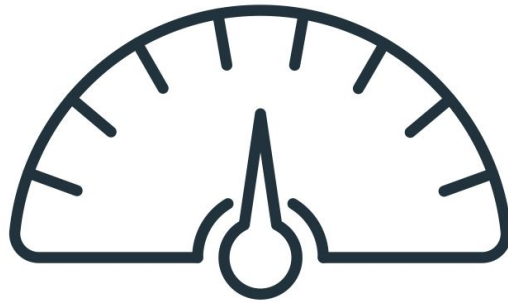


Blackwire™

WireGuard in HDL

100+ Gbit/s

on an FPGA NIC



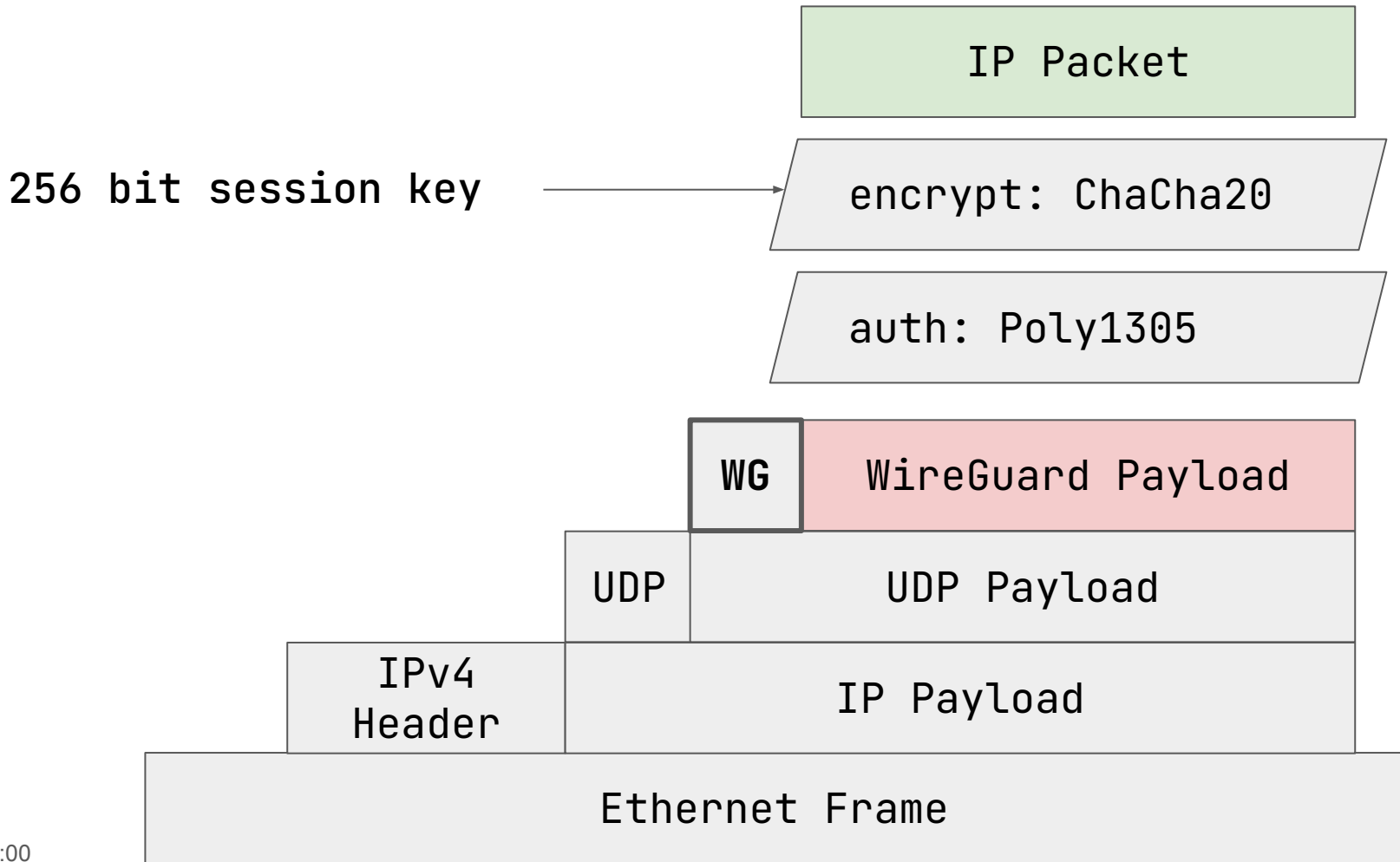
Leon Woestenberg

WIREFGUARD®

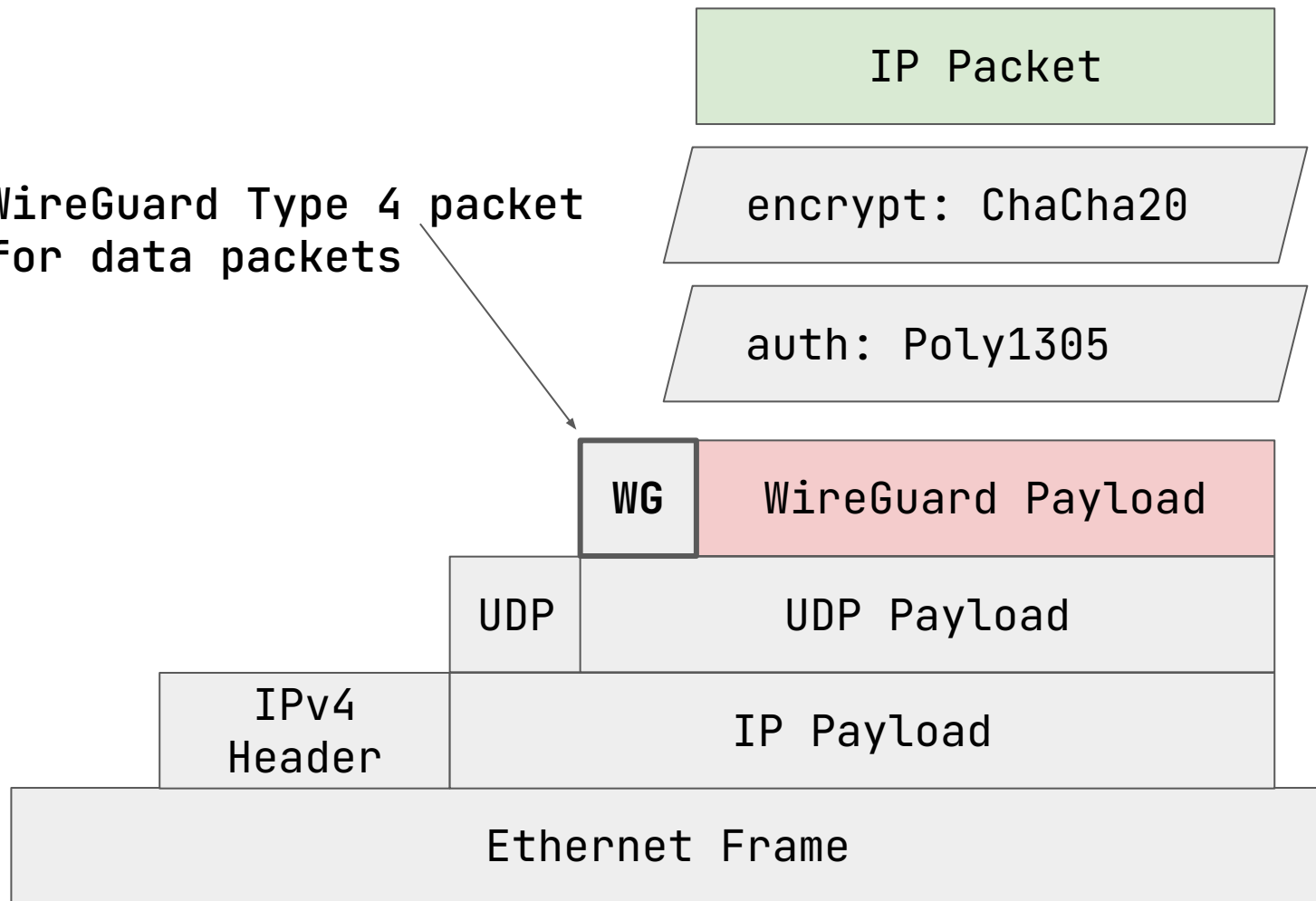
WIREGUARD[®]

FAST, MODERN, SECURE VPN TUNNEL

- state-of-the-art clean-slate protocol for building network overlays with encryption and authentication, IP/IP, many use cases, trusted crypto.
- a *WireGuard* **peer** is a host in the VPN, identified by its public key. each peer can potentially reach every remote peer directly: **meshing**
- a *WireGuard* **endpoint** is the public (or outer) UDP address. may change when endpoint **roams**; the tunnel stays up.
- each peer has a set of **allowed IP** address prefixes within tunnel. **multicasting** is possible



WireGuard Type 4 packet
for data packets



Why WireGuard on an FPGA?



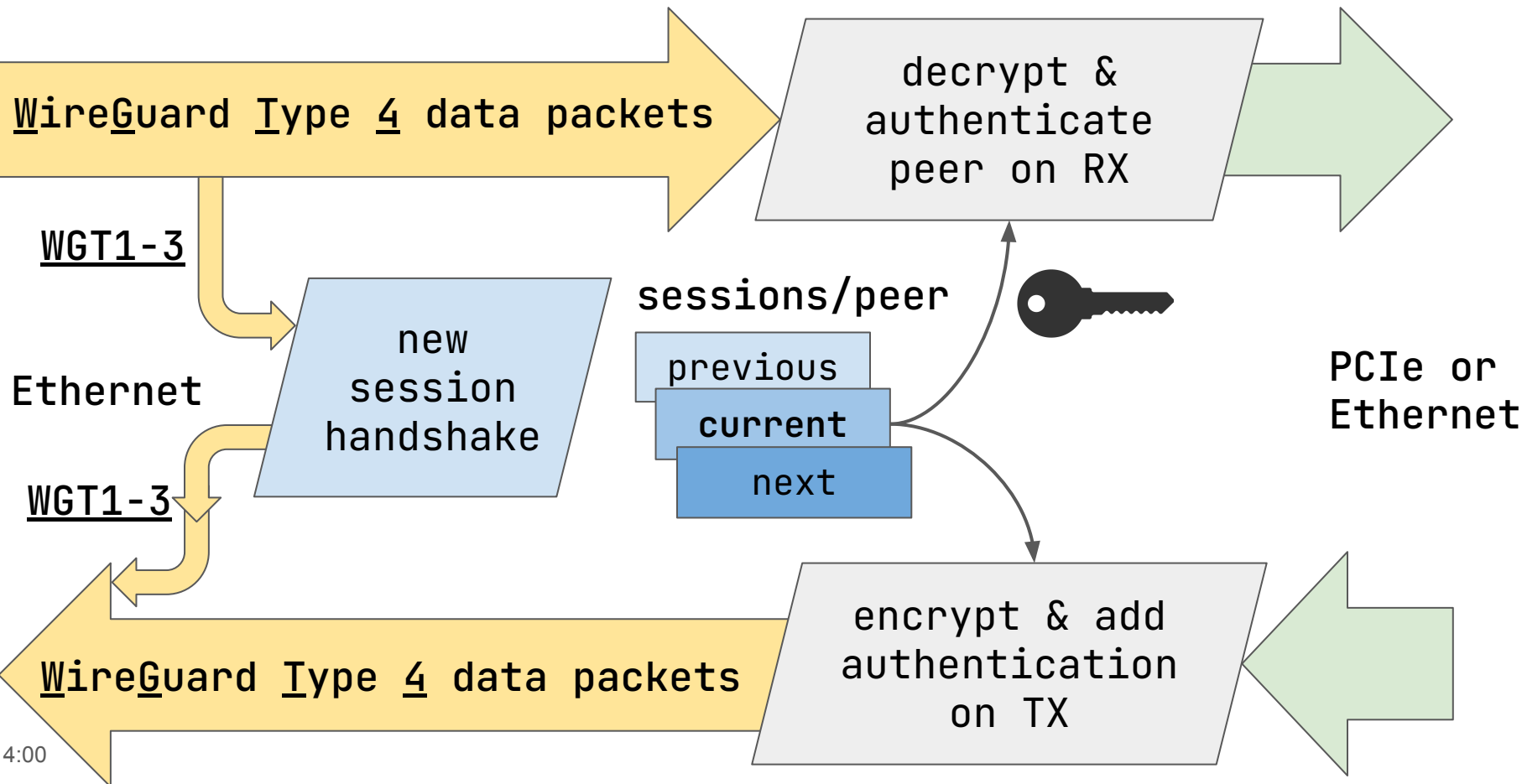
- Speed growth curve of Ethernet is steeper than that of CPUs. Offloading 'instructure tasks' such as VPN/encryption and authentication to smartNIC FPGAs becomes cost effective.
- Deterministic guaranteed behaviour vs. best-effort on CPU.

Example use case:

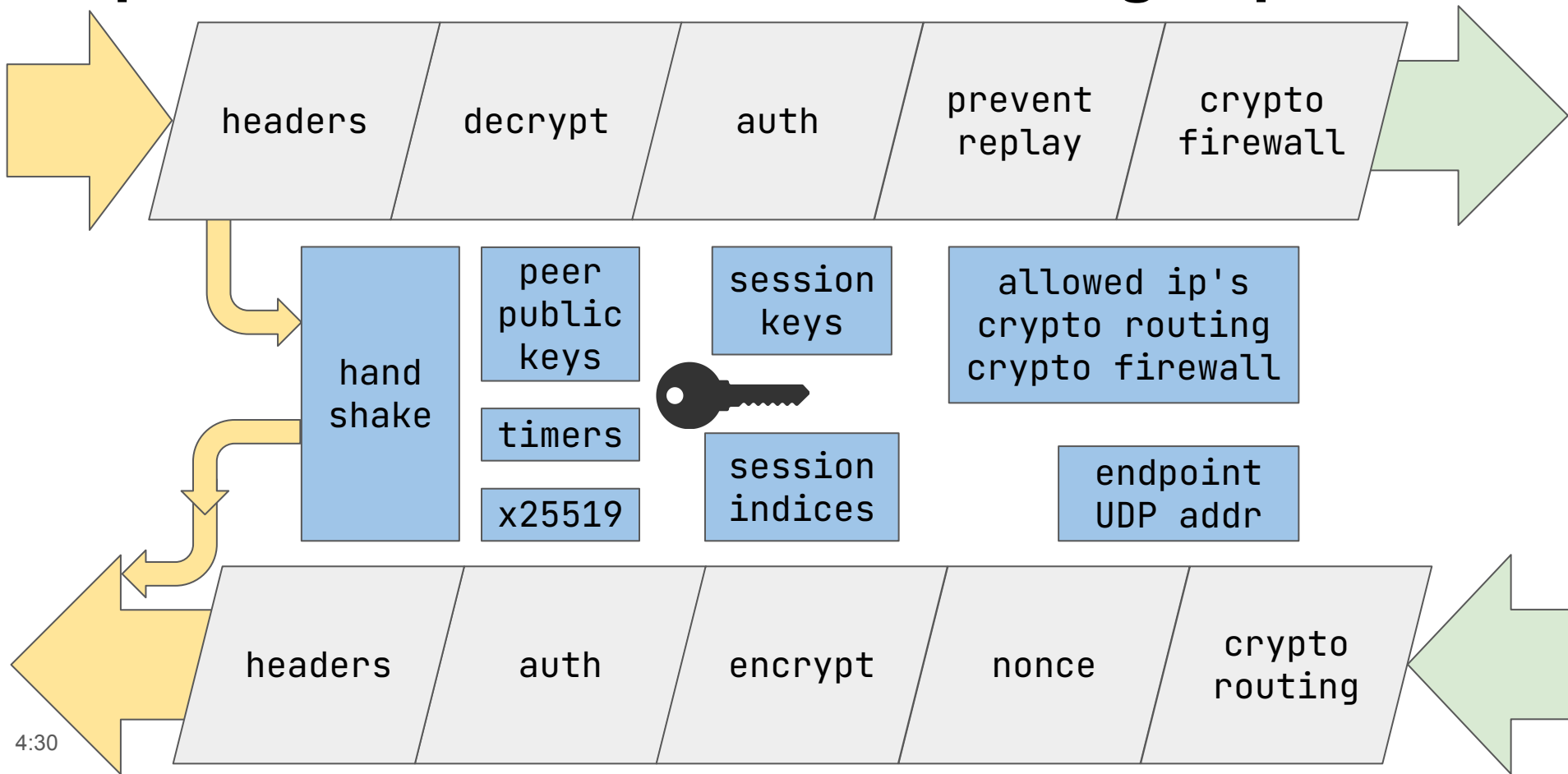


- 9th February 2023 – The Video Services Forum (VSF), has further enhanced the Reliable Internet Streaming Transport (RIST) protocol with the use of WireGuard VPN in RIST devices.

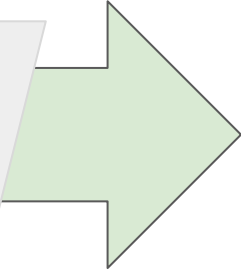
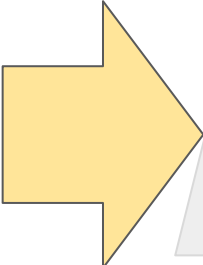
Map WireGuard into our FPGA design space




Map WireGuard into our FPGA design space




Map WireGuard; Talk Numbers, challenges



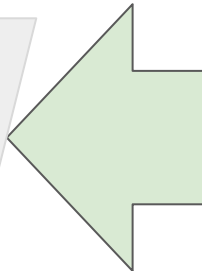
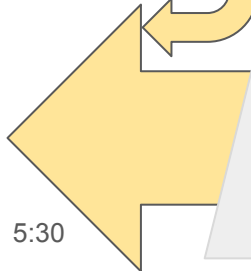
100 Gbit/s, 512 bits wide AXI Streaming 250 MHz
worst case one packet header in each clock cycle



roughly one handshake per minute
per active peer connection

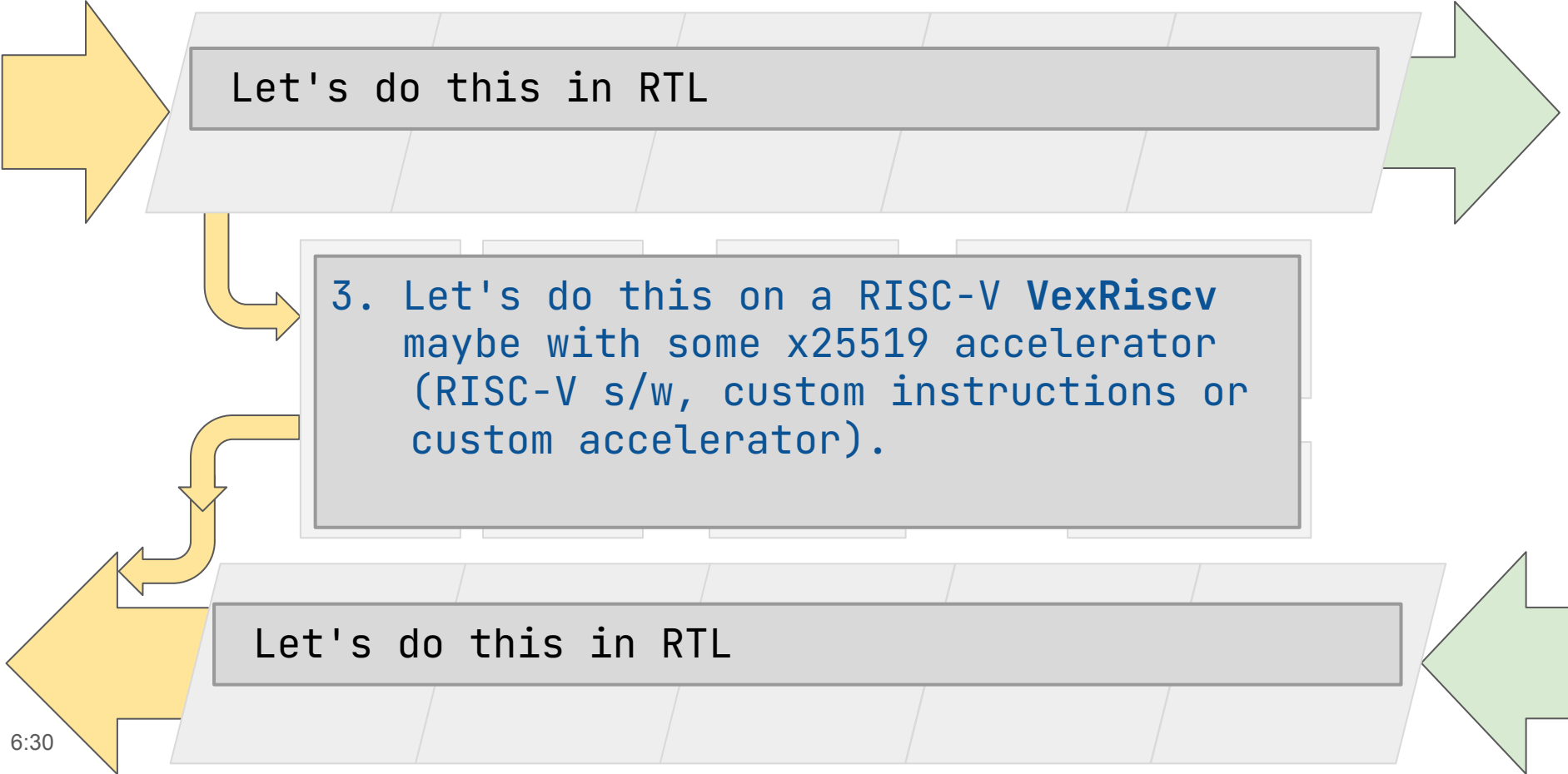


1024 peers \rightarrow 58 ms budget per handshake
take conservative design budget: 25 ms?



100 Gbit/s, 512 bits wide AXI Streaming 250 MHz
worst case one packet header in each clock cycle

Initial design choices for implementation

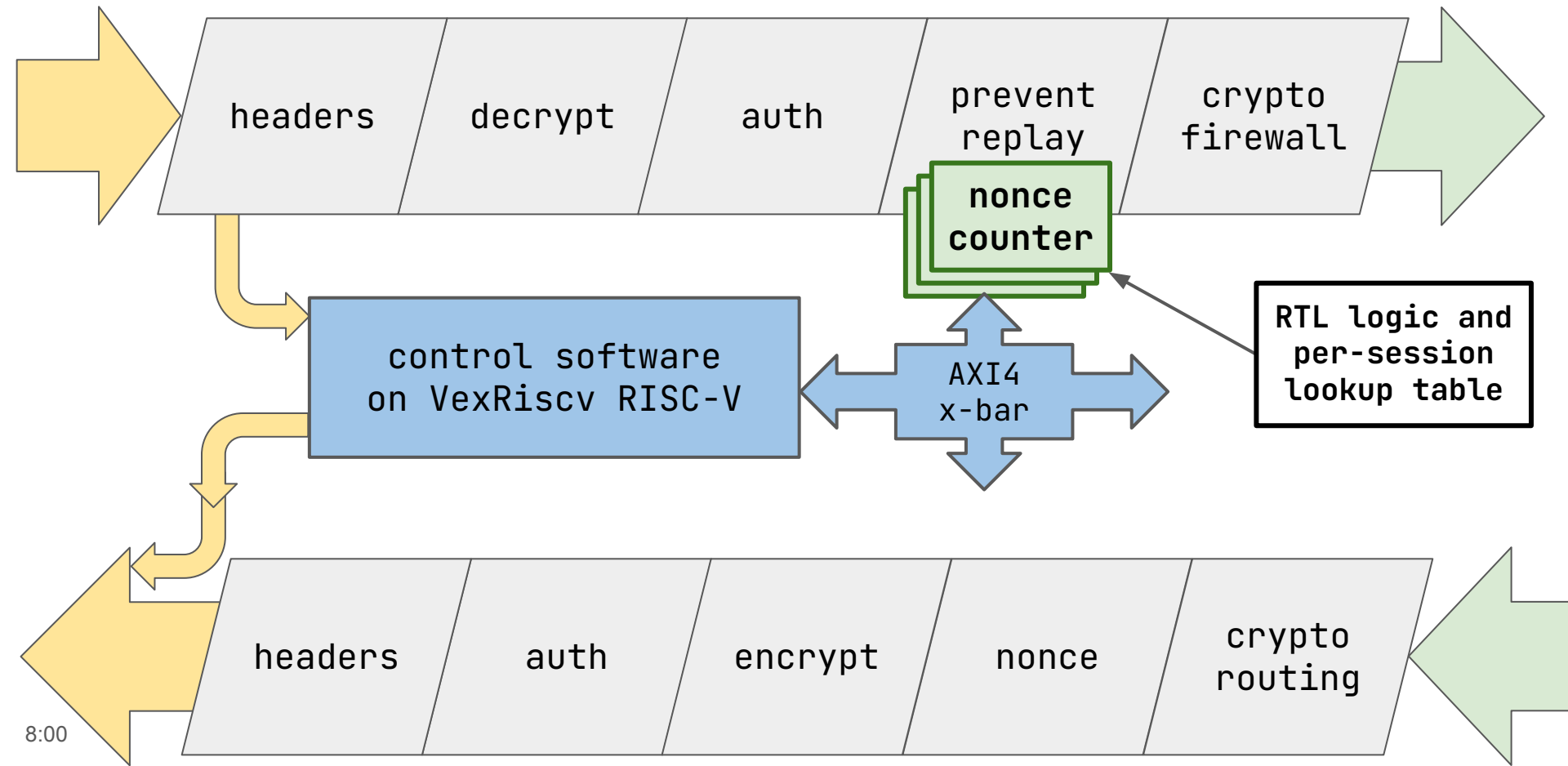


Let's do this in RTL

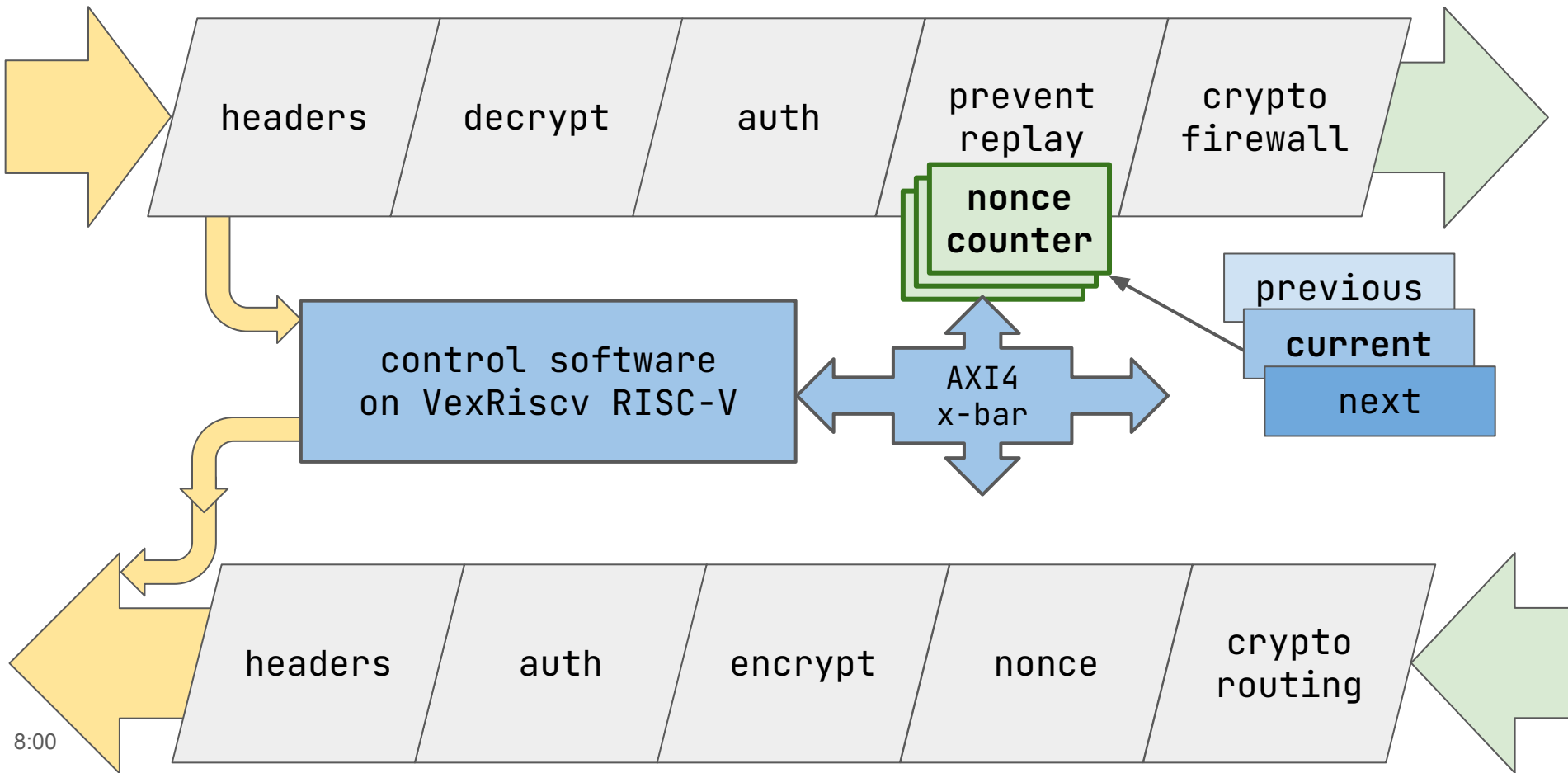
3. Let's do this on a RISC-V **VexRiscv** maybe with some x25519 accelerator (RISC-V s/w, custom instructions or custom accelerator).

Let's do this in RTL

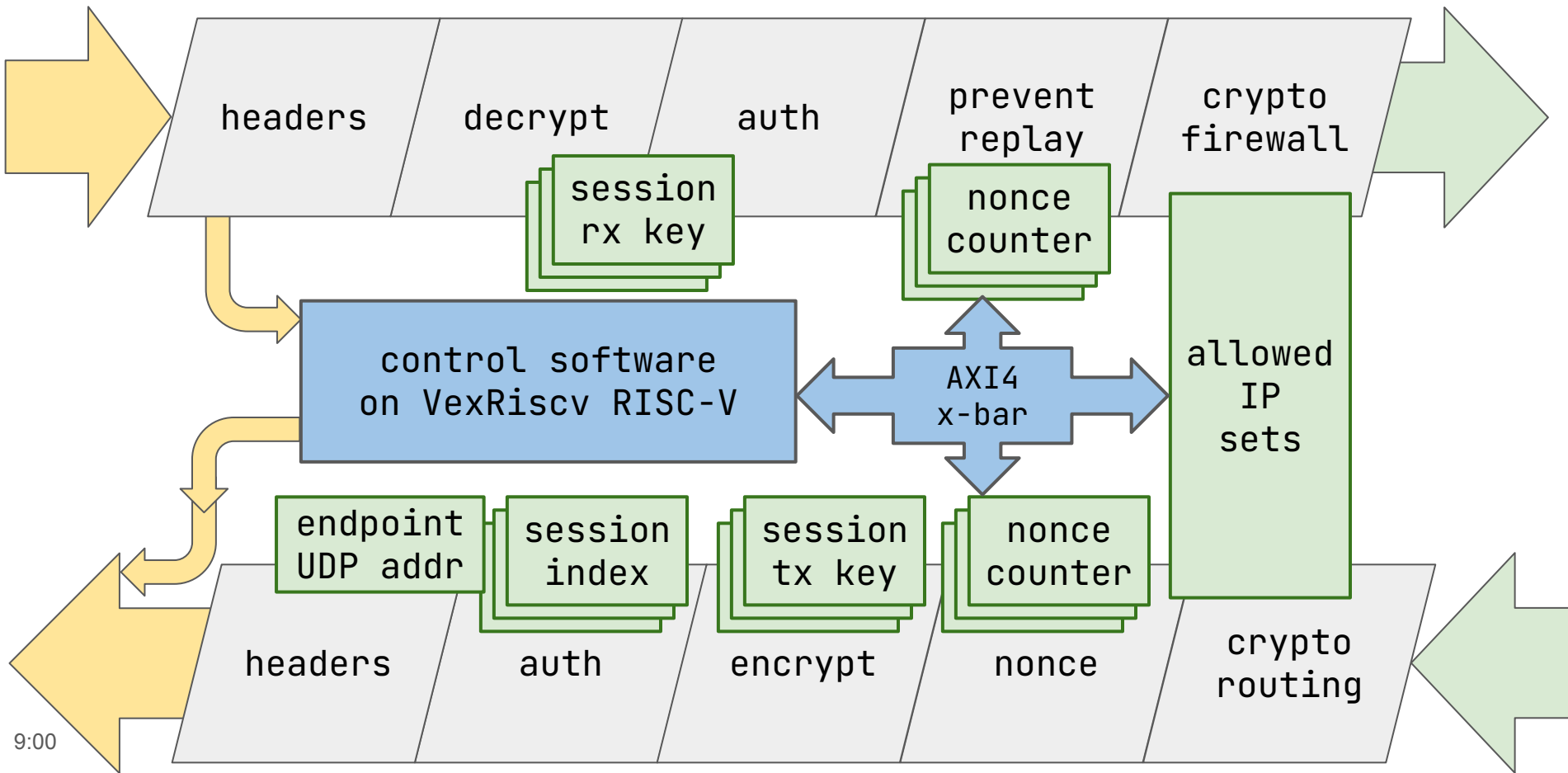
Design choices: (De)coupling data/control



Design choices: (De)coupling data/control



Design choices: (De)coupling data/control



SpinalHDL: design at RTL and system level



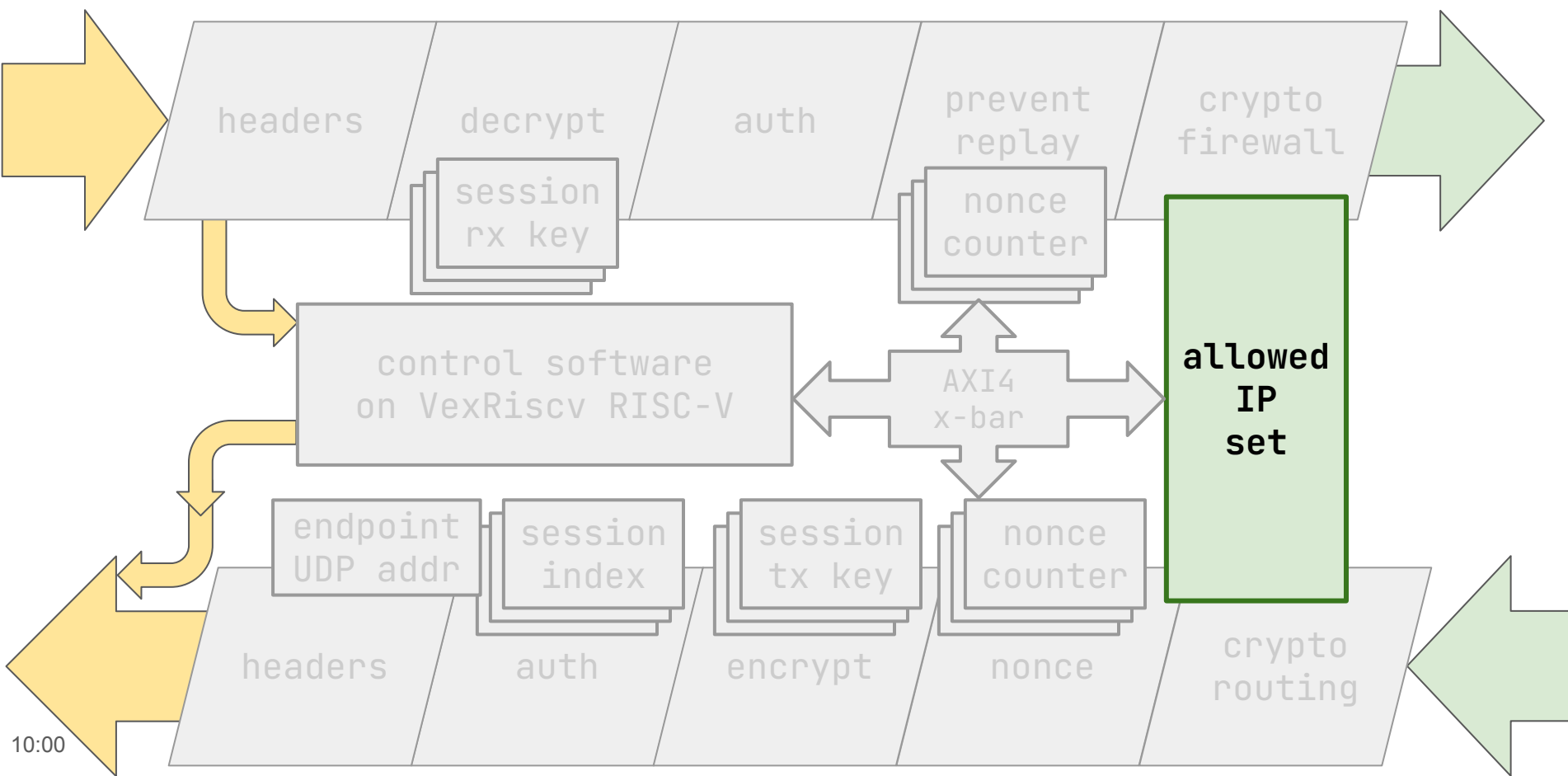
`SpinalHDL; RTL + zero-cost abstractions from lib`

3. Let's do this on a RISC-V VexRiscv
and build everything with **SpinalHDL**

Programmatically generate RTL, but
also SoC design at much higher level!

`SpinalHDL; RTL + zero-cost abstractions from lib`

Example: Allowed IP address prefix lookup

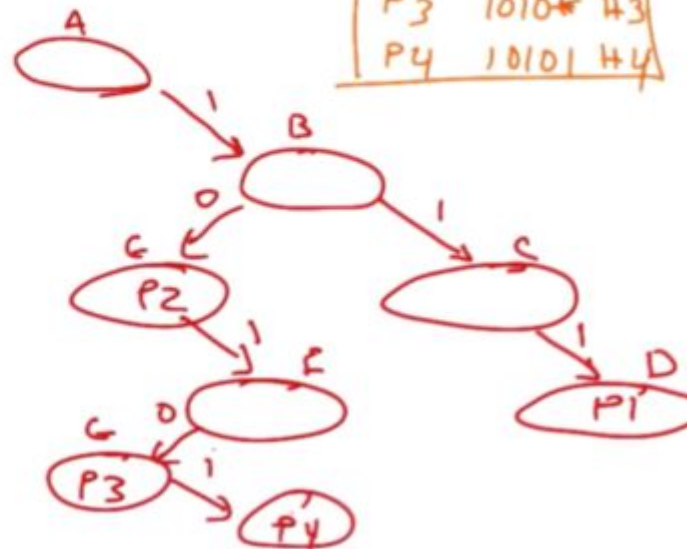


Allowed IP lookup: Longest prefix match using binary tree

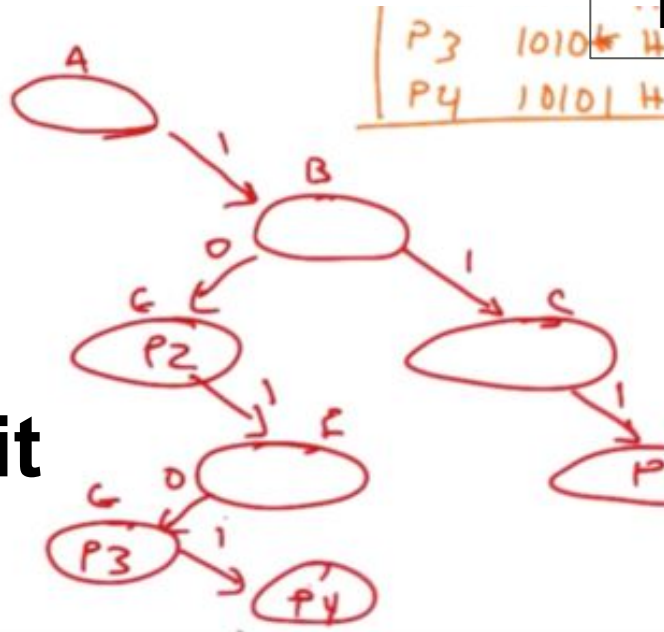
Address Lookup Using Tries

- Prefixes "spelled out" by following path from root
- To find the best prefix spell out address in trie.

P1	111*	H1
P2	10*	H2
P3	1010*	H3
P4	10101	H4



Pipeline: one IP address bit per clock



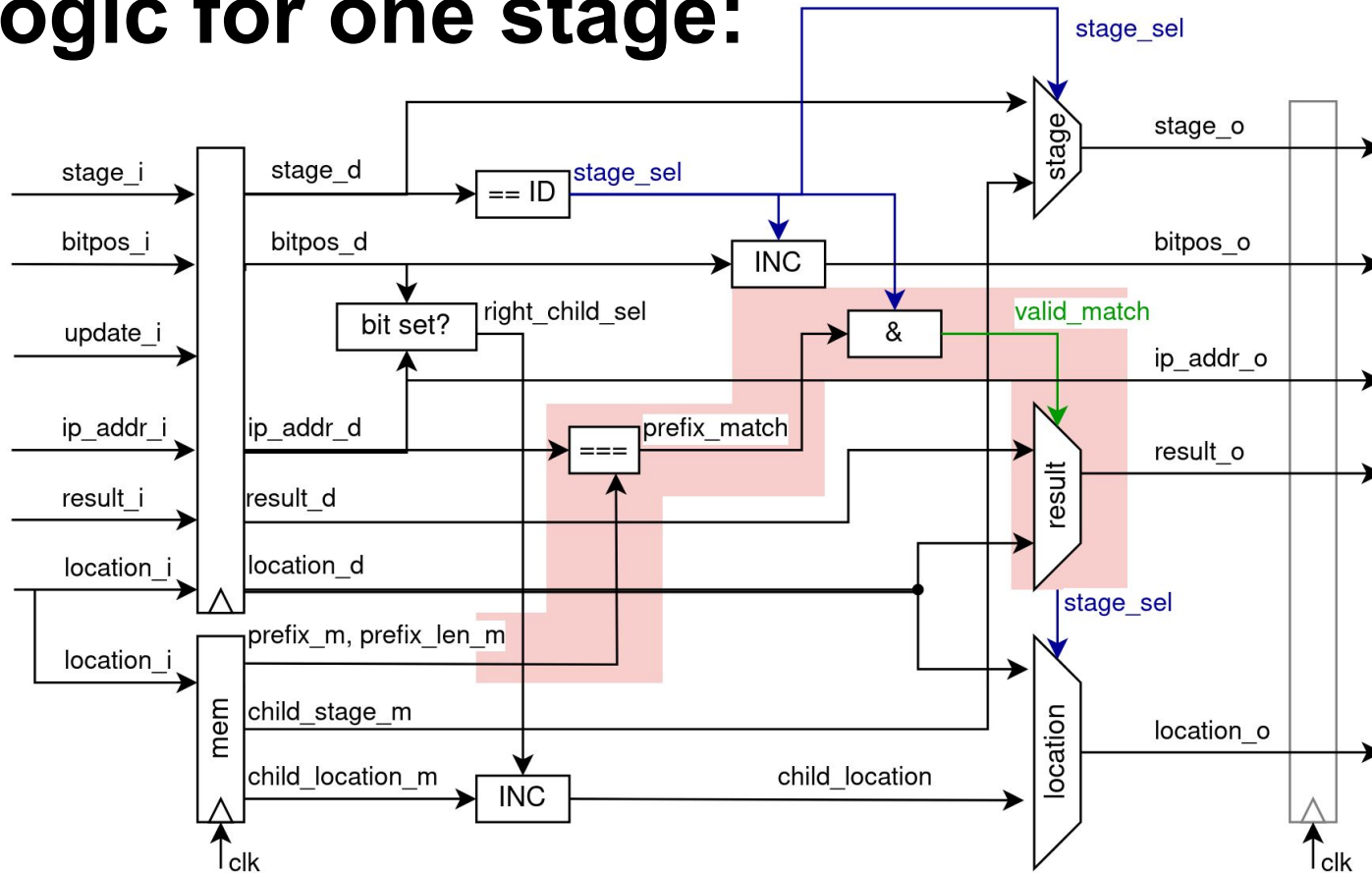
Match Addr w/ Prefix Set (Tree)

Lookup Stage 0	RAM
Lookup Stage 1	RAM
Lookup Stage 2	RAM
Lookup Stage 3	RAM
...	...
Lookup Stage 31	RAM

Result: longest prefix match

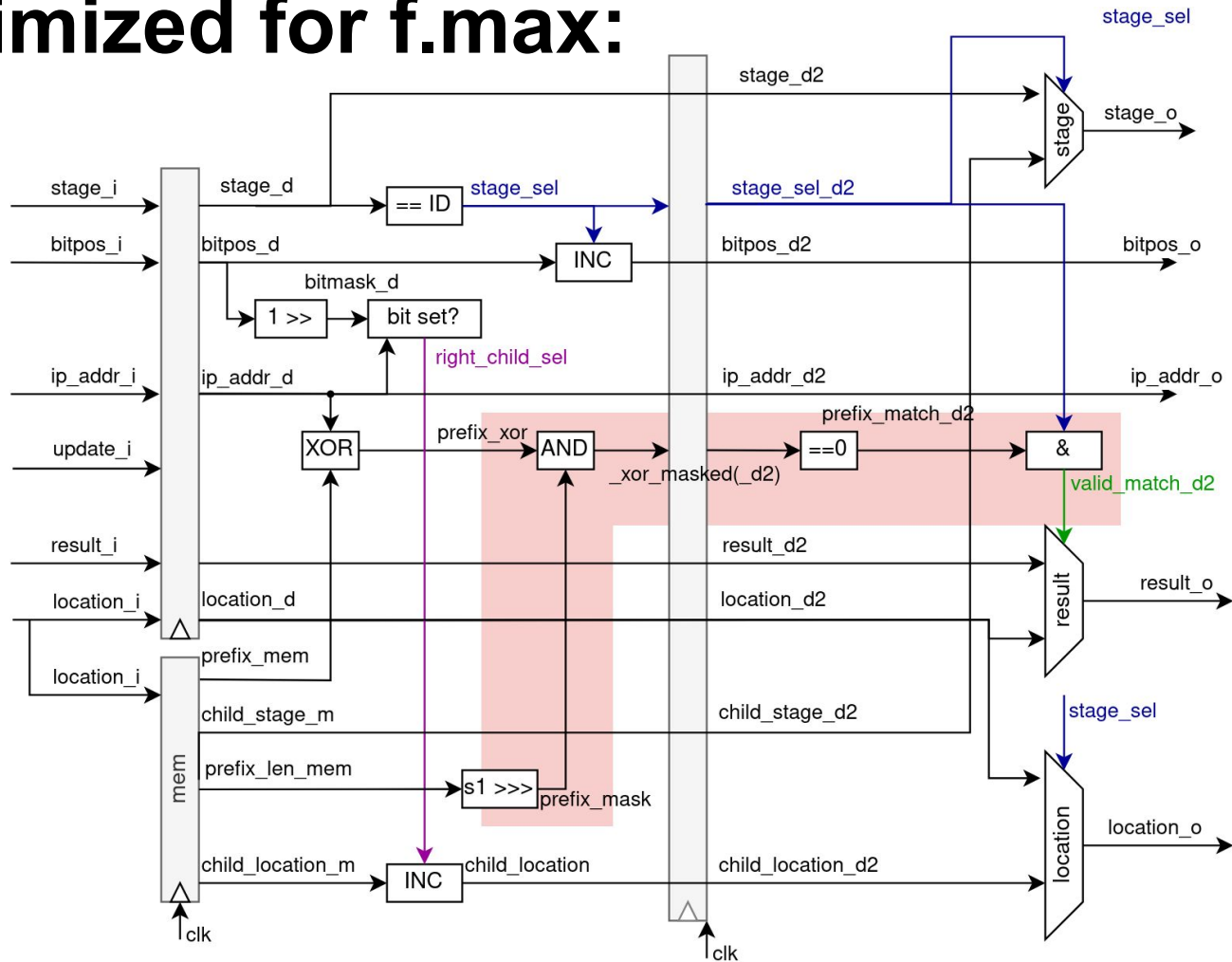
- Multi stage pipeline, one stage for each address bit.
- 32 bits/stages for IPv4, 128 for IPv6, 129 for both.
- Each stage has a memory (LUT, distributed or BRAM).
- Use optimally balanced tree (reduce worst case RAM use!).

Logic for one stage:

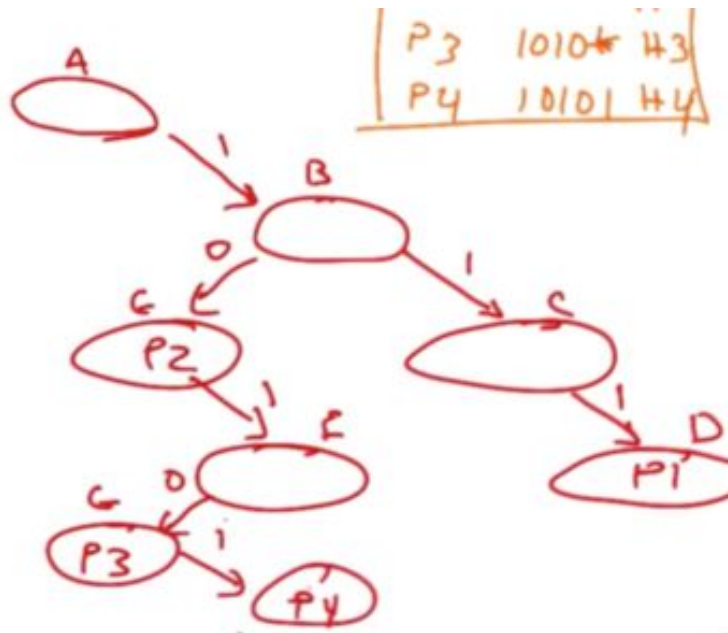


Vivado: report_design_analysis -logic_level_distribution

Optimized for f.max:



Match Addr w/ Prefix Set for RX and TX



RX	TDP RAM #0	TX
RX	TDP RAM #1	TX
RX	TDP RAM #2	TX
RX	TDP RAM #3	TX
	...	
RX	TDP RAM #31	TX

Result: longest prefix matches

- Two pipeline stages per bit
- Balanced combinatorial logic (≤ 4 levels) between registers
- 400 MHz on Ultrascale+
- True dual port RAM (RX reads, TX reads, RISC-V writes).
- **800 million Allowed IP address prefix lookups per second.**

Blackwire builds upon top-notch OSH:

- Blackwire **uses SpinalHDL** (Thanks Charles Papon!)
 - ◆ write cycle efficient RTL in less code
 - ◆ zero cost (no overhead) abstractions
 - ◆ the Spinal (building blocks) library is a piece of art
- Blackwire **uses Corundum** (Thanks Alex Forencich!)
 - ◆ SGDMA NIC design for PCIe and Ethernet FPGA boards
 - ◆ comes with Linux Kernel device driver

and tools... Verilator, GHDL, CocoTB, GTKWave, SymbiYosys, ...
thanks to everyone committing to those projects.

Thanks for explaining **formal verification** (Thanks Matt Venn!)

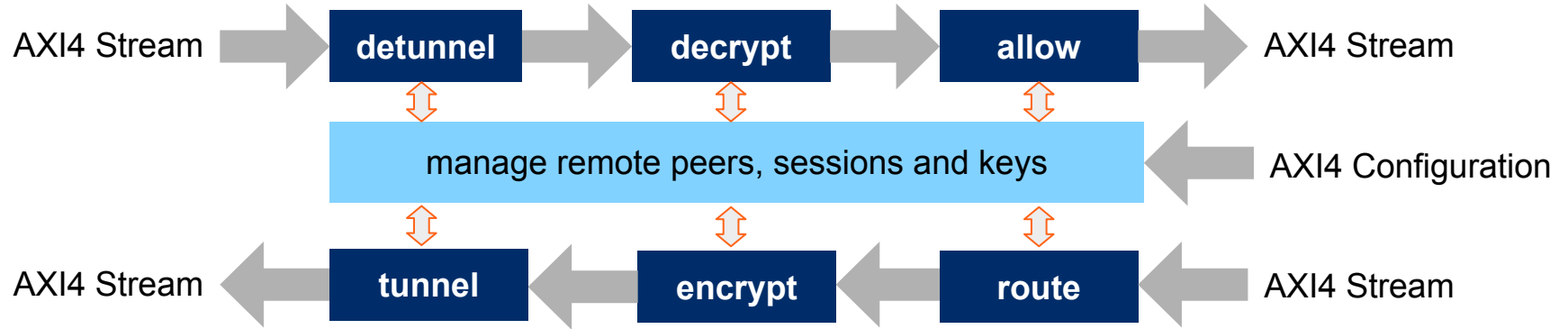
Blackwire Project Status (9/2023)

- Open sourced HDL on GitHub, some WIP to follow
<https://github.com/brightai-nl/BlackwireOverview>

FAQ: Actual code repositories listed in README!

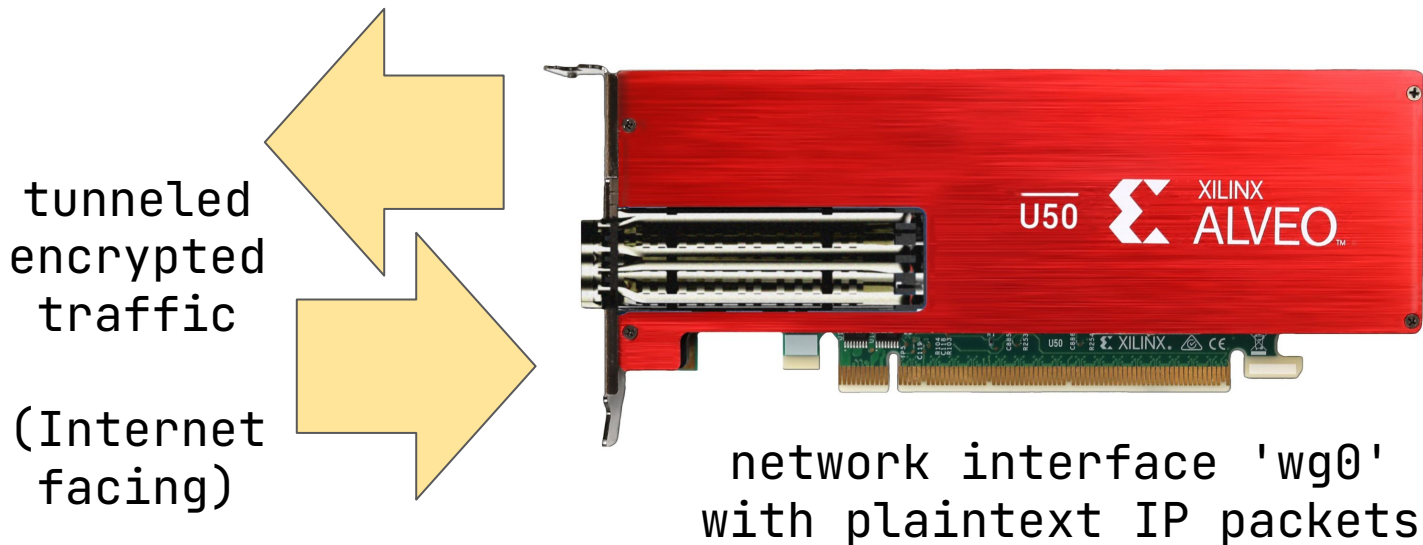
- Are We WireGuard Yet (AWWY)?
 - ◆ 75% done;
 - ◆ 25% to do, see README on GitHub for TODOs.

Blackwire IP Core



→ <https://github.com/brightai-nl/BlackwireOverview>

Blackwire: 'wg0' but implemented on FPGA



Blackwire: integrated in network infrastructure

tunneled
encrypted
traffic

plaintext
traffic



control interface
over PCIe (or other
interface)

Blackwire WireGuard

Thanks! *Questions?*

- <https://github.com/brightai-nl/BlackwireOverview>
- Q&A e-mail: Leon Woestenbergh <leon@brightai.nl>

Complementary Slides

Blackwire FPGA Resources ~ (for 100 Gbit)

Alveo U50 example with Corundum + WireGuard (BRAM, no URAM)

RX path is 128 Gbit/s, TX path is 64 Gbit/s in this design

Name	¹	CLB LUTs (871680)	CLB Registers (1743360)	CARRY8 (108960)	F7 Muxes (435840)	F8 Muxes (217920)	CLB (108960)	LUT as Logic (871680)	LUT as Memory (403200)	Block RAM Tile (1344)	URAM (640)	DSPs (5952)
▼ N fpga		31.39%	32.04%	18.67%	0.36%	0.11%	58.96%	26.90%	9.71%	25.78%	6.09%	21.91%

Resources (roadmap: move more registers into BRAM/URAM)

Name	¹	CLB LUTs (871680)	CLB Registers (1743360)	CARRY8 (108960)	F7 Muxes (435840)	F8 Muxes (217920)	CLB (108960)	LUT as Logic (871680)	LUT as Memory (403200)	Block RAM Tile (1344)	URAM (640)	DSPs (5952)
▼ I app.app_block_inst (mqnic_app_block)		211501	462593	19298	649	8	50192	182601	28900	153	21	1300

add the following numbers for 100 Gbit/s full duplex:

subtract the following numbers for ~60 Gbit/s full duplex

> I packetTx_tx (BlackwireTransmit)	67906	156822	6397	185	4	17931	59587	8319	20.5	0	432
-------------------------------------	-------	--------	------	-----	---	-------	-------	------	------	---	-----



Olof Kindgren (He/Him) (He/Him) • 1st

1mo ...

Award-winning Engineer and Actor at Qamcom

This is super interesting. I have been looking at doing exactly the same thing. Is this a proprietary or open source implementation? Would love to read some more about the work

Like ·  6 | **Reply** · 3 Replies



Olof Kindgren (He/Him) (He/Him) • 1st

1mo ...

Award-winning Engineer and Actor at Qamcom

This is super interesting. I have been looking at doing exactly the same thing. Is this a proprietary or open source implementation? Would love to read some more about the work

Like ·  6 | Reply · 3 Replies



Olof Kindgren (He/Him) (He/Him) • 1st

1mo ...

Award-winning Engineer and Actor at Qamcom

Ah! It looks like it is written in SpinalHDL too :D I'm sure [Charles Papon](#) must be happy to see that :)

Like ·  1 | Reply