Implementing Image Processing Pipelines in a Hardware/Software Environment

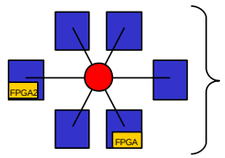
Northeastern University

Heather Quinn[1], Dr. Miriam Leeser
Northeastern University
[1]hquinn@ece.neu.edu

Dr. Laurie Smith King
College of the Holy Cross

Motivation: Accelerate image processing tasks through efficient use of FPGAs. Combine already designed components at runtime to implement series of transformations (pipelines)

Our Environment



- A heterogeneous network of workstations (NOW)
- FPGAs are expensive, available on some hosts but not others
- NOW provide coarse-grained parallelism, FPGAs provide fine-grained parallelism

- Software processing
  - Designed in Java
  - Done on a single host
- Hardware Processing
  - Designed with JHDL (from BYU)
  - Input and output image in FPGA's on-board memory
    - Input image communicated from host to FPGA at beginning of processing
    - Output image communicated from FPGA to host at end of processing
    - Host and FPGA connected through PCI Bus

Hardware Systems

- Using hardware incurs execution costs not present in software systems
  - Hardware initialization,
  - Communicating image, and
  - Reprogramming

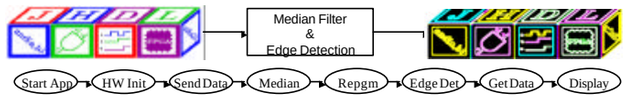
Efficient Use of FPGAs

- Software algorithm's runtime for small images less than the hardware costs
  - Profiling the hardware and software runtimes for different image sizes determines the crossover point
  - Deciding at runtime to execute in software or hardware is simple for one algorithm processing one image

Image Processing Pipelines

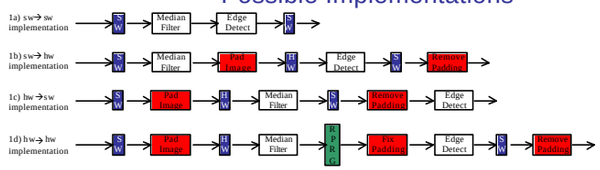
- Series of image processing algorithms applied to an image
- Each algorithm has a software and hardware implementation
- Finding the crossover point for a pipeline is complicated
  - Exponential number of implementations
  - Reprogramming costs
- Need a strategy to find a fast pipeline implementation at runtime

An Example



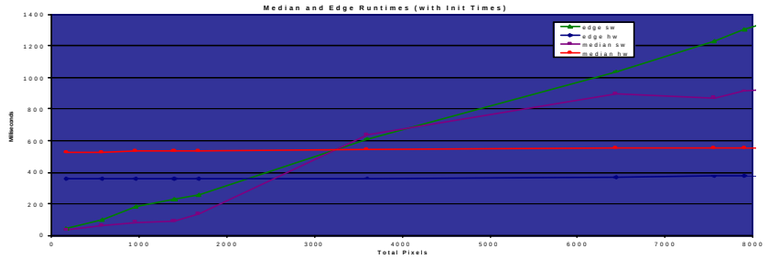
Need pipeline implementations that minimize reprogramming and communication costs

Possible Implementations

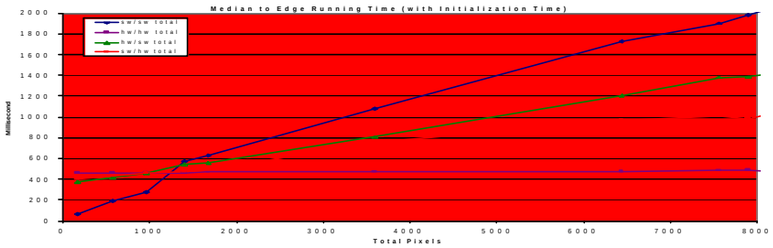


- Blue boxes are hw/sw boundaries
- Red boxes are fixing image edges
- Green Boxes are reprogramming

Median Filter and Edge Detection Profiles



Median Filter→Edge Detection Profiles



Problem Statement

- Inputs: a profiled library of image processing implementations (components), a pipeline, and an image
- Output: an assignment of each component to a hardware or software implementation

The Library of Components

- Each algorithm has two implementations: hardware and software
- Each implementation has known runtimes for a set of images
  - Interpolation used for rest of image sizes
- Each hardware implementation has a known area size
- All components are image in/image out

Assumptions

- Reprogramming and Communication costs incurred at sw/hw boundaries
- Might need to fix image edge in between components
- Problems sizes of 20 or fewer stages
- 500 ms to make a decision

Solving Pipeline Assignment

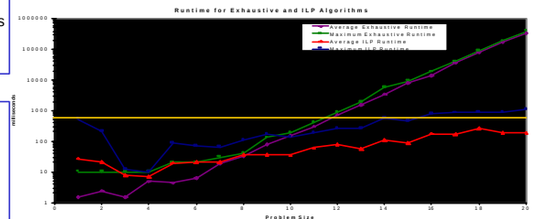
- Exhaustive Search
  - Find: Optimal solutions
  - How: Search entire problem space
  - Algorithm Runtime: $O(2^N)$, where N is the number of pipeline stages
- ILP
  - Find: Optimal solutions
  - How: AMPL model running on CPLEX
  - Need: ILP formulation of the problem statement
  - Algorithm Runtime: Unknown
- Greedy
  - Find: Sub-optimal solutions
  - How: Make optimal decisions for each pipeline stage based on hardware area usage and speedup values
  - Algorithm Runtime: O(N), where N is the number of pipeline stages
- Local Search
  - Find: Sub-optimal solutions
  - How: Improve upon initial solutions (found through Greedy or randomly)
  - Algorithm Runtime: runs for user supplied amount of time

Experiments

- Synthetic components arranged into pipelines of length 1 to 20
- Exhaustive algorithm run to completion
  - Used as a baseline for solution quality
  - Timed to find 500 ms boundary
- ILP solver constrained to 500 ms
  - Ability to solve dependent on components
- Local Search returns best solution found within time limit

Results

- Exhaustive: optimal solutions for 11 stages
- Optimal solutions in 500 ms
  - Exhaustive: up to 11 stages
  - ILP: all pipelines up to 13 stages, some pipelines up to 18 stages, and none larger
- Sub-optimal solutions in 500 ms
  - Greedy and local search: all problem sizes
- Strategy
  - Exhaustive and ILP up to 13 stages
  - Greedy or local search for more than 13 stages



Future Work

- ADAPT: Algorithm that calls exhaustive, ILP and local search algorithms to solve pipeline assignment problem based on problem size
- Decision Time: Study how the amount of time allotted affects ADAPT results
- Virtex II Pro: Add scheduling support for using embedded Power PC cores

Publications

L. Smith King, H. Quinn, M. Leeser, D. Galatopoullos and E. S. Manolakos, "Run-time Execution of Reconfigurable Hardware in a Java Environment", *International Conference on Computer Design*, September 2001.

H Quinn, M. Leeser, and L. Smith King, "Accelerating Image Processing in a Software/Hardware Environment", MAPLD International Conference, September 2002.