Getting the best from the Ultra96 in the IoT and Embedded Vision Arena without being a FPGA Expert

ADAM TAYLOR – WWW.MICROZEDCHRONICLES.COM
For those unfamiliar with the U96

- Based upon 96 Boards Consumer Specification
- Heterogeneous SoC – Zynq MPSoC ZU3EG
  - 2 GB RAM
  - SD Card
- Runs Linux – Generally developed using PetaLinux flow
  - PetaLinux is not a Linux Distribution but allows us to create Linux solutions using Xilinx Git or open source communities
Introduction to Heterogeneous SoC

- Processing System (PS)
  - Boots first
  - Contains processors, fixed peripherals, clocks, memory, and memory controllers
  - Dedicated silicon Limited configurability

- Programmable logic (PL)
  - Based on UltraScale architecture logic Details available in other topic clusters and documentation
  - Contains dedicated silicon resources: DSP48e, block RAM, high-speed serial, XADC, PCIe core, etc.

- Interconnects
  - AXI based
  - Clocks and Resets
  - Other
PS in detail

- Quad/dual-core ARM Cortex-A53 processor cluster
- Dual-core ARM Cortex-R5 real-time processor cluster ARM Mali-400MP graphics processor
- Memory controllers (DDRx and SMC)
- Peripherals
  - High-speed peripherals: USB 3.0, SATA 3.0, PCIe Gen 2
  - technology, DisplayPort
  - Low-speed peripherals: CAN, UART, I2C
- Security, power management, safety, and reliability
Some lesser known – but useful sub systems

- Helps us Secure our System
- Shared Acceleration Functions
- AES 256- GCM
- 4096 RSA Multiplier
- SHA-384 engines
- Key Management
- Use CSU DMA to transfer data
Why PL?

▶ Programmable Logic - It is not just for performance
  ▶ Flexibility of IO structures enables with the right Phy – Any to Any Interfacing
    ▶ High Speed ADC, DAC & Imagers, IO Standards Updates, Legacy and Bespoke
  ▶ Can be used for acceleration of functions from the PS to the PL
  ▶ Removal of traditional bottle necks
  ▶ Increases determinism
  ▶ Reduces Latency
  ▶ More Responsive Solution
PL / PS Applications

- PS/PL combination offers significant benefits for
  - Internet of Thing applications – Home Automation
  - Embedded Vision applications – Surveillance, Vision Guided Robotics
  - Machine Learning applications – intelligence at the edge

- Increasingly in products these are converging in products
Developing for the PL is hard?

- Traditional approaches to PL development are very low level
  - VHDL or Verilog – working at the register transfer level. State Machines, Counters etc
  - Development time is significant
  - Verification time is significant
  - Gap between the system models and Implementation Significant
IP Integrator

- Leverage IP blocks
- Configure via Dialogs and SW
- Significant range of IP available
- Need to be able to drive Vivado
Create Your Own

- High Level Synthesis
  - Use C, C++ or OpenCL to create Implementable Hardware elements which can be added into IPI Integrator.
  - Uses a subset of C (no system calls) with simple bounds and rules.
  - Performance in PL is optimised by the use of Pragma
  - Need to be aware of basic techniques
    - Un Rolling Loops
    - Pipelining
    - Fracturing Memory
Supporting Libraries

- Math
- Vivado HLS IP Library
- Linear Algebra Library
- Arbitrary Precision Data Types
- reVISION Acceleration stack
- HLS Video Libraries
Example of HLS

```c
int foo(char x, char a, char b, char c) {
    char y;
    y = x*a+b+c;
    return y
}
```
Example HLS

- Sobel Filter – 60 frames per second
- Uses HLS Video Libraries
- 9 Lines of code

```c
#pragma HLS dataflow
hls::AXIvideo2Mat(INPUT_STREAM, img_0);
hls::CvtColor<HLS_BGR2GRAY>(img_0, img_1);
hls::GaussianBlur<3,3>(img_1, img_2);
hls::Duplicate(img_2, img_2a, img_2b);
hls::Sobel<1,0,3>(img_2a, img_3);
hls::Sobel<0,1,3>(img_2b, img_4);
hls::AddWeighted(img_4,0.5, img_3,0.5,0.0, img_5);
hls::CvtColor<HLS_GRAY2RGB>(img_5, img_6);
hls::Mat2AXIvideo(img_6, OUTPUT_STREAM);
```
What is SDSoC?

- SDSoC is a system optimising compiler which allows us to optimise
  - Zynq PS / PL
  - Zynq MPSoC PS / PL
  - MicroBlaze

- What does this mean?
  Following the creation of a platform, we can develop our application in C, C++ and accelerate functions from executing in the Processor to being implemented in programmable logic.
SDSoC Development Flow

The SDSoC Development Environment

- C/C++/OpenCL application development experience
- System-level profiling
- Full system optimizing compiler
- Expert use model for platform developers & system architects
Under the hood

- Accelerates the Function using Vivado HLS
- Analyses Communication
- Establishes AXI Communications
- Generates Software Stub
Device support: Zynq®-7000 device, Zynq UltraScale+™ MPSoC & MicroBlaze

ARM compiler tool chain support: Linaro-based gcc compiler tool chains

Target OS support: Linux (kernel 4.x, Xilinx branch), bare metal, and FreeRTOS 9.0.1

QEMU and RTL co-simulation: Linux and Windows 64-bit host support

OpenCL compilation flow support
So How does it work

1. Accelerator function is called
2. Configures DMA to move data
3. Data is sourced from DDR, OCM, or L1/L2 to an input buffer
4. DMA
5. Data is moved to appropriate place
6. Accelerator
7. Buffer might be local memory, BRAM, or FIFO
8. DMA Runs
9. The accelerator loads the output buffer
10. Accelerator function completes user code continues

Once the transfer is complete, the DMA signals the processor
What is the best to accelerate

- Some obvious things that cannot be accelerated
  - Pre compiled Libraries, OS Calls etc

- Intensive algorithms are a good candidate

- Need to consider the time it takes for data movement to and from the PL

- Use Amdahl’s law
  \[ S = \frac{1}{(1 - a) + a/p} \]
Amdahl's law

- $S$: overall performance improvement
- $\alpha$: percentage of the algorithm that can be sped up with hardware acceleration
- $1 - \alpha$: percentage of the algorithm that cannot be improved.
- $p$: is the speedup due to acceleration ($\%$).

$$S = \frac{1}{(1 - \alpha) + \alpha/p}$$

- Set $\alpha$ to 0.1 and select speed up - even with large acceleration $P$ defined, speed up is close to 1
- Set $\alpha$ to 0.5 and select same speed up – close to factor of two improvement.
Do you want to know more?

- Check out my Hackster projects - hackster.io/adam-taylor
- Follow the MicroZed Chronicles – weekly blogs on Zynq, Zynq MPSoC Universe
- If you want to learn more and experience hands on labs on the Ultra96
- **Hello Ultra96! Getting Started with the Ultimate SoC Board**
- **Three Labs**
  - Lab 1 - Create a simple hello world
  - Lab 2 - Create a PetaLinux OS
  - Lab 3 - Create a simple user application
- Register here?
- https://events.hackster.io/Ultra96
Questions ?