Vrije Universiteit Brussel

Performance Analysis of a Real-Time Video Processing System

Frank Vanbever

27 January 2014

Contents

- Introduction
- Platform Overview
  - Zynq-7000
  - ZC-702 Base TRD
- High Level Synthesis
  - Vivado HLS
- Performance Analysis
  - Roofline Model
  - Memory Architecture
  - Directives
  - Resource Consumption
  - Memory Bandwidth

Introduction

- Until recently computing performance could be gained maintaining the sequential programming paradigm
  - Primarily through Moore's law
  - multiple-issue, pipelining and *out-of-order* execution perpetuated this
- Eventually a *power wall* was hit. The size of MOSFETs couldn't keep on decreasing whilst the frequency decreased.
- The solution was to turn to parallel processors, containing more than one processing unit working at a time.
  - This caused a new problem: How could this parallel hardware be used efficiently.
  - A new programming paradigm is necessary

Zynq-7000

- The Zynq-7000 is a *System On Chip* or SoC which consists of a dual core ARM Cortex-A9 and Xilinx programmable logic
  - These are called the Processing System (PS) and Programmable Logic (PL)
- The processing system consists of the Application Processor Unit (APU), a memory controller and numerous I/O peripherals
- The Programmable Logic is based on Artix-7 or Kintex-7 technology dependent on the type

Interconnect

- The interconnections in the Zynq SoC use the Advanced Extensible Interface (AXI) protocol which is part of ARM's Advanced Microcontroller Bus Architecture (AMBA) v3.0
- The interconnections between the PS and PL play an important role in the system's performance
- **AXI_HP** The system has four **H**igh **P**erformance interconnections connecting PL masters to DDR and OCM memories using double FIFO buffers
- **AXI_GP** The system has four **G**eneral **P**urpose ports divided into 2 master ports and 2 slave ports.
- **AXI_ACP** The **A**ccelerator **C**oherency **P**ort connects the PL directly with the APU caches through the Snoop Control Unit with optional coherency

Base Targeted Reference Design 14.5

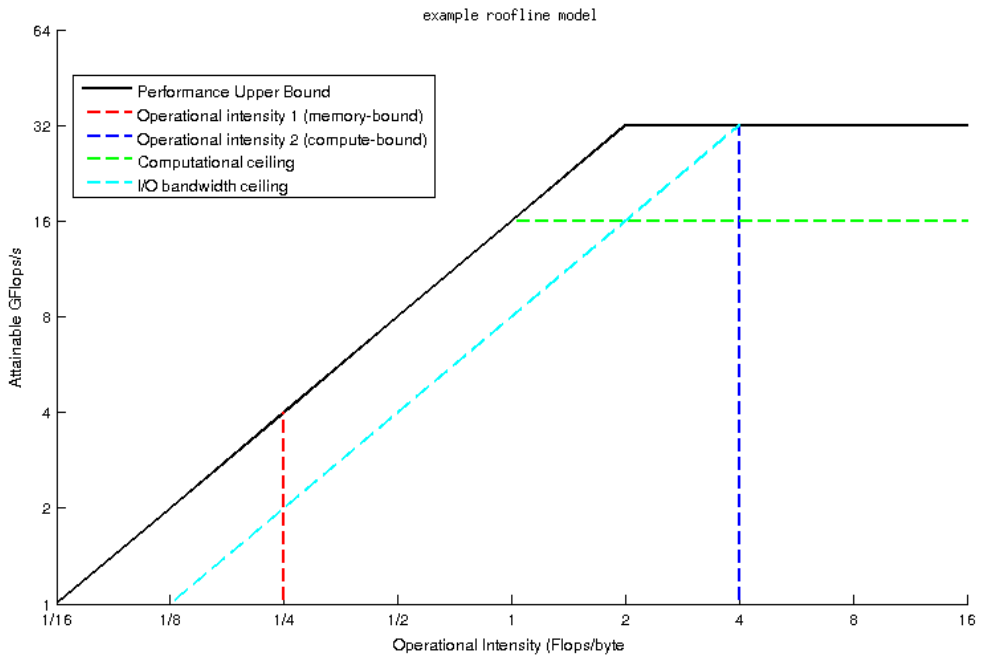
- A preexisting system was used as the base for the experiments
- On the hardware side it consists of a complete image processing pipeline
  - This pipeline can be split into 3 stages, reading and writing to/from memory, each using their own DMA controller
- The software side consists of a bootloader, the Xilinx Linux Kernel and a number of applications
- The goal of this system is to apply the Sobel operator to the video in real-time

High Level Synthesis

- A recurring theme is the relative difficulty of implementing an algorithm on FPGA
  - Both development and synthesis take considerably longer
- HLS tools partially alleviate these problems
  - Typically a C,C++ or SystemC program is converted into VHDL or Verilog
- This improves programmer productivity
- Vivado HLS was used for this thesis
  - C code is compilable allowing pre-synthesis verification testbenches, which in turn can be compiled to SystemC for post-synthesis verification
  - A multitude of optimizations and interfaces are available
  - Synthesis reports and analysis perspective contains information about timing, latency and resource consumption

Roofline Model

- The roofline model is a visial model that gives insight into factors influencing performance
  - It is based on the observation that bandwidth is the constraining resource on system performance
  - The model is a plot of the peak computational performance as a function of the computational performance



Memory Architecture

- Buffers influence the computational intensity, 2 types are used
  - **Linebuffers** buffer a number of lines of the application, in this case 3. Implemented as BRAMS for their dual port nature
  - **Memory Windows** buffer the values on which the Sobel operation is applied. Implemented as registers to give instantaneous access to all values at the same time
- The implementation reads and writes one 32 bpp pixel every iteration. The sobel operator get applied 3 times giving following expression for the CI

$$CI = \frac{3\times(H\times W)}{4\times2\times(H\times W)} = \frac{3}{8}$$

Directives

- The impact of directives following on the system's performance was tested and compared to the original configuration
  - **Pipelining** allows for the concurrent execution of the operations in the algorithm. After the initial latency a new output can be written after II cycles
  - **Loop Flattening** is a loop level optimization that combines nested loops into 1 loop. Disabling it prevents timing violations and increased initiation interval
  - **Dependence** provides the compiler with extra information on the intra- or inter-dependencies in the code. Notifying the compiler of a mistakenly identified dependency accessing the linebuffer removes the need for an extra cycle
- Only 2 configurations were found to satisfy the system constraints.

Resource Consumption & Memory Bandwidth

- The resource consumption determines the scalability of the system. In the case of the TRD the AXI HP interfaces were found to be the constraining factor allowing 3 possible instances
- As more AXI HP interfaces are used the available bandwidth increases
  - Only using 4 interfaces the DDR RAM bandwidth becomes the constraining factor
- In the case of the TRD the throughput remains constrained by the core

Conclusion

- The performance of a Zynq based system is influenced by a number of factors
  - The memory implementation influences the Computational Intensity
  - The number and type of AXI interconnects used play a decisive role in the bandwidth
- HLS synthesis tools present a serious productivity improvement
- The necessary DMA controllers impact the available resources

The Zynq platform presents an interesting hybrid between an FPGA and a CPU, especially when considered in an embedded systems context.