

cesnet
"...."

NDK+OFM:

Rapid Development of Accelerated
Applications for FPGA SmartNICs

Jiří Matoušek, Daniel Kondys
CESNET

8 May 2023 (at FCCM 2023 in Los Angeles, CA, USA)

■ Jiří Matoušek

- researcher at CESNET
- assistant professor at FIT BUT (Brno, Czech Republic)



■ Daniel Kondys

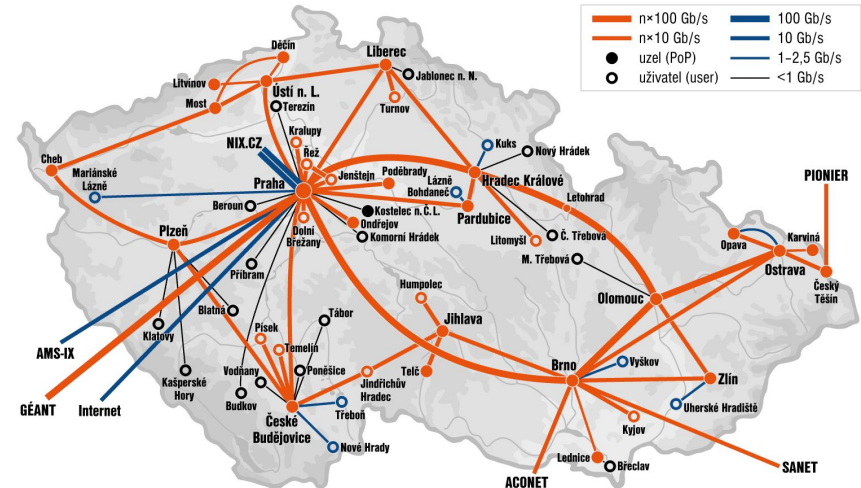
- developer in FPGA R&D team at CESNET

■ Jakub Cabal

- head of the FPGA R&D team at CESNET



- CESNET is an association of universities of the Czech Republic and the Czech Academy of Sciences
- Operates and develops the national e-infrastructure for science, research and education which encompasses
 - computer network
 - computational grids
 - data storage
 - collaborative environment



a.k.a. Security and Administration Tools Department at CESNET

- More than 60 people in R&D (about 28 FTE)
- Cooperation with universities (CTU, BUT) and students
- Applied research and tools based on unique technology
 - hardware acceleration
 - network monitoring & traffic analysis
 - Cyber Threat Intelligence
 - DDoS mitigation
 - configuration tools



- One of the first 100 Gb FPGA acceleration cards

- Czech Head Prize 2016 - Industrie award



- NETCONF tools awarded by ONF

- now used by telco operators in the network infrastructure

- P4 compiler for FPGAs acquired by Intel



- Flowmon Networks spin-off acquired by Kemp

- founded in 2007 based on the research results of Scampi and Geant projects (Netflow/IPFIX probes)



- DDoS mitigation system deployed by NIX



Flowmon
A Kemp Company




- What is your experience with FPGA SmartNICs?
- Why did you register for this tutorial?
- What are your expectations?
- What would you like to take away?



- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal line at the bottom of the slide, consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.

The logo for cesnet, featuring the word "cesnet" in a white, lowercase, sans-serif font. Below the text is a graphic element consisting of a series of white dots of varying sizes arranged in a pattern that suggests a network or data flow.

cesnet
"...."

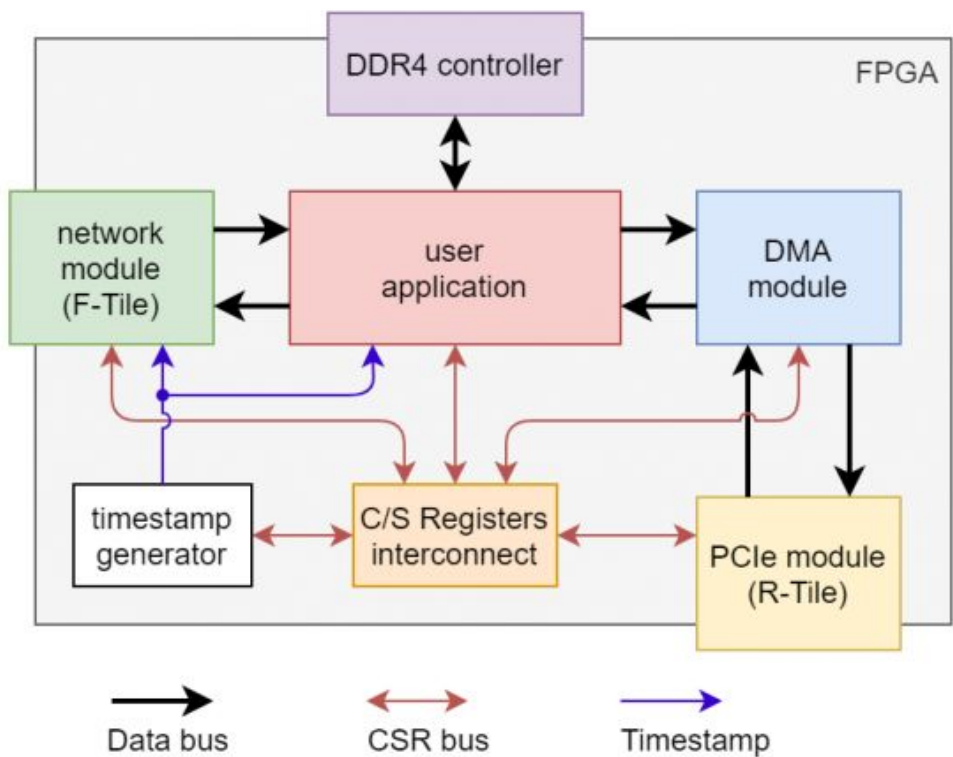
NDK Introduction

[presentation]

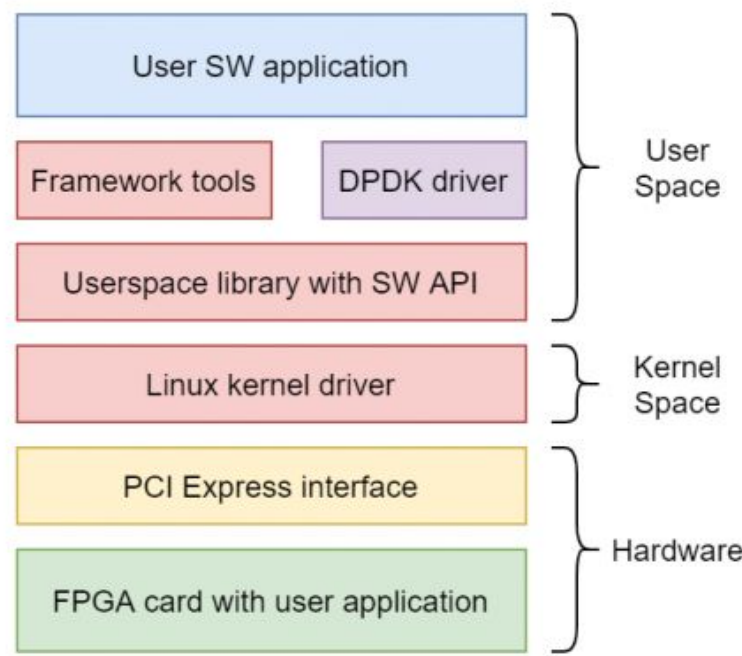
- Growing number of available/deployed FPGA SmartNICs
 - [Intel](#), [AMD/Xilinx](#), [Cisco](#), [Napatech](#), [Silicom](#), [BittWare](#), etc.
- Need for rapid development of applications targeting these devices
- Common low-level operations on FPGA SmartNICs
 - transmitting/receiving data via network interface (typically Ethernet)
 - transmitting/receiving data via host interface (typically PCIe)
 - writing/reading data via external memory interface (typically DDR)
- Implementation of low-level operations is difficult for users

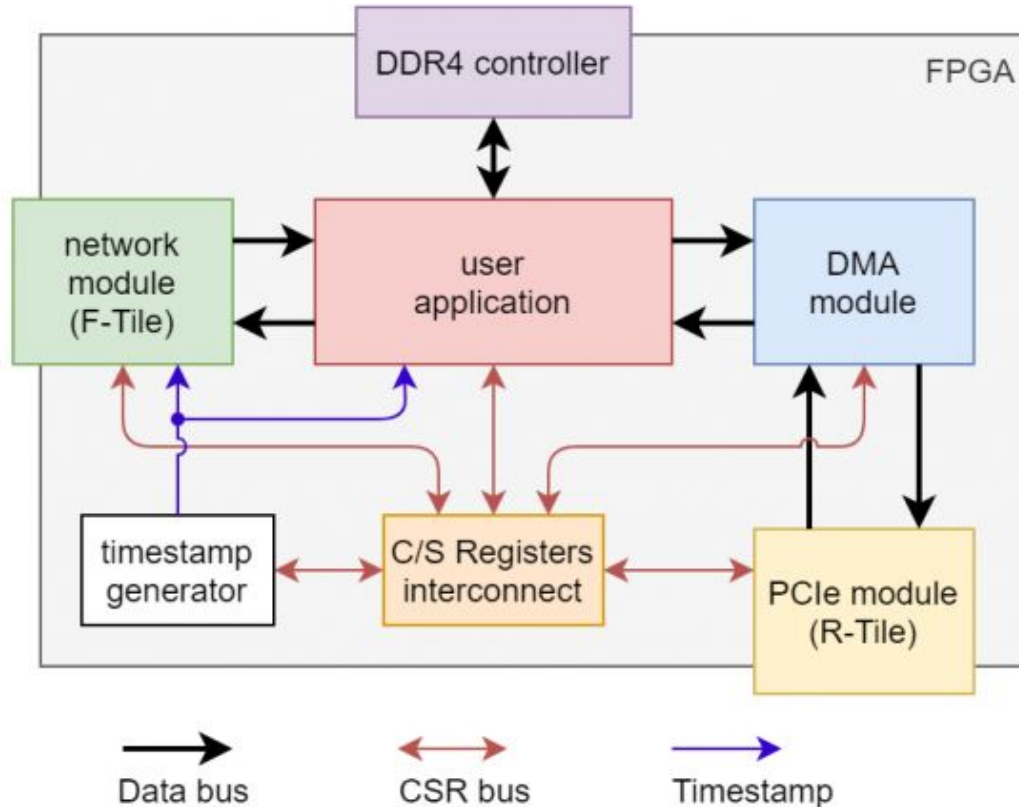
- [Network Development Kit \(NDK\)](#)
 - open-source framework for fast and easy development of accelerated applications for FPGA SmartNICs
 - based on 20+ years of active research and development at CESNET
- **NDK key features**
 - support for high-speed network interfaces (up to 400 GbE)
 - support for high-speed host interfaces (up to PCIe gen5 x16)
 - ready-to-use software stack (driver, library, tools) and API (C, Python)
 - Data Plane Development Kit (DPDK) support
 - single `make` command to create the entire FPGA bitstream

FPGA design architecture



Software stack





■ FPGA architecture modules

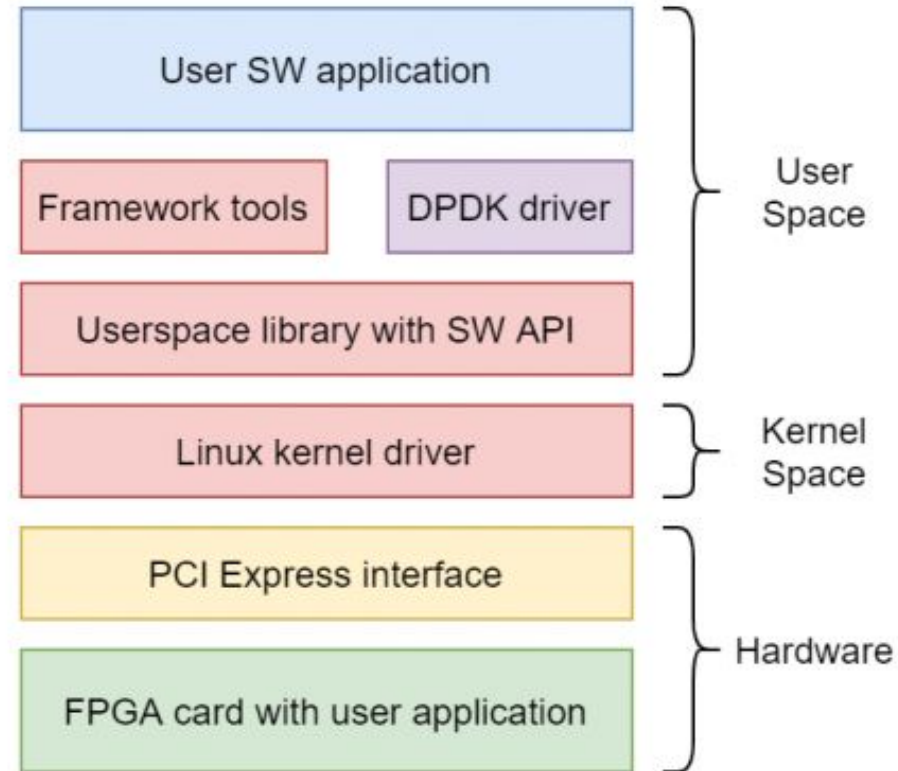
- user application (e.g., minimal) (*not part of NDK*)
- network module (e.g., F-Tile)
- DMA module (e.g., Medusa) (*not open-source*)
- PCIe module (e.g., R-Tile)
- memory controller (e.g., DDR4)
- timestamp generator
- C/S registers interconnect

- NDK FPGA design architecture is divided into
 - common part same for all supported FPGA cards ([ndk-core](#))
 - specific part for each supported FPGA card
- Supported FPGA cards
 - ReflexCES XpressSX AGI-FH400G ([ndk-card-agi-fh400g](#))
 - Intel Stratix 10 DX FPGA Development Kit ([ndk-card-dk-dev-1sdx-p](#))
 - Intel Agilex I-Series FPGA Development Kit ([ndk-card-dk-dev-agi027res](#))
 - Silicom fb4CGg3@VU9P and fb2CGg3@VU9P (both in [ndk-card-fb4cgg3](#))
 - Silicom fb2CGhh@KU15P ([ndk-card-fb2cghh](#))
 - Bittware IA-420F card ([ndk-card-ia-420f](#))

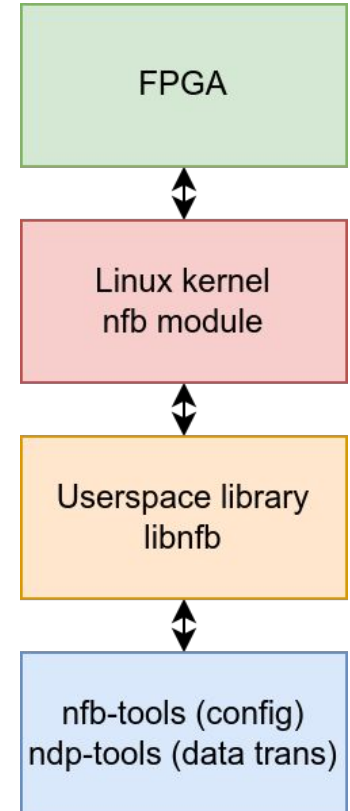
- Various basic modules for FPGA designs ([ofm](#))
 - see [OFM Introduction](#) part of this tutorial
- Minimal/reference NDK application ([ndk-app-minimal](#))
 - see [ndk-app-minimal Introduction](#) part of this tutorial
- Build system for easy simulation/verification, synthesis, implementation, and bitstream generation ([ofm/build](#))
 - see [Advanced Topics](#) part of this tutorial

■ Software stack components

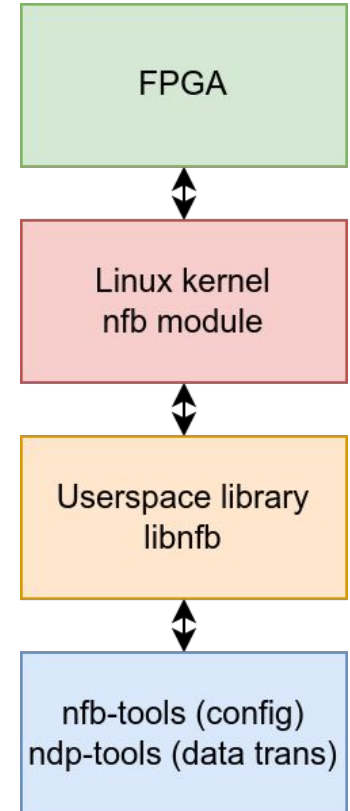
- Linux kernel driver
(nfb = NDK FPGA board)
- userspace library with C/Python API
(libnfb)
- NDK SW tools
- DPDK driver
- user application *(not part of NDK)*



- [nfb-tools](#) (nfb = NDK FPGA board)
- Tools to read/write configuration and status data via CSR bus
 - nfb-info – basic info about card and current design
 - nfb-boot – load bitstream into card
 - nfb-eth – configure/read status of network module
 - nfb-dma – configure/read status of DMA module
 - nfb-tsu – configure timestamp generator
 - nfb-bus – configure/read status of arbitrary register



- [ndp-tools](#) (ndp = NDK Data Plane)
- Tools to read/write packet data using DMA
 - ndp-read – read packets from FPGA
 - ndp-generate – send generated packets to FPGA
 - ndp-receive – read packets from FPGA into PCAP
 - ndp-transmit – send packets from PCAP to FPGA
 - ndp-loopback – read packets from FPGA and send them back



- RPM packages available via [COPR](#)

```
sudo dnf copr enable @CESNET/nfb-framework
```

```
sudo dnf install nfb-framework python3-nfb
```

- nfb driver
 - libnfb library (including C and Python API)
 - both nfb-* and ndp-* tools
- NDK software can also be built from sources ([ndk-sw](#))
 - follow [Build instructions](#) in README.md
 - DPDK driver available via mainline [DPDK repository](#)

- Design in FPGA is characterized using [DeviceTree](#) (DT)
 - build system composes DT string (dts) and translates it to DT blob (dtb), which is stored in the design and can be read from software
- DTS example

```
ref_name: my_comp {  
    reg = < $BASE_ADDRESS 0x40 >;  
    compatible = "netcope,my_comp";  
    version = < 0x00010004 >;  
    type = "reduced";  
};
```


- Precise network monitoring
 - flow monitoring with deep packet inspection
- Network security applications
 - IDS/IPS acceleration - Suricata pre-filter and bypass
 - Anti DDoS - mitigation of volumetric attacks
- High-frequency trading
 - algorithmic trading with very low response delay
- And many others...



- J. Cabal, J. Sikora, Š. Friedl, M. Špinler, and J. Kořenek, "[FPL Demo: 400G FPGA Packet Capture Based on Network Development Kit](#)," FPL 2022, pp. 474-474, doi: 10.1109/FPL57034.2022.00090.
- J. Kubálek, J. Cabal, M. Špinler, and R. Iša, "[DMA Medusa: A Vendor-Independent FPGA-Based Architecture for 400 Gbps DMA Transfers](#)," FCCM 2021, pp. 258-258, doi: 10.1109/FCCM51124.2021.00045.
- L. Kekely, J. Cabal, V. Puš and J. Kořenek, "[Multi Buses: Theory and Practical Considerations of Data Bus Width Scaling in FPGAs](#)," DSD 2020, pp. 49-56, doi: 10.1109/DSD51259.2020.00020.

- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal line at the bottom of the slide consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.

The logo for cesnet, featuring the word "cesnet" in a white, lowercase, sans-serif font. Below the text is a stylized graphic consisting of a series of white dots of varying sizes arranged in a pattern that suggests a network or data flow.

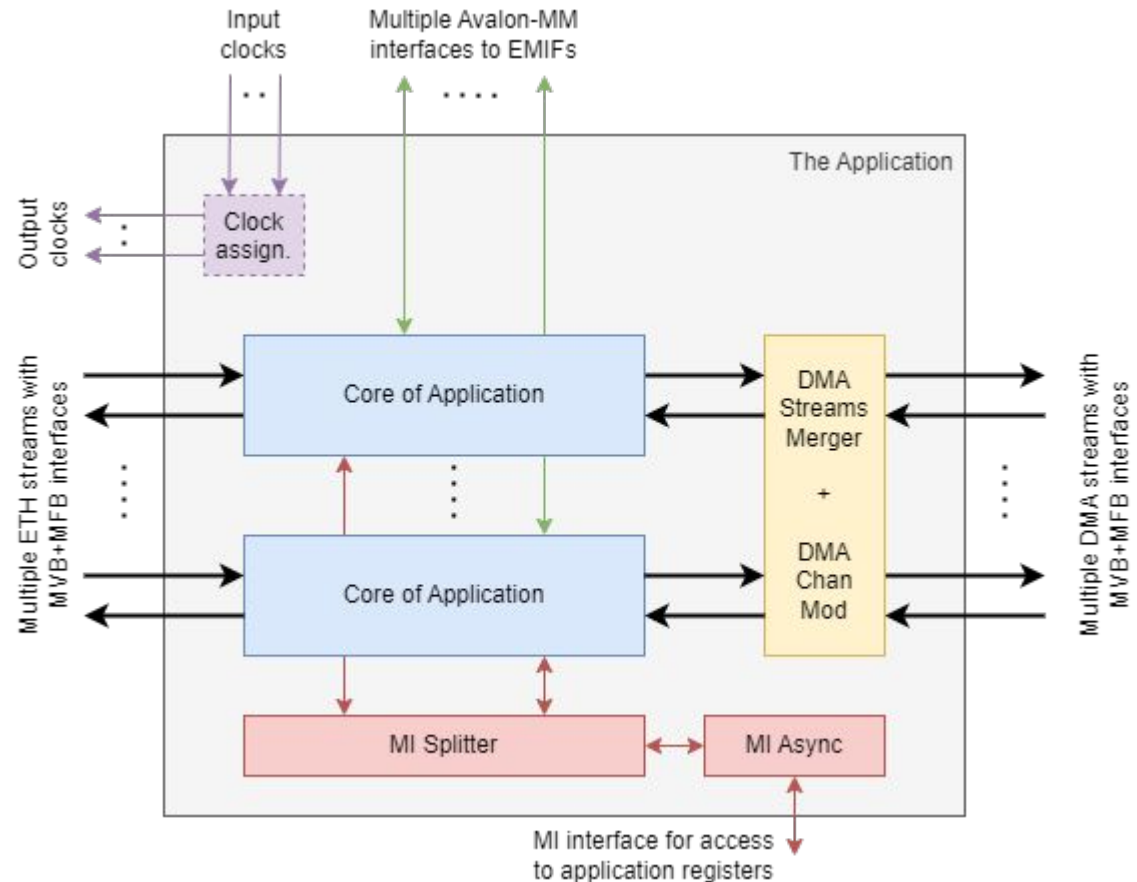
cesnet
"...."

ndk-app-minimal Introduction

[presentation]

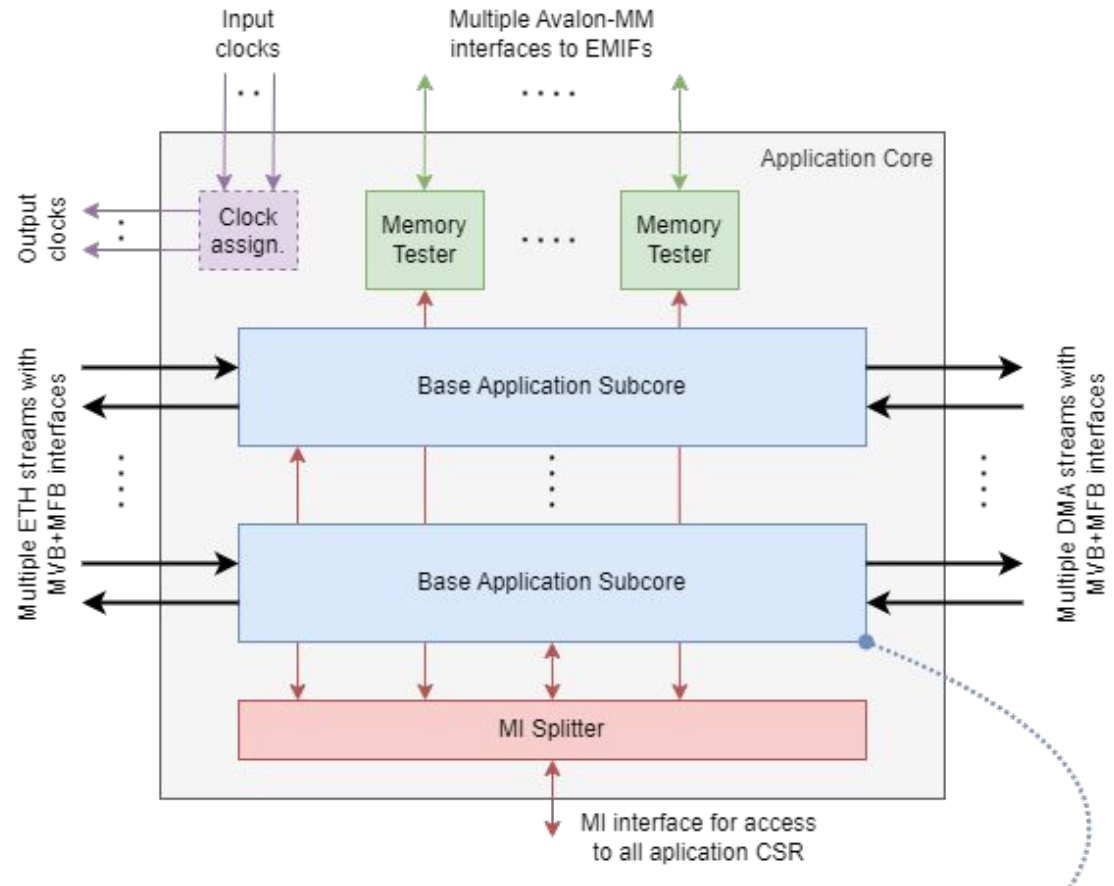
- [ndk-app-minimal](#) represents simple example of NDK-based FPGA application
 - possible starting point for user-defined FPGA applications based on NDK
- Does not process network packets, just routes them through application core
- Documentation
 - [The Application](#) section within NDK Architecture
 - [Minimal NDK application](#)

- ETH streams
 - MVB+MFB interfaces
- DMA streams
 - MVB+MFB interfaces
- APP registers interface
 - MI interface
- EMIF interfaces
 - Avalon-MM interfaces
- Clock INs and OUTs



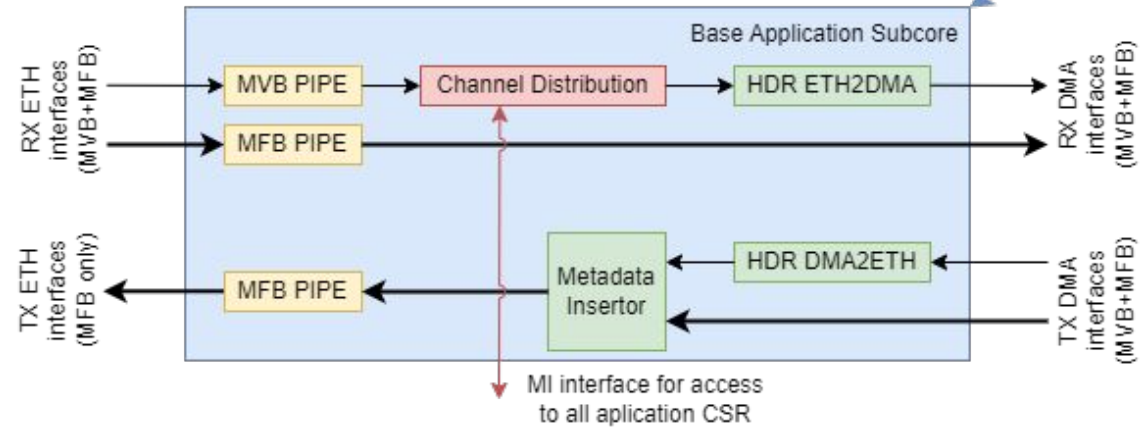
- The application interfaces
 - details of data format can be found in [documentation](#)
 - proprietary interfaces MVB, MFB, and MI will be introduced in [OFM Introduction](#) part of this tutorial
- DMA Streams Merger + DMA Chan Mod
 - deal with situation when ETH streams \neq DMA streams
- MI Async
 - clock-domain-crossing unit for MI interface

- Single Base Application Subcore per ETH stream
- Single Memory Tester per EMIF interface
- Separate MI interface for each
 - Base Application Subcore
 - Memory Tester



- MFB only TX ETH interfaces

- other interfaces utilize MVB+MFB



- TX direction channels (DMA→ETH)

- Static mapping


- RX direction channels (ETH→DMA)

- Dynamic mapping (configurable by user)

- [ndk-app-minimal](#) integrates all other NDK repositories
 - app/ implementation of application
 - build/ build scripts for supported FPGA cards
 - conf/ build system configuration
 - doc/ documentation system + top-level documentation
 - ndk/ NDK platform sources (multiple repositories)
 - cards/ top-level designs for supported FPGA cards (e.g., ndk-card-agi-fh400g)
 - core/ common NDK core (ndk-core)
 - modules/ special modules and IPs (e.g., ndk-mod-dma-medusa)
 - ofm/ Open FPGA Modules - basic components and build system (ofm)
 - tests/ test scripts for Jenkins, etc.
- note that NDK cannot be built without application core

- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal line at the bottom of the slide consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.

The logo for cesnet, consisting of the word "cesnet" in a white, lowercase, sans-serif font, with a stylized graphic of four small white squares arranged in a row below it.

cesnet
"...."

Playing with ndk-app-minimal

[hands-on]

1. Start provided VirtualBox VM
2. Open terminal
3. Connect via SSH to given "preklad" server using given credentials
 - `ssh tutorial{1-5}@preklad{11-13, 21-24}.liberouter.org`

4. Read through "[How to start](#)" section of ndk-app-minimal README.md
5. Clone [ndk-app-minimal](#) repository with at least [ofm](#), [core](#), and [agi-fh400g](#) submodules
 - `git clone https://github.com/CESNET/ndk-app-minimal.git`
 - `cd ndk-app-minimal`
 - `git submodule update --init ndk/ofm`
 - `git submodule update --init ndk/core`
 - `git submodule update --init ndk/cards/agi-fh400g`

6. Read through "[How to start](#)" section of ndk-app-minimal documentation
7. Build default ndk-app-minimal design for agi-fh400g card
 - `cd build/agi-fh400g`
 - `make`
8. In the meantime, study details of Gen Loop Switch module (next slide) and read through "[NDK testing](#)" section of ndk-app-minimal documentation

- Module for easy debugging and measurements of FPGA designs
 - [documentation](#) available for instance as part of ndk-app-minimal documentation
- Basic features
 - two configurable packet generation modules
 - two configurable loopback paths
 - four units for throughput measurements
- By default, GLS module is enabled in ndk-app-minimal
- GLS operation can be controlled using available [Python script](#)

9. Once the demo server Solaris is available, ask Daniel Kondys for supervision
10. Use nfb-boot tool to load default ndk-app-minimal design into 400G FPGA card in Solaris server
 - `nfb-boot -f0 your_ndk_firmware.nfw`
11. Use nfb-info tool for verification of successful design booting
 - `nfb-info`

12. Test R/W access to the scratch registers

- `nfb-bus -p /firmware/mi_bus0/mi_test_space <address>`
- `nfb-bus -p /firmware/mi_bus0/mi_test_space <address> <data>`
- `nfb-bus -p /firmware/mi_bus0/mi_test_space <address>`

13. Enable Network Module in "PMA local loopback mode"

- `nfb-eth -Pc "+PMA local loopback"`
- `nfb-eth -e1`


14. Measure ndk-app-minimal throughput using GLS module

- `python3 gls_mod.py 1`



- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal line at the bottom of the slide consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.


cesnet
"...."

[lunch break]



- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal line at the bottom of the slide consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.

The logo for cesnet, featuring the word "cesnet" in a white, lowercase, sans-serif font. Below the text is a graphic element consisting of a series of white dots of varying sizes arranged in a pattern that suggests a network or data flow.

cesnet
"...."

OFM Introduction

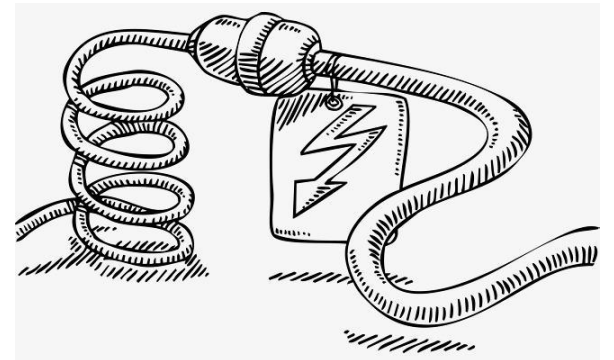
[presentation]

- Open FPGA Modules (OFM)
 - open-source library of basic FPGA modules for (not only) high-speed processing of data
 - heavily utilized in NDK and ndk-app-minimal
 - based on 20+ years of active research and development at CESNET
- Most common interfaces of modules within OFM repository
 - multi buses (MFB, MVB)
 - memory interface bus (MI)

- New generation of highly parametric proprietary buses
- Key benefit: support for multiple packets per clock cycle
 - necessary for processing 400G Ethernet data streams in FPGA
- Two different buses that are often used together
 - Multi-Frame Bus (MFB): designed for transfers of packet data
 - Multi-Value Bus (MVB): designed for transfers of (packet) meta-data
- Available documentation introduces MFB and MVB very well
 - [MFB specification](#)
 - [MVB specification](#)

- Memory Interface Bus (MI bus)
 - low-performance bus for SW access to control/status registers and memories
 - sometimes mentioned as MI32 bus (due to its default data width of 32 bits)
- Available documentation introduces MI bus very well
 - [MI bus specification](#)

- set of basic components for use with common buses
 - implementation: [comp/mfb tools](#), [comp/mvb tools](#), [comp/mi tools](#), etc.
 - documentation: [MFB Tools](#), [MVB Tools](#), [MI Tools](#), etc.
 - not all components are documented!
- similar components stored together within each group
 - data flow transformations (flow)
 - data storage (storage)
 - debugging (debug)
 - etc.




- set of basic components for general use
 - implementation: [comp/base](#)
 - documentation: [Basic Tools](#) (not all components are documented!)
 - some components are optimized for specific FPGA architecture(s)
- similar components stored together
 - combinatorial logic (logic)
 - shift registers (shreg)
 - general (mem) and FIFO (fifo) memories
 - etc.



- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

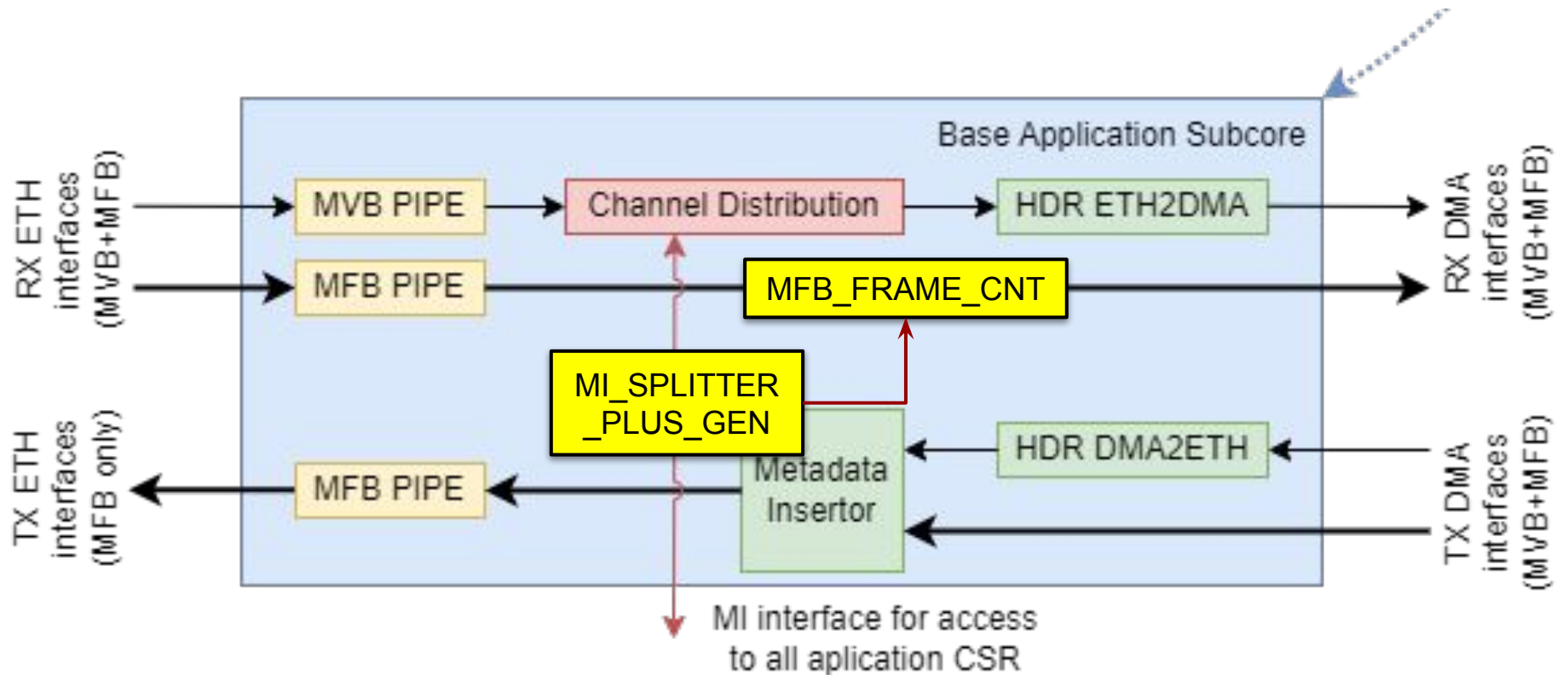
 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal bar at the bottom of the slide, consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.



Implementing Custom NDK Application


[hands-on]



- Extend ndk-app-minimal with 64-bit counter of RX MFB frames
 - use MFB_FRAME_CNT (not documented) from ofm/mfb_tools directory
- Make the counter value accessible via MI bus at address offset 0x100 from Base Application Subcore base address
 - use MI_SPLITTER_PLUS_GEN to split address space of Base Application Subcore between MVB_CHANNEL_ROUTER_MI and MFB_FRAME_CNT
- Make sure to edit all relevant files
 - app/top/app_subcore.vhd
 - app/top/Modules.tcl
 - app/top/DevTree.tcl

- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal bar at the bottom of the slide, consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.



cesnet
"...."

SW API Introduction

[presentation]



- NDK userspace library provides two APIs for user applications
 - C API: native libnfb API
 - Python API: Cython-based libnfb API
- Typical use cases of C API
 - applications with high-performance requirements
 - NDK SW tools (both nfb-tools and ndp-tools)
- Typical use cases of Python API
 - majority of user applications
 - testing/debugging scripts

- Comprises two sets of API functions
 - [Base API](#): basic operations over NFB (init/deinit, registers read/write, etc.)
 - [NDP API](#): manipulation of high-speed queues for packet transmissions
- Further materials
 - [quick start guides](#) as well as [complete examples](#) for both Base and NDP API
 - example use of Base and NDP API in [nfb-tools](#) and [ndp-tools](#), respectively

■ Base API example

```
#include <nfb/nfb.h>

int main(int argc, char *argv[]) {
    struct nfb_device *dev = nfb_open("0");

    int node = nfb_comp_find(dev, "netcope,rxmac", 0);
    struct nfb_comp *comp = nfb_comp_open(dev, node);

    const int RXMAC_EN = 0x20;
    int en = nfb_comp_read32(comp, RXMAC_EN) & 0x01;
    if (!en)
        nfb_comp_write32(comp, RXMAC_EN, 0x01);

    nfb_comp_close(comp);
    nfb_close(dev);
    return 0;
}
```

■ NDP API example

```
#include <nfb/nfb.h>
#include <nfb/ndp.h>

int main(int argc, char *argv[]) {
    struct nfb_device *dev = nfb_open("0");

    struct ndp_queue *txq = ndp_open_tx_queue(dev, 0);
    ndp_queue_start(txq);

    struct ndp_packet packet = {.data_length = 128,
                                .header_length = 16};

    ndp_tx_burst_get(txq, &packet, 1);
    packet.data[14] = 0x08;
    ndp_tx_burst_flush(txq);

    ndp_close_tx_queue(txq);
    nfb_close(dev);
    return 0;
}
```

- Work-in progress, but already useable for real applications
 - basic operations over NFB
 - manipulation of high-speed queues for packet transmissions
 - ethernet port manipulation
- Aims to provide at least the same functionality as C API
- Documentation is to be published in the near future, however
 - [complete examples](#) are already available within ndk-sw repository
 - OFM contains handful or real use cases: [Memory Tester](#), [Rate Limiter](#), etc.

■ Basic example

```
import nfb

dev = nfb.open()

node = dev.fdt_get_compatible("netcope,eth")[0]
phandle = node.get_property("rxmac").value
node = dev.fdt_get_phandle(phandle)

comp = dev.comp_open(node)

RXMAC_EN = 0x20
en = comp.read(RXMAC_EN) & 0x01
if (not en):
    comp.write(RXMAC_EN, 0x01)
```

■ Data transfer example

```
import nfb


dev = nfb.open()
ndp = dev.ndp

txq = ndp.tx[0]
txq.start()

pkt = bytes([0]*128)
hdr = bytes([0]*16)
txq.send([pkt], hdrs=hdr)
```

- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal bar at the bottom of the slide, consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.



Controlling NDK Application from SW

[hands-on]

- Test extended ndk-app-minimal using GLS module in mode 1
 - use design from "[Implementing Custom NDK Application](#)" hands-on session
 - follow the same steps as in "[Playing with ndk-app-minimal](#)" hands-on session
- Read out the number of packets generated in GLS module
 - `nfb-bus -p /firmware/mi_bus0/dbg_gls0/mfb_gen2eth 0x20`
- Read out the number of packets observed by counter of RX MFB frames in Base Application Subcore
 - `nfb-bus -p /firmware/mi_bus0/application/app_core_minimal_0/<name> 0x0`
- Compare numbers read out from GLS module and RX MFB frame counter




- Implement Python script that performs the same operations as specified on the previous slide
- Useful Python API examples
 - [01-basics.py](#)
 - [03-eth.py](#)
- Not all operations can be done via Python API
 - where necessary, use module [os](#) allowing to execute command in subshell

```
import os
os.system(<command>)
```

- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal bar at the bottom of the slide, consisting of a series of small blue squares of varying heights and positions, creating a pixelated or digital effect.

The logo for cesnet, featuring the word "cesnet" in a white, lowercase, sans-serif font. Below the text is a graphic element consisting of a series of white dots of varying sizes arranged in a pattern that suggests a network or data flow.

cesnet
"...."

Advanced Topics

[presentation]


- Set of scripts for simple simulation, verification, synthesis, and implementation of individual modules as well as full applications
 - Tcl scripts for controlling EDA tools
 - Makefile scripts for tasks automation
- Available within OFM repository
 - [implementation](#)
 - [documentation](#)
- Key elements from user's perspective
 - definition of module's sources within [Modules.tcl](#) file
 - scripts for individual modules/applications (directories sim, uvm, synth, etc.)

- Simulation based on advanced testing techniques
 - constrained pseudo-random generation of test vectors
 - analysis of functional coverage
 - invariants definition and checking
 - scoreboarding
- OFM provides infrastructure for functional verification according to UVM (Universal Verification Methodology)
 - [implementation](#)
 - [documentation](#)

- Documentation of NDK platform and OFM modules is compiled from sources using [Sphinx](#)
- VHDL sources are compiled using Sphinx-vhdl extension
 - [implementation](#)
 - [documentation](#)
- Two modes of documentation building
 - automatic (default)
 - manual

- [presentation] NDK Introduction
 - [presentation] ndk-app-minimal Introduction
 - [hands-on] Playing with ndk-app-minimal

 - [lunch break]

 - [presentation] OFM Introduction
 - [hands-on] Implementing Custom NDK Application
 - [presentation] SW API Introduction
 - [hands-on] Controlling NDK Application from SW
 - [presentation] Advanced Topics
- 
- A decorative horizontal bar at the bottom of the slide, consisting of a series of small blue squares of varying heights and widths, creating a pixelated or digital effect.

cesnet
"...."

[Q&A session]



cesnet
"...."

THANK YOU FOR YOUR ATTENTION