Software Acceleration in FPGA Fabric: Deploying C/C++ Code with AMD Vitis HLS on a ZynqUS+ Platform

04.10.2023

AVNET
Reach Further™

Contact

Email:
marco.hoefle@avnet.eu

LinkedIn:
https://www.linkedin.com/in/marco-hoefle/

Agenda

01 **Intro**

02 **Vivado**: Platform Generation

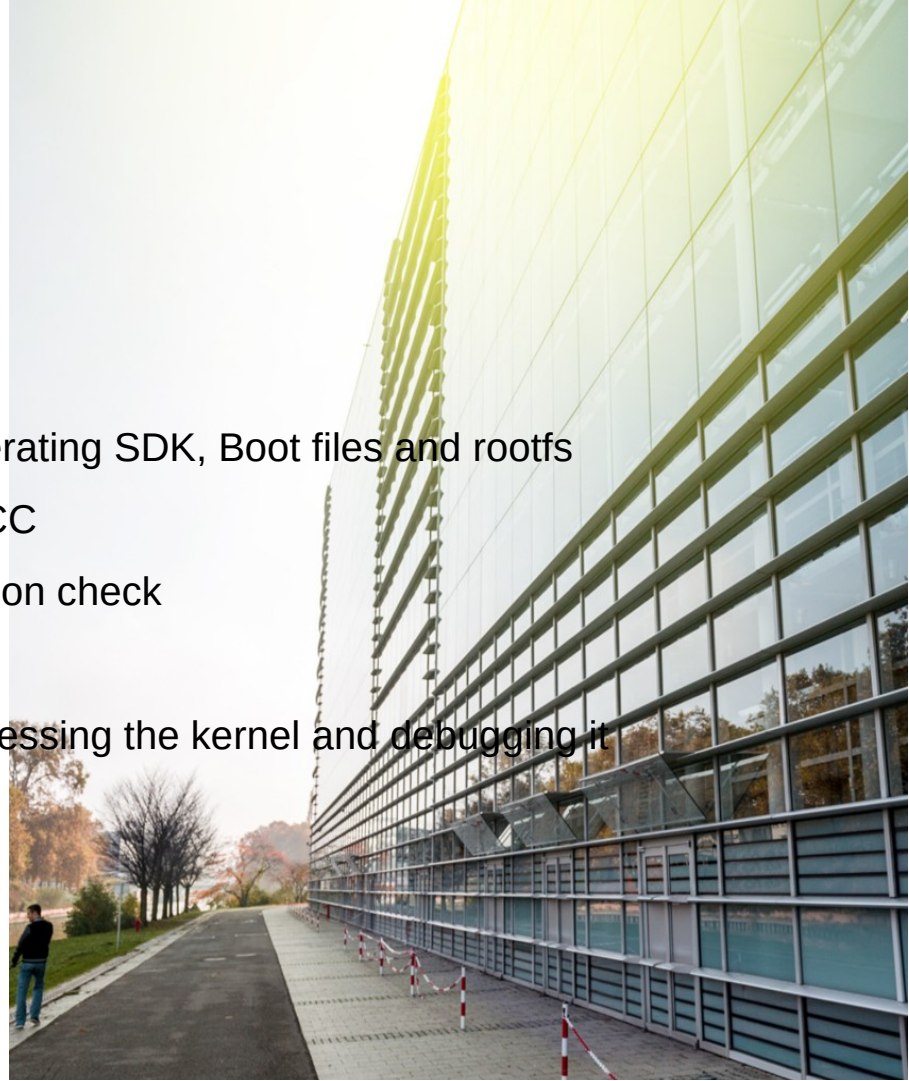
03 **PetaLinux**: Importing Vivado XSA, Generating SDK, Boot files and rootfs

04 **Vitis HLS Tcl**: Simulating Kernel with GCC

05 **Vitis HLS GUI**: Code stepping and function check

06 **Vitis Tcl**: Building Kernel and XCLBIN

07 **Vitis IDE**: Developing Host Code for Accessing the kernel and debugging it



3. High Level Synthesis

3. HLS

Xilinx HLS (High-Level Synthesis) is a tool provided by AMD / Xilinx.
HLS enables designers to write their digital hardware designs using
high-level languages, such as C, C++, or SystemC
instead of traditional low-level hardware description languages like VHDL or Verilog.

HLS exsits since Vivado 2014

3. HLS

What is HLS and HLS Benefits

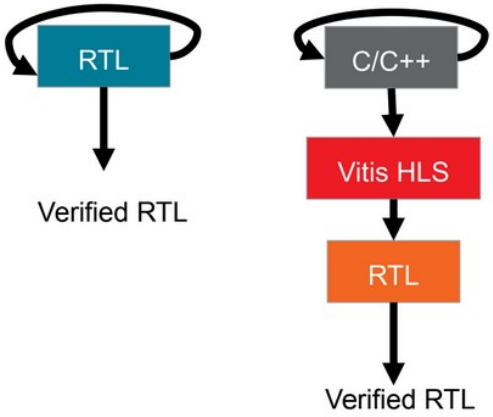
- Automated C/C++ to RTL Conversion
- Allows Significantly Faster Design Iterations
- Significantly Accelerates Simulation – Important For Wireless, Video Applications

3. HLS

Vitis™ HLS: Significantly Accelerates Simulation

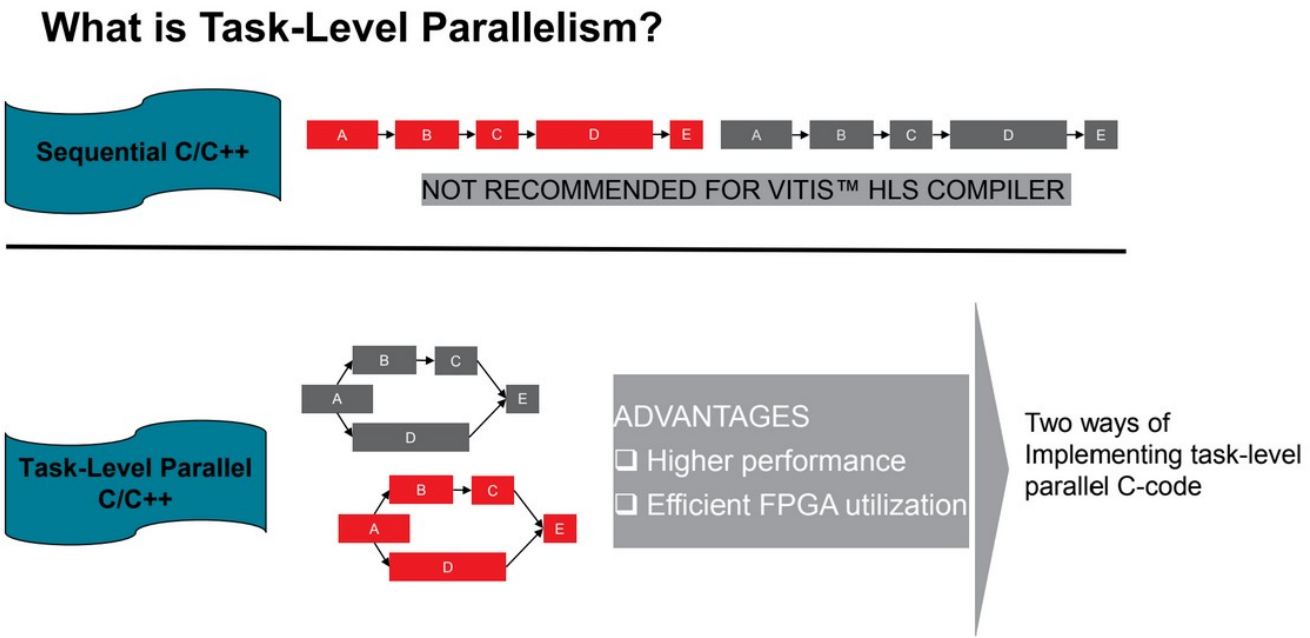
Especially Important in Video and Wireless type of applications – where certain frames of video and certain number of packets have to be simulated

Input	RTL Simulation Time	C-Simulation Time	Acceleration
10 frames of video data	~ 2 days	10 seconds	~12,000X



5

3. HLS


What is Task-Level Parallelism?
Sequential C/C++
A
B
C
D
E
A
B
C
D
E
NOT RECOMMENDED FOR VITIS™ HLS COMPILER
Task-Level Parallel C/C++
B
C
A
E
D
B
C
A
E
D
ADVANTAGES
Higher performance
Efficient FPGA utilization
Two ways of Implementing task-level parallel C-code

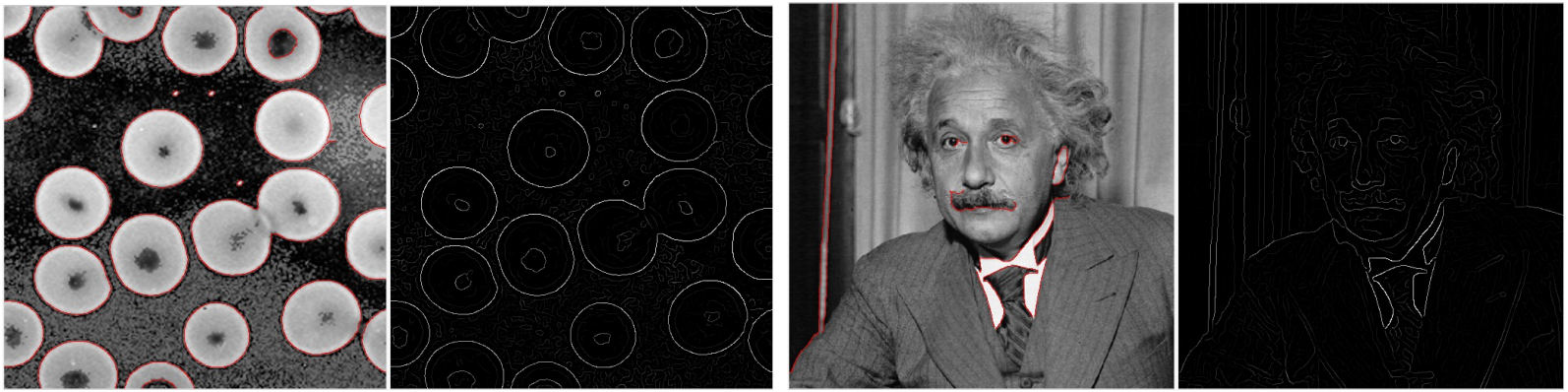
3. HLS

- Vitis HLS User Guide(UG1399)
- Vitis HLS Getting Started Tutorial
- Vitis Tutorials
- Vitis HLS Introductory Examples
- Vitis Libraries

4. Practical Examples

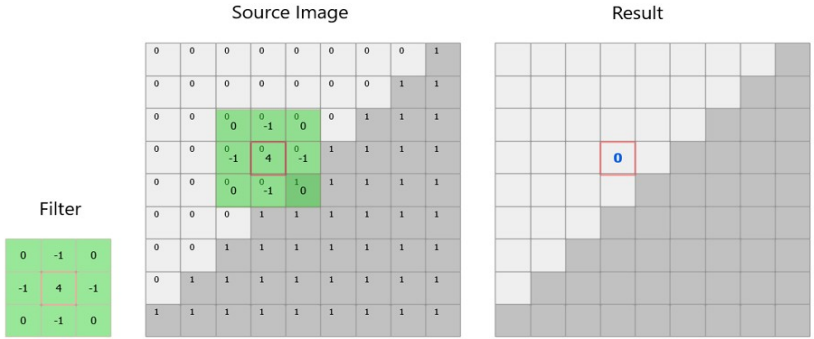
4. Practical Examples

Image filtering

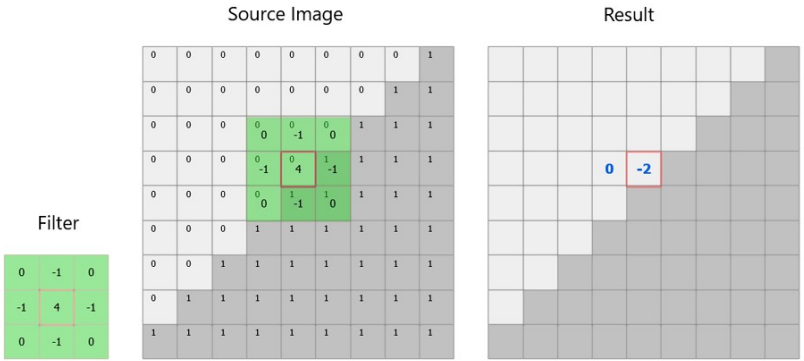


4. Practical Examples

Image filtering



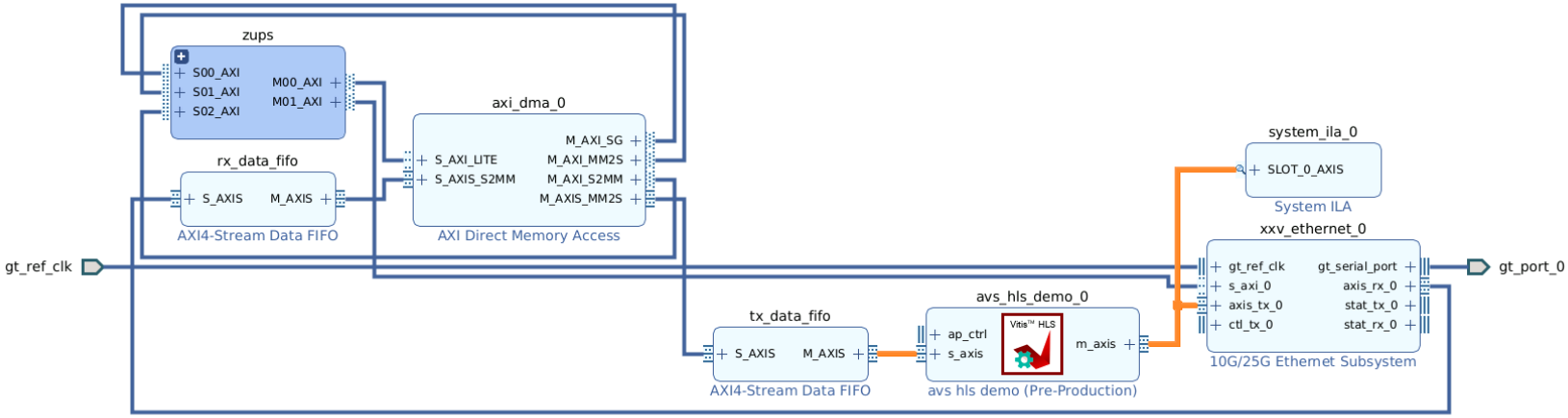
0*0 + 0*-1 + 0*0 + 0*-1 +
0*4 + 0*-1 + 0*0 + 0*-1 +
1*0 = 0



0*0 + 0*-1 + 0*0 + 0*-1 +
0*4 + 1*-1 + 0*0 + 1*-1 +
1*0 = -2

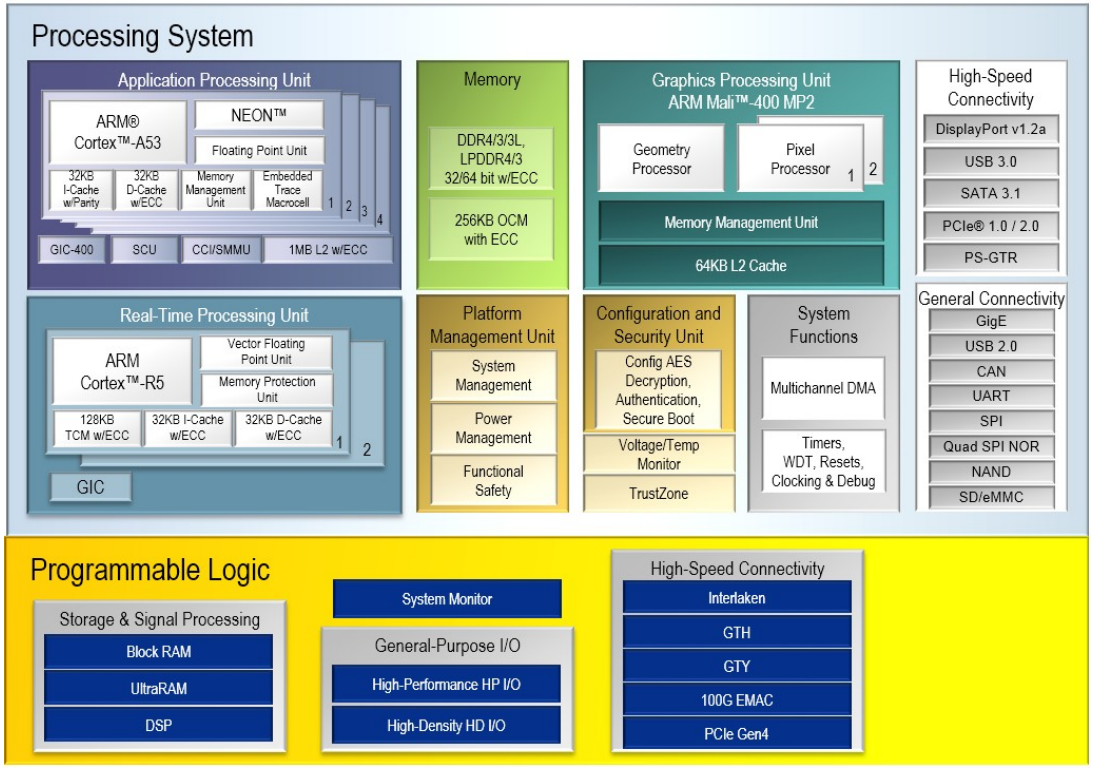
4. Practical Examples

Network acceleration



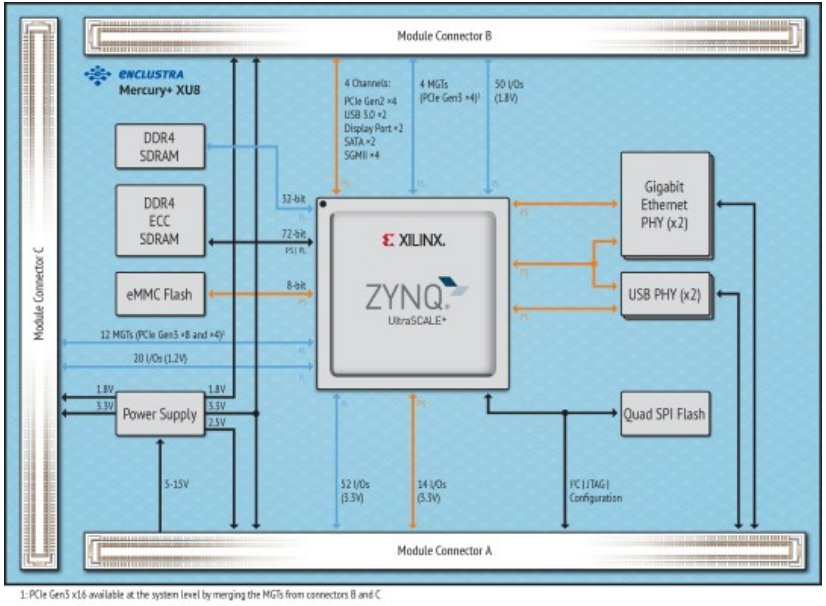
2. Hardware

2. Hardware: AMD Zynq UltraScale+ Architecture



2. Hardware: Enclustra Mercury+ XU8 SoM + ST1 carrier

Zynq UltraScale+ XCZU7EV MPSoC



Thank you!