Distributed FPGA for enhanced Image Processing

Bildverarbeitung mit Vivado HLS

Noah Hütter & Jan Stocker

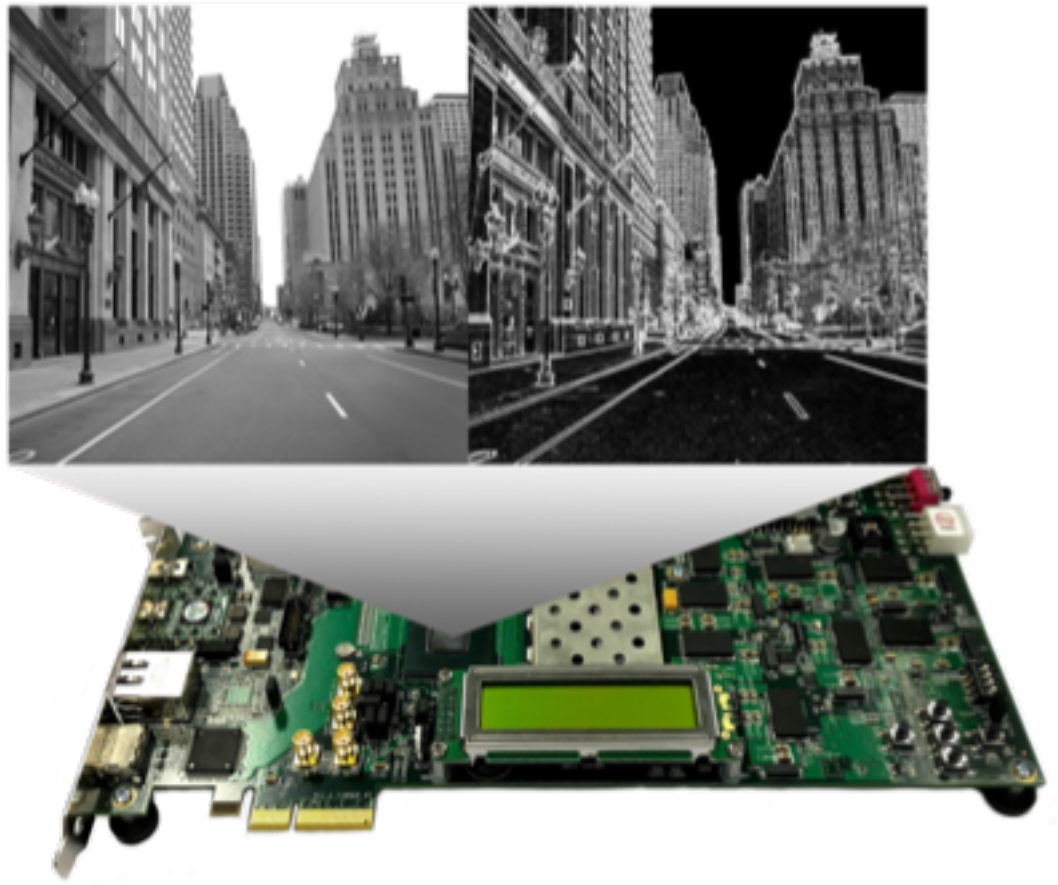
Inhaltsverzeichnis

- Projekt
- Vivado HLS Überblick
- Datenverarbeitung mit Vivado HLS
- Fazit

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

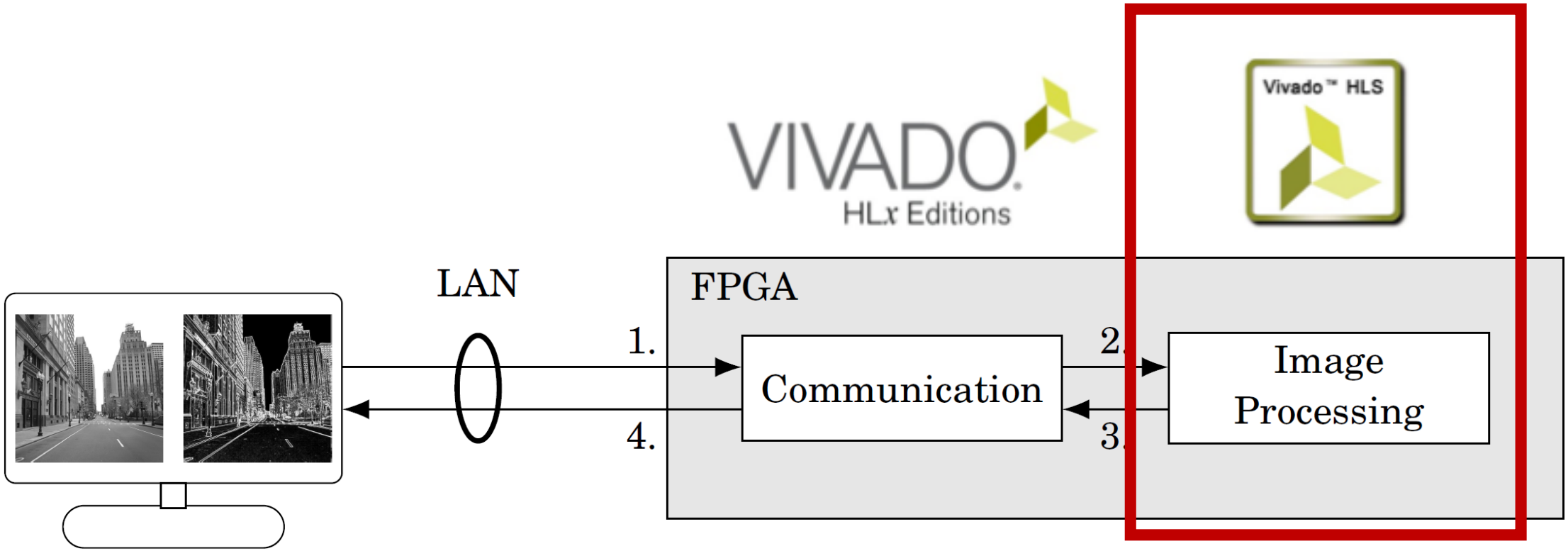
Ausgangslage

- 1500 MP Bilder
- Preprocessing
- Distributed FPGA Lösung



University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Aufbau



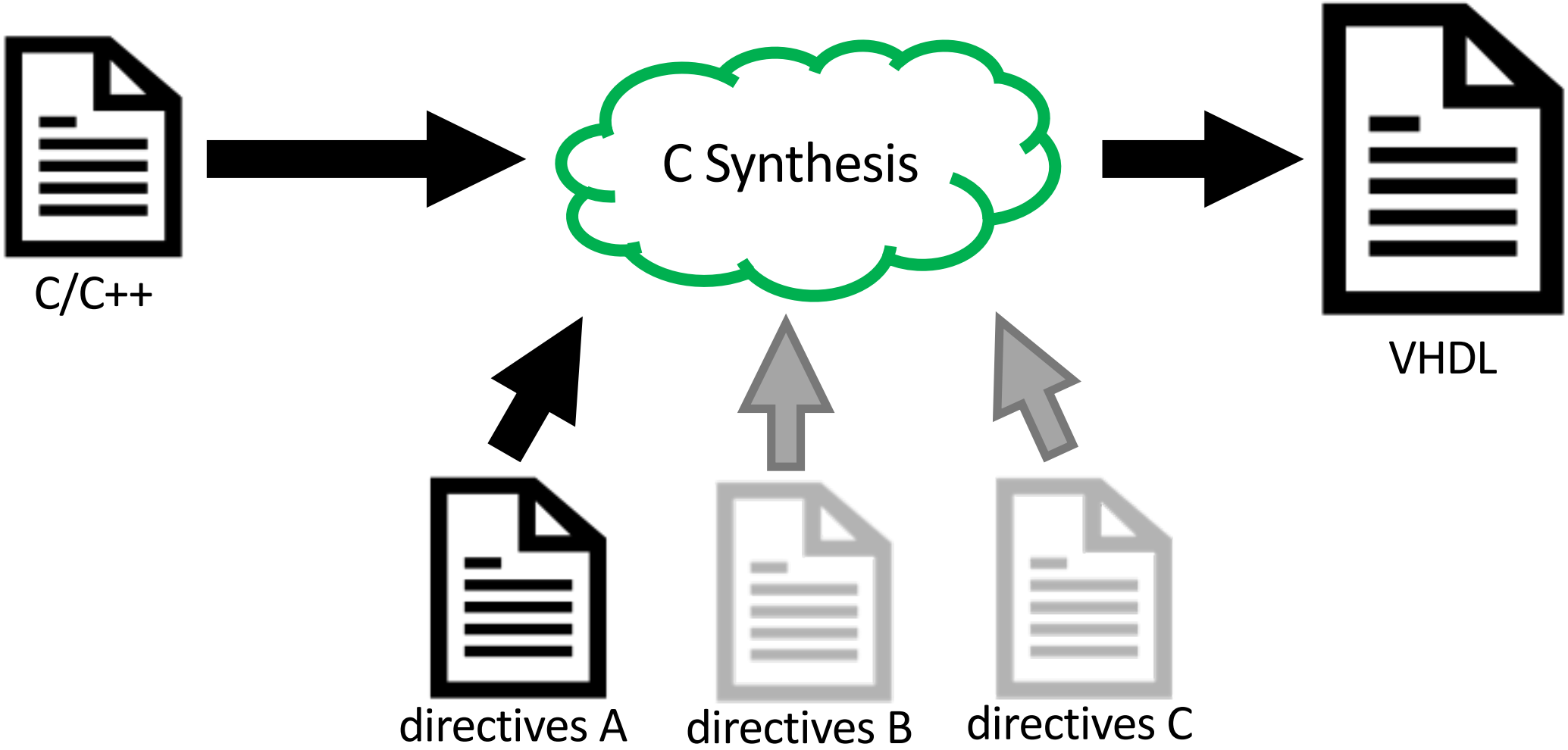
University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Vivado HLS: Übersicht

- Konzept: C/C++ als Sprache für Hardwarebeschreibung
- Design Flow
- Erweiterungen in C/C++
- GUI
- Pragmas

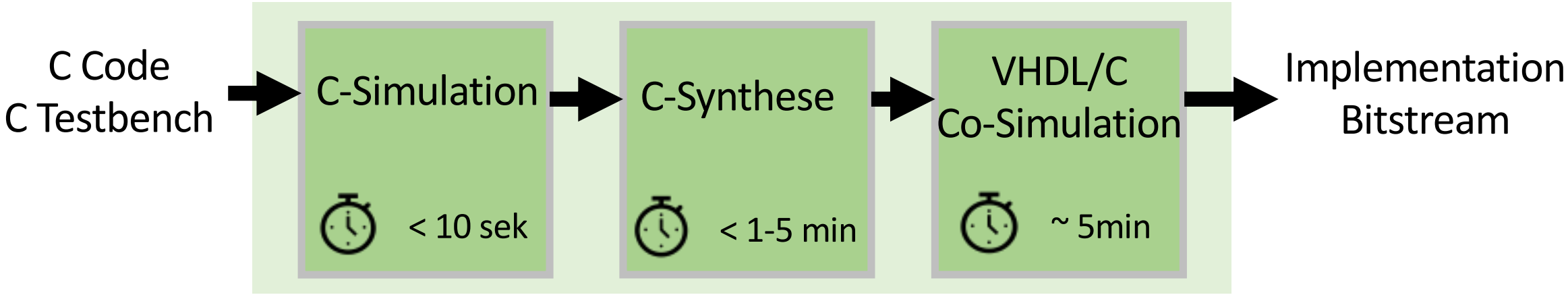
University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Konzept



University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Design Flow



C-Simulation	Schneller Bearbeiten/Kompilieren/Testen Kreislauf
C-Synthese	C Code wird in VHDL Code übersetzt
VHDL/C Co-Simulation	VHDL Code wird simuliert und die Resultate mit der C-Simulation verglichen

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Erweiterungen in C/C++

Arbiträre Datentypen

```
ap_uint<12>
```

C++ Klassen

```
hls::stream
```

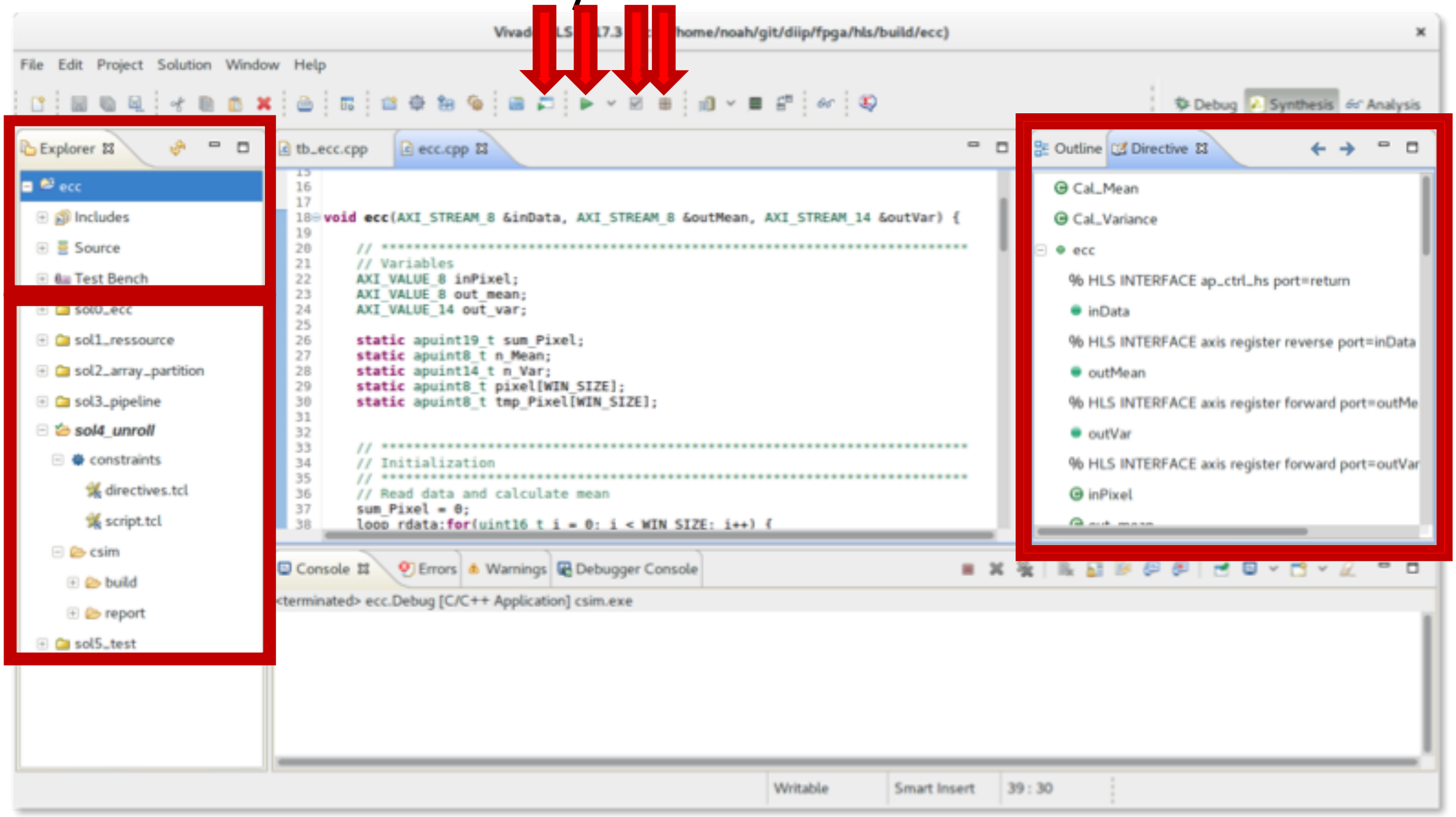
Directives (Pragmas)

- Definition der Schnittstellen
- Loop unrolling/ pipelining
- Array partitioning/reshaping
- Dataflow
- Resource control

→ Deren Verständnis ist der Schlüssel zum Erfolg mit Vivado HLS

GUI: Synthese

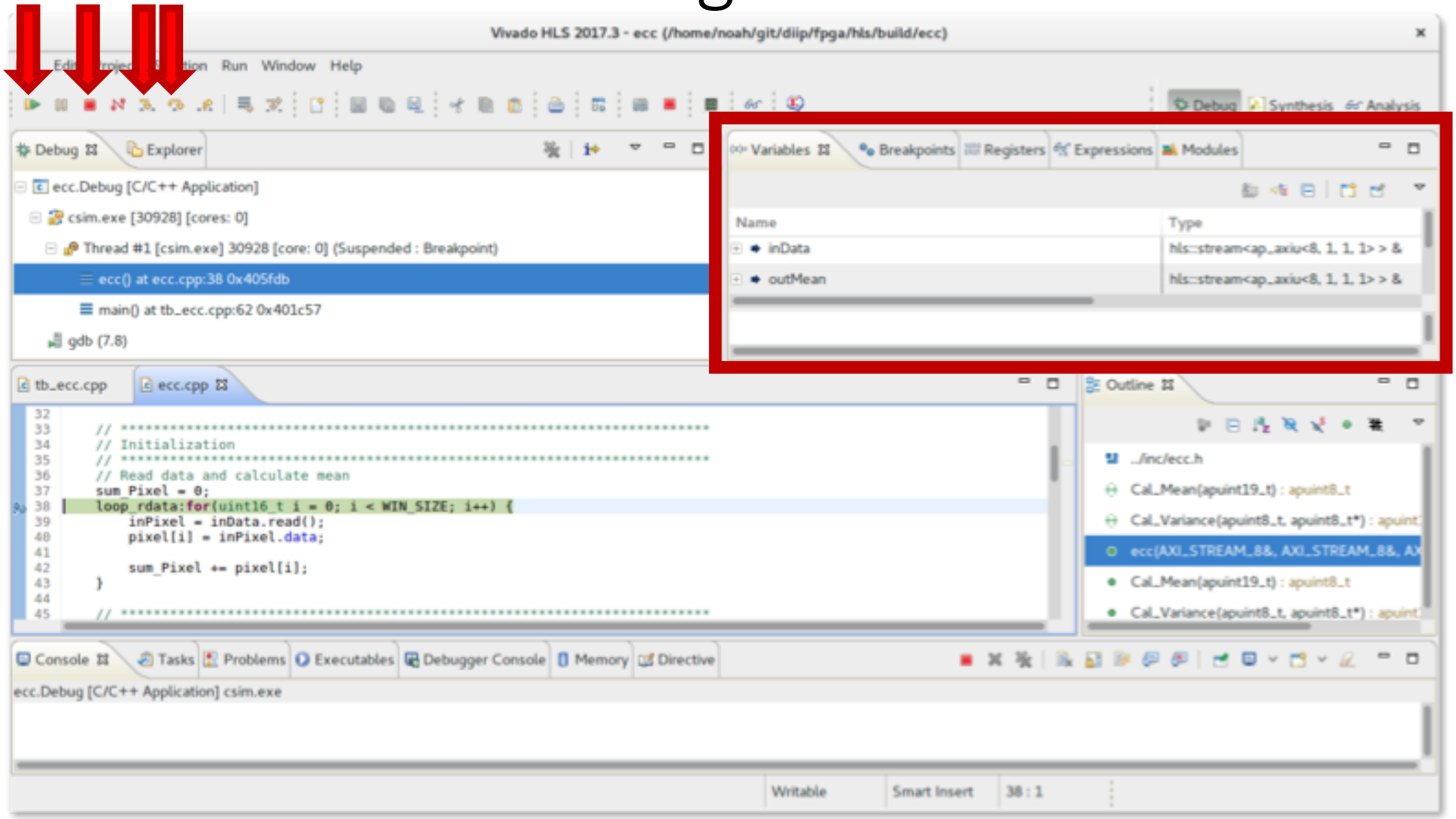
University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering



Institute of Microelectronics

GUI: Debug

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

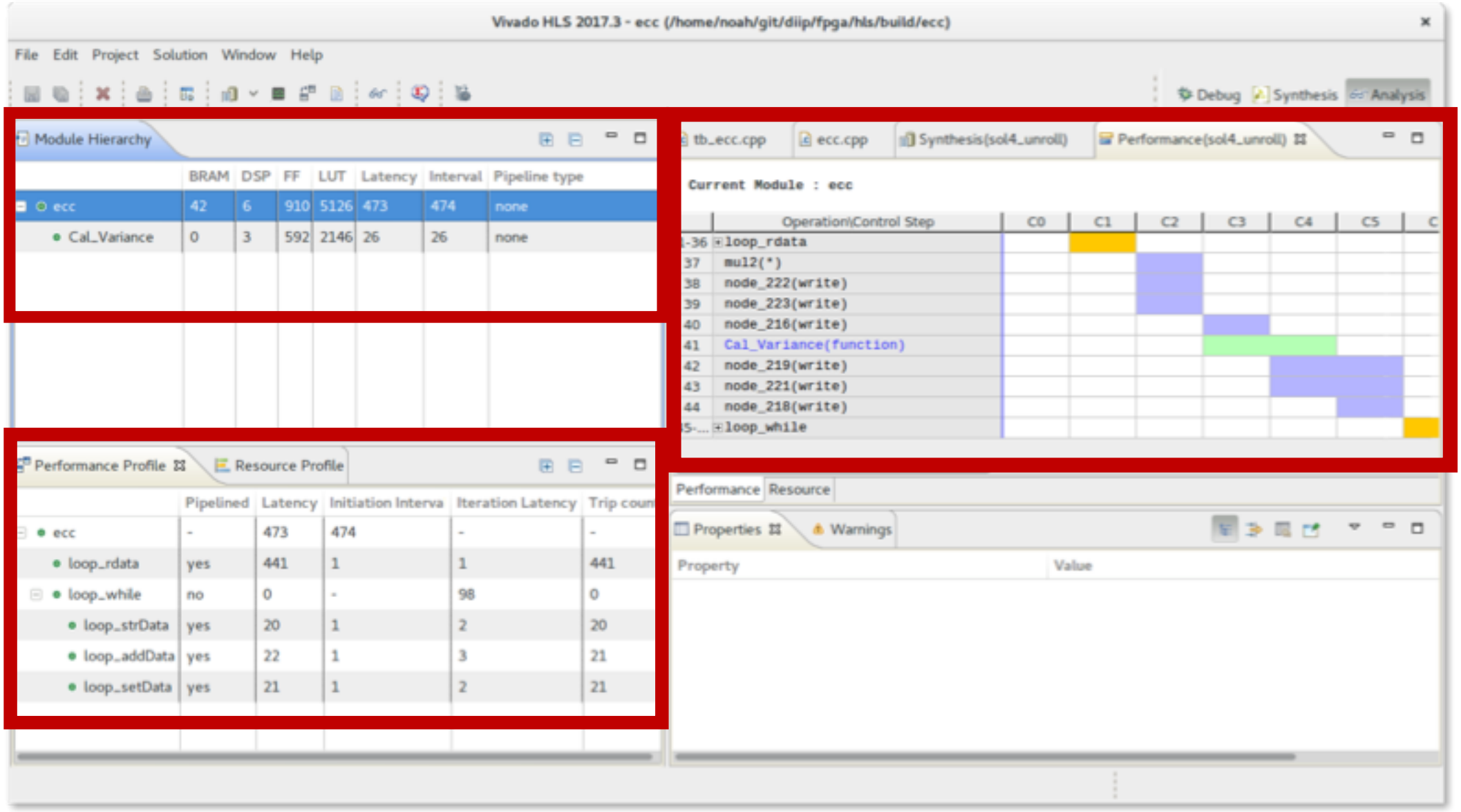


Noah Hütter & Jan Stocker

ime Institute of Microelectronics

GUI: Analyse

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

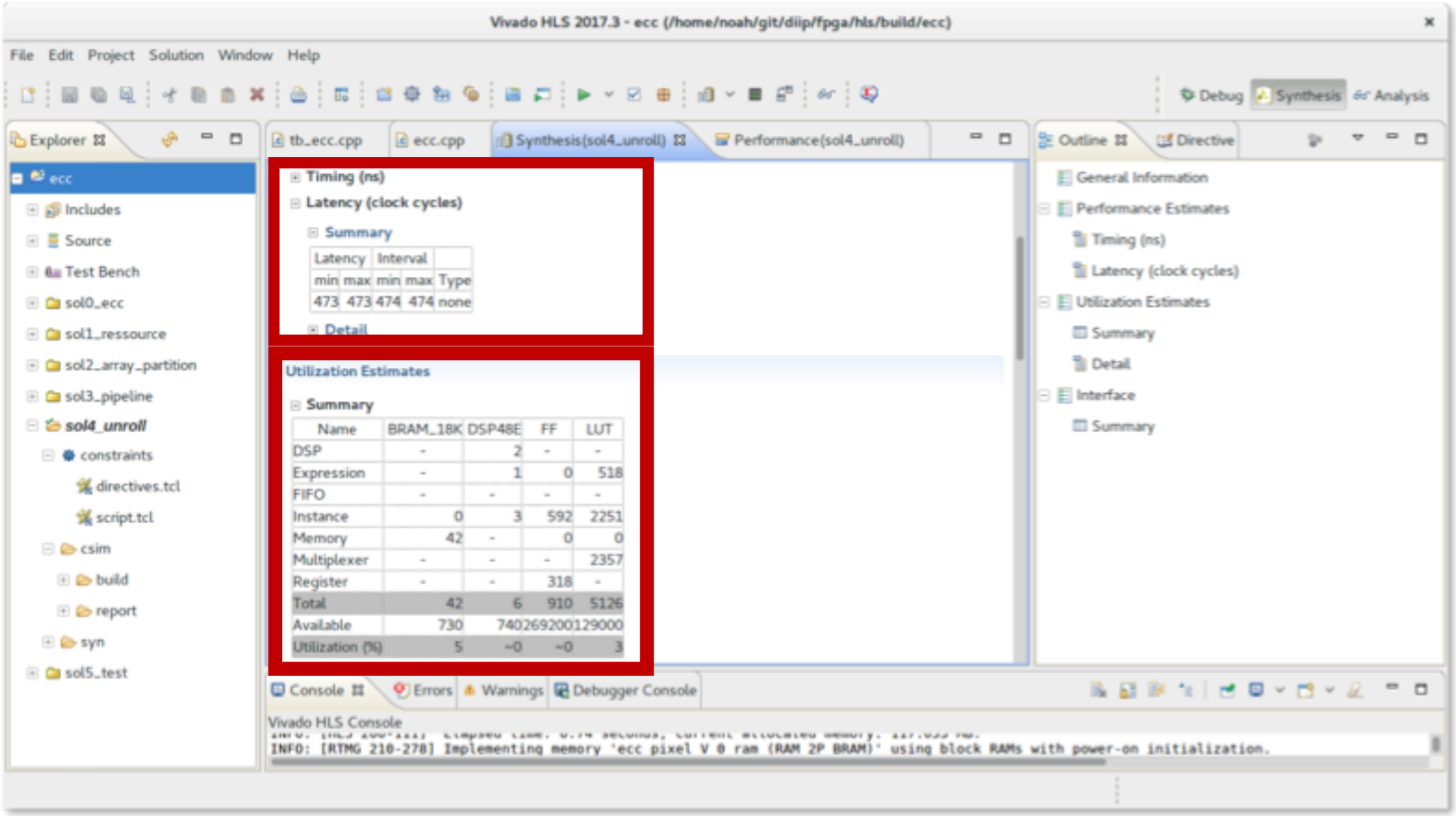


Noah Hütter & Jan Stocker

ime Institute of Microelectronics

GUI: Synthesis report

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

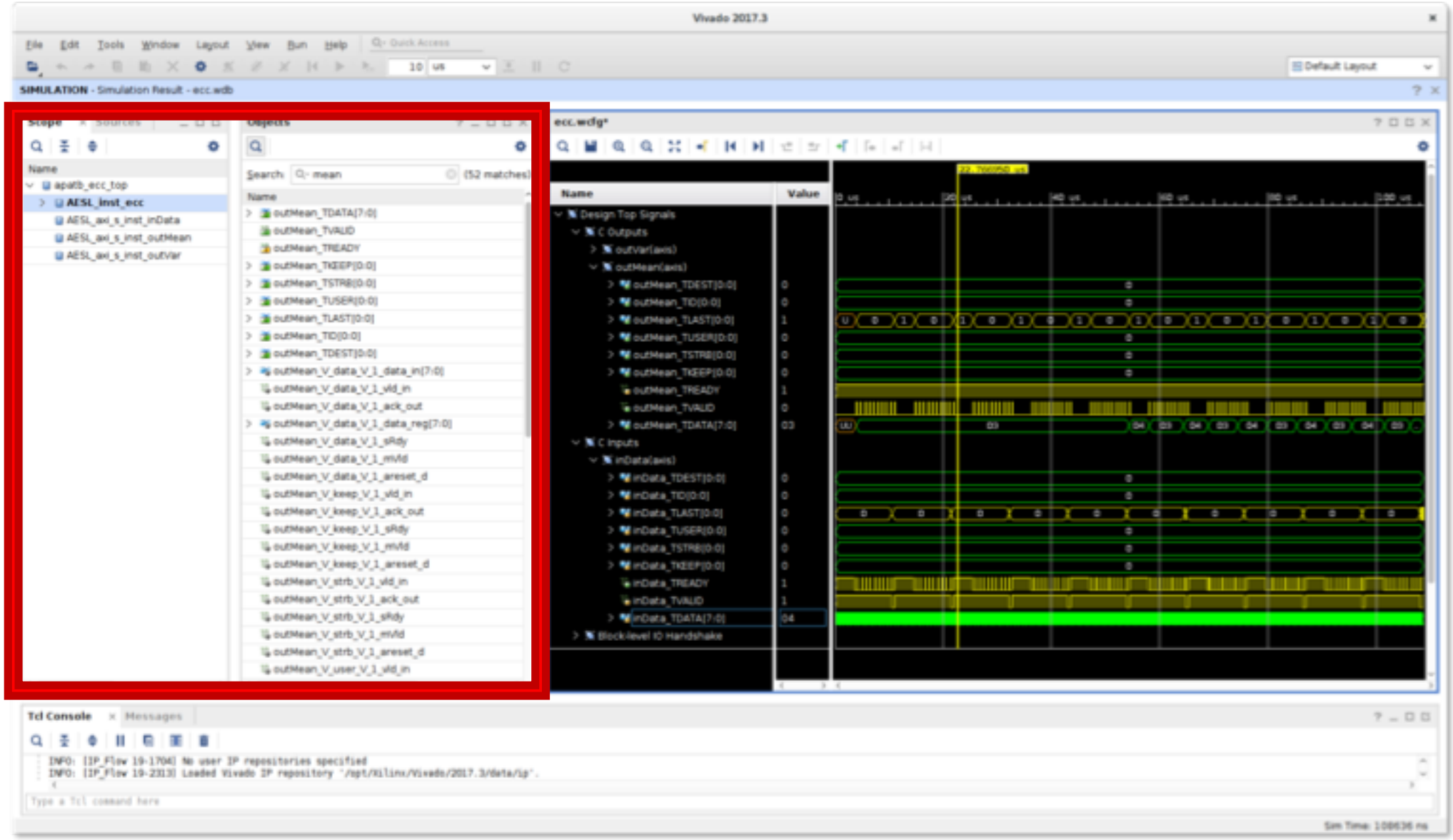


Noah Hütter & Jan Stocker

Institute of Microelectronics

GUI: Wave

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering



Noah Hütter & Jan Stocker

Institute of Microelectronics

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Pragmas

Interface
- interface
- protocol
- stream

Ressource
- resource
- array_map
- array_partition
- array_reshape
- data_pack
- dependence
- allocation

Throughput/Area
- unroll
- pipeline
- inline
- dataflow
- loop_flatten
- loop_merge

Verschiedene
- clock
- reset
- loop_tripcount

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Pragma: INTERFACE

Control
- ap_ctrl_none
- ap_ctrl_hs

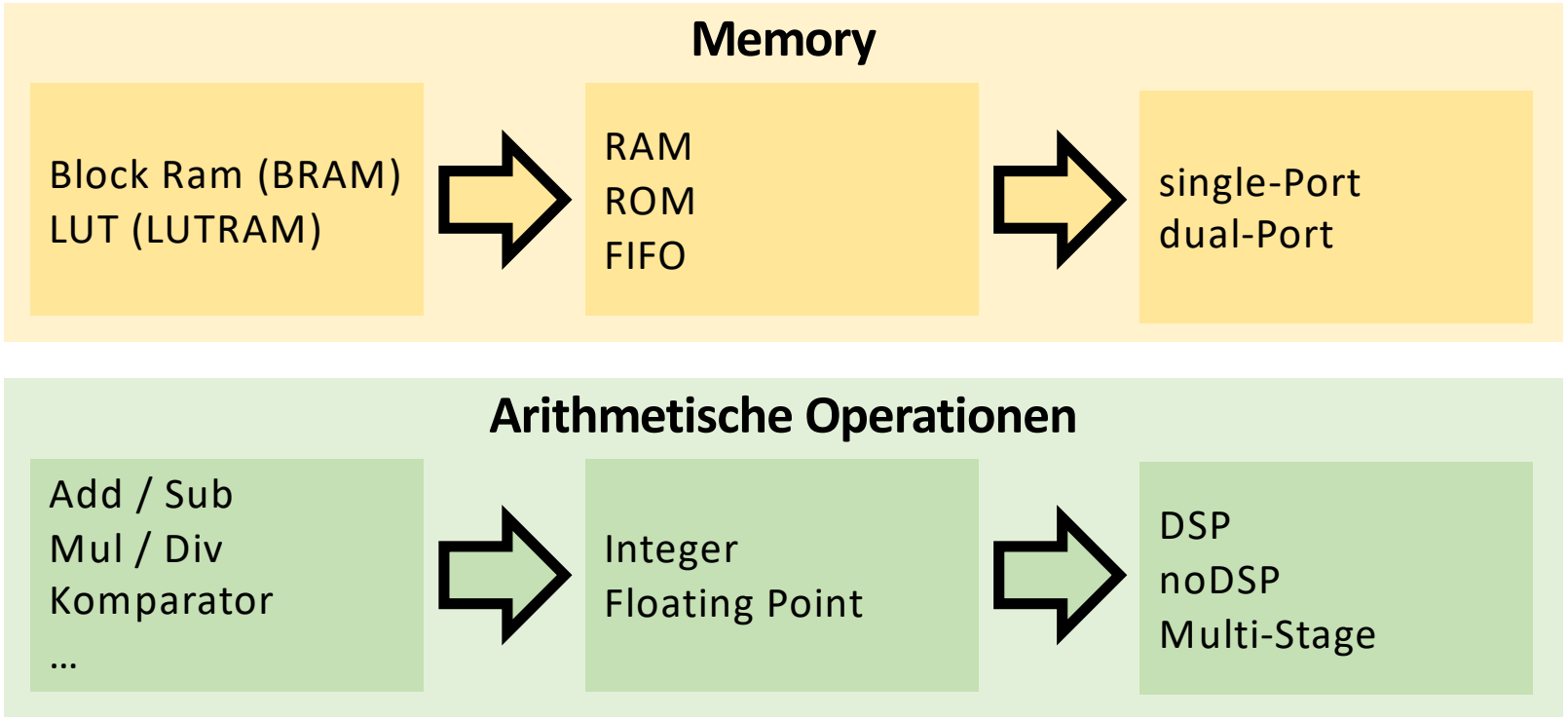
AXI4
- axis
- m_axi
- s_axilite

Memory
- ap_fifo
- bram

```
#pragma HLS INTERFACE axis register forward port=outMean
```

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

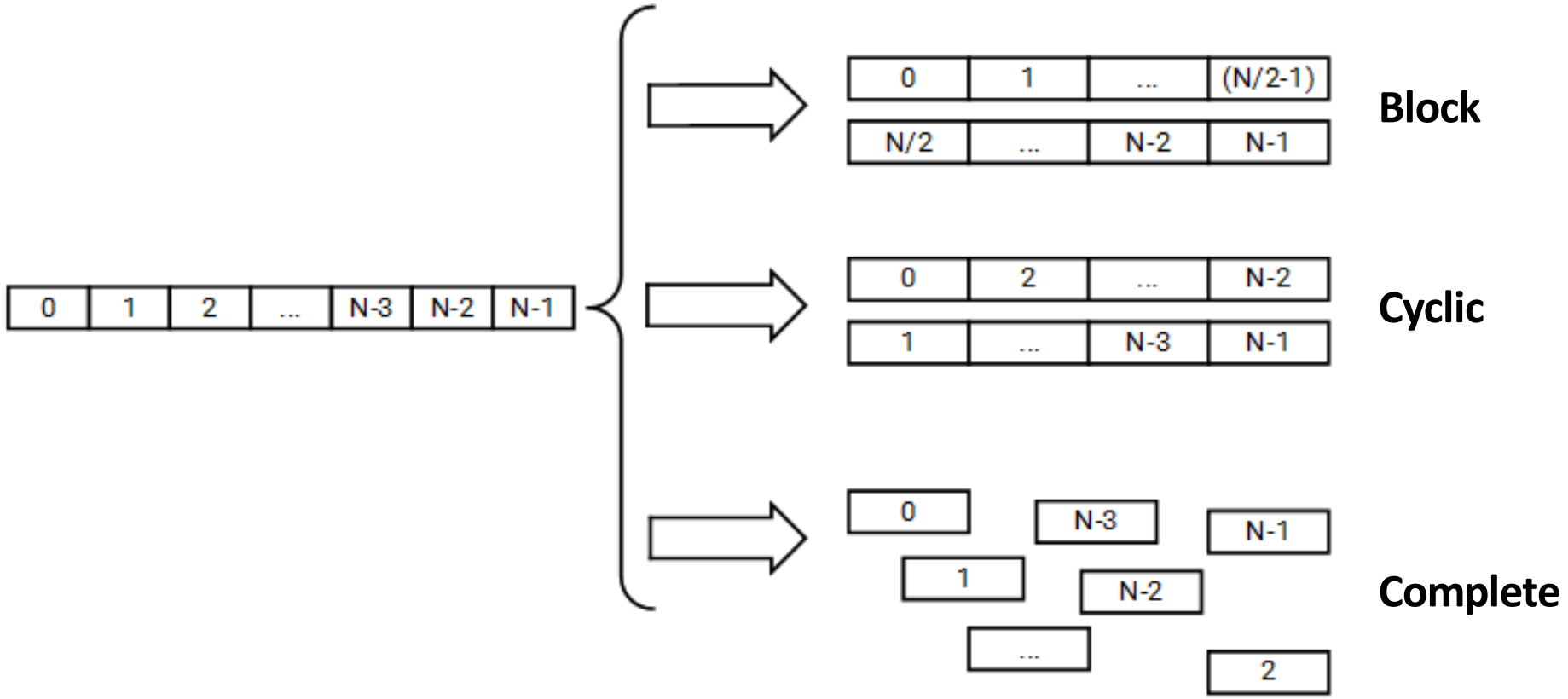
Pragma: RESSOURCE




```
#pragma HLS RESOURCE variable=pixel core=RAM_2P_BRAM
```

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

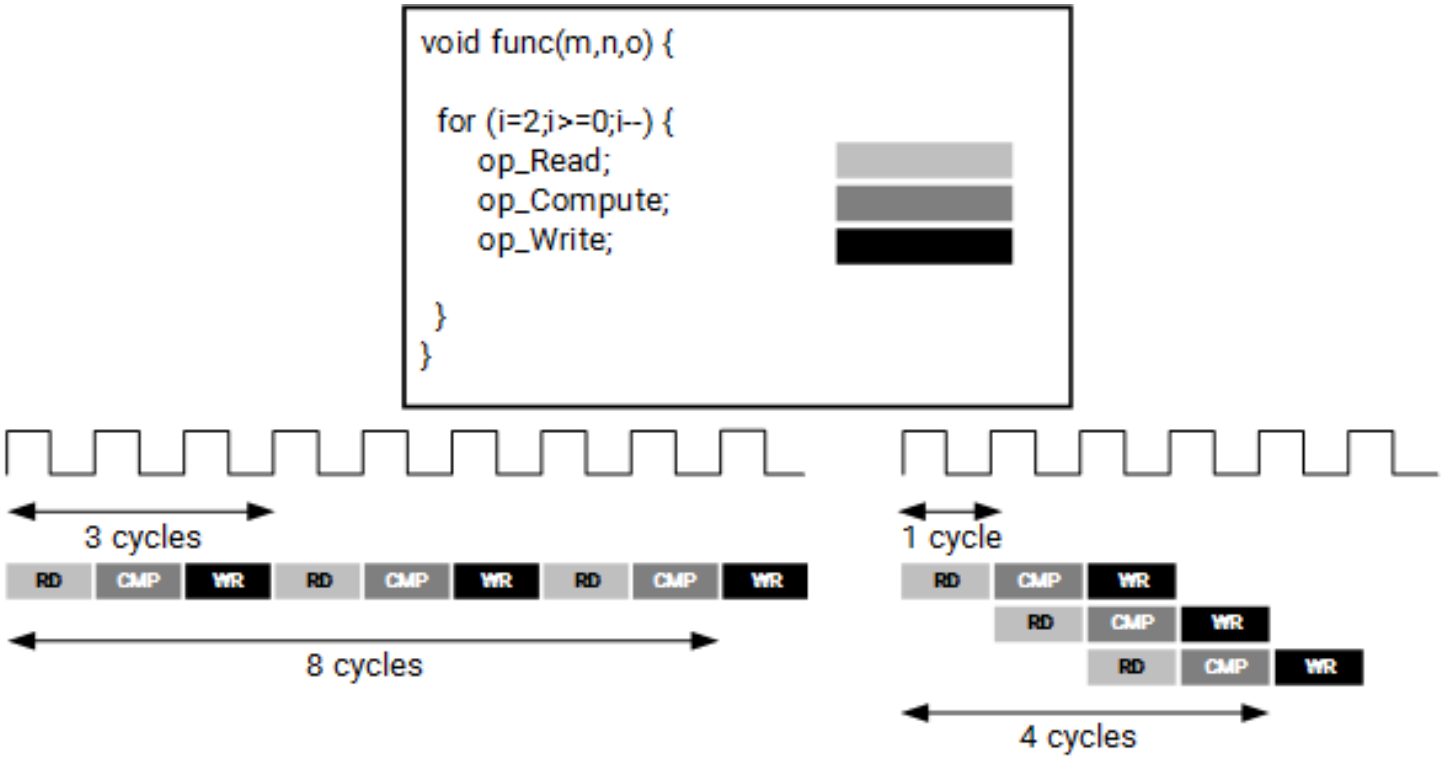
Pragma: ARRAY_PARTITION




```
#pragma HLS ARRAY_PARTITION variable=pixel dim=1 cyclic factor=21
```

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

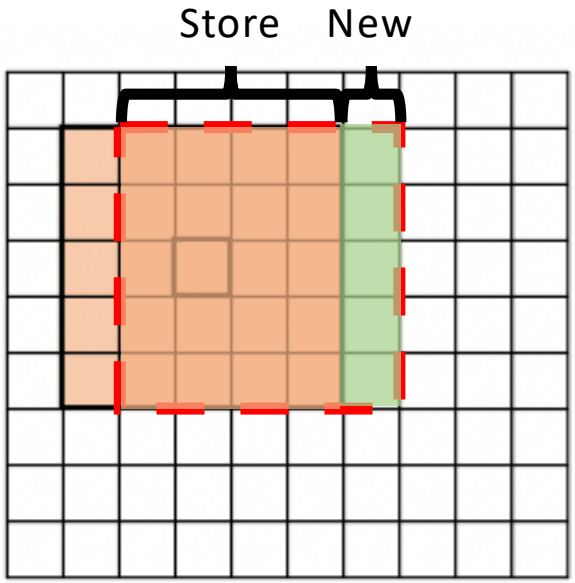
Pragma: PIPELINE / UNROLL



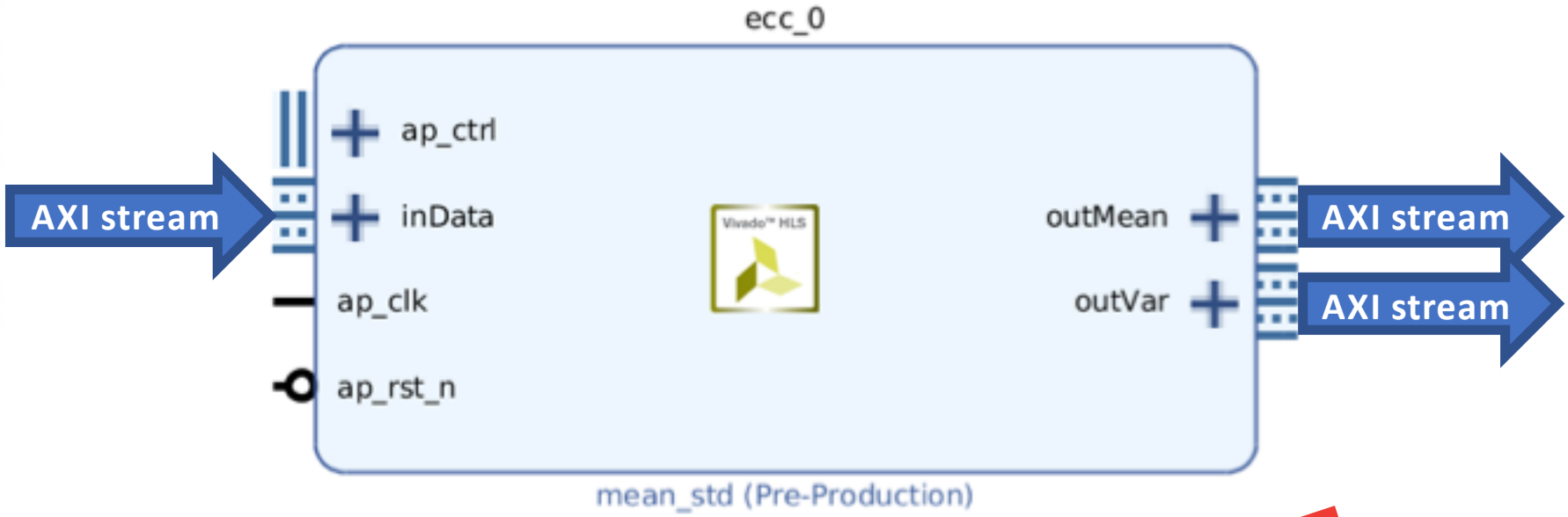

```
#pragma HLS PIPELINE
#pragma HLS UNROLL factor=21
```

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Ausgangslage



$$\mu = \frac{\sum_{i=1}^{N} p_i}{N} \qquad \sigma^2 = \frac{\sum_{i=1}^{N} (p_i - \mu)^2}{N - 1}$$



ime Institute of Microelectronics

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Problem

- Jedes Pixel wird zweimal gebraucht
- Zuerst Mittelwert, dann Varianz berechnen

$$\mu = \frac{\sum_{i=1}^{N} p_i}{N} \qquad \sigma^2 = \frac{\sum_{i=1}^{N} (p_i - \mu)^2}{N - 1}$$

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Solutions

Interface

```
#pragma HLS INTERFACE axis register reverse port=inDATA
```

Array Partition

```
#pragma HLS ARRAY_PARTITION variable=pixel cyclic factor=21 dim=1
```

Pipeline Loops

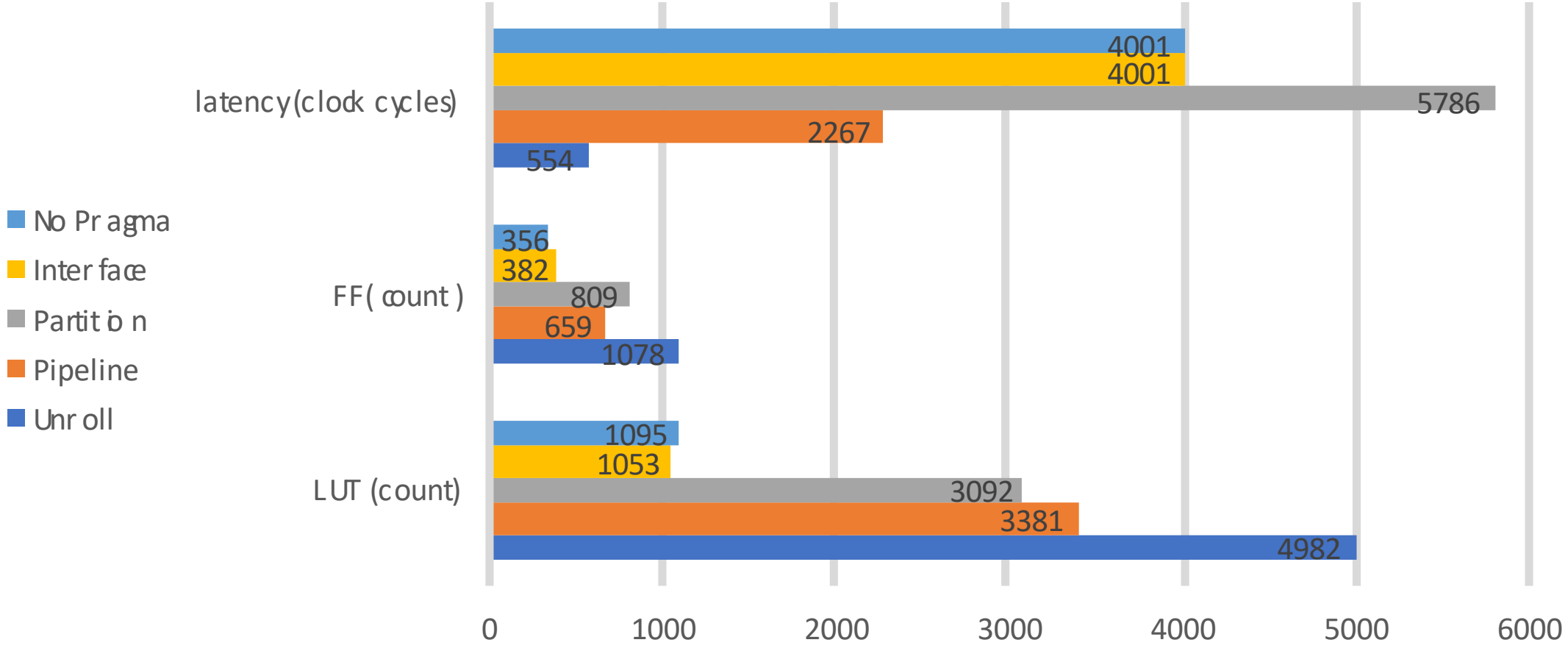
```
#pragma HLS PIPELINE
```

Unroll Loops

```
#pragma HLS UNROLL factor=21
```

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Vergleich

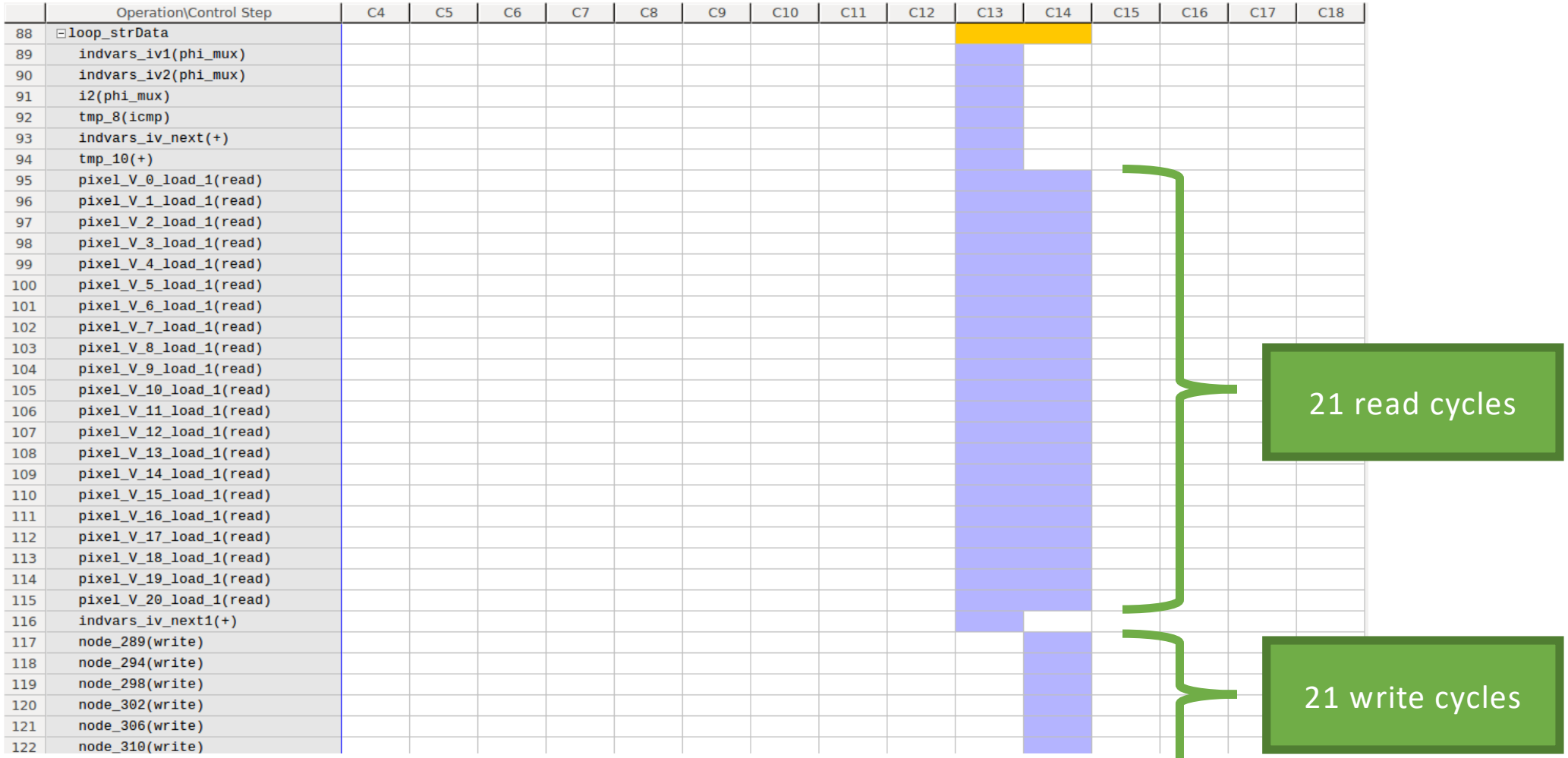


Noah Hütter & Jan Stocker

ime Institute of Microelectronics

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Analyse



Fazit

Vorteile

- Schneller Design Flow
- Lösungen Vergleichen
- Portabel

Nachteile

- Mächtiges Tool das es zu beherrschen gilt
- Nicht nur einfach C/C++ Code

Fazit

Stolpersteine

- Kleine Änderungen können zu grossen Unterschieden in Zeit und Ressourcennutzung führen
  → **Nur ein Pragma ändern**

- Hardwarenahes Denken wird vernachlässigt
  → **Flache Hierarchie im C/C++ Code**

Fazit

Unsere Erfahrungen

- Pragmas müssen richtig eingesetzt werden
- Mit wenig VHDL Kenntnissen zum Erfolg
- AXI Interfaces schnell umgesetzt
- Auch FSM möglich

Distributed FPGA for enhanced Image Processing

Bildverarbeitung mit Vivado HLS

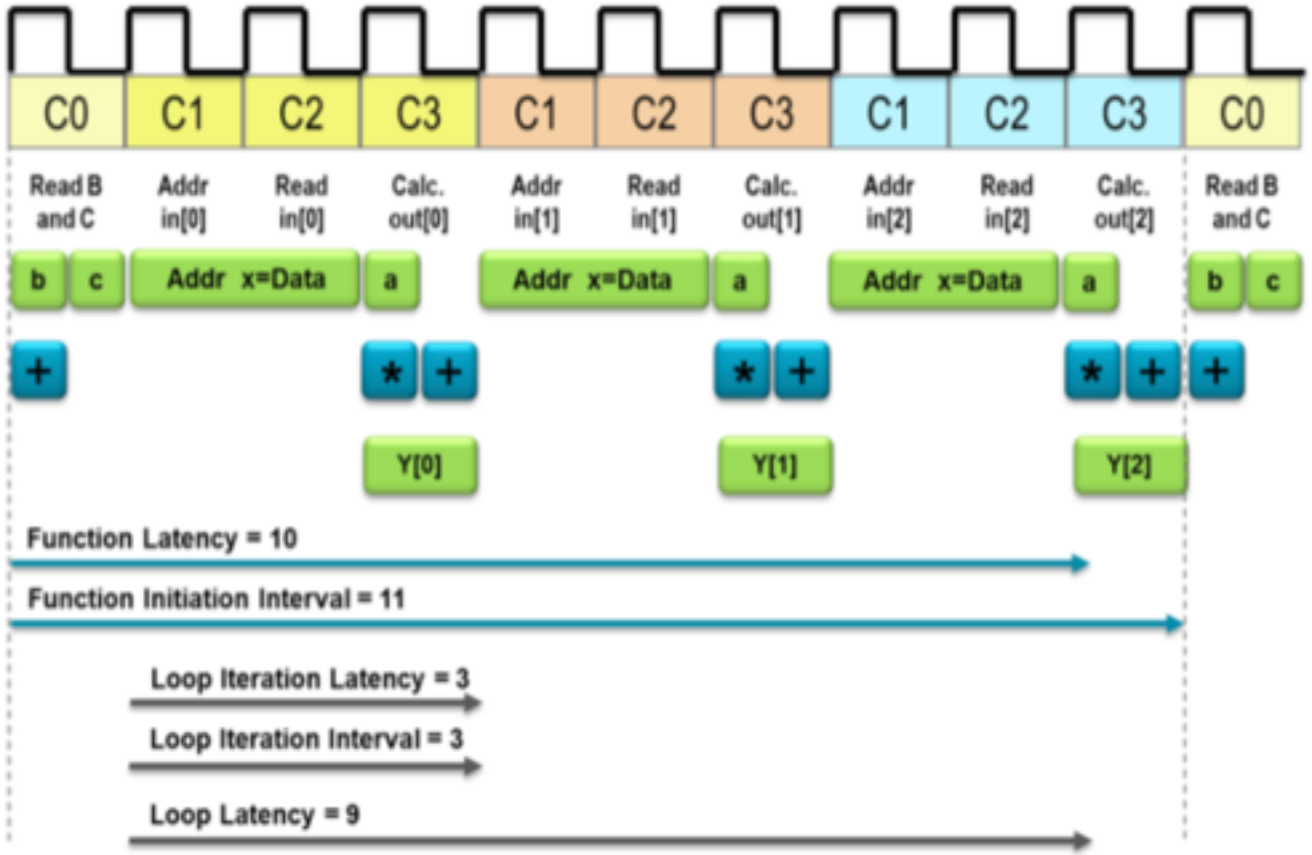
Noah Hütter noahhuetter@gmail.com
Jan Stocker janstocker@gmx.ch

UG902 Vivado HLS User Guide PDF
UG871 Vivado HLS Tutorial PDF
HLS Pragmas Site
Präsentation & Code Site

bit.do/ECC18

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Latency and Initiation Interval



Xilinx – UG902 High-Level Synthesis

University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

FSM mit Vivado HLS

```
void fsm() {

    static enum dState {D_IDLE = 0, D_WRITE, D_READ} state;

    switch(state) {
        case D_IDLE:
            if ( next_state_condition )
                state = D_WRITE;
            // Do stuff
            break;
        case D_WRITE:
            if ( next_state_condition )
                state = D_READ;
            // Do stuff
            break;
        case D_READ:
            if ( next_state_condition )
                state = D_IDLE;
            // Do stuff
            break;
    }
}
```