

Using the ClearSpeed CSX700 processor in Video and Image processing applications.

Author: Tim Styles
ClearSpeed Technology Ltd

Introduction

Modern imaging applications require more processing than is currently available using a serial DSP. Often a farm of processors is used, leading to a bulky and inefficient solution. FPGA devices can provide the necessary performance, but at the expense of increased development cost and reduced flexibility.

ClearSpeed's processor architecture is ideally suited to image processing and machine vision applications. The CSX700 processor chip offers world leading power efficiency and a level of performance that makes real time processing for complex image processing algorithms possible in mobile and power sensitive environments where other solutions would not prove capable.

The ClearSpeed Software Development Kit (SDK) includes a C compiler which together with a state of the art profiler and math libraries means that existing applications and algorithms written in C may be easily ported and optimized to run on the ClearSpeed architecture, enabling large reductions in development time and cost.

Built around the CSX700, the Advance e710 PCIe card enables development, laboratory demonstration and proof of concept prior to development of your hardware.

ClearSpeed has partners actively working on:

- Machine Vision
- Target Tracking
- Intelligent Cameras
- Video Analytics

CSX700 Processor

The CSX700 is a multi-threaded array processor (MTAP) with a high degree of replication for efficiency and reliability. Performance is achieved from the high level of internal parallelism. Each one of the 192 processing elements on the CSX700 includes a floating point multiplier, a floating point adder, an integer ALU, a 16-bit integer MAC, 6KB of RAM and a 128 byte register file. The register file allows single byte accesses, optimum for 8-bit image processing operations.

Image processing

A typical image processing application will distribute an image across the array of processing elements such that each element processes one tile. A 1280x720 pixel image can be distributed into the 6 KB of RAM on each PE for processing in a single pass, while an HD 1920x1080 image can be processed in two passes.

Once the image has been distributed, the image tiles are processed in parallel. A data path between neighbouring PEs allows values to be compared or accumulated across all the tiles. For example: Auto exposure, Auto contrast, White balance and Gamma correction.

Image tiles can be distributed with an overlap to enable processing using neighbouring pixels. For example: Noise filtering, Scaling, Edge enhancement, Corner detection, Auto Focus and Perspective correction.

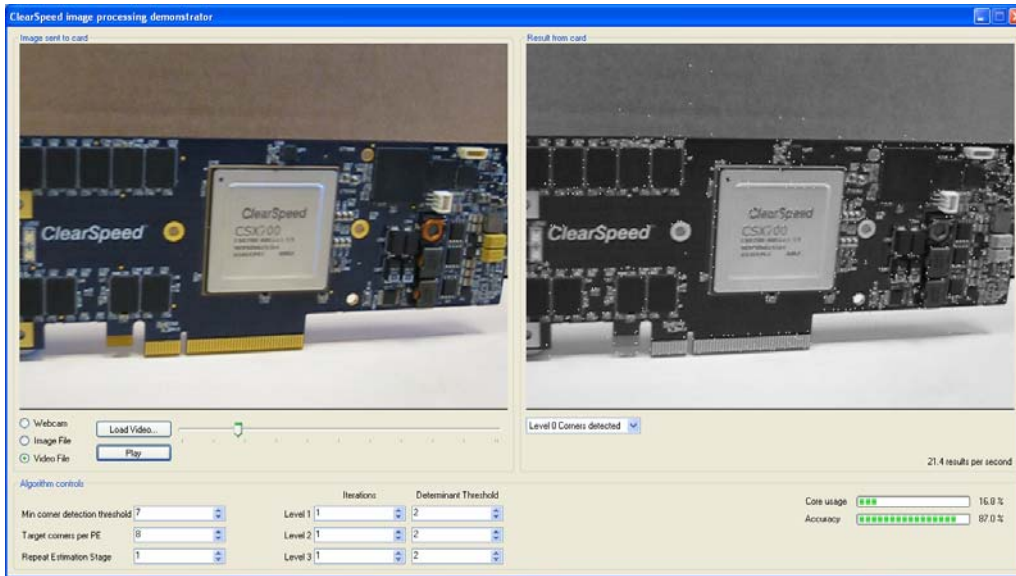


Figure 1: Screen shot showing corner detection. All processing is performed on the CSX700 in real time, and the algorithm parameters can be adjusted via the GUI while the demonstration is running.

Video processing

A typical video processing application will distribute a new frame across the array of processing elements while also keeping the previous frame on the array. Two 800x600 frames can be distributed into the 6 KB of RAM on

each PE, while two HD 1920x1080 frames can be processed in four passes.

Once the frame has been distributed, the tiles from the two frames are processed in parallel. For example: Motion detection, Feature tracking, Video stabilization and Video encoding.

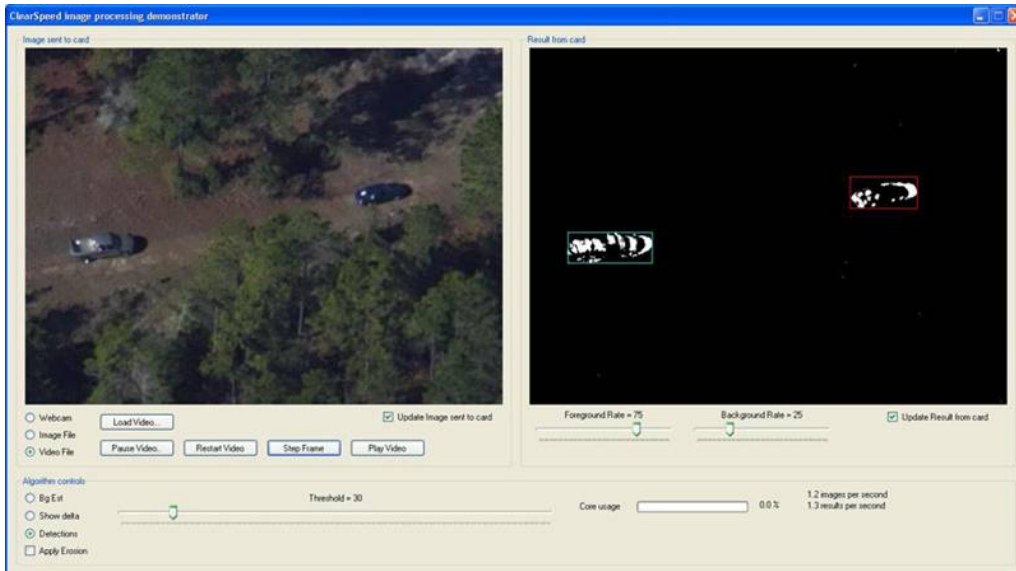


Figure 2: Screen shot showing background removal. The video footage is from a moving camera, so the processing must track the background movement and highlight objects moving relative to the background.

Advantages


The performance of the CSX700 is similar to that of a state-of-the-art FPGA, but with the following advantages common to software programmable devices:

- The most suitable algorithm can be selected, dependent on the data, with no delay or loss of efficiency.
- The algorithm can be easily developed and debugged in real time on the same processor that is used in the final product (see demonstrator screen shots).
- The same hardware is used for each step in the algorithm, so there are no idle stages typical in a fixed function pipeline.
- The core frequency can be scaled to achieve a desired performance while minimizing power consumption.
- Floating point hardware support avoids the need to manage integer overflow and underflow.

- Image processing example using compiler generated code developed in 3 weeks:
- 200 us to transfer a 640x480 image on to the PE array
- 460 us to resample the image to 320x240 with a 5x5 Gaussian filter
- 1500 us to find corner features in a 320x240 image using the FAST algorithm
- 1500 us to track corners in a 320x240 image using Lucas Kanade algorithm and 3 resolutions

Example

The CSX700 is programmed in C, with a small extension to identify variables being processed on the array. In the following example, loop control variables x and y are processed by the array controller, while the image tiles src_tile and dst_tile are scaled by the processing elements in the array.



```

// Stores in 'dst_tile' a down sampled version of the image in 'src_tile'.
// 'dst_tile' and 'src_tile' can be the same, and the top of the tile will
// be overwritten with the result.
// Down sampling uses a 3x3 gaussian filter, so the size of the input
// tile must be at least 2n+1 on each side, for an output of n on each
// side. 'result_size' defines the size to be processed (2n -> n) and
// 'source_width' defines the actual width of the input tile buffer (>2n)
//
void scale_down( poly unsigned char * dst_tile,
                 poly const unsigned char * const src_tile,
                 unsigned short result_size,
                 unsigned short source_width )
{
    char x, y;

    for (y=0; y < result_size; y++)
    {
        char yy = (2*y);

        // Pointers at the start of row 2y and the following two rows
        poly unsigned char * r = &src_tile[(yy+0) * source_width];
        poly unsigned char * s = &src_tile[(yy+1) * source_width];
        poly unsigned char * t = &src_tile[(yy+2) * source_width];

        // Calculate the weighted sum of pixels to the left
        poly short sideA = *(r++) + (2 * *(s++)) + *(t++);
        poly short middle, sideB;

        for (x=0; x < result_size-1; x+=2)
        {
            // Calculate the weighted sum of the centre pixel and
            // the pixels above and below
            middle = (2 * *(r++)) + (4 * *(s++)) + (2 * *(t++));

            // Calculate the weighted sum of pixels to the right
            sideB = *(r++) + (2 * *(s++)) + *(t++);

            // Store the weighted sum of all nine pixels
            *(dst_tile++) = (sideA + middle + sideB) >> 4;

            // Use the previous weighted sum of pixels to the right
            // as the new weighted sum of pixels to the left

            // Calculate the weighted sum of the centre pixel and
            // the pixels above and below
            middle = (2 * *(r++)) + (4 * *(s++)) + (2 * *(t++));

            // Calculate the weighted sum of pixels to the right
            sideA = *(r++) + (2 * *(s++)) + *(t++);

            // Store the weighted sum of all nine pixels
            *(dst_tile++) = (sideA + middle + sideB) >> 4;
        }

        // One more iteration if result_size is odd
        if (x < result_size)
        {
            middle = (2 * *(r++)) + (4 * *(s++)) + (2 * *(t++));
            sideB = *(r++) + (2 * *(s++)) + *(t++);
            *(dst_tile++) = (sideA + middle + sideB) >> 4;
        }
    }
}

```

Figure 3: Example code extract. The full source code is available from the video processing page under the applications tab at www.clearspeed.com

Specification

CSX700 Performance

- 250MHz core clock frequency
- 96 GFLOPS single or double precision
- 75 GFLOPS sustained double precision DGEMM
- 16 GFLOPS sustained complex floating point 2D FFT
- 9W typical power dissipation
- 192 Gbytes/s internal memory bandwidth
- 2 x 4 Gbytes/s external memory bandwidth
- 4 Gbytes/s chip-to-chip bandwidth

CSX700 Features

- Single byte accessible register file and high bandwidth PE memory
- PCIe x16 host interface

- 2 x 64-bit DDR2 DRAM interface with ECC support
- 256 Kbytes on-chip scratchpad memory
- On-chip instruction and data caches
- ECC protection on all on and off-chip memory
- ClearConnect NoC provides on-chip and inter-chip data network
- Host debug port
- 64-bit virtual, 48-bit physical addressing
- On-chip DMA controller

For high volume applications ClearSpeed IP is ideal for creating a low cost, even lower power processor capable of high resolution image processing in real time, with a performance level tailored to your requirement. The technology is supported by a mature C based SDK that enables a large reduction in development time and cost.

For further information about the full-range of products provided by ClearSpeed Technology, visit www.clearspeed.com, or contact info@clearspeed.com.

Copyright 2010 ClearSpeed Technology Ltd. The information contained herein is subject to change without notice.
ClearSpeed shall not be liable for technical or editorial errors or omissions contained herein.

ClearSpeed, Advance, and CATS are trademarks or registered trademarks of ClearSpeed Technology Ltd. All other marks are the property of their respective owners.