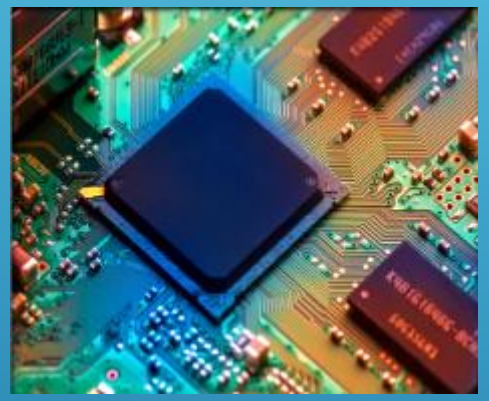
CHEVIN TECHNOLOGY



Designing Reusable, Portable and Secure IP for FPGA

- IP-XACT (IEEE1685) Standard for IP Portability and Reusability
- RTL Encryption (IEEE1735 v2) – Protecting Intellectual Property
- Secure by Design and HW-based Authentication/Authorization
- Case Study – STFC's Configurable Scientific Data Processor

arm SIEMENS UK Research and Innovation Si SILICON CATALYST A member of ADS

Si CHIP START UK the semiconductor incubator

BittWare a molex company

ALPHA DATA

intel partner Gold

AMD Adaptive Computing Partner

CHERI

NASA Jet Propulsion Laboratory California Institute of Technology

IP-XACT – IEEE 1685-2022

Schema developed by SPIRIT Consortium/Accelera (First version 2009)

Backed by ARM, Cadence, Synopsys, Mentor, NXP, STM, TI and others

Features provided by XML Standard

- Interfaces - Bus protocols, e.g. AXI4 and user-defined
- Interconnect - Hierarchical Design, Clock Domains
- Memory Maps - Constraints
- Version Control - VLNV (Vendor Library, Name Version)

Benefits

- Automation – Connect RTL Instances, Create Memory Maps
- Encapsulation - Manage Complexity
- Reduce Risk from Error-prone Manual Steps
- Increase Reusability – reduced effort in design and test

Supported by

https://www.accellera.org/images/downloads/standards/ip-xact/IPXACT-2022_user_guide.pdf

VENDOR IDE TOOLS

Productivity Tools from EDA Vendors

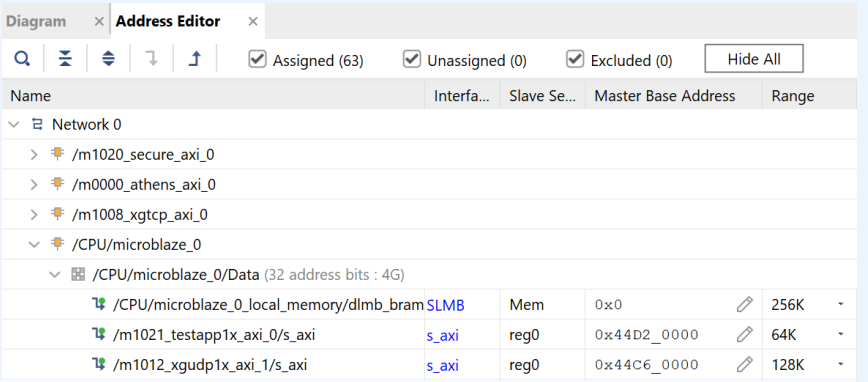
- Automatic Connection and Address Handling
- Hierarchical Design / Manage Complexity
- Hardware/Software Co-Development & Integration

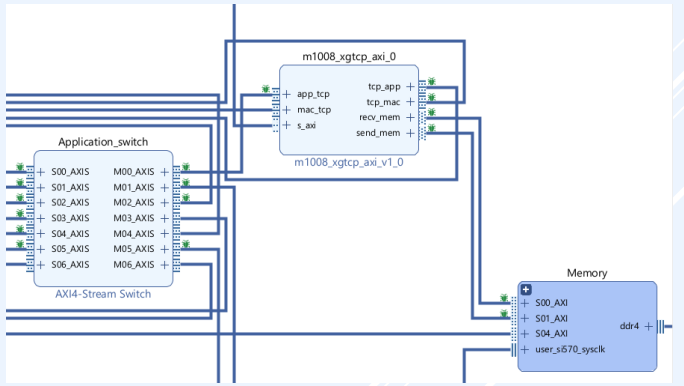
Benefits

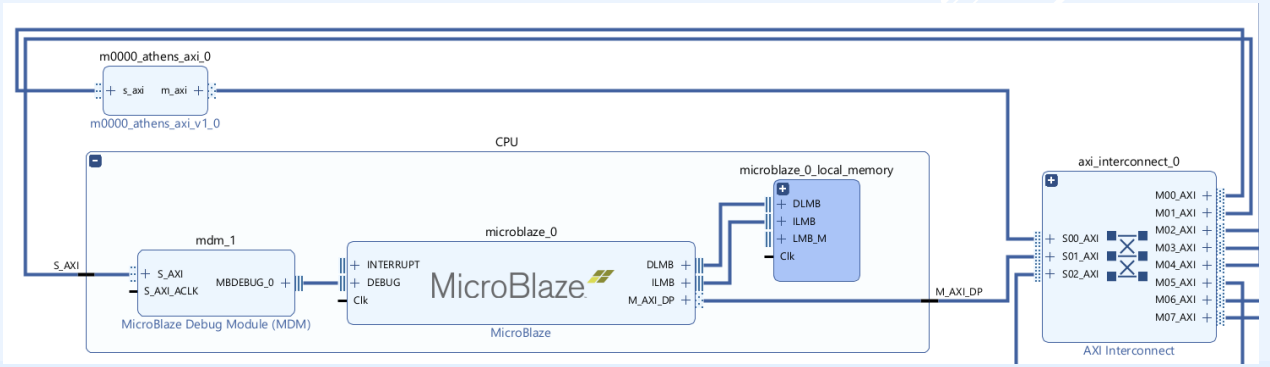
- Faster Results - Error Checking
- Scalable – Reusable – Portable
- Visual Representation – Version Control

Cons

- Vendor Lock-in – IP version updates
- Design Bloat – Large tcl files – Large IP Files
- Obscured Complexity – Run-time







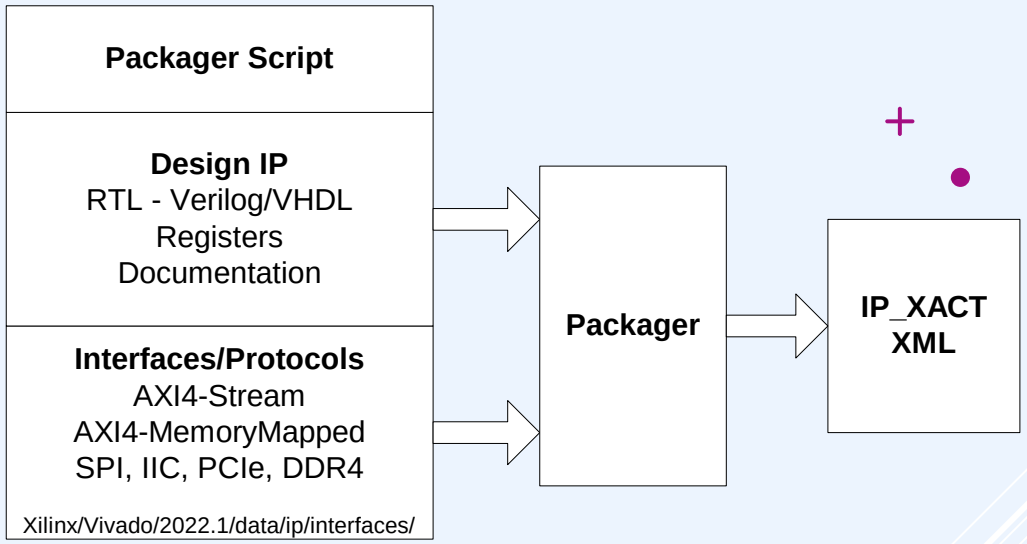
IP PACKAGER

Interfaces Library (also an XML File)

AbstractionDefinition – e.g. spi_rtl

Ports

- **BusType – VLNV – name:SPI**
- **LogicalName – SS_I, SS_O, SPISEL**
- **Connectivity – onMaster, onSlave**
- **Presence – optional/required**
- **Width – integer value 1..N**
- **Direction – in, out**



Packager Script (tcl file)

- Design IP Files Verilog/VHDL
- Clock Definitions – Clock name, FreqHz
- Bus Interfaces – Associated Clock Domains
- Memory Maps – e.g. AXI Lite registers 64KiB
- Core name, version , revision

IP COMPONENTS

Reusable IP Block

- IP Source Code
- Component XML

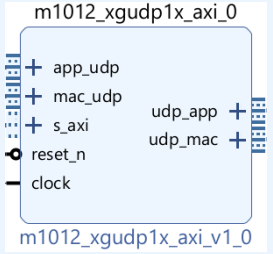
IP-XACT describes Interfaces

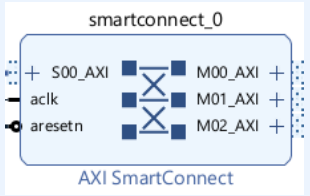
- IP + XML + GUI scripts tcl
- Master/Slave Interfaces
- Mapping Logical ⇔ Physical

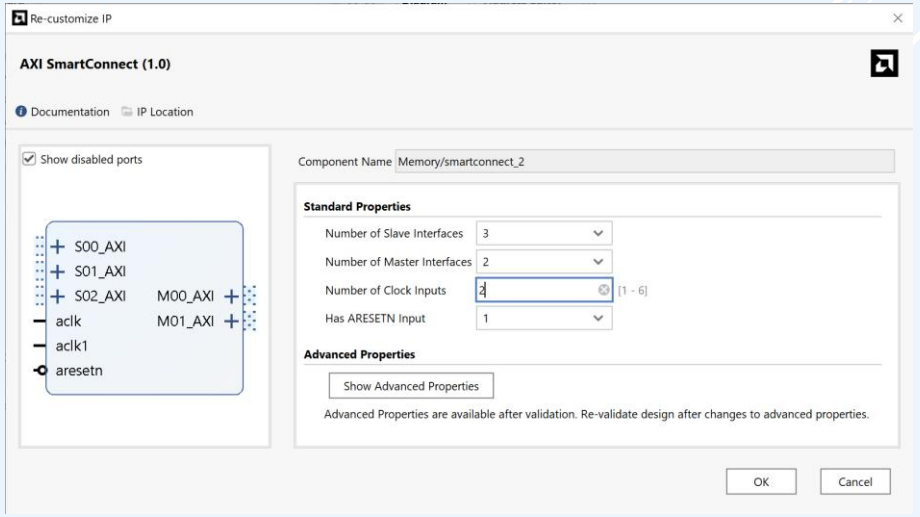
Vivado output (bd files)

Provided by EDA Tools Vendor

- GUI customizable
- User defined







IP BLOCK REPO

IP Repository

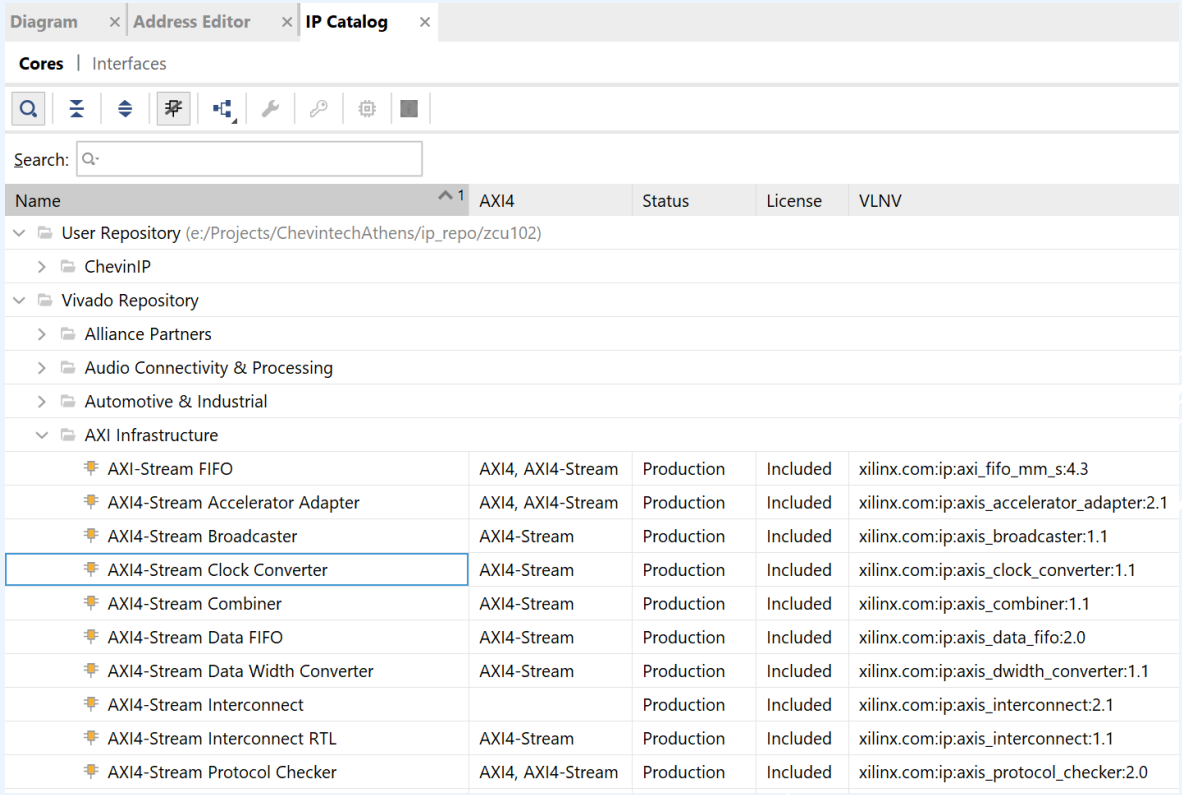
- RTL - one or more files
- XML Component
- TCL script for GUI parameters

EDA Tools Vendor Repo

- C:/Xilinx/Vivado/2022.1/data/ip/Xilinx

User Repo

- User's Packaged IP – any directory



VIVADO BLOCK DESIGNER

Commonly used interfaces

AXI4-S 64bit Stream with tkeep (byte lanes) and tlast (frames)

- IP-XACT <spirit:busInterfaces> spirit:name="axis“

AXI4-MM 64bit Memory Mapped – write/read to DDR Memory

AXI4-Lite 32bit Simple Memory Mapped – burst length=1

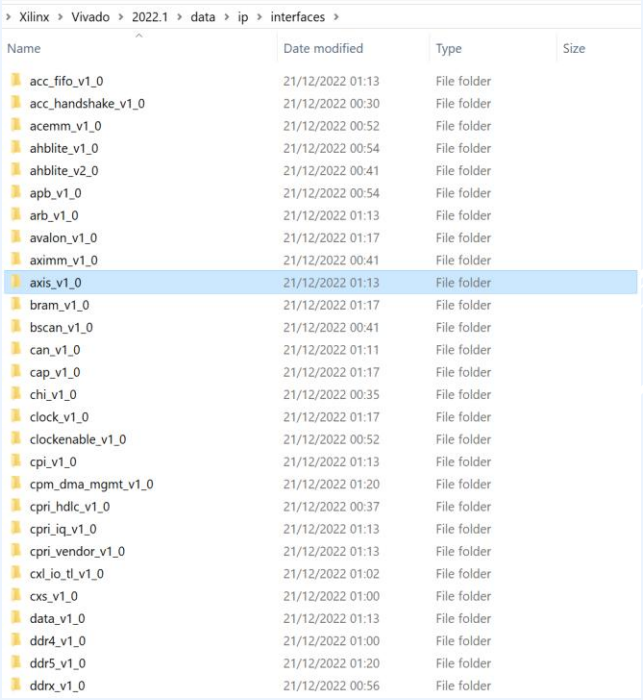
- IP-XACT <spirit:busInterfaces> spirit:name="aximm “

AXI-S Optional items HAS_TKEEP, HAS_TLAST, HAS_TREADY

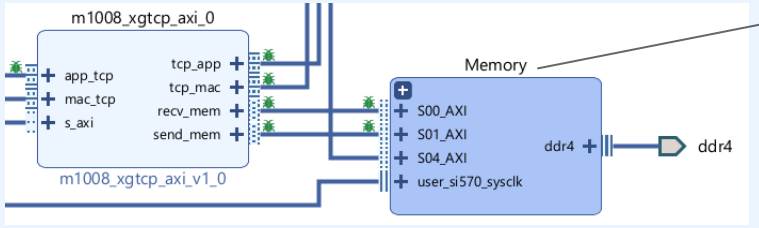
- IP-XACT <spirit:name>HAS_TKEEP</spirit:name>
- IP-XACT <spirit:name>HAS_TLAST</spirit:name>
- IP-XACT <spirit:name>HAS_TREADY</spirit:name>

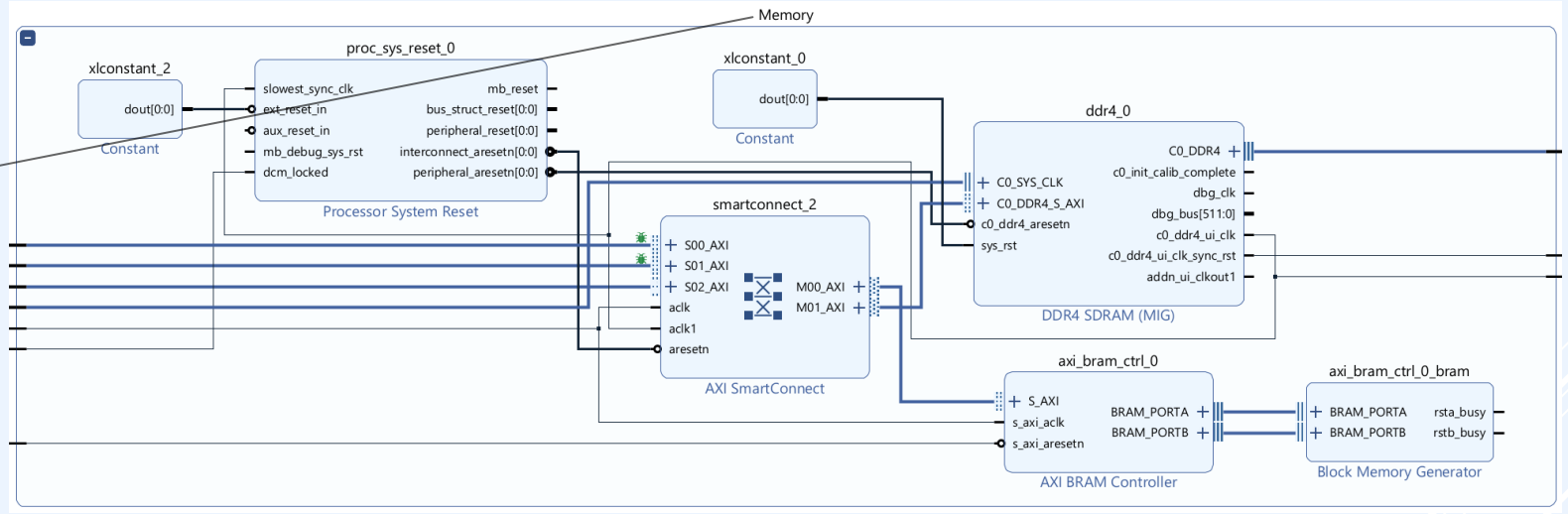
Bus Interface Definitions

- Vivado has 100’s installed
- User defined - create from examples



IP BLOCK DESIGN GUI





IP Block Instances

- Graphical Design Input or Scripts from command line
- Everything entered is IP, even constants
- Must follow IP-XACT, IP + XML
- Hierarchy – supported

Output

- Block Design .bd file - describes connection YAML
- Address Mapping – e.g. AXI4-Lite Base Address/Size

Scripts to save design to file

- write_bd_tcl design_name.tcl
- write_project_tcl design_name.tcl

Script to reproduce design

- source design_name.tcl

Re-create entire project from script

IEEE 1735-2014 V2 ENCRYPTION

IEEE Standard for Encrypted Electronic Design Intellectual Property

Background

- Encrypt HDL files – Verilog, VHDL, SystemVerilog
- AES + RSA key – Vendor Specific Public Key
- IEEE 1735-Version 2 - commonly used by EDA Tools and Vendors
- Protect IP – Black Box, Authorized EDA tool only can open

Benefits

- Distribute HDL designs in secure format
- Reduce risk of reverse engineering
- Encapsulation - Manage Complexity
- Control Licenced Use – Simulation, Build, Expiry-dates

IEEE

Supported by

arm cadence SIEMENS SYNOPSYS intel AMD

https://standards.ieee.org/ieee/1735/7237/

OVERVIEW

What it does

- Encrypts HDL - so only target EDA tool can open it

What it doesn't do

- Prevent Unauthorized use by other than customers
- Prevent Unauthorized use on any other devices
- Prevent Unauthorized use beyond intended purpose
- Prevent Seeing what's inside
- Prevent Replacing parts of the design
- Protect against Trojans/ Spoofing / MiM

So, what is it good for?

```
// Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
// ------------------------------------------------------------------------------
// Tool Version: Vivado v.2022.1 (win64) Build 3526262 Mon Apr 18 15:48:16 MDT 2022
// Date        : Thu Feb 22 12:05:12 2024
// Host        : running 64-bit major release  (build 9200)
// Command     : write_verilog -force -mode design -file m1012_xgudp1x_axi.v
// Design      : m1012_xgudp1x_axi
// Purpose     : This is a Verilog netlist of the current design or from a specific
//               cell of the design. The output is an IEEE 1364-2001 compliant
//               Verilog HDL file that contains netlist information obtained from
//               the input design files.
// Device      : xczu9eg-ffvb1156-2-e
// ------------------------------------------------------------------------------
`timescale 1 ps / 1 ps
`pragma protect begin_protected
`pragma protect version = 1
`pragma protect encrypt_agent = "XILINX"
`pragma protect encrypt_agent_info = "Xilinx Encryption Tool 2022.1"
`pragma protect key_keyowner="Xilinx", key_keyname="xilinxt_2021_07", key_method="rsa"
`pragma protect encoding = (enctype="BASE64", line_length=76, bytes=256)
`pragma protect key_block
VmFsYmIa6lOWaJjybrf25hoM1egwfzDyTaAktTIZ6W6dnt6cCqo8N/vMiaH8i7KUcq+qkNwO2Xms
p6gNwKT3B/bvYVnDndKBtmZZDv9PyPGGJd2ikakb1HFQZV7t2hcMxuTGRYirRwTBLny1uOSmUcYv
73hM3uMyGTa657P5ocKcWkoP8iT9doACwycwpgJPzBlzb+ac6DL6K/uAy3abkdYwYZyk5OHqyPQZ
PF8I8gMOfnhjtwjjDzRRJOIPfKrIV2P3SxZhCqFhGWUhNfTVxloT0zLeKXoWCGgulYnE3K39JsVZ
UG8YuXDp9Ny+CrwnTEWvFg3/ecy0OAbtKKQfcw==

`pragma protect data_method = "AES128-CBC"
`pragma protect encoding = (enctype = "BASE64", line_length = 76, bytes = 10692272)
`pragma protect data_block
rLmZQxnmBv0ctAZLIThaR6p5M9N9Lzt/i6GhEHVKLkViE81hDxHn3Kt3dYyvGWuVi6QtL1oOW1uB
7IXkaS/Yhov2U2twe2OxS6OIGdAeGWVAP8EL7Oqbm26Y2Gd7uk1ZbSbjjK2WXnjAIZzYzbc3xV9/
MMKCzBxm5sHLcs5wKRFkcHyeW6AJw0JtRHFXbxfEkJsHjRK78/6KrNEYv9bdjpRfRXnlMBjYAcOe
KVK0YXErp8+ovGOafdeFGotKBNBEk85x7CzcvoRJ4xKBsKHofKARzWu6yVqHHMpU0/PrYJDvzdpc
+PMJz0RJw/oqYGy0E72+RguErex8Z7Veb8GmmhPySOOvl6hBa+uu8l9fttxUS19IOi5mRos4qsnU
8NAuw4ma+kmMqzBfs0rBO2D6pZ9GGSqCHoC8w7u7Gu2FXBfCJlIWZVree4gDwE4oPXHT5xC75Kz4
T/1+ekdzP6UI7RaOY1pJIR7CP9tJo06JfeypXAh5Y2RAB5ZeEudHWL3hDTOWa6Wc0y+jXRXHooOB
q368Bc6b40LfIo6FJJa9HU1VhdwdWKaXqplKVeYY7KAt88Tsq5e2vETIWMUf5hCcNW7cyjl75Wh2
```

ONLY EDA TOOLS CAN OPEN

However, once opened in EDA tool

- Signal - names, paths, width
- Hierarchy - names, interfaces, order, clock domain
- LUTs content - truth table

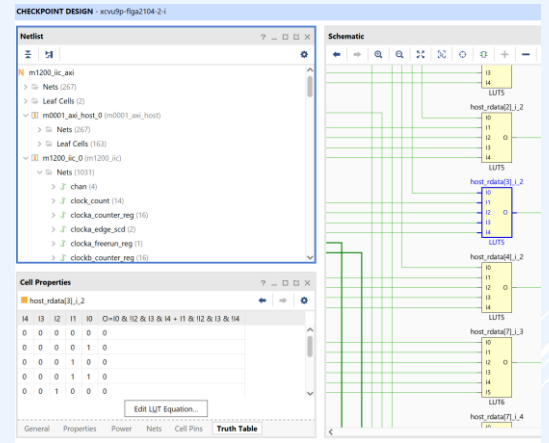
Schematic Viewer exposes design to reverse Engineering

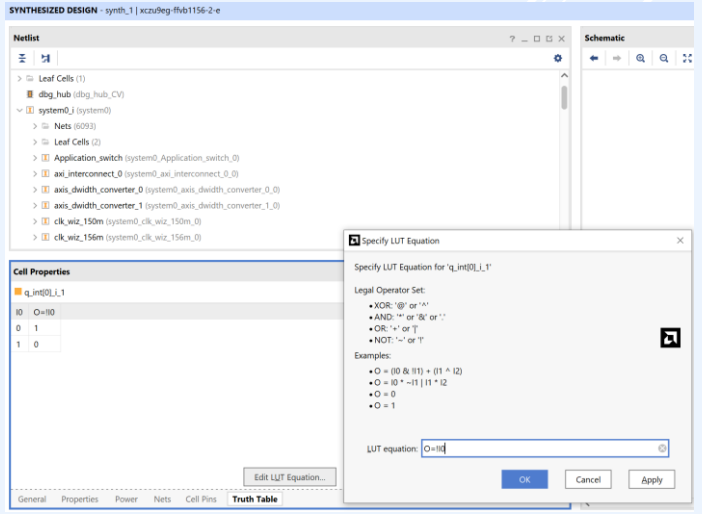
- Hierarchy can be explored
- Signals can be traced source/destination
- LUT Truth Tables can be read

Logic can be edited / inserted

- Truth Table Editor dialog box makes it simple
- Replace entire blocks of netlist – with later compile order

“Oh dear – design is unsecured“





SECURING THE NETLIST

Remedies in Source Code & Netlist

- Signals – obfuscate signal names
- Hierarchy – flatten hierarchy, obfuscate block names

Remedies in EDA Tool Directives

- LUTs content – prevent viewing truth table
- Simulation – prevent viewing or make sim -only

Design can still be copied, re-targeted, re-purposed

- Black Box - RTL code netlist open to unauthorized access

"Design can't be changed, but it could be spoofed"

Obfuscate

```
m_update_count <= m_update_count+1;
case to_integer(m_update_count) is
  when 7 =>
    m_update <= '1';
    m_update_count <= (others=>'0');

  when 6 =>
    m_raddr_update <= m_raddr_next;

  when 3 =>
    m_update <= '0';

  when others =>
end case;
```

→

```
zplwsujmroxk <= zplwsujmroxk + 1 ;
case to_integer ( zplwsujmroxk ) is
when 7 =>
rohcaivgsekm <= '1' ;
zplwsujmroxk <= ( others => '0' ) ;
when 6 =>
zytbirgeudwq <= cvfylbimwdxe ;
when 3 =>
rohcaivgsekm <= '0' ;
when others =>
end case ;
```

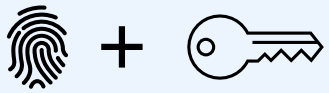
synth_design -top design_name -part xcvu9p-flga2104-2-i -mode out_of_context –retiming **-flatten_hierarchy full**

write_vhdl -force -mode funcsim -file file_name.vhd

SECURING THE HARDWARE

Make Netlist Hardware Specific – Check in SW

- Add a PUF – AMD/Xilinx DNA_PORT, Altera CHIP_ID
- Software reads PUF – accepts only authorized HW
- Vulnerable to attacks if check is run in software



Prevent attacks – Check it in HW

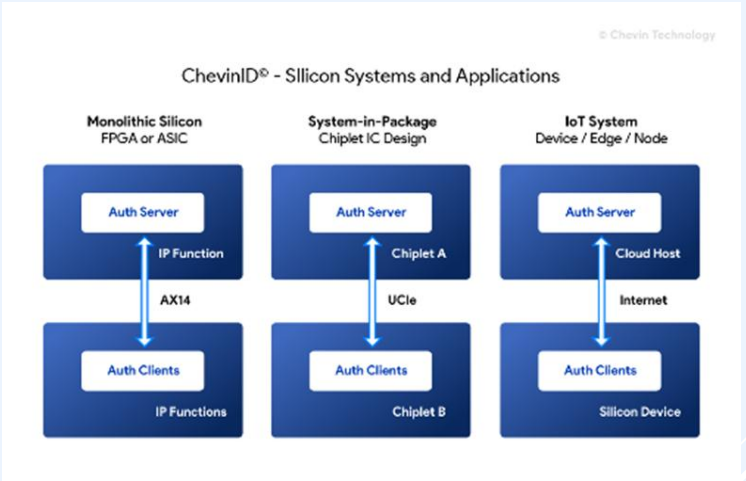
- PUF Identifier stored in netlist to identify HW
- Black Box - RTL code netlist open to unauthorized access

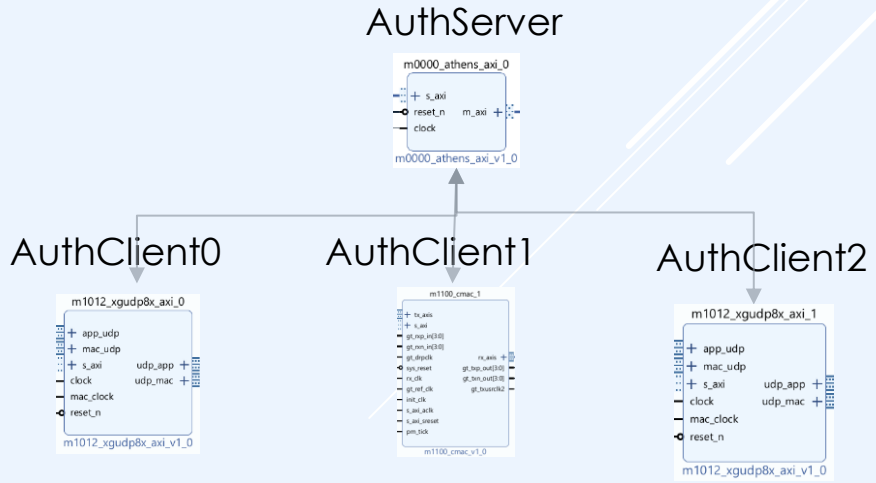
“Design is more secure, but we have only one PUF”

CHEVIN ID

Hardware-based Authentication

- PUF Unique Identifier verified with Message Authentication Code
- Root of Trust + AuthN/AuthZ
- Vendor Agnostic ASIC/FPGA/Chiplet
- CRA legislation coming into force for IoT (Dec 11 2027)
- MAC/HASH is inherently PQC safe
- Resilience - Prevents single point of failure from Attack
- Local authorization requests/acknowledge





“Verify Hardware Authenticity”

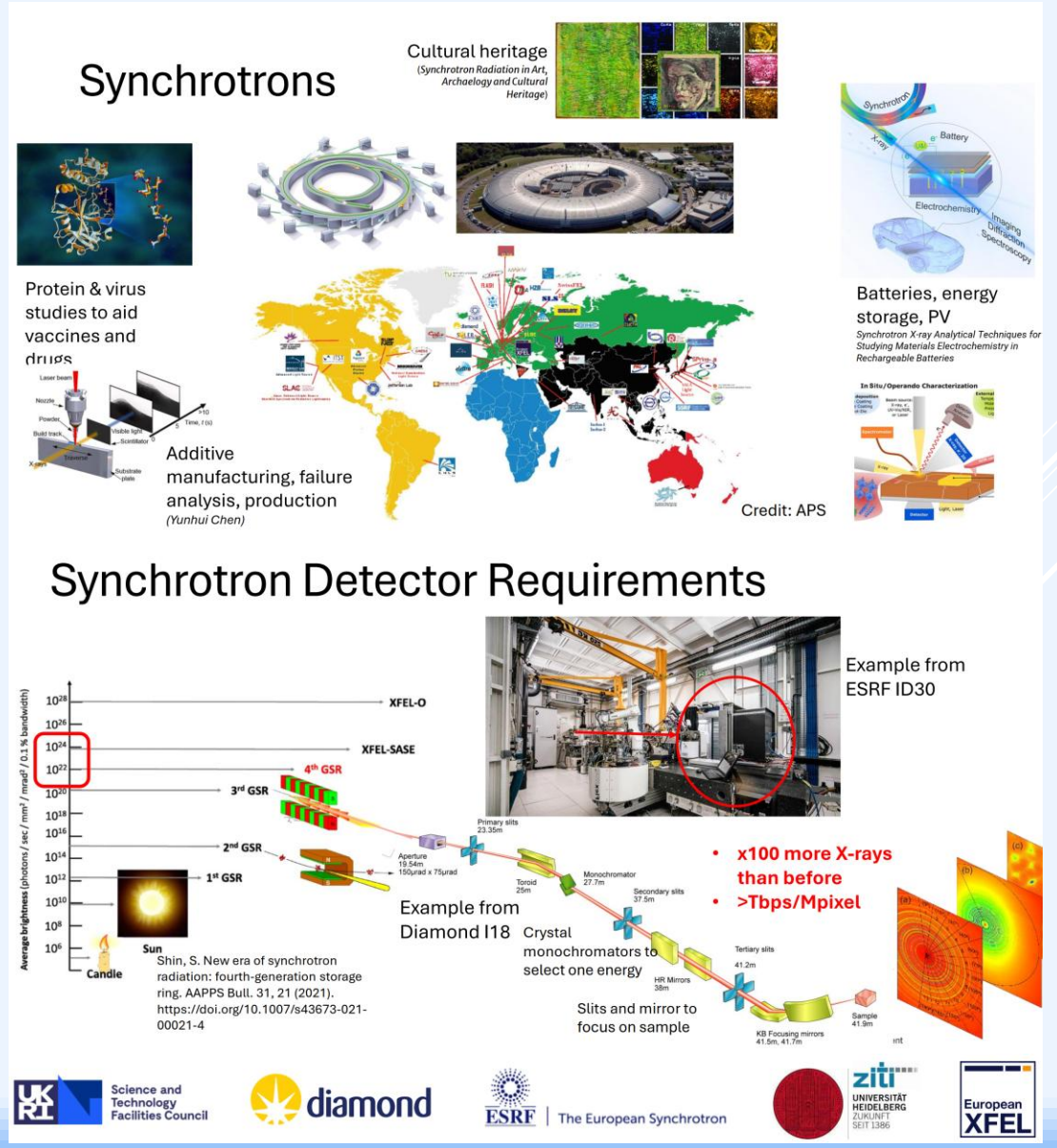
CASE STUDY – DATA ACQUISITION

STFC's Harwell Science & Innovation Campus

- Flexibility - Programmable Logic Solution with FPGA
- COTS Hardware for capture
- COTS Alveo and processing
- Standard Ethernet for transfer
- NVMe Storage

COTS Benefits

- Lower Costs
- Shorter development time
- Fast deployment



UDP IEEE 768

Use Cases

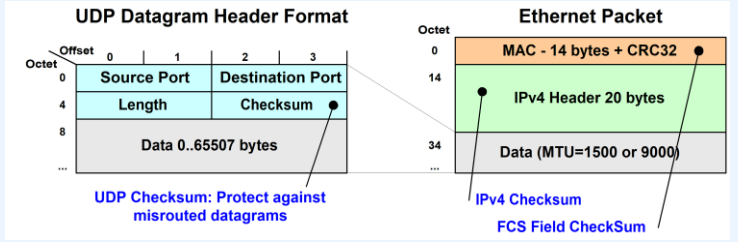
- Radar & EW – ADC/DAC Acquisition
- Satellite RF/IF – VITA49 & DIFI
- Mobile 5G RRU – eCPRI
- Video, Streaming, Gaming - RTP

Pros

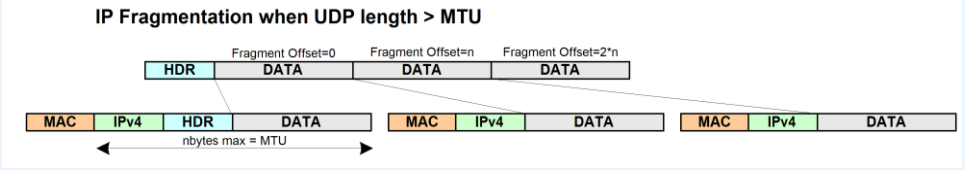
- Light Weight alternative to TCP
- Simple – No "established" sockets
- No acknowledge – Send and forget
- Compact - No resend buffer required

User Datagram Protocol UDP/IP

- Point to point
- Broadcast/Multicast
- Datagram Packet - All or nothing
- Socket SRC/DST, Len & Checksum
- Length supported 0..65507 bytes



- Fragmentation supports large datagrams, > MTU



UDP IEEE 768

Cons

- Packet loss causes missing data
- Packet Duplication
- Packets Out-of-Order
- No Sequence number
- No Congestion Control
- No Flow Control

"I could tell you a joke about UDP, but you probably wouldn't get it"

Resend Mechanism *can* be added

- Sender sends and increments Sequence
- Receiver acknowledges Sequence
- Sender
- Take care not to "reinvent TCP"

Minimum Requirements

- Source/Destination IP address resolution
- ARP – request/reply – ARP table
- ICMP – Ping is optional but "nice to have"

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Xilinx_44:a8:09	Broadcast	ARP	64	Who has 192.168.128.101? (ARP Probe)
2	0.000002784	IntelCor_a7:29:7a	Xilinx_44:a8:09	ARP	64	192.168.128.101 is at 90:e2:ba:a7:29:7a
3	0.000026278	192.168.128.11	192.168.128.101	ICMP	74	Echo (ping) request id=0x0003, seq=0/0, ttl=128 (reply in 4)
4	0.000031597	192.168.128.101	192.168.128.11	ICMP	74	Echo (ping) reply id=0x0003, seq=0/0, ttl=128 (request in 3)
5	0.000057139	192.168.128.11	192.168.128.101	UDP	1066	32000 → 16000 Len=1024
6	0.000058003	192.168.128.11	192.168.128.101	UDP	1066	32001 → 16001 Len=1024
7	0.000058867	192.168.128.11	192.168.128.101	UDP	1066	32002 → 16002 Len=1024
8	0.000059731	192.168.128.11	192.168.128.101	UDP	1066	32003 → 16003 Len=1024
9	0.000060595	192.168.128.11	192.168.128.101	UDP	1066	32004 → 16004 Len=1024
10	0.000061459	192.168.128.11	192.168.128.101	UDP	1066	32005 → 16005 Len=1024
11	0.000062323	192.168.128.11	192.168.128.101	UDP	1066	32006 → 16006 Len=1024
12	0.000063187	192.168.128.11	192.168.128.101	UDP	1066	32007 → 16007 Len=1024

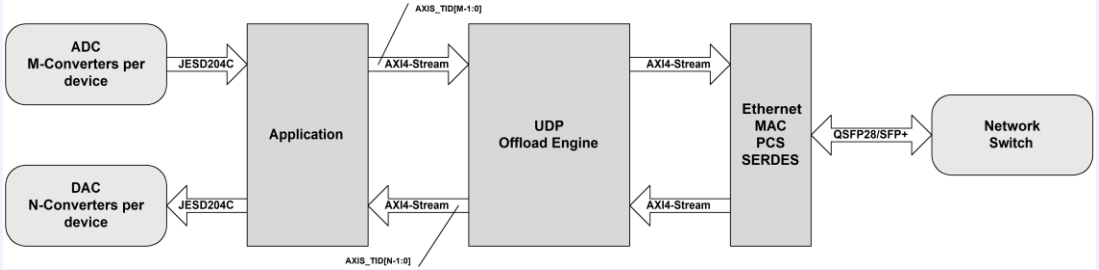
Flow Control Optional

- IEEE 802.3x Ethernet MAC PAUSE frames support flow control via Layer2 multicast address 01-80-C2-00-00-01, "ask the sender to pause"
- IEEE 802.1Qbb, Priority Flow

AXI4 HARDWARE FOR UDP

AXI4-Stream

- AXIS_TDATA(511:0) – Payload byte lanes
- AXIS_TKEEP(63:0) – Payload byte enables
- AXIS_TLAST – UDP payload frame delimiter
- AXIS_TID – Identify sender by Socket/Channel
- AXIS_TDEST – Routing identifier
- AXIS_TREADY/TVALID – Flow Control



Application Layer

- RTPv2 IEEE 3550 (2003)
- Lightweight/Proprietary

UDP + ARP Cache

- Map ADC/DAC Channels to UDP Sockets
- Map Sockets to ARP Cache Table

Flow Control Options

- L5 – Flow Control using SEQ/ACK
- L2 - Ethernet PAUSE frames

```python
#Sender

import socket
ip = "192.168.0.100"
port = 1000
msg = b"Hello World"

print(f'Sending {msg} to {ip}:{port}')
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(msg, (ip,port))
```

"Hello world"

```python
#Receiver

import socket
ip = "192.168.0.100"
port = 1000

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((ip,port))

print(f'Start Listening to {ip}:{port}')

while True:
    data,addr = sock.recvfrom(1024)
    print(f"received message {data}")
```

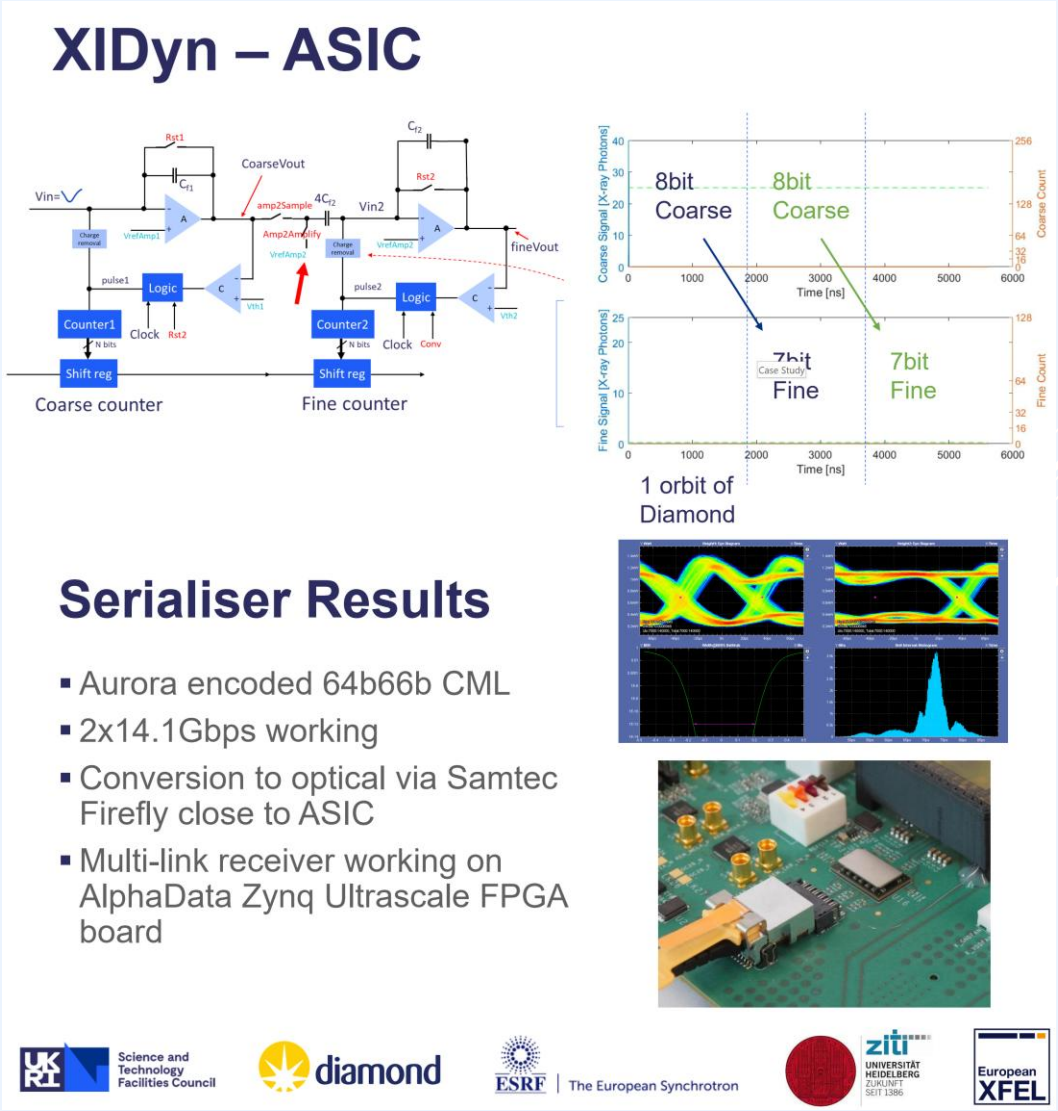
DATA ACQUISITION FROM ASIC

Aurora64b66b Serial Transceiver IP Core

- Point to point, Link Layer Protocol
- Compact IP Core – Hard Macro + logic
- Vendor provided (free of charge)
- Arbitrary data rate – limited only by MGT

GTY Transceivers

- Support max 26.5Gbps
- Drive optical /copper directly
- Four individual channels per GTY Quad
- Flexible reference clock



DATA ACQUISITION HARDWARE

Capture 24 channels of 2-20Gbps Aurora data

Alpha Data ADM-PCIE-9V5 1200k LUT FPGA

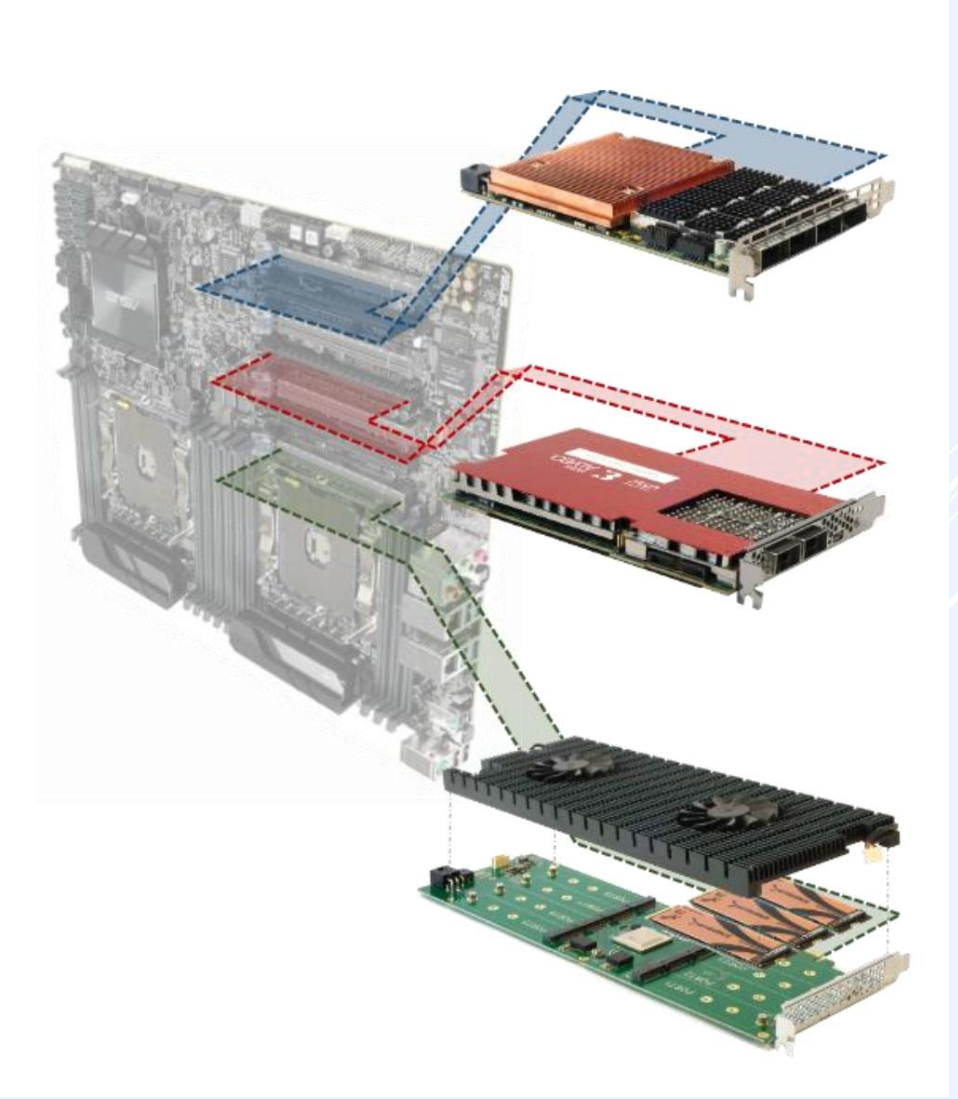
- AMD Ultrascale+ with 8x 100G MACs
- Full length- full height PCIe
- 4x QSFP-DD + Firefly - 900Gbps capacity

Data Processing

- Alveo - Processing & Compression

Data Storage

- NVMe - SSD



DESIGNING FOR MULTI-100'S GBIT/S

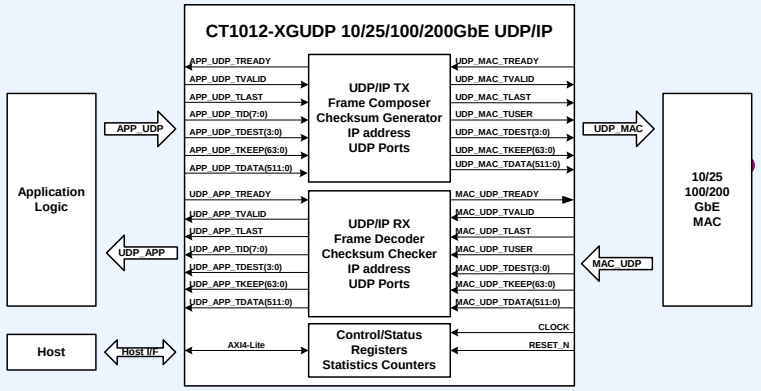
ASIC side - Aurora Transceivers (24x14Gbps)

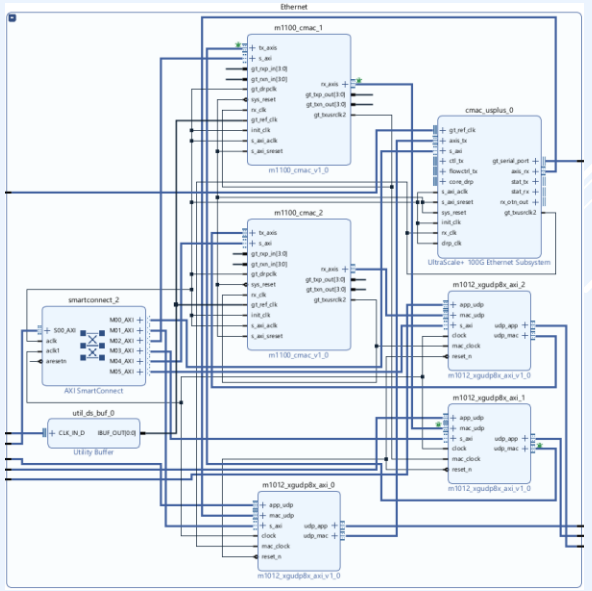
- Send/Receive – Programmable 3-20Gbps
- Link Layer Error detection – but no correction/resend
- Utilization 6000 LUTs / 4 BRAM

Ethernet side - UDP Offload Engine (3x100Gbps)

- Send / Receive @ 100Gbps using AMD CMAC hard macro
- Multiple UDP Sockets – arbitrary number of sockets/sessions
- Jumbo Packets MTU 9000 bytes
- IP Fragmentation – support UDP datagrams to 64k bytes
- Utilization 65k LUTs / 200BRAM (support max len recv packets)
- ARP resolution – MAC/IP
- ICMP - Ping

"Stream direct to Ethernet"





CHEVIN TECHNOLOGY

Meet the team

ChevinID - Silicon Security

Embedded in *Every* Chiplet

Uniquely identify hardware & authorize device features to boost your Silicon Supply Chain security

Svein-Egil Nielsen
Executive Advisor

David Harold
Executive Advisor
Fractional CMO

Steinn Gustafsson
Founder & CEO
BEng. Electronic Eng.

Karen Rogers
Co- Founder & COO
BA Business

David Marsden
Marketing &
Communications Advisor

Mark Champion
Defense Market Advisor

Chevin Technology
The Bradfield Centre
184 Cambridge Science Park
Milton Road,
Cambridge, CB4 0GA

www.chevintechnology.com
steinn@chevintechnology.com

A member of ADS

arm

CHERI Alliance Founder

ALPHA DATA

UKRI Science and Technology Facilities Council

SIEMENS

BittWare a molex company