Parallel Graphical Processing in Ada

Michael Ward

Dependable Computing Systems Centre

BAE Systems

Mward@cs.york.ac.uk

Synopsis

- Image manipulation systems
- The problem
- Solution
- Implementation
- Demo
- Suggestion
- Conclusions

Image Manipulation Systems

- Take a video stream input, alter the image in some way, and output a video stream.
  - Simple manipulations (e.g. sharpen, edge detect) can be done to the stream
  - Complex manipulations (image morphing or warping) are done on entire frames.

IMS Implementations

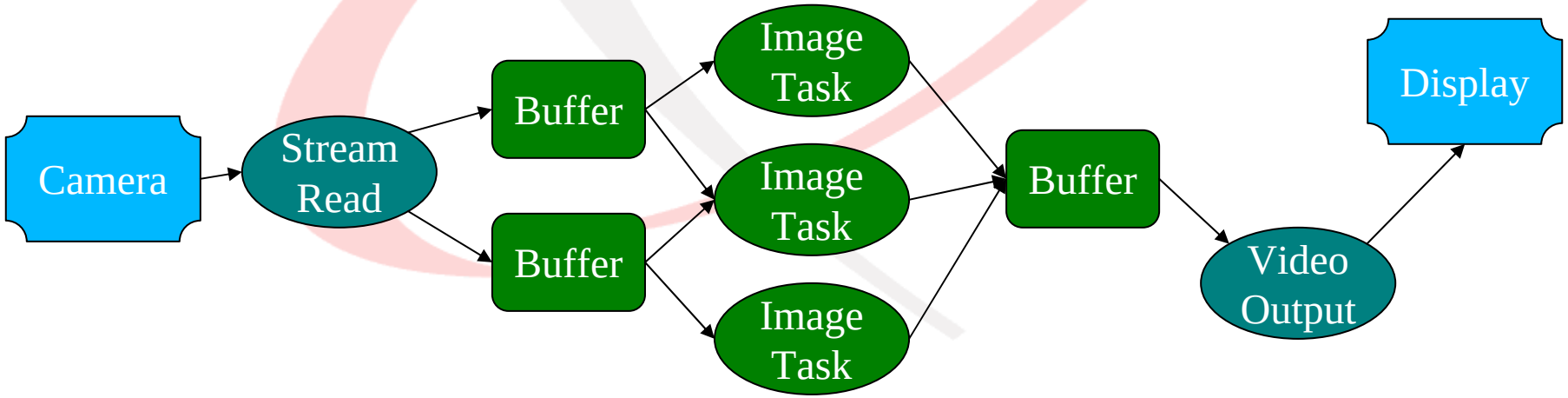
- Traditional implementation uses a processor
  - Either graphical or general purpose
  - Programmed using graphics libraries
- For a dependable image system
  - Need to use trusted source/compiler
  - Running on trusted hardware
- But there are limits to the performance of these systems
  - Driving high-resolution, high-refresh graphics is hard
  - Moving beyond the limits of dependability

The Problem

- To design an image manipulation system capable of dealing with high-resolution, high-refresh graphics systems
  - Must be dependable
    - Safe language to be used – preferably SPARK / Ravenscar
  - Minimal delay on the output image
- Initial demo to implement an image warp
  - For distortion correction
    - E.g. correcting for imperfect optics
    - E.g. pre-distortion for display on shaped surfaces

Solution Suggestion

- Input and output image buffers
  - Split to allow multiple accesses at once
- Parallel image processing tasks
  - Each processes part of the image
- Implemented on FPGA using YHAC



YHAC

- York Hardware Ada Compiler
  - Compiles Ada programs to hardware circuits
  - Allows parallel implementations of concurrent systems
  - Produced circuit is traceable back to the source code
  - Circuit can be easily analysed for resource usage
    - Time and space
    - All done at source code level
  - Provides provable single threaded performance equivalent to mid-range processors

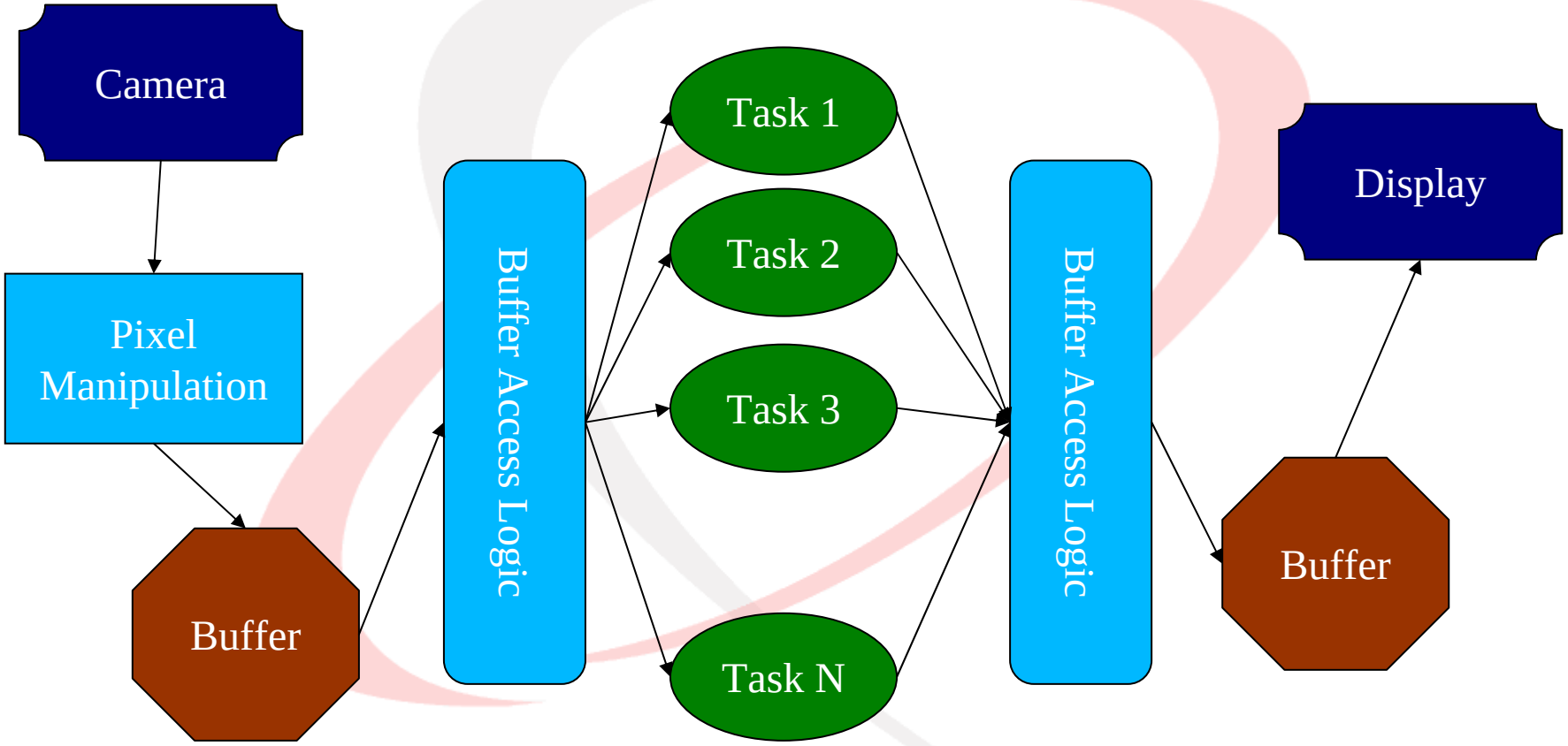
Solution Issues

- Ada does not have a broadcast semantic
  - So buffer filling is harder
    - Cannot broadcast to multiple POs
    - More time required per input pixel
      - But have a definite limit to timing of the input stream
- Cannot release processing tasks at same time
  - Increases system latency
  - More so due to Ravenscar restrictions
- Need cycle accurate timing on output
  - YHAC provides deterministic timing
    - But no definite timing

Implementation

- An image manipulation system
- Takes a video stream and ‘warps’ it
  - To allow for, e.g. optics correction or surface projection
  - In this case a good image is ‘fish-eyed’
- System generates a 512x480 image @ 65Hz
  - Restricted by available memory and display adaptor
  - Image sourced from a composite video camera
- Warping is done frame by frame
  - Uses two image buffers (hence memory restriction)
  - Delays image by one frame

Implementation



Implementation

- Image processing handled by 9 separate tasks
  - Compiled from Ada source using YHAC
  - Each handles part of the image
    - Currently implements a single transform
    - Pre-computes transform for high performance on
  - Communicate with rest of system through internal ports
- System uses dedicated hardware to capture input stream to buffer, and generate output from buffer
  - Provides some pre-processing of the image
    - Conversion to RGB, clipping to size
  - Also provides pipelined access to the buffers
    - Can't be done from within the Ada compiler
  - Scalable to allow more tasks to be used
    - For larger, faster streams, or more complex transforms.

Language Suggestion

- We have shown a use for a ‘broadcast’ and parallel entry call semantic
  - Broadcast to allow the same data to be transmitted to multiple protected objects.
  - Parallel entry call to allow multiple tasks to receive data at the same time
- Multiple Entry:
  - Similar to entry semantic except:
    - Guard is a boolean variable – reset to false on access
    - Must be read-only - hence need for guard reset
  - When guard is released, all tasks waiting gain access to PO
    - In parallel system, all calls become active as readers
    - In a concurrent system, behaves the same way as a traditional ‘last one out closes the door’ implementation on entry queue.

Conclusions

- Have demonstrated that Ada can be used to implement high-integrity and high-performance graphics
- Produced a scalable and flexible graphics framework that can be used for any graphical processing application
- Raised the idea of a parallel entry call semantic for the Ada language