

# ClearSpeed Acceleration – Is It Right For Your Application?

Kristopher R. Buschelman,  
ClearSpeed Technology, Inc.  
Engineering Manager: CSXL

## What does ClearSpeed's coprocessor do?

- **New, parallel co-processor, the CSX600**
  - Accelerate compute-intensive math libraries on general purpose CPUs
  - Can be integrated next to the main CPU on the motherboard
  - Or installed on add-in cards, e.g., PCI-X, PCI Express
  - Or embedded, e.g., aerospace, auto, medical, defense
- **Significantly accelerate libraries & applications**
  - BLAS, LAPACK
  - MATLAB, Mathematica, Amber, etc.
  - In-house codes using SDK & porting code
- **ClearSpeed Advance board is aimed at the server market**
  - dual CSX600
  - Over 66 GFLOPS for DGEMM
  - 133MHz PCI-X, PCIe (x8)
  - Low power – around 35 Watts

## Overview

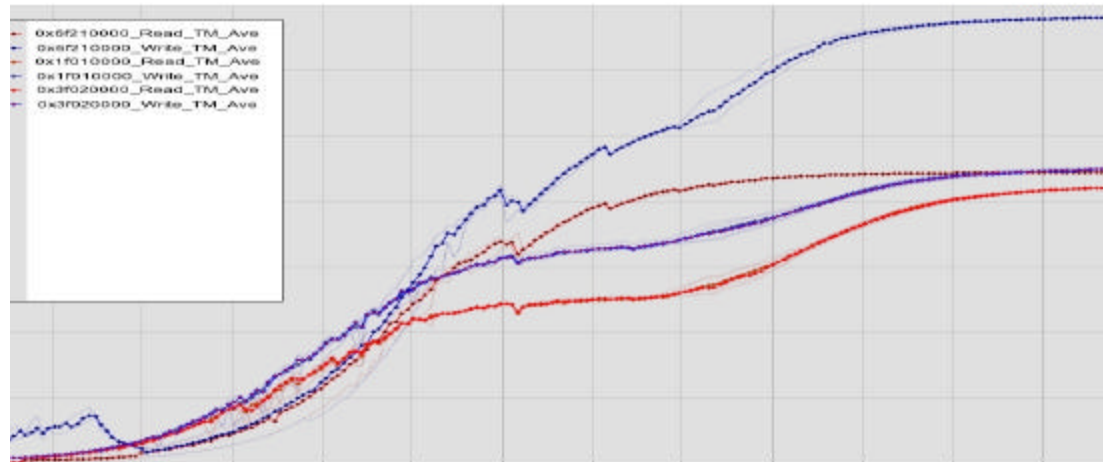
- **Common Sources of Bottlenecks**
  - PCI Bus
  - On-board Memory
- **SIMD Computing**
  - What is it?
  - What's special about ClearSpeed's implementation
- **A few example kernels**

## PCI Bus

- **ClearSpeed boards utilize either PCI-X or PCIe busses**
  - PCI-X 133 MHz: 1 GB/s Peak
  - PCIe x8: 1.6 GB/s Peak
- **Available memory on board**
  - 1 GB of 200 MHz DDR2 SDRAM shared by 2 CSX600 processors
- **Must consider both the transfer rate AND the available memory**
  - If application requires more memory, then more communication to the board is necessary
- **Infinitely fast board**
  - $\text{Time} = \text{Bus Speed} * \text{Total data size transferred}$

## PCI Bus

- **Driver performance is very machine specific and depends on transfer size, direction, etc.**
  - Transfer Size vs. Transfer Rate



- See Runtime User's Guide for current driver performance

## On-board Memory

- **2 level memory hierarchy**
  - 1 GB “mono” shared memory
  - 6 kB “poly” memory per processing element (PE)
    - $6 \text{ kB/PE} * 96 \text{ PE} = 576 \text{ kB per CSX600}$
- **Peak bandwidth between levels**
  - $3.2 \text{ GB/s} * 2 \text{ chips} = 6.4 \text{ GB/s}$
- **Must consider both the transfer rate AND the available memory**
  - If application requires more memory, then more communication to the board is necessary
- **Infinitely fast PEs**
  - $\text{Time} = \text{Bus Speed} * \text{Total data size transferred}$
- **Secondary considerations**
  - Burst size: 64 Bytes/PE (i.e., 8 doubles)
  - Transfers can be smaller, but at reduced efficiency

## SIMD Computing

- **What is SIMD?**
  - Single Instruction, Multiple Data
    - Each PE sees the same instruction stream
    - Each PE issues “load”, “multiply”, etc., simultaneously
    - But acts on different data per PE
  - PARALLEL COMPUTATION
- **ClearSpeed SIMD is enhanced by:**
  - Local memory for each PE
    - data management is easier within “poly” memory
    - does not require adjacent access for all 96 elements involved in the computation from shared memory pool
  - PEs can be enabled/disabled
    - not required to use all PEs always
    - useful for handling “boundaries”

## SIMD Array

- **96 PEs per CSX600**
  - 210 MHz
  - double precision multiply-accumulate per cycle
  - 4 cycle pipeline depth for multiply and accumulation
    - For top performance, use operations on 4 element vectors on each PE
- **Nearest neighbor communication**
  - “swazzle” path topology is a line or ring
  - Bandwidth: 8 Bytes per cycle between reg files
    - $8 \cdot 96 \cdot 210 = 161 \text{ GB/s}$
    - Useful for fine grained communication

## Possible Kernel

- **Partial Differential Equations**
  - Some are memory bandwidth limited, so not a good candidate for ClearSpeed acceleration
    - small stencil implies little computation per grid point
    - wide, sparse stencil implies large active data set
- **But, some PDE simulations are good candidates**
  - require a small grid, so can run entirely in PE memory (computational finance)
  - have large, dense stencils
    - large amounts of computation per grid point
    - sufficiently small active data set
  - implicit time stepping
    - large systems of equations solved via direct methods
    - direct solvers utilize dense linear algebra kernels
      - (i.e., **DGEMM**)

## Good example kernels

- **Dense Linear Algebra**
  - Matrix-Matrix products (DGEMM)
    - Low memory bandwidth required = high data re-use
    - Inner kernel: Matrix-multivector product
      - 96x96 matrix, x4 vectors
        - » 96x96 matrix due to 96 PEs
        - » 4 vectors due to multiply/accumulate pipeline depth
- **Monte Carlo (computational finance)**
  - “Embarrassingly parallel” task distribution
  - Very little data requirement
- **Molecular Dynamics (Amber, BUDE)**
  - Large numbers of identical tasks can be found
  - Requires small working data sets

## Keys to Success

- **Clearly, parallelism is essential**
- **Proper management of the “poly” memory is also critical to success.**
  - Application must accept memory bandwidth limits
    - PCIe or PCI-X
    - On-board memory heirarchy
  - SDK enables asynchronous data transfers
    - permits efficient “double buffering” to manage data streams, accomodating the size limit
  - Application must employ a small working data set
    - less than 576 kB, distributed across 96 PEs
    - also aware of 1 GB shared memory limit
- **While developing ClearSpeed applications, use the ClearSpeed Visual Profiler to discover what’s actually happening on the board!**

## Oh, yeah

- **Don't forget about the host processor also**
  - Today's multicore hosts are very useful for managing "other tasks" that are not accelerated by ClearSpeed
  - Many applications can overlap these tasks with ClearSpeed accelerated tasks
  - And, profile your host portion of the application as well using any of a variety of tools
    - including ClearSpeed Visual Profiler for CSAPI utilization