Ashwin K. Avula
Yucheng Chang

ECE 554
Alvin & The Chip-Monks

Garry Chen
Alvin Cheng

Project Description

Overview:

- Implementation of a General-Purpose 32-Bit RISC-V CPU and dedicated DSP Coprocessor for acceleration of common image-processing tasks on an Altera DE-1 FPGA
- Independent Image & Video memories for efficient storage, access, and visualization of 256x256 4-Bit RGB Images
- Custom UART Bootloader system allowing for reprogramming of all device memories, avoiding frequent compilation of the FPGA
- Memory-aware design for full utilization of fast FPGA on-chip Block SRAM

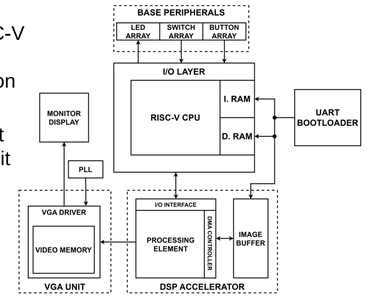

Figure 1: Top-Level Device Components

RISC-V Base 32-I Core Architecture

Features:

- 32x32b Register File supporting Register File Bypassing
- Harvard Memory Model (8 KB Instruction + 8 KB Data RAMs)
- Byte-Addressable Data Memory supporting Byte, Halfword, Word Accesses
- Execute→Execute, Memory→Execute, Memory→Memory, and Execute→Decode Forwarding
- Branch & Jump handling in DECODE for reduced STALL cycles

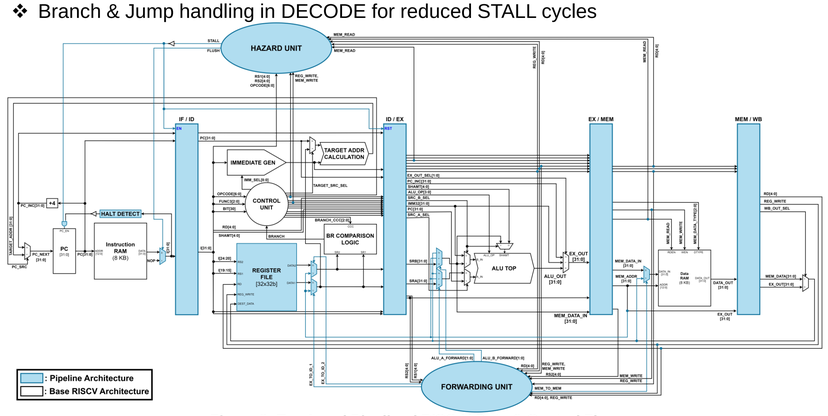

Figure 2: Top-Level Pipelined RISC-V Data & Control Flow

CPU Memory-Mapped I/O & UART Bootloader

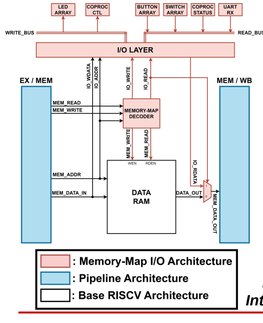

Figure 3: Top-Level I/O Layer Interface with CPU Memory Stage

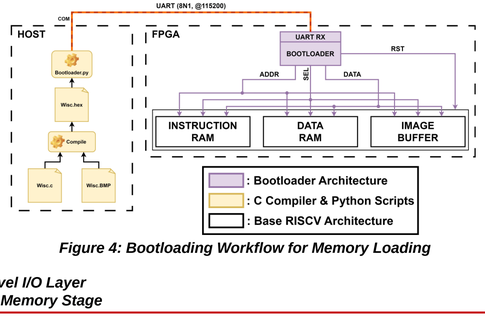

Figure 4: Bootloading Workflow for Memory Loading

Image Coprocessor Architecture

Image Buffer & Storage Format:

- Image Buffer allowing for fast dual-port access, while maximizing utilization of FPGA M10K Banks
- Pixel data streaming from Image Buffer by DMA, and cached for optimal data reuse in Processing Element
- Image Data Prefetching into Row Cache by DMA, thus <u>reducing Processing Element stalls</u>

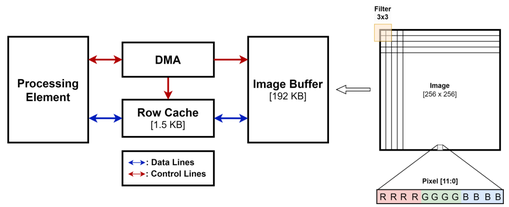

Figure 5: Direct Memory Access (DMA) Controller and Coprocessor Dataflow

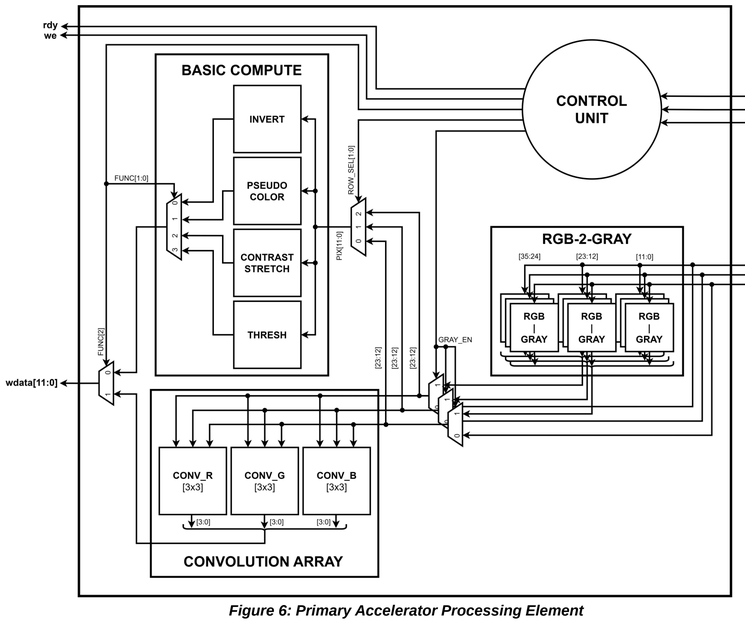

Figure 6: Primary Accelerator Processing Element

FUNC [2:0]	Coprocessor Operation	Description
000	INVERT	Bitwise Pixel Inversion
001	COLOR	Contrast-Based Pseudo-Coloring of Grayscale Image
010	CONTRAST	Contrast Stretching / Normalization of Image
011	THRESH	Noise Reduction using Pixel value Thresholding
100	GAUSSIAN	3x3 Gaussian Filter Convolution
101	EDGE_DET	3x3 Sobel Filter Convolution
110	SHARPEN	3x3 Sharpening Convolution
111	NOP	Image passthrough for display without operation

Table 1: Supported Coprocessor Operations

Accelerator Processing Element:

- Primary Processing Element containing RGB→Gray Conversion, Convolution, and Basic Compute Units
- Single-Cycle compute allowing for calculation of an output pixel per cycle
- Fixed channel-wise convolutional array for 3x3 Filters, Stride = 1
- Output of Processing Element written into VGA Video memory for display and into Row Cache for Image Buffer writeback

RISC-V ISA Extension for Coprocessor Support

Dedicated Load-Word-Coprocessor (LWCP) Instruction:

- **Behavior:** LWCP behaves like a LW Instruction, however upon reaching the MEM stage, LWCP stalls upstream stages until Coprocessor returns DONE response
- **Usage:** LWCP instruction must follow a SW instruction that invokes the Memory-Mapped Coprocessor, ensuring that the CPU will receive a DONE response in the future

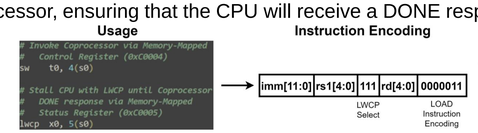

Figure 7: Usage & Encoding for Dedicated LWCP Instruction

Validation & Application

- **Verification:**
  - Complete CPU verification in RISC-V toolchain environment
  - Coprocessor verification by comparing expected output generated in Python with accelerator output images
- **Software:** C and Assembly programs written for performing all operations using CPU-Only, CPU + Coprocessor, and CPU + Coprocessor + LWCP Instruction device combinations
- **Application:** Operations performed on various 256x256 RGB and grayscale medical images
- **Implementation:** Deployed onto Altera DE-1 FPGA @ 50MHz and image output to Monitor display via VGA driver

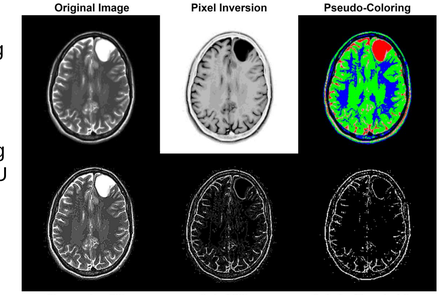

Figure 8: Example Coprocessor Output

Discussion

Results:

- Implementation of Coprocessor resulted in <u>significant program speedup, reduced application code, and less traffic to Memory</u>

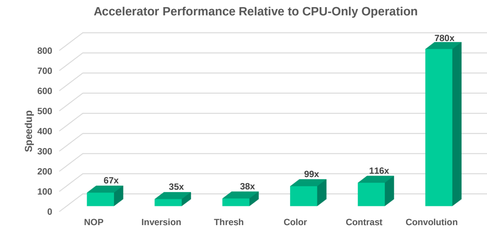

Figure 9: CPU + Accelerator Cycle Speedup Compared to CPU-Only Operation

- Implementation of LWCP Instruction avoids polling of Coprocessor Memory-Mapped Registers, and stalling allows for idling of CPU, thus <u>reducing application code and minimizing CPU Dynamic Power Consumption</u> during Coprocessor operation
- UART Bootloader allowed for on-the-fly programming of any device memory, <u>avoiding frequent inefficient recompilation of device and memory onto FPGA</u>

Future Improvements:

- Image compression in hardware allowing for storage of images with greater resolution
- Programmable convolutional array allowing for Convolutional Neural Network inferencing on Coprocessor