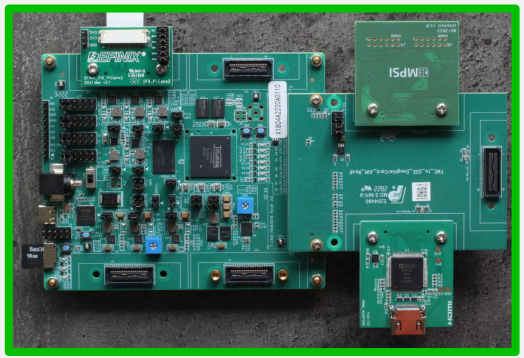
Scope | IDE-based vs. generic | Platforms

Lattice CrossLink-NX eval board (cleb)

- Device LIFCL-40-9BG400C
- Host: PC / NUC (UART-over-USB)
- ✗ DDR memory



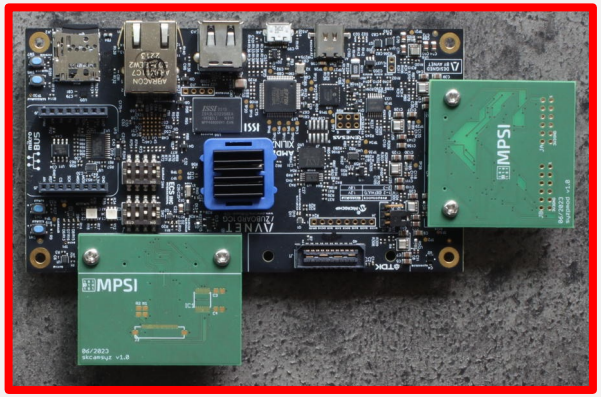
Efinix Titanium Ti180 dev kit (tidk)



- Device Ti180M484I3
- Host: Buildroot Linux on soft RISC-V (AXI/32)
- ✓ 256 MB LPDDR4 SDRAM

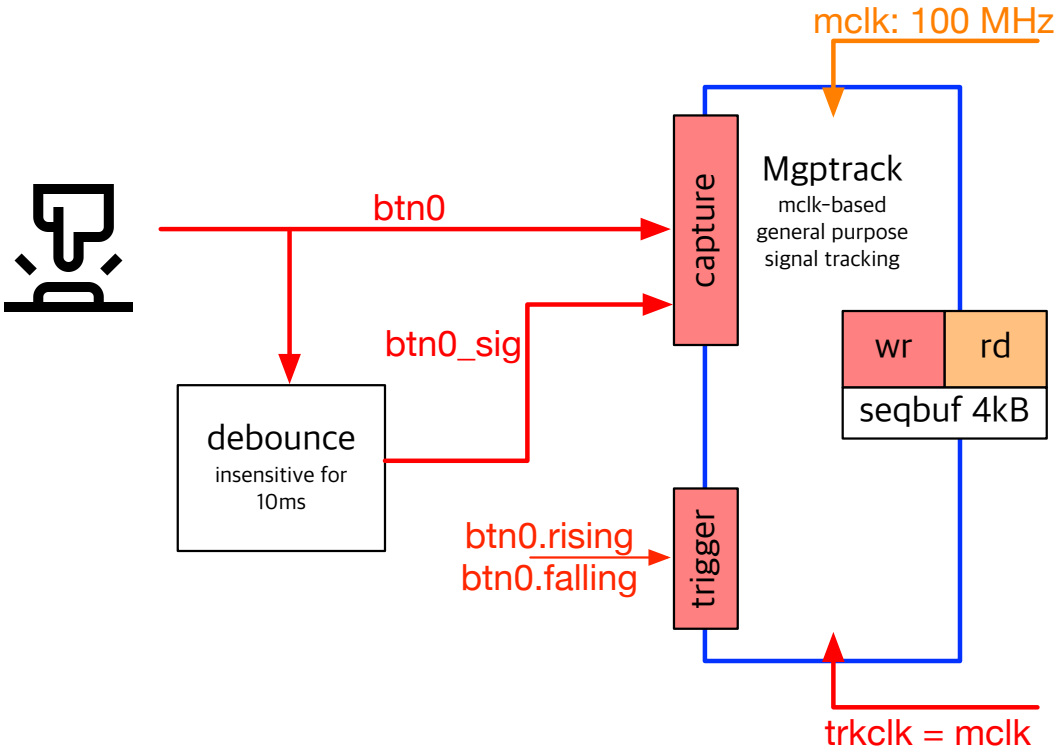
Avnet ZUBoard 1CG (zudk)

- Based on Zynq UltraScale+ 1CG
- Device ZU1CG-1SBVA484I
- Host: Yocto Linux on ARMv8 (AXI/64)
- ✓ 1GB DDR4 SDRAM



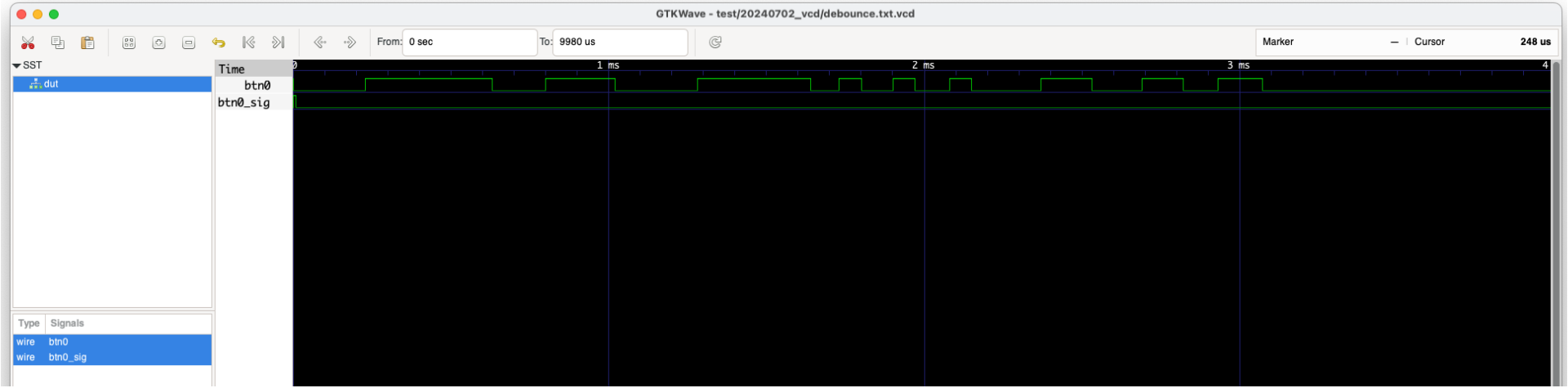
Example #1: Pushbutton De-bounce

Context | Capture | Gptrack_Easy_v1_0 | Data format | Readout



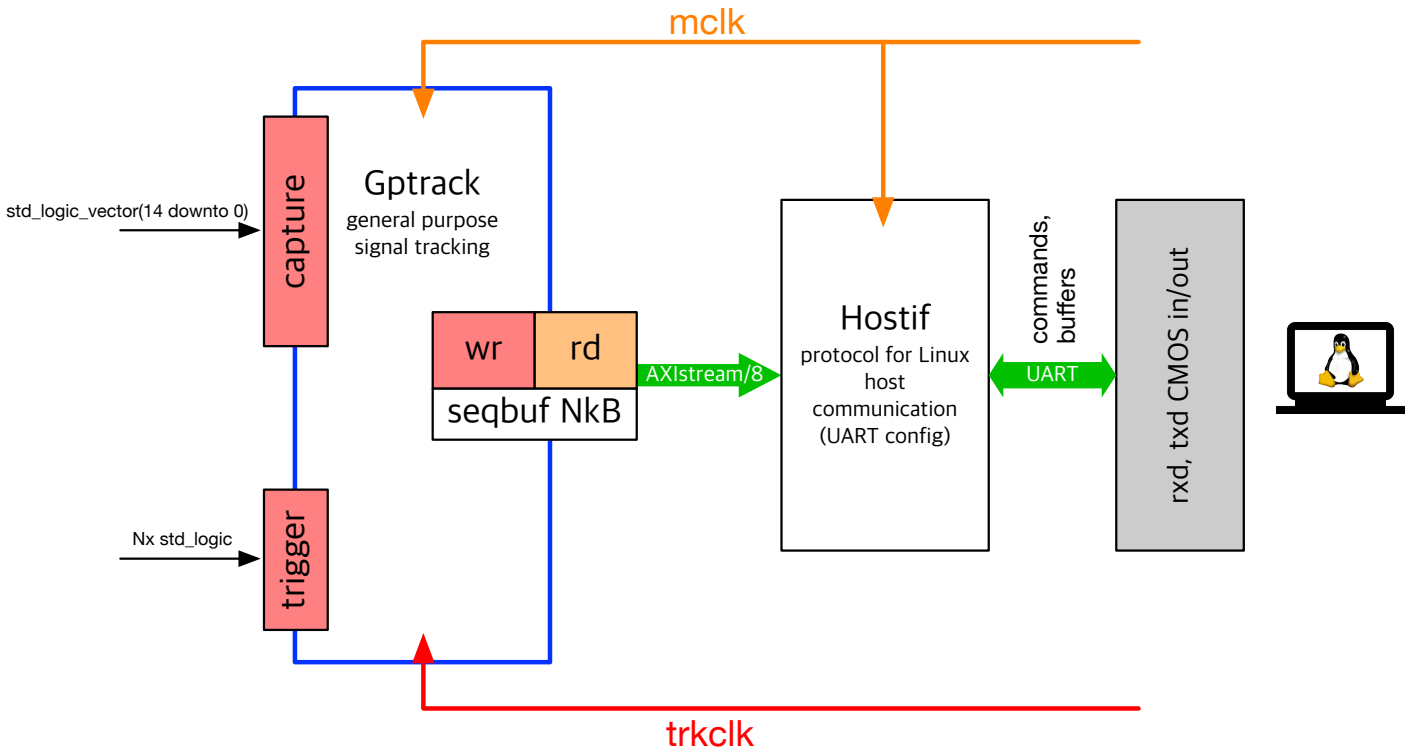
Example #1: Pushbutton De-bounce

Context | Capture | Gptrack_Easy_v1_0 | Data format | Readout



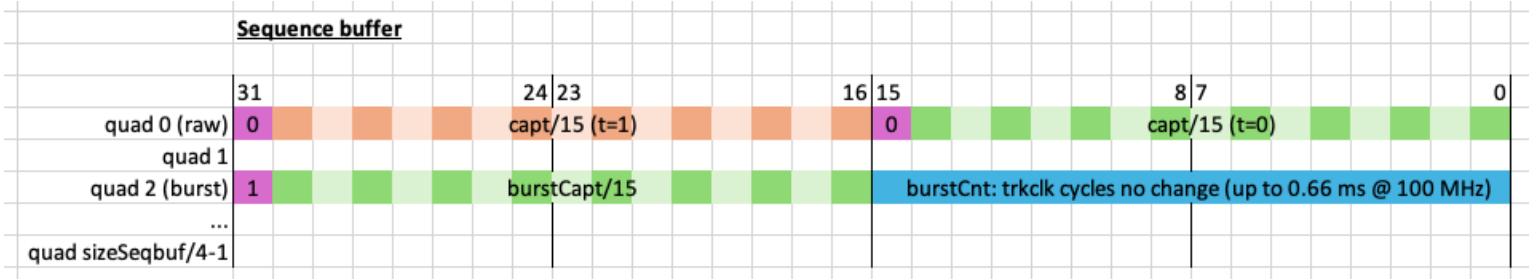
Example #1: Pushbutton De-bounce

- 15-bit capture port
- Selectable trigger
- Variable size sequence buffer
- Tracking clock (trkclk)
- Master clock for readout (mclk)
- Adaptive readout width



Example #1: Pushbutton De-bounce

- 32-bit words, raw vs. burst flag, two captures (raw) vs. capture + duration (burst)



Example #1: Pushbutton De-bounce

- Three commands for trigger selection, acquisition start and status polling

```
mgptrack.select(
  staTixVTrigger={void,btn0.rising,btn0.falling,ackInvTkclksrcSetTkst.rising},
  stoTixVTrigger={void,btn0.rising;btn0.falling;ackInvTkclksrcSetTkst.rising})

mgptrack.set(rng={false,true},TCapt=[uint32])

(tixVState{idle,arm,acq,done}) = mgptrack.getInfo()
```

- One buffer transfer for resulting sequence buffer

```
(file) = seqbufMgptrackToHostif.read(reqlen[uint32])
```

Example #2: CPU-FPGA Transaction

Context | Capture | Fsmtrack_Easy_v1_0 | Data format | Readout

- PHY-agnostic CPU host interface with byte protocol to invoke commands / write/read buffers

runCmd general case (parameters inv/ret: K/L)

	byte 0	byte 1	...						
→	Buffer::INV	tixVController	tixVCommand	invparlen 31	...	...	... 0	CRC 15	... 0
←	ACK/NAK								
→	invpar byte 0	...	... N-1	CRC 15	... 0				
←	ACK/NAK								
→	Buffer::RET	0x00	Bufop::POLL	0x00	0x00	0x00	0x04	CRC 15	... 0
←	ACK/NAK								
←	tixVController	tixVCommand	retparlen 15	... 0	CRC 15	... 0			
→	ACK/NAK								
←	retpar byte 0	...	... N-1	CRC 15	... 0				
→	ACK/NAK								

Legend

→ host to FPGA		← FPGA to host	
handshake	addressing	length	payload

mpsitech — ssh -l root 192.168.17

```
root@ST1ME-XU1-6CG-1E-D11E-G1:~/whiznium/bin/fstdterm# ./Fstdterm mplet
Welcome to the interactive terminal for Fstd version 0.2.3, unit UntFstd
UntFstdMplet [disconnected] >> showRxtx
UntFstdMplet [disconnected] >> connect
UntFstdMplet >> tkclksrc.getTkst()
        tkclksrc.getTkst()
tx 0x010B0000000000491A
rx 0x55
tx 0x00001000000040060
rx 0x55
rx 0x0B0000049C18
tx 0x55
rx 0x0002616FC74F
tx 0x55
         = (tkst=156015)
```

Example #2: CPU-FPGA Transaction

Context | Capture | Fsmtrack_Easy_v1_0 | Data format | Readout

- FSM states for individual sub-transfers and mapping to ID (C enum style)

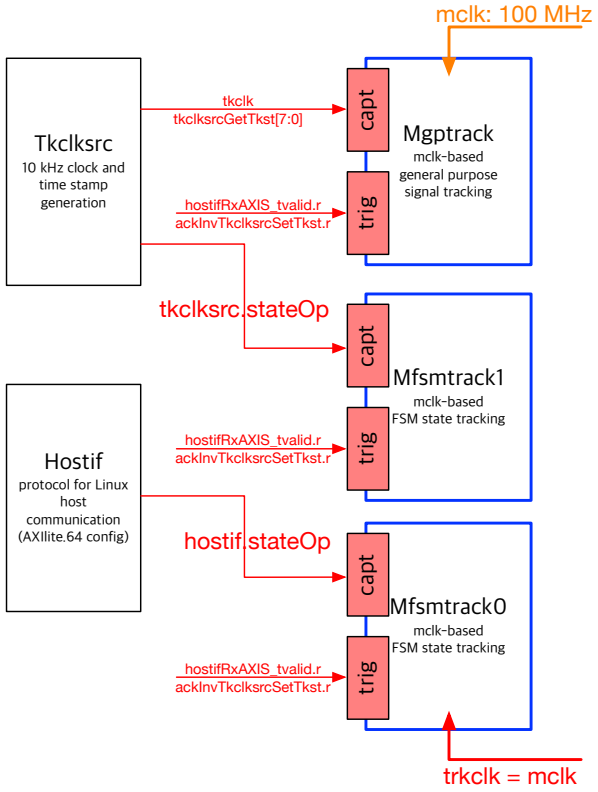
```
type stateOp_t is (
  stateOpInit,
  stateOpIdle,
  stateOpRxopA, stateOpRxopB, stateOpRxopC, stateOpRxopD, stateOpRxopE,
  stateOpTxpollA, stateOpTxpollB, stateOpTxpollC, stateOpTxpollD, stateOpTxpollE, stateOpTxpollF, stateOpTxpollG,
  stateOpTxA, stateOpTxE, stateOpTxB, stateOpTxC, stateOpTxD, stateOpTxF, stateOpTxG,
  stateOpTxbufA, stateOpTxbufB, stateOpTxbufC, stateOpTxbufD, stateOpTxbufE, stateOpTxbufF, stateOpTxbufG, stateOpTxbufH,
  stateOpRxA, stateOpRxB, stateOpRxC, stateOpRxD,
  stateOpRxbufA, stateOpRxbufB, stateOpRxbufC, stateOpRxbufD, stateOpRxbufE, stateOpRxbufF,
  stateOpInv
);
```

```
stateOp_dbg <= x"00" when stateOp=stateOpInit
    else x"10" when stateOp=stateOpIdle
    else x"20" when stateOp=stateOpRxopA
    else x"21" when stateOp=stateOpRxopB
    else x"22" when stateOp=stateOpRxopC
    else x"23" when stateOp=stateOpRxopD
    else x"24" when stateOp=stateOpRxopE
    else x"30" when stateOp=stateOpTxpollA
    else x"31" when stateOp=stateOpTxpollB
    ...
    else x"80" when stateOp=stateOpInv
    else (others => '1');
```

Example #2: CPU-FPGA Transaction

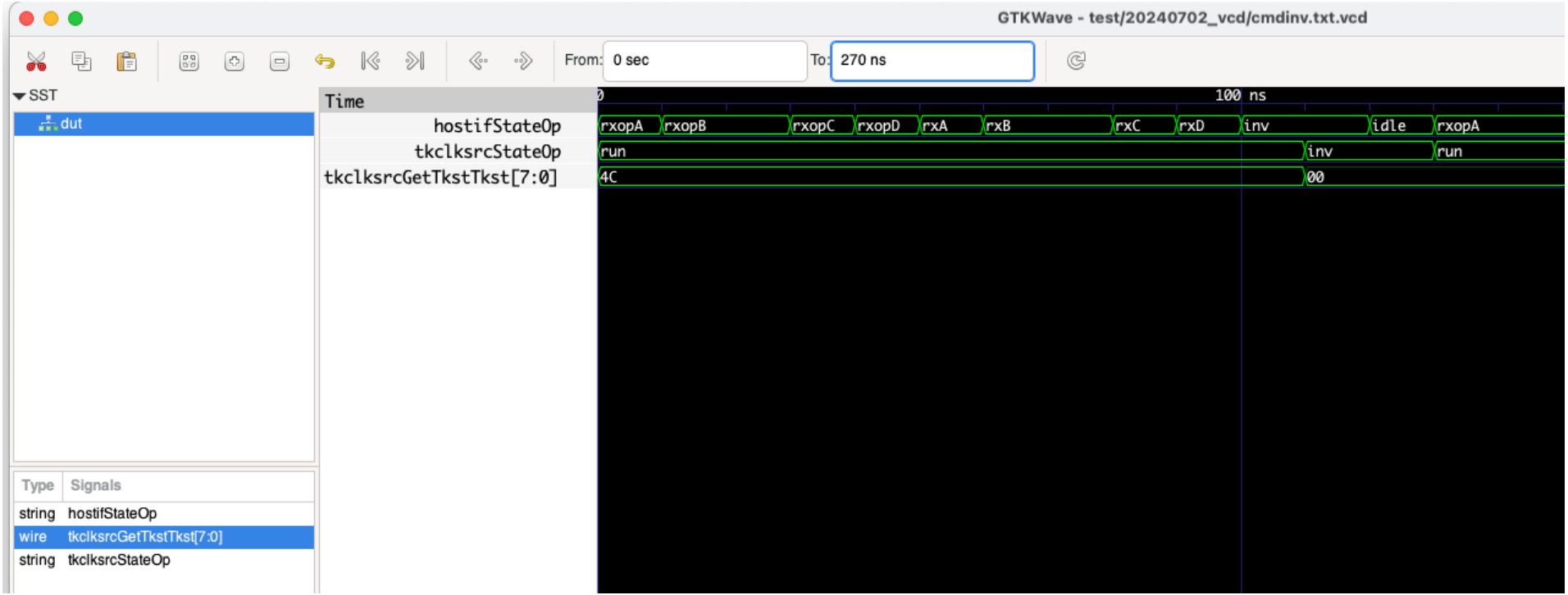
Context | Capture | Fsmtrack_Easy_v1_0 | Data format | Readout

- DUT comprises host interface and connected module 10 kHz clock source
- Trigger on first received byte from CPU
- Capture FSM progress and tkst (time stamp) update
- CPU-triggered command is `tkclksrc.setTkst(tkst=0)`



Example #2: CPU-FPGA Transaction

Context | Capture | Fsmtrack_Easy_v1_0 | Data format | Readout



Example #2: CPU-FPGA Transaction

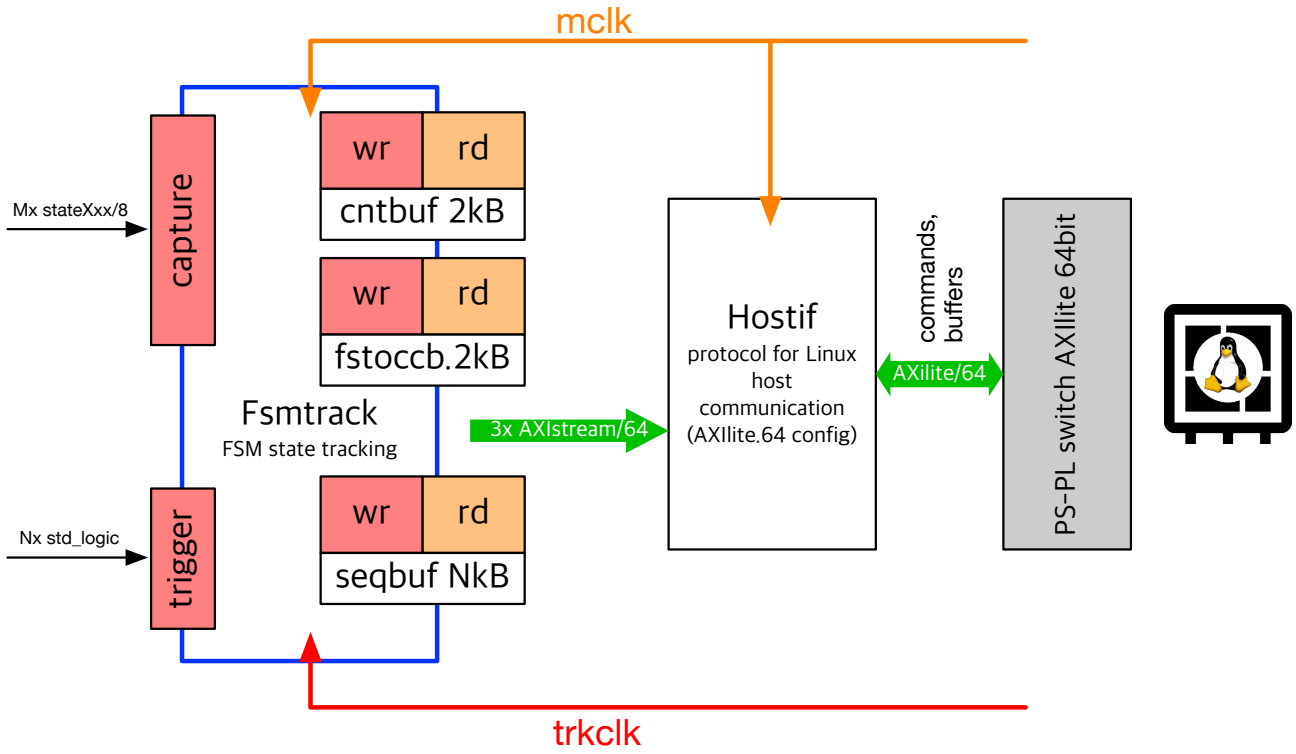
- FSM state statistics output (different example)

```
states covered: 0x00 (init), 0x01 (idle), 0x10 (runA), 0x11 (runB), 0x20 (done)

states missed: 0x12 (runC), 0x30 (reset)

first state occurrence in ms:
[0.000000] 0x00 (init)
[0.000010] 0x01 (idle)
[0.003320] 0x10 (runA)
[0.003348] 0x11 (runB)
[0.019200] 0x20 (done)

state occurrence and share:
0x00 (init): 88.0% (1760/2000)
0x01 (idle): 0.1% (2/2000)
0x10 (runA): 3.3% (66/2000)
0x11 (runB): 6.6% (132/2000)
0x20 (done): 2% (40/2000)
```

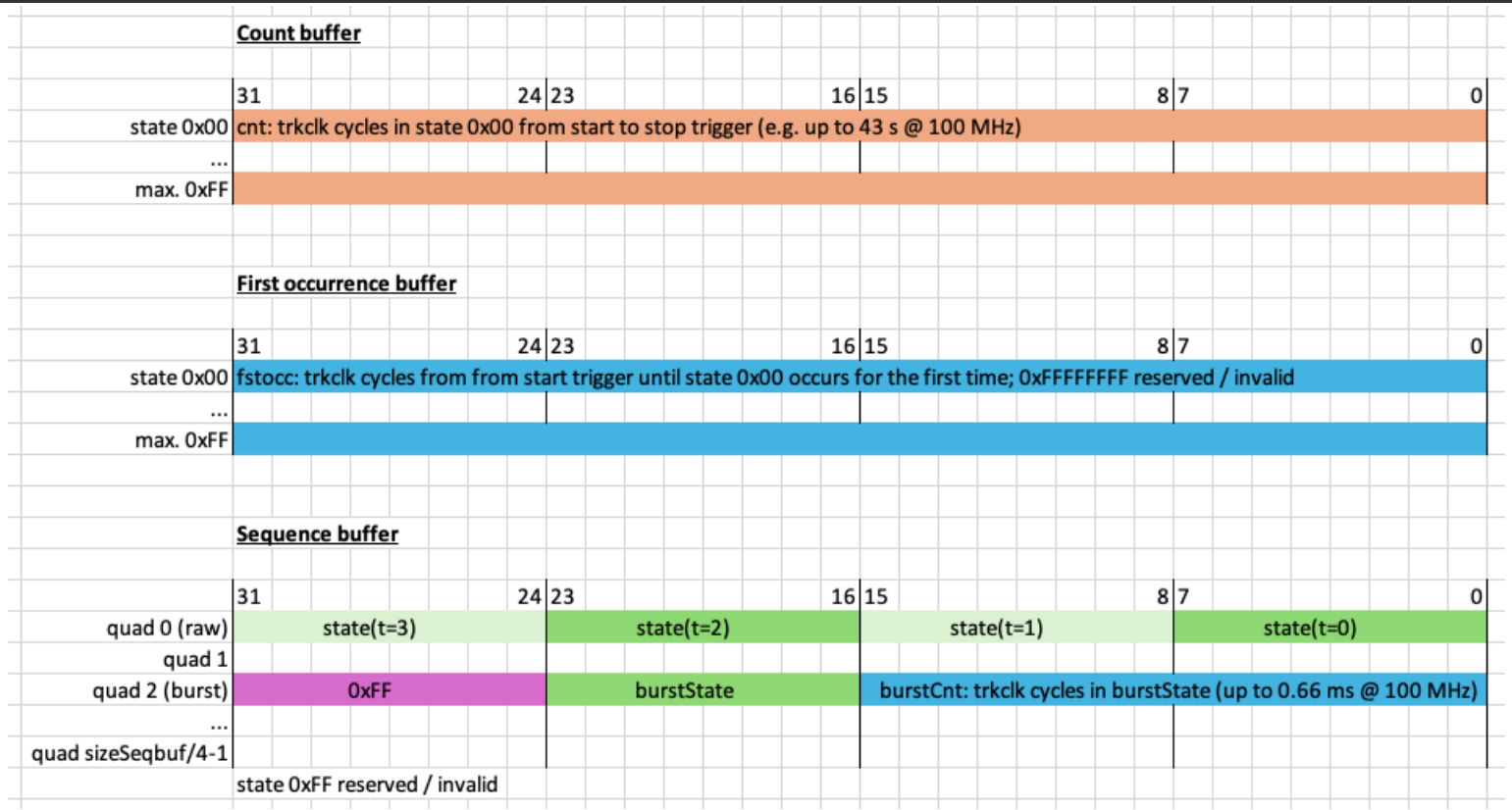
Example #2: CPU-FPGA Transaction

Context | Capture | Fsmtrack_Easy_v1_0 | Data format | Readout

- Mux'ed capture port
- Selectable trigger
- Count buffer
- First occurrence buffer
- Variable size sequence buffer
- Tracking clock (trkclk)
- Master clock for readout (mclk)
- Adaptive readout width



Example #2: CPU-FPGA Transaction



Example #2: CPU-FPGA Transaction

- Three commands for source/trigger selection, acquisition start and status polling

```
mfsmtrack0.select(tixVSource={hostifOp},
  staTixVTrigger={void,hostifRxAXIS_tvalid,ackInvTkclksrcSetTkst},
  stoTixVTrigger={void,hostifRxAXIS_tvalid,ackInvTkclksrcSetTkst})

mfsmtrack0.set(rng={false,true},TCapt=[uint32])

(tixVState{idle,arm,acq,done},coverage[blob32]) = mfsmtrack0.getInfo()
```

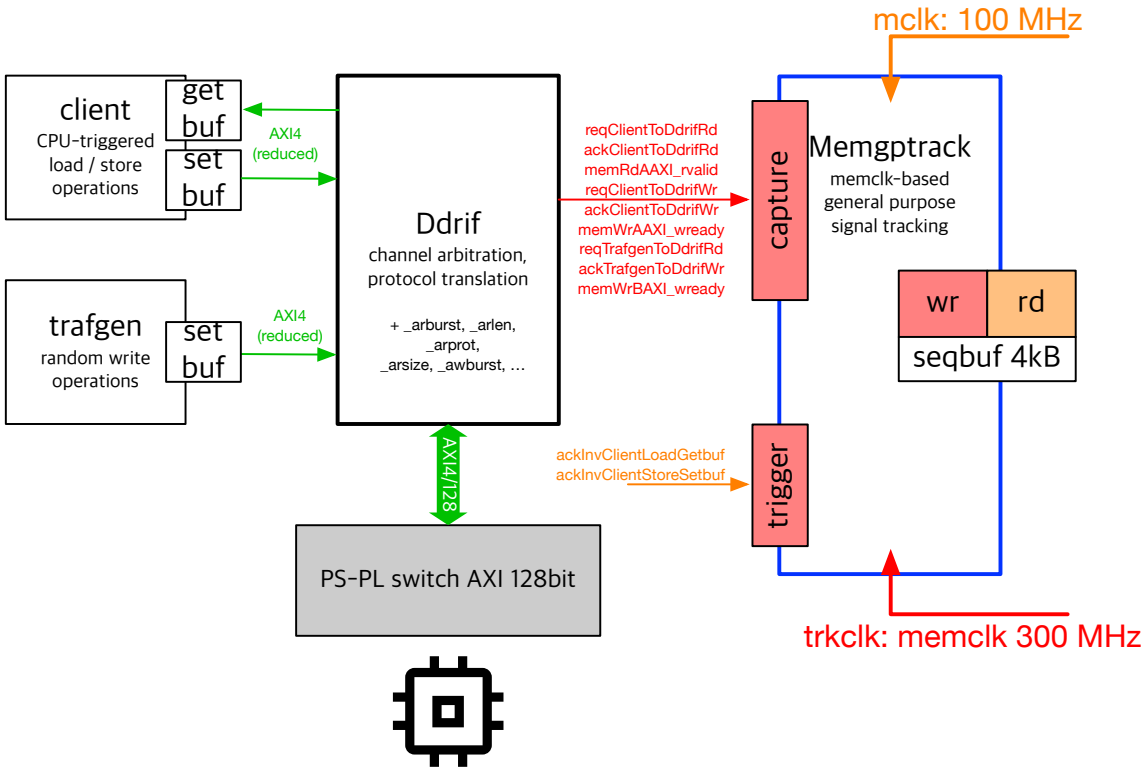
- Three buffer transfers for resulting count, first occurrence and sequence buffers

```
(file) = cntbufMfsmtrack0ToHostif.read(reqlen[uint32])
(file) = fstoccbufMfsmtrack0ToHostif.read(reqlen[uint32])
(file) = seqbufMfsmtrack0ToHostif.read(reqlen[uint32])
```

Example #3: Time-multiplexed DDR Memory Access

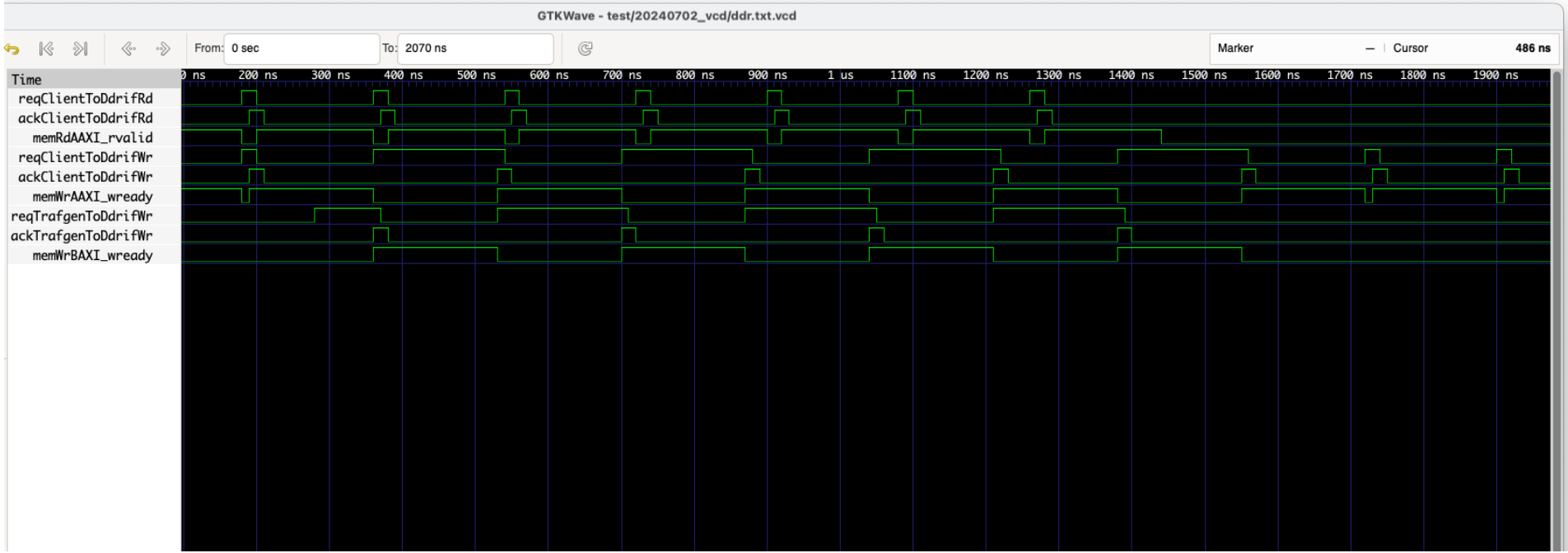
Context | Capture | CDC & Performance

- Ddrmux module template to share DDR memory access across multiple read/write ports
- AXI4 burst mode transfers (here: 16 beats @ 16 byte)
- Deterministic transfers from / to client buffers perturbed by random trafgen traffic
- Capture request / acknowledge handshakes



Example #3: Time-multiplexed DDR Memory Access

Context | Capture | CDC & Performance



Example #3: Time-multiplexed DDR Memory Access

Context | Capture | CDC & Performance

- Trigger signals are CDC'ed before the input mux, otherwise (true) dual port RAM's achieve the CDC between tracking and read-out

- All inputs are registered, timing closure achieved at up to 434 MHz (on Xilinx MPSoC)

- Most critical (not in this example): count buffer due to read _and_ write on trkclk

WhizniumDBE Module Templates

- **WhizniumDBE** module templates used across the presented examples (cf. https://github.com/mpsitech/wted-WhizniumDBE-Template-Evaluation-Device)

Module template	Features
Axislave_v1_0 *)	AXIlite to AXI stream, var. width
Crcspec_v3_0	16-/32-bit poly CRC with var. width AXI stream
Ddrmux_Easy_v1_0	Multiplexed DDR memory access with burst transfers across subset of AXI4, transfer statistics accumulation with CPU read-out CROSS-VENDOR PHY-AGNOSTIC
Debounce_v1_0	Digital pushbutton debouncer
Dpram_efnx_v1_0	Dual-port RAM generator for Efinix with asymmetric port widths
Dpebram_lttc_v1_2 *)	Dual-port EBRAM wrapper for Lattice
Dpbram_xlnx_v8_4	Dual-port BRAM wrapper for Xilinx
Fsmtrack_Easy_v1_0	FSM tracker with trigger and capture manifolds, compression and statistics accumulation, CPU read-out CROSS-VENDOR

WhizniumDBE Module Templates

Module template	Features (cont'd)
Gptrack_Easy_v1_0	15-bit general purpose tracker with trigger manifold and compression, CPU read-out CROSS-VENDOR
Hostif_Easy_v1_0	Connection to CPU host with byte protocol decoding PHY-AGNOSTIC
Ident_Easy_v1_0	Block / IP identification including version and Git hash with CPU read-out
Rgbled_v1_0	Driver for RGB888 in to PWM LED out
Timeout_v1_0 *)	Watchdog timer
Tkclksrc_Easy_v1_0	10 kHz clock source for time-stamping and timeout tasks with CPU read-out
Top_v1_0	Clock and reset generation, top-level routing CROSS-VENDOR
Uartrx/tx_v2_0 *)	UART RX/TX to AXI stream

- Plus, an awesome 3rd party core: neoTRNG for random number generation (BSD3 license) https://github.com/stnolting/neoTRNG

*) auto - instantiated

The Whiznium Universe & Developer Lifestyle

Concept | Key features | Workflow

WhizniumDBE (“Device Builder’s Edition”) is

- NOT high-level synthesis (HLS), not a compiler
- NOT your typical generator framework
- NOT a visual design tool

The Whiznium Universe & Developer Lifestyle

Concept | Key features | Workflow

WhizniumDBE (“Device Builder’s Edition”) is

- NOT
- NOT
- NOT

WhizniumDBE is

… a user-extensible framework written in C++, that for a given RTL design

… interprets its structure and features, specified in text-based model files

… composes and maintains a fine-grained RTL model *) in a SQL database

… then is able to write VHDL and C++ code based on it

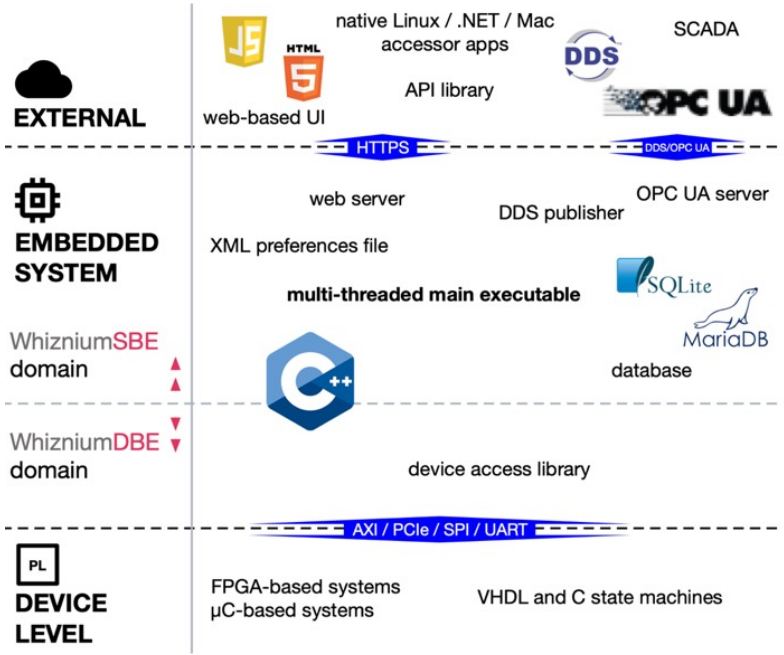
… taking into account manual code contributions of previous design versions

*) from hierarchical structure down to FSM’s incl. state transitions, CDC fabric, generics/ports/signals/variables

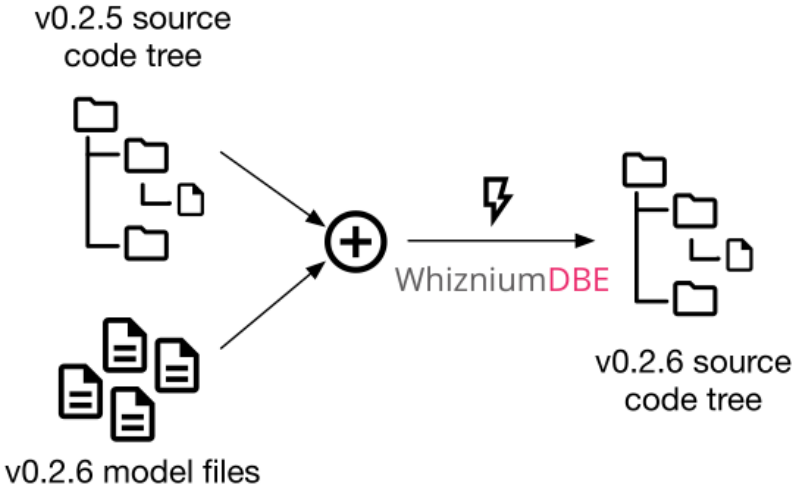
The Whiznium Universe & Developer Lifestyle

Concept | Key features | Workflow

- Clean, ergonomic, source code structure (“tasteful naming conventions”, etc.)
- Parametrized templates for standard components (e.g. SPI, GPIO, CRC, Git-Ident; 35 and counting)
- Custom templates can interact with the model / module surroundings while a design is composed (not just simple files with placeholders)
- The applicable vendor(‘s primitives) can be an auto-derived template parameter
- Scope extends beyond the FPGA world with **WhizniumSBE** (Service Builder’s Edition)



- Regular RTL workflow (including use of vendor IDE's) augmented by "source code tree iteration"

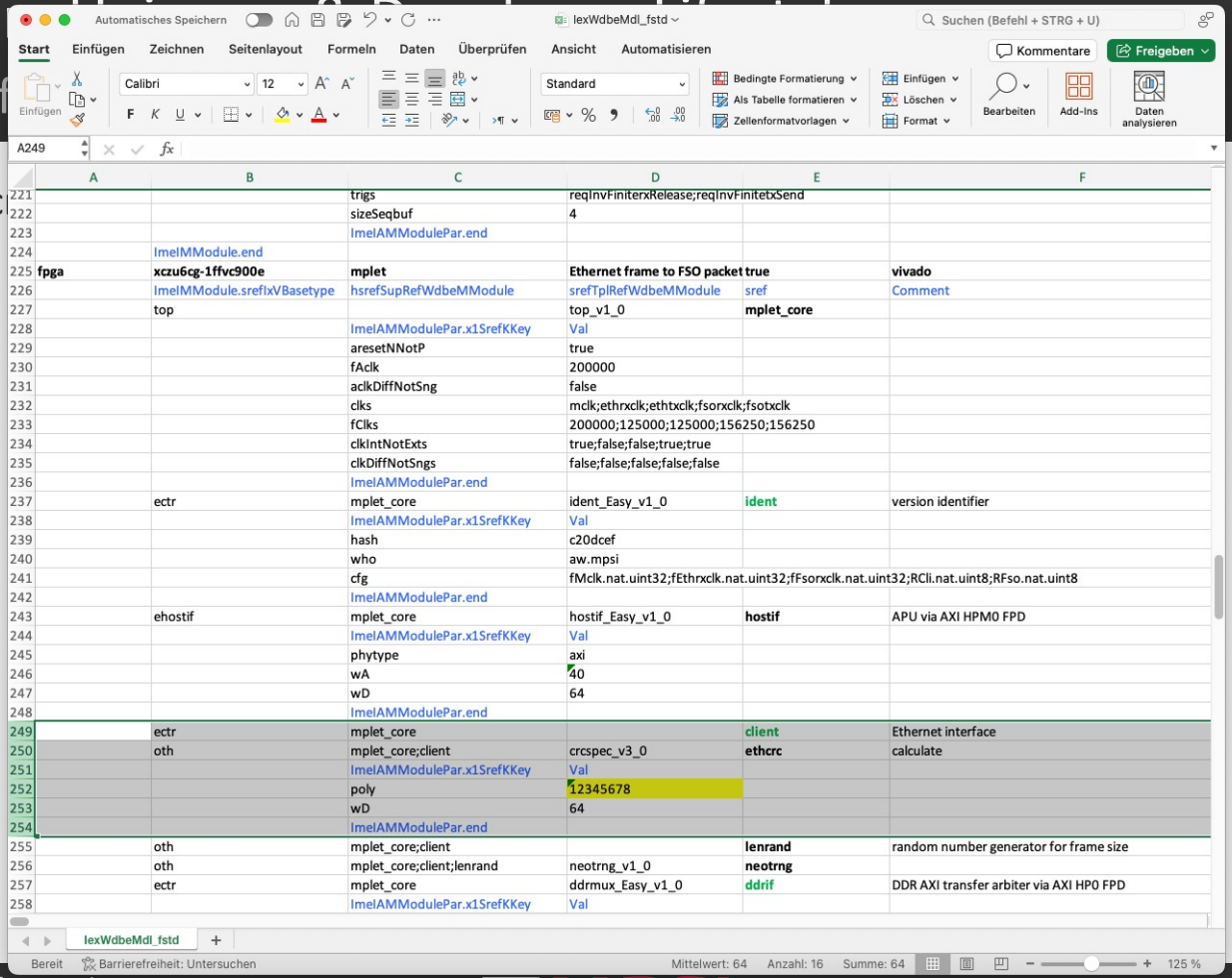


- Modular description (IexWdbeMdl.txt), CRC template, polynomial changed

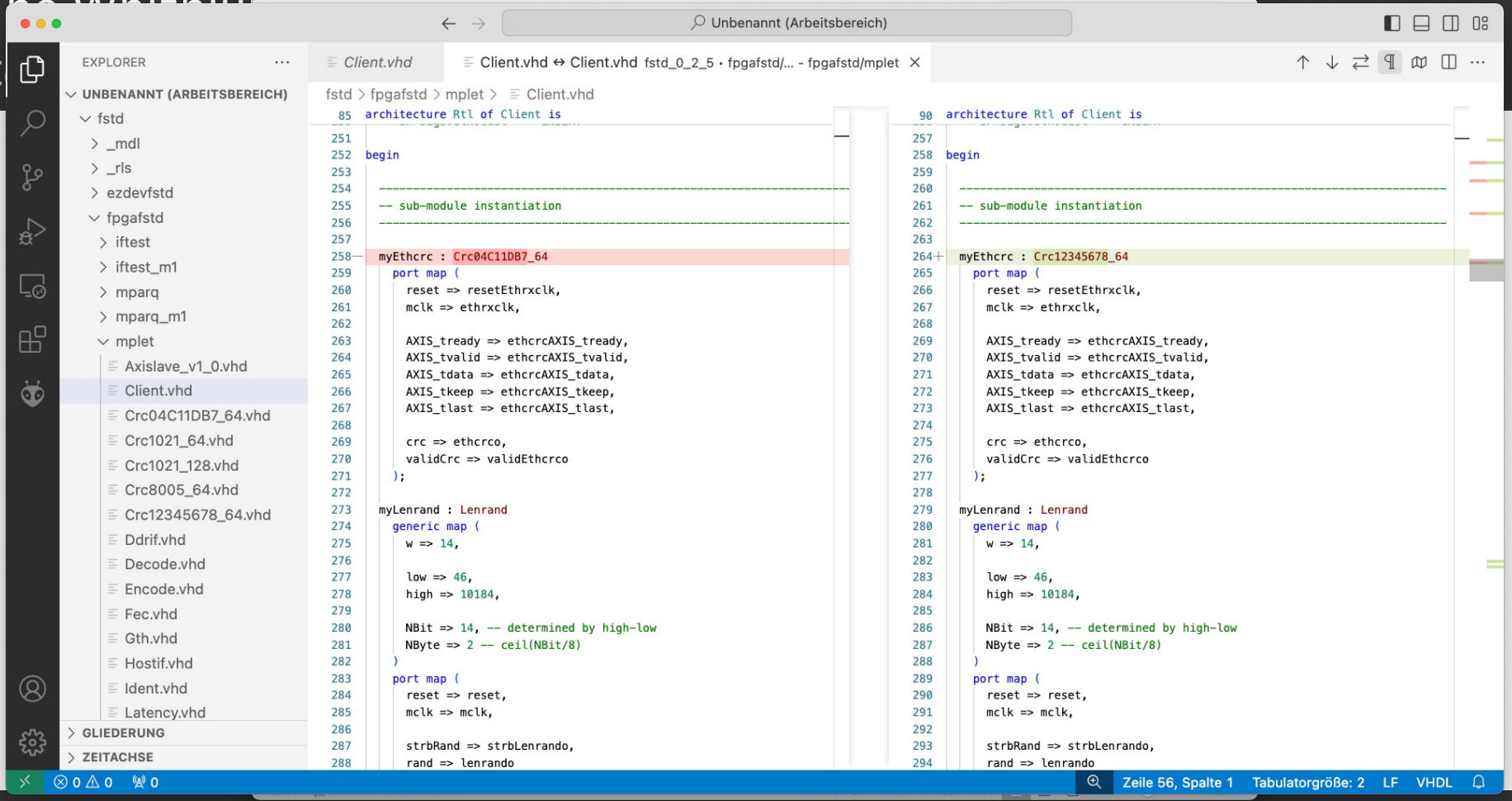
The Whizniu

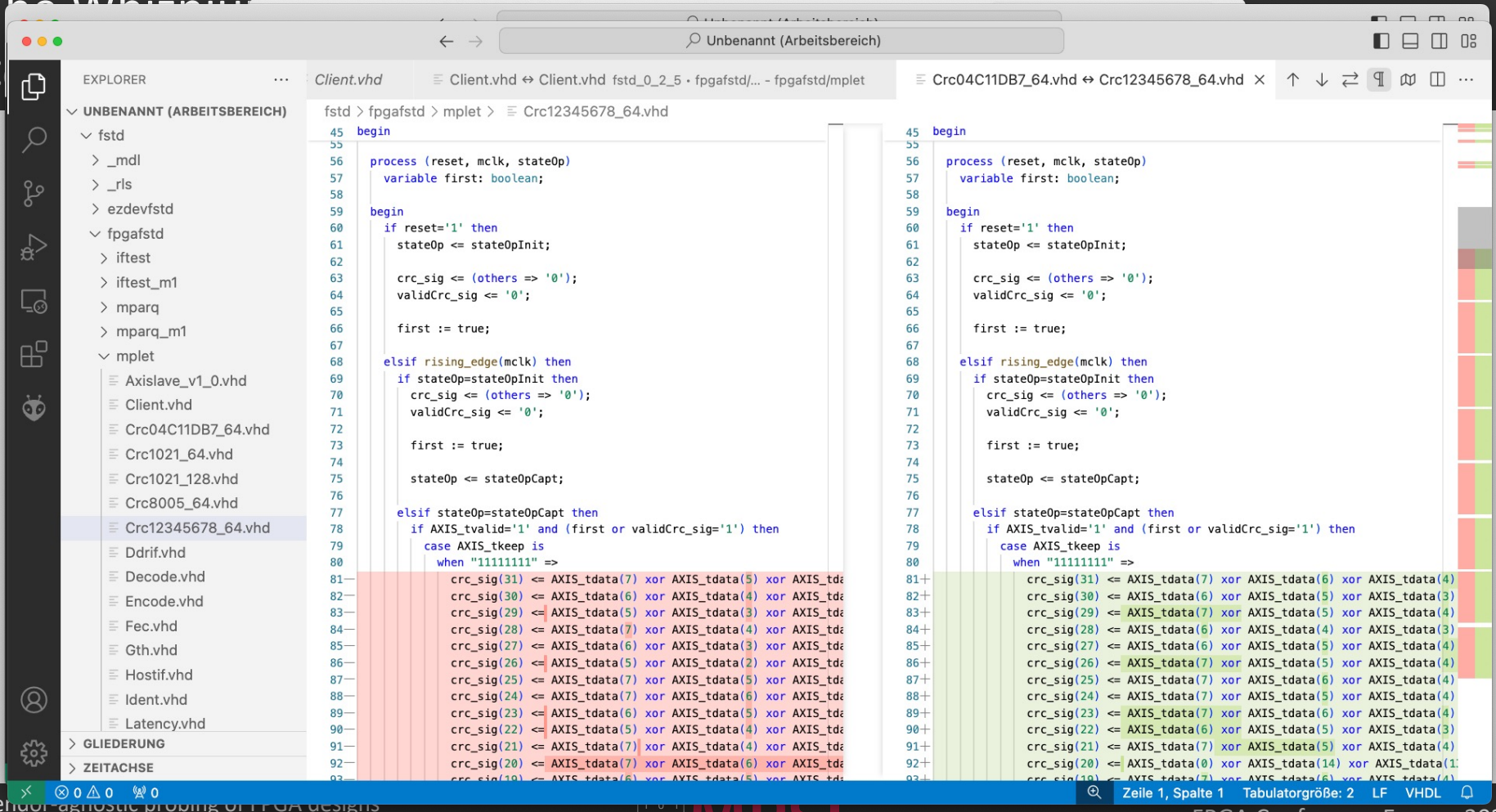
Concept | Key f

- Modular desc



	A	B	C	D	E	F
221			trigs	reqInvFiniterxRelease;reqInvFinitetxSend		
222			sizeSeqbuf	4		
223			ImelAMModulePar.end			
224		ImelMModule.end				
225	**fpga**	**xczu6cg-1ffvc900e**	**mplet**	**Ethernet frame to FSO packet**	**true**	**vivado**
226		ImelMModule.srefIxVBasetype	hsrefSupRefWdbeMModule	srefTplRefWdbeMModule	sref	Comment
227		top		top_v1_0	**mplet_core**	
228			ImelAMModulePar.x1SrefKKey	Val		
229			aresetNNotP	true		
230			fAclk	200000		
231			aclkDiffNotSng	false		
232			clks	mclk;ethrxclk;ethtxclk;fsorxclk;fsotxclk		
233			fClks	200000;125000;125000;156250;156250		
234			clkIntNotExts	true;false;false;true;true		
235			clkDiffNotSngs	false;false;false;false;false		
236			ImelAMModulePar.end			
237		ectr	mplet_core	ident_Easy_v1_0	ident	version identifier
238			ImelAMModulePar.x1SrefKKey	Val		
239			hash	c20dcef		
240			who	aw.mpsi		
241			cfg	fMclk.nat.uint32;fEthrxclk.nat.uint32;fFsorxclk.nat.uint32;RCli.nat.uint8;RFso.nat.uint8		
242			ImelAMModulePar.end			
243		ehostif	mplet_core	hostif_Easy_v1_0	**hostif**	APU via AXI HPM0 FPD
244			ImelAMModulePar.x1SrefKKey	Val		
245			phytype	axi		
246			wA	40		
247			wD	64		
248			ImelAMModulePar.end			
249		ectr	mplet_core		client	Ethernet interface
250		oth	mplet_core;client	crcspec_v3_0	**ethcrc**	calculate
251			ImelAMModulePar.x1SrefKKey	Val		
252			poly	12345678		
253			wD	64		
254			ImelAMModulePar.end			
255		oth	mplet_core;client		**lenrand**	random number generator for frame size
256		oth	mplet_core;client;lenrand	neotrng_v1_0	**neotrng**	
257		ectr	mplet_core	ddrmux_Easy_v1_0	ddrif	DDR AXI transfer arbiter via AXI HP0 FPD
258			ImelAMModulePar.x1SrefKKey	Val		


Unbenannt (Arbeitsbereich)
EXPLORER
UNBENANNT (ARBEITSBEREICH)
fstd
_mdl
_rls
ezdevfstd
fpgafstd
iftest
iftest_m1
mparq
mparq_m1
mplet
Axislave_v1_0.vhd
Client.vhd
Crc04C11DB7_64.vhd
Crc1021_64.vhd
Crc1021_128.vhd
Crc8005_64.vhd
Crc12345678_64.vhd
Ddrif.vhd
Decode.vhd
Encode.vhd
Fec.vhd
Gth.vhd
Hostif.vhd
Ident.vhd
Latency.vhd
GLIEDERUNG
ZEITACHSE
Client.vhd
Client.vhd ↔ Client.vhd fstd_0_2_5 • fpgafstd/... - fpgafstd/mplet
fstd > fpgafstd > mplet > Client.vhd
85 architecture Rtl of Client is
251
252 begin
253
254 --
255 -- sub-module instantiation
256 --
257
258 myEthcrc : Crc04C11DB7_64
259 port map (
260 reset => resetEthrxclk,
261 mclk => ethrxclk,
262
263 AXIS_tready => ethcrcAXIS_tready,
264 AXIS_tvalid => ethcrcAXIS_tvalid,
265 AXIS_tdata => ethcrcAXIS_tdata,
266 AXIS_tkeep => ethcrcAXIS_tkeep,
267 AXIS_tlast => ethcrcAXIS_tlast,
268
269 crc => ethcrco,
270 validCrc => validEthcrco
271);
272
273 myLenrand : Lenrand
274 generic map (
275 w => 14,
276
277 low => 46,
278 high => 10184,
279
280 NBit => 14, -- determined by high-low
281 NByte => 2 -- ceil(NBit/8)
282)
283 port map (
284 reset => reset,
285 mclk => mclk,
286
287 strbRand => strbLenrando,
288 rand => lenrando
90 architecture Rtl of Client is
257
258 begin
259
260 --
261 -- sub-module instantiation
262 --
263
264 myEthcrc : Crc12345678_64
265 port map (
266 reset => resetEthrxclk,
267 mclk => ethrxclk,
268
269 AXIS_tready => ethcrcAXIS_tready,
270 AXIS_tvalid => ethcrcAXIS_tvalid,
271 AXIS_tdata => ethcrcAXIS_tdata,
272 AXIS_tkeep => ethcrcAXIS_tkeep,
273 AXIS_tlast => ethcrcAXIS_tlast,
274
275 crc => ethcrco,
276 validCrc => validEthcrco
277);
278
279 myLenrand : Lenrand
280 generic map (
281 w => 14,
282
283 low => 46,
284 high => 10184,
285
286 NBit => 14, -- determined by high-low
287 NByte => 2 -- ceil(NBit/8)
288)
289 port map (
290 reset => reset,
291 mclk => mclk,
292
293 strbRand => strbLenrando,
294 rand => lenrando
0 0 0
Zeile 56, Spalte 1
Tabulatorgröße: 2
LF
VHDL



Vendor-agnostic probing of FPGA designs
For efficient real-time debugging

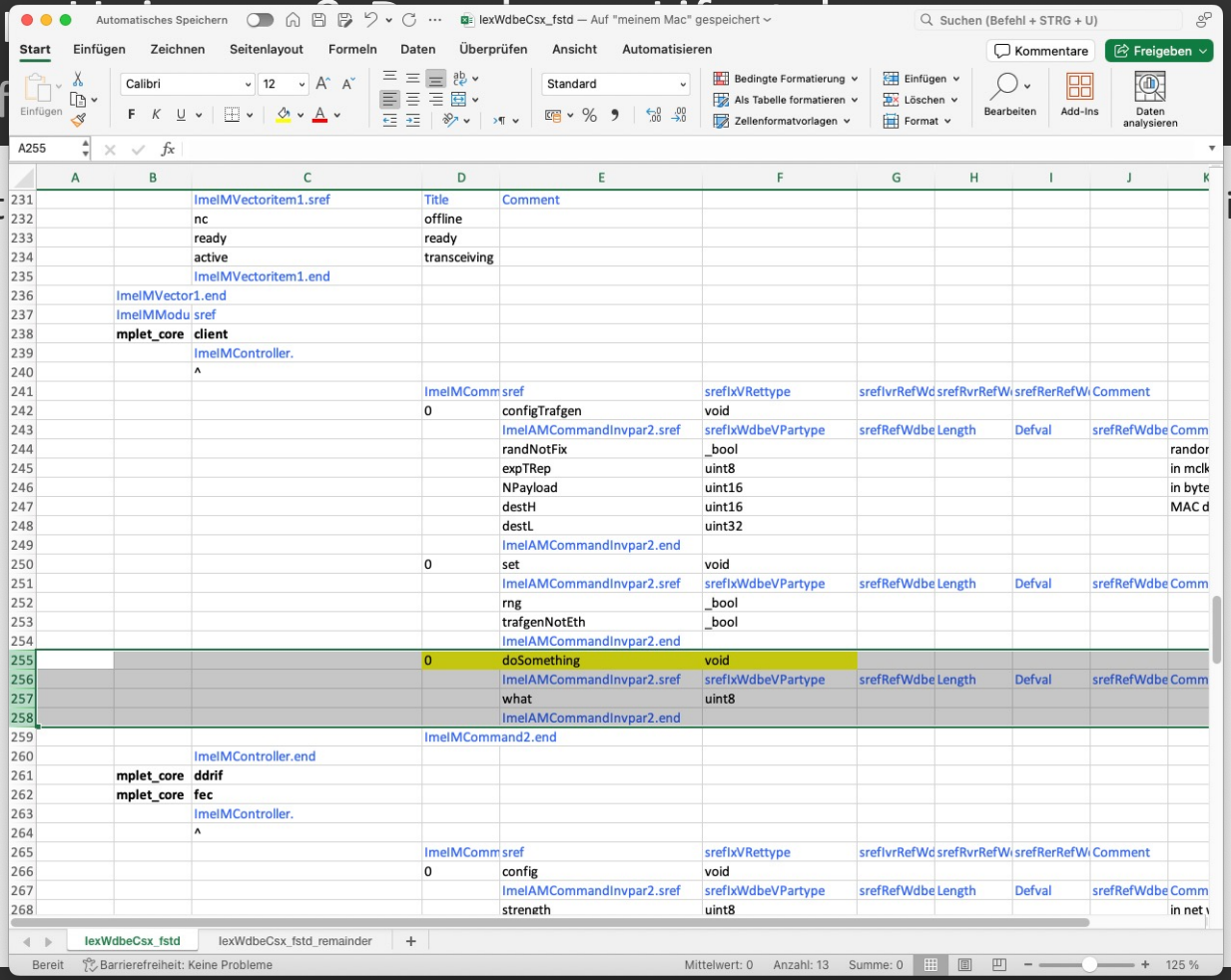
FPGA Conference Europe 2024

- Command set and buffer transfers (IexWdbeCsx.txt), command added to module “Client”

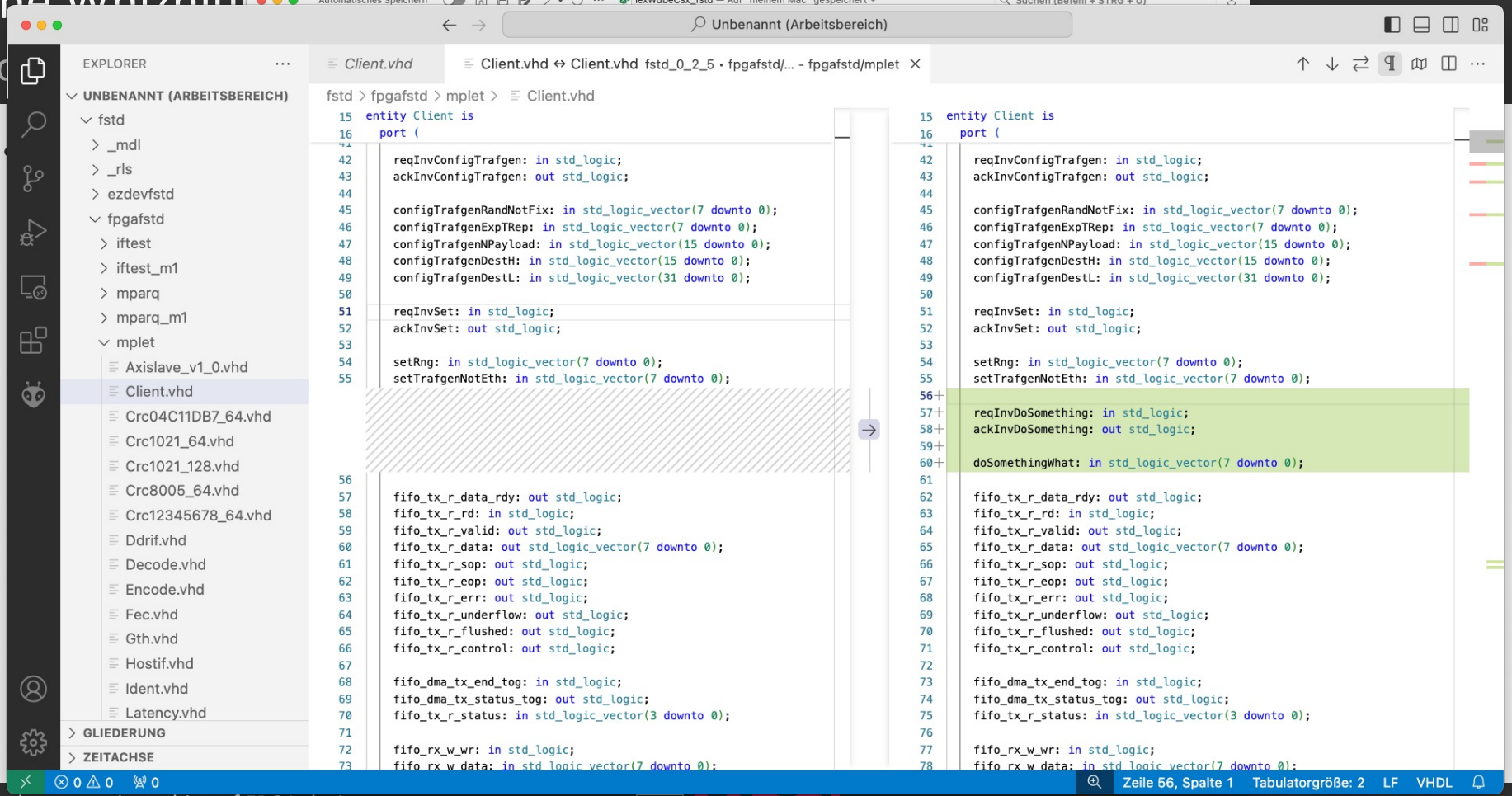
The Whizniu

Concept | Key f

- Command set ient”



	A	B	C	D	E	F	G	H	I	J	K
231			ImelMVectoritem1.sref	Title	Comment						
232			nc	offline							
233			ready	ready							
234			active	transceiving							
235			ImelMVectoritem1.end								
236		ImelMVector1.end									
237		ImelMModu	sref								
238		**mplet_core**	**client**								
239			ImelMController.								
240			^								
241				ImelMComm	sref	sreflxVRettype	sreflvrRefWo	srefRvrRefW	srefRerRefW	Comment	
242				0	configTrafgen	void					
243					ImelAMCommandInvpar2.sref	sreflxWdbeVPartype	srefRefWdbe	Length	Defval	srefRefWdbe	Comm
244					randNotFix	_bool					randor
245					expTRep	uint8					in mclk
246					NPayload	uint16					in byte
247					destH	uint16					MAC d
248					destL	uint32					
249					ImelAMCommandInvpar2.end						
250				0	set	void					
251					ImelAMCommandInvpar2.sref	sreflxWdbeVPartype	srefRefWdbe	Length	Defval	srefRefWdbe	Comm
252					rng	_bool					
253					trafgenNotEth	_bool					
254					ImelAMCommandInvpar2.end						
255				0	doSomething	void					
256					ImelAMCommandInvpar2.sref	sreflxWdbeVPartype	srefRefWdbe	Length	Defval	srefRefWdbe	Comm
257					what	uint8					
258					ImelAMCommandInvpar2.end						
259				ImelMCommand2.end							
260			ImelMController.end								
261		**mplet_core**	**ddrif**								
262		**mplet_core**	**fec**								
263			ImelMController.								
264			^								
265				ImelMComm	sref	sreflxVRettype	sreflvrRefWo	srefRvrRefW	srefRerRefW	Comment	
266				0	config	void					
267					ImelAMCommandInvpar2.sref	sreflxWdbeVPartype	srefRefWdbe	Length	Defval	srefRefWdbe	Comm
268					strength	uint8					in net

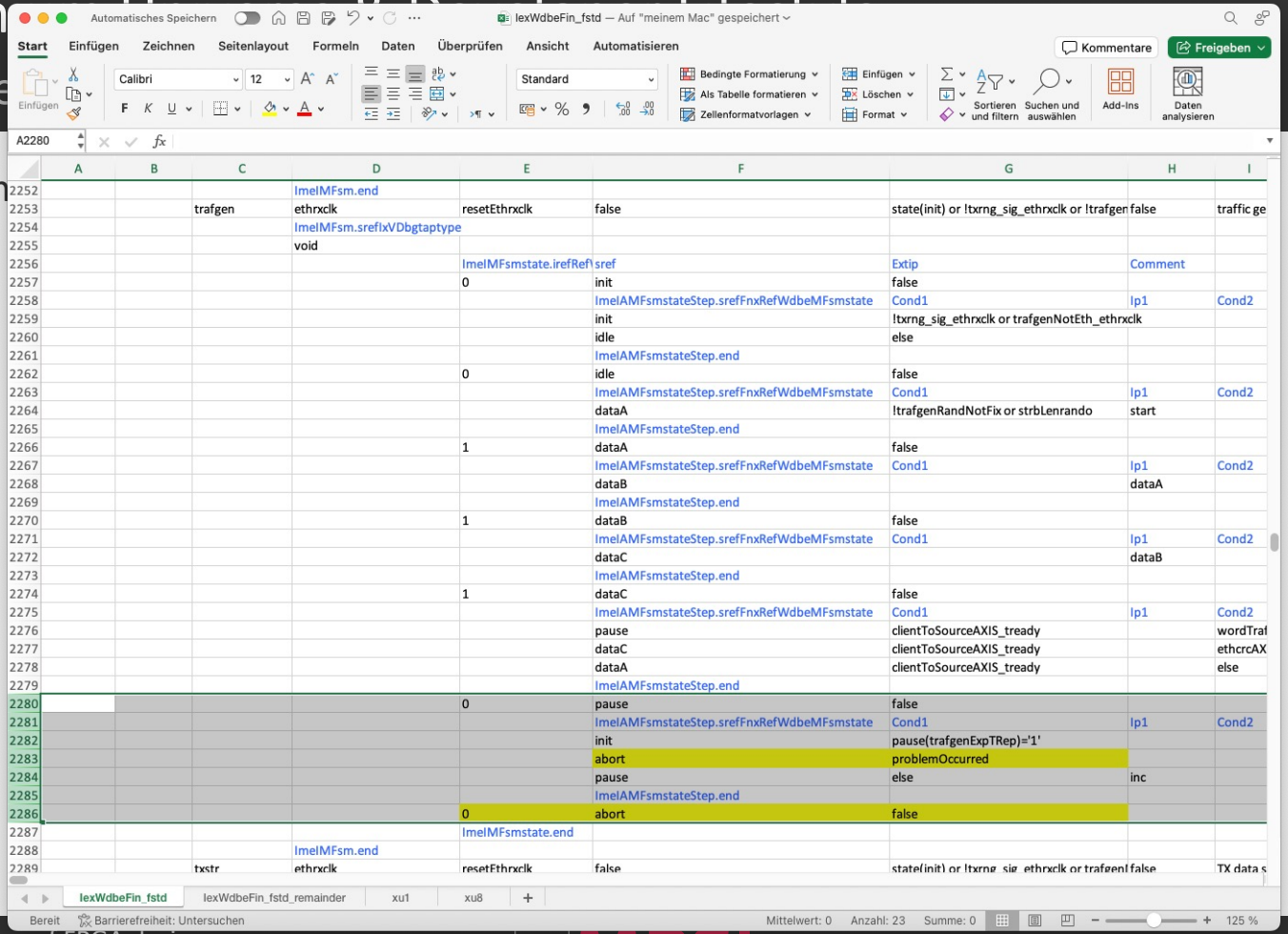

Automatisches Speichern
lexWdbeCsx_fstd — Auf "meinem Mac" gespeichert
Suchen (Befehl + STRG + U)
Unbenannt (Arbeitsbereich)
EXPLORER
UNBENANNT (ARBEITSBEREICH)
fstd
_mdl
_rls
ezdevfstd
fpgafstd
iftest
iftest_m1
mparq
mparq_m1
mplet
Axislave_v1_0.vhd
Client.vhd
Crc04C11DB7_64.vhd
Crc1021_64.vhd
Crc1021_128.vhd
Crc8005_64.vhd
Crc12345678_64.vhd
Ddrif.vhd
Decode.vhd
Encode.vhd
Fec.vhd
Gth.vhd
Hostif.vhd
Ident.vhd
Latency.vhd
GLIEDERUNG
ZEITACHSE
Client.vhd
Client.vhd ↔ Client.vhd fstd_0_2_5 • fpgafstd/... - fpgafstd/mplet
fstd > fpgafstd > mplet > Client.vhd
entity Client is
port (
reqInvConfigTrafgen: in std_logic;
ackInvConfigTrafgen: out std_logic;
configTrafgenRandNotFix: in std_logic_vector(7 downto 0);
configTrafgenExpTRep: in std_logic_vector(7 downto 0);
configTrafgenNPayload: in std_logic_vector(15 downto 0);
configTrafgenDestH: in std_logic_vector(15 downto 0);
configTrafgenDestL: in std_logic_vector(31 downto 0);
reqInvSet: in std_logic;
ackInvSet: out std_logic;
setRng: in std_logic_vector(7 downto 0);
setTrafgenNotEth: in std_logic_vector(7 downto 0);
reqInvDoSomething: in std_logic;
ackInvDoSomething: out std_logic;
doSomethingWhat: in std_logic_vector(7 downto 0);
fifo_tx_r_data_rdy: out std_logic;
fifo_tx_r_rd: in std_logic;
fifo_tx_r_valid: out std_logic;
fifo_tx_r_data: out std_logic_vector(7 downto 0);
fifo_tx_r_sop: out std_logic;
fifo_tx_r_eop: out std_logic;
fifo_tx_r_err: out std_logic;
fifo_tx_r_underflow: out std_logic;
fifo_tx_r_flushed: out std_logic;
fifo_tx_r_control: out std_logic;
fifo_dma_tx_end_tog: in std_logic;
fifo_dma_tx_status_tog: out std_logic;
fifo_tx_r_status: in std_logic_vector(3 downto 0);
fifo_rx_w_wr: in std_logic;
Zeile 56, Spalte 1
Tabulatorgröße: 2
LF
VHDL

- Custom fine structure (IexWdbeFin.txt), FSM state added

The Whizn

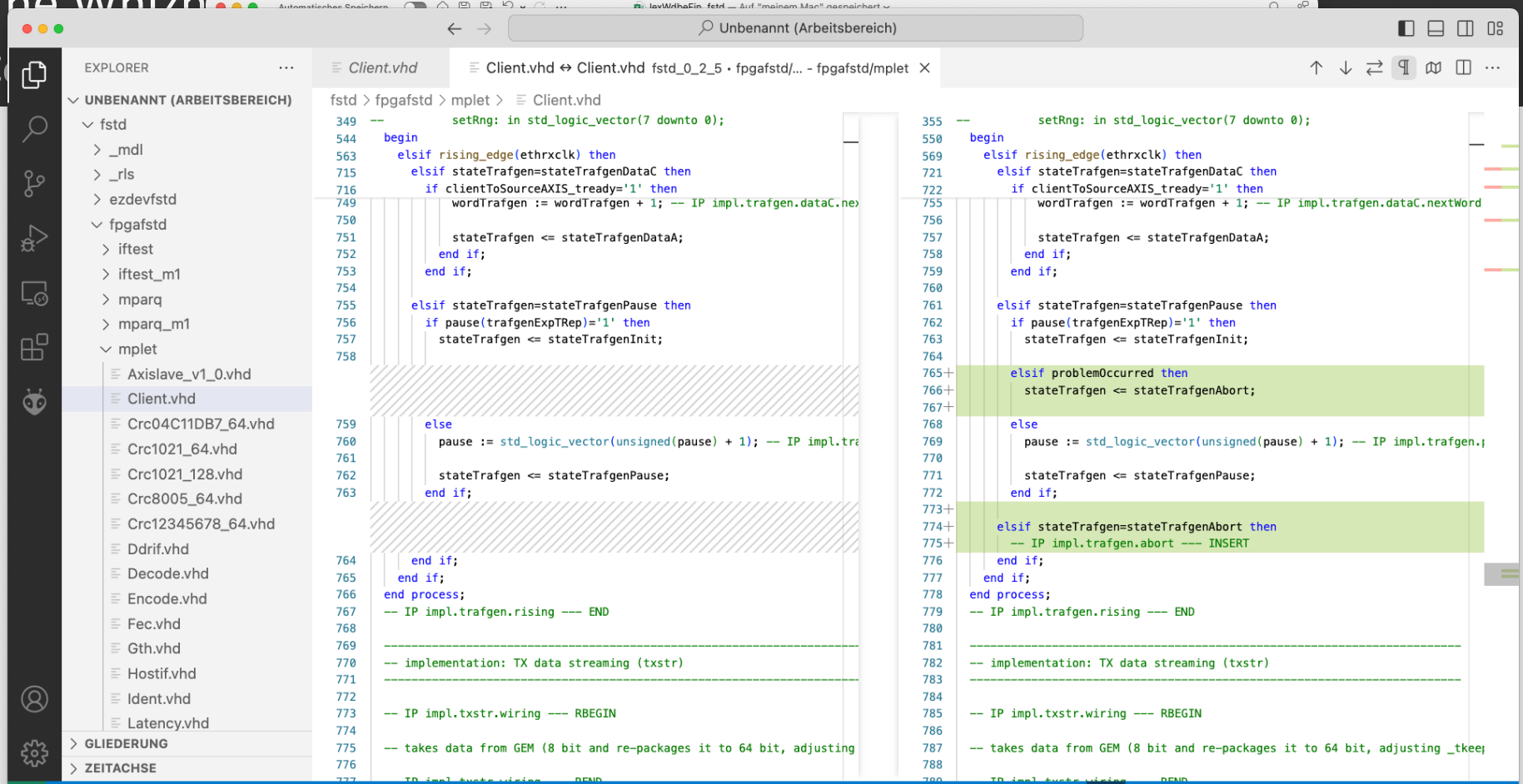
Concept | Ke

- Custom fin



lexWdbeFin_fstd — Auf "meinem Mac" gespeichert

	A	B	C	D	E	F	G	H	I
2252				ImeIMFsm.end					
2253			trafgen	ethrxclk	resetEthrxclk	false	state(init) or !txrng_sig_ethrxclk or !trafgen	false	traffic ge
2254				ImeIMFsm.sreflxVDbgtaptype					
2255				void					
2256					ImeIMFsmstate.irefRef\sref		Extip	Comment	
2257					0	init	false		
2258						ImeIAMFsmstateStep.srefFnxRefWdbeMFsmstate	Cond1	Ip1	Cond2
2259						init	!txrng_sig_ethrxclk or trafgenNotEth_ethrxclk		
2260						idle	else		
2261						ImeIAMFsmstateStep.end			
2262					0	idle	false		
2263						ImeIAMFsmstateStep.srefFnxRefWdbeMFsmstate	Cond1	Ip1	Cond2
2264						dataA	!trafgenRandNotFix or strbLenrando	start	
2265						ImeIAMFsmstateStep.end			
2266					1	dataA	false		
2267						ImeIAMFsmstateStep.srefFnxRefWdbeMFsmstate	Cond1	Ip1	Cond2
2268						dataB		dataA	
2269						ImeIAMFsmstateStep.end			
2270					1	dataB	false		
2271						ImeIAMFsmstateStep.srefFnxRefWdbeMFsmstate	Cond1	Ip1	Cond2
2272						dataC		dataB	
2273						ImeIAMFsmstateStep.end			
2274					1	dataC	false		
2275						ImeIAMFsmstateStep.srefFnxRefWdbeMFsmstate	Cond1	Ip1	Cond2
2276						pause	clientToSourceAXIS_tready		wordTraf
2277						dataC	clientToSourceAXIS_tready		ethcrcAX
2278						dataA	clientToSourceAXIS_tready		else
2279						ImeIAMFsmstateStep.end			
2280					0	pause	false		
2281						ImeIAMFsmstateStep.srefFnxRefWdbeMFsmstate	Cond1	Ip1	Cond2
2282						init	pause(trafgenExpTRep)='1'		
2283						abort	problemOccurred		
2284						pause	else	inc	
2285						ImeIAMFsmstateStep.end			
2286					0	abort	false		
2287					ImeIMFsmstate.end				
2288				ImeIMFsm.end					
2289			txstr	ethrxclk	resetEthrxclk	false	state(init) or !txrng_sig_ethrxclk or trafgen	false	TX data s

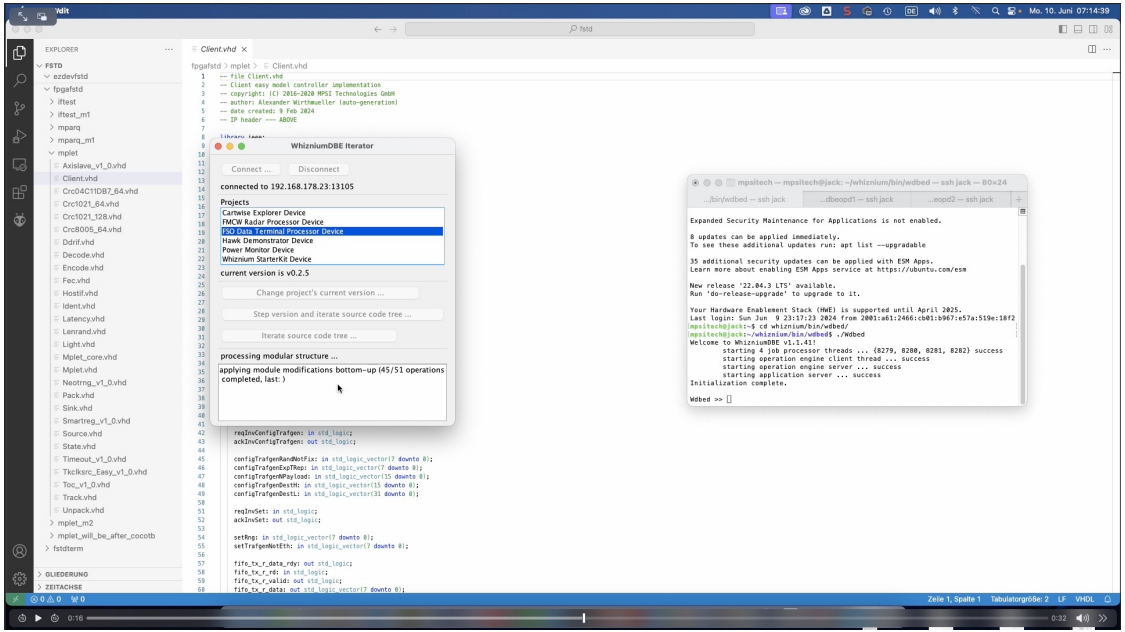

Unbenannt (Arbeitsbereich)
EXPLORER
UNBENANNT (ARBEITSBEREICH)
fstd
_mdl
_rls
ezdevfstd
fpgafstd
iftest
iftest_m1
mparq
mparq_m1
mplet
Axislave_v1_0.vhd
Client.vhd
Crc04C11DB7_64.vhd
Crc1021_64.vhd
Crc1021_128.vhd
Crc8005_64.vhd
Crc12345678_64.vhd
Ddrif.vhd
Decode.vhd
Encode.vhd
Fec.vhd
Gth.vhd
Hostif.vhd
Ident.vhd
Latency.vhd
GLIEDERUNG
ZEITACHSE
Client.vhd
Client.vhd ↔ Client.vhd fstd_0_2_5 • fpgafstd/... - fpgafstd/mplet
fstd > fpgafstd > mplet > Client.vhd
-- setRng: in std_logic_vector(7 downto 0);
begin
elsif rising_edge(ethrxclk) then
elsif stateTrafgen=stateTrafgenDataC then
if clientToSourceAXIS_tready='1' then
wordTrafgen := wordTrafgen + 1; -- IP impl.trafgen.dataC.nextWord
stateTrafgen <= stateTrafgenDataA;
end if;
end if;
elsif stateTrafgen=stateTrafgenPause then
if pause(trafgenExpTRep)='1' then
stateTrafgen <= stateTrafgenInit;
elsif problemOccurred then
stateTrafgen <= stateTrafgenAbort;
else
pause := std_logic_vector(unsigned(pause) + 1); -- IP impl.trafgen.
stateTrafgen <= stateTrafgenPause;
end if;
elsif stateTrafgen=stateTrafgenAbort then
-- IP impl.trafgen.abort --- INSERT
end if;
end if;
end process;
-- IP impl.trafgen.rising --- END
-- implementation: TX data streaming (txstr)
-- IP impl.txstr.wiring --- RBEGIN
-- takes data from GEM (8 bit and re-packages it to 64 bit, adjusting _tkeep
Zeile 56, Spalte 1
Tabulatorgröße: 2
LF
VHDL

- **WhizniumDBE** Iterator (cross-platform Java tool accessing the **WhizniumDBE** daemon via it JSON API)

The Whiznium Universe & Developer Lifestyle

Concept | Key features | Workflow

- **WhizniumDBE** Iterator (cross-platform Java tool accessing the **WhizniumDBE** daemon via it JSON API)



Full Video: https://content.mpsitech.cloud/FPGAEurope2024/WhizniumDBE_step.mp4

Conclusion & Outlook

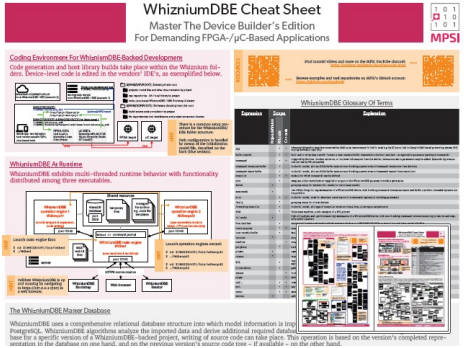
- PHY- and vendor-agnostic probing of FPGA designs using Open Source software is feasible
- … and long-overdue to replace proprietary solutions
- Mid-range (and larger) devices have sufficient capacity to implement this functionality

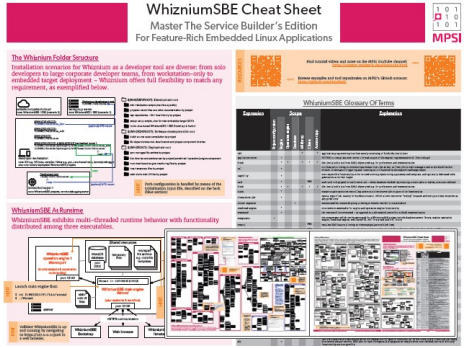
- TBD 1: integration of read-out and .vcd generation into the Open Source **WhizniumDBE** core library; also support merging results across clock domains
- TBD 2: addition of a **WhizniumSBE** capability to control the probing blocks from an (auto-generated) web UI

- Not on the roadmap: the ”O” part of “Virtual I/O”. While probing is universal, stimuli aren’t … and it is easy to design a project-specific, CPU-connected module using **WhizniumDBE**

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” webinar series on Whiznium is on YouTube (google it)
- For advanced users WhizniumSBE/DBE cheat sheets are available which serve as reference for writing model files





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” webinar series on Whiznium is on YouTube (google it)
- For advanced users WhizniumSBE/DBE cheat sheets are available which serve as reference for writing model files
- (New) home for cross-vendor cores
  https://mpsitech.github.io/Laser-Scanner-By-Platform
- Some more presentations on the topic
  https://www.mpsitech.com/documentation/presentations

Thank You!

Questions?

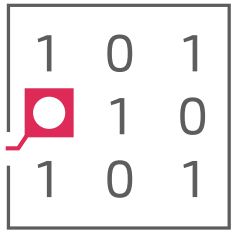
Also, feel free to connect.

- 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



Gptrack / Fsmtrack model entry

IexWdbeMdl v1.1.28					
ImeIMUnit.srefIxVB	srefSilRefWdbeMUnit	sref	Title	Easy	srefKToolch
fpga	**lifcl-40-9bg400c**	**cleb**	**Lattice CrossLink-NX Evaluatic**	**true**	**radiant**
	ImeIMModule.srefIxVBasetype	hsrefSupRefWdbeMModule	srefTplRefWdbeMModule	sref	Comment
	edbgctr	**wted_core**	**fsmtrack_Easy_v1_0**	**mfsmtrack0**	**track host interface FSM**
		ImeIAMModulePar.x1SrefKKey	Val		
		trkclk	mclk		
		ratioTrkclk	1		
		capts	hostif.op		
		trigs	hostifRxAXIS_tvalid.rising;ackInvTkclksrcSetTkst.rising		
		sizeSeqbuf	4		
		ImeIAMModulePar.end			
	edbgctr	**wted_core**	**fsmtrack_Easy_v1_0**	**mfsmtrack1**	**track other mclk-based FSM's**
		ImeIAMModulePar.x1SrefKKey	Val		
		trkclk	mclk		
		ratioTrkclk	1		
		capts	tkclksrc.op		
		trigs	hostifRxAXIS_tvalid.rising;ackInvTkclksrcSetTkst.rising		
		sizeSeqbuf	4		
		ImeIAMModulePar.end			
	edbgctr	**wted_core**	**gptrack_Easy_v1_0**	**mgptrack**	**track mclk-based signals**
		ImeIAMModulePar.x1SrefKKey	Val		
		trkclk	mclk		
		ratioTrkclk	1		
		capts	tkclk;rgb0_r;rgb0_g;rgb0_b;btn0;btn0_sig;tkclksrcGetTkstTkst[7..0]		
		trigs	btn0.rising;btn0.falling;ackInvTkclksrcSetTkst.rising		
		sizeSeqbuf	4		

Hostif model entry

ImeIMUnit.srefIxVE	srefSilRefWdbeMUnit	sref	Title	Easy	srefKToolch
fpga	**lifcl-40-9bg400c**	**cleb**	**Lattice CrossLink-NX Evaluati**	**true**	**radiant**
	ImeIMModule.srefIxVBasetype	hsrefSupRefWdbeMModule	srefTplRefWdbeMModule	sref	Comment
	ehostif	**wted_core**	**hostif_Easy_v1_0**	**hostif**	**connection to host via FT2232H**
		ImeIAMModulePar.x1SrefKKey	Val		
		phytype	uart		
		fSclk	115200		
		ImeIAMModulePar.end			

fpga	**ti180m484i3**	**tidk**	**Efinix Titanium Ti180 develo**	**true**	**efinity**
	ImeIMModule.srefIxVBasetype	hsrefSupRefWdbeMModule	srefTplRefWdbeMModule	sref	Comment
	ehostif	**wted_core**	**hostif_Easy_v1_0**	**hostif**	**connection to RISC-V soft-cores running Linux via AXIlite**
		ImeIAMModulePar.x1SrefKKey	Val		
		phytype	axi		
		wA	32		
		wD	32		
		ImeIAMModulePar.end			

fpga	**xczu1cg-1sbva484e**	**zudk**	**Avnet Zynq UltraScale+ MPSo**	**true**	**vivado**
	ImeIMModule.srefIxVBasetype	hsrefSupRefWdbeMModule	srefTplRefWdbeMModule	sref	Comment
	ehostif	**wted_core**	**hostif_Easy_v1_0**	**hostif**	**connection to Cortex-A53's running Linux via FPD AXIlite**
		ImeIAMModulePar.x1SrefKKey	Val		
		phytype	axi		
		wA	40		
		wD	64		
		ImeIAMModulePar.end			

Efinix vs. Xilinx Fsmtrack