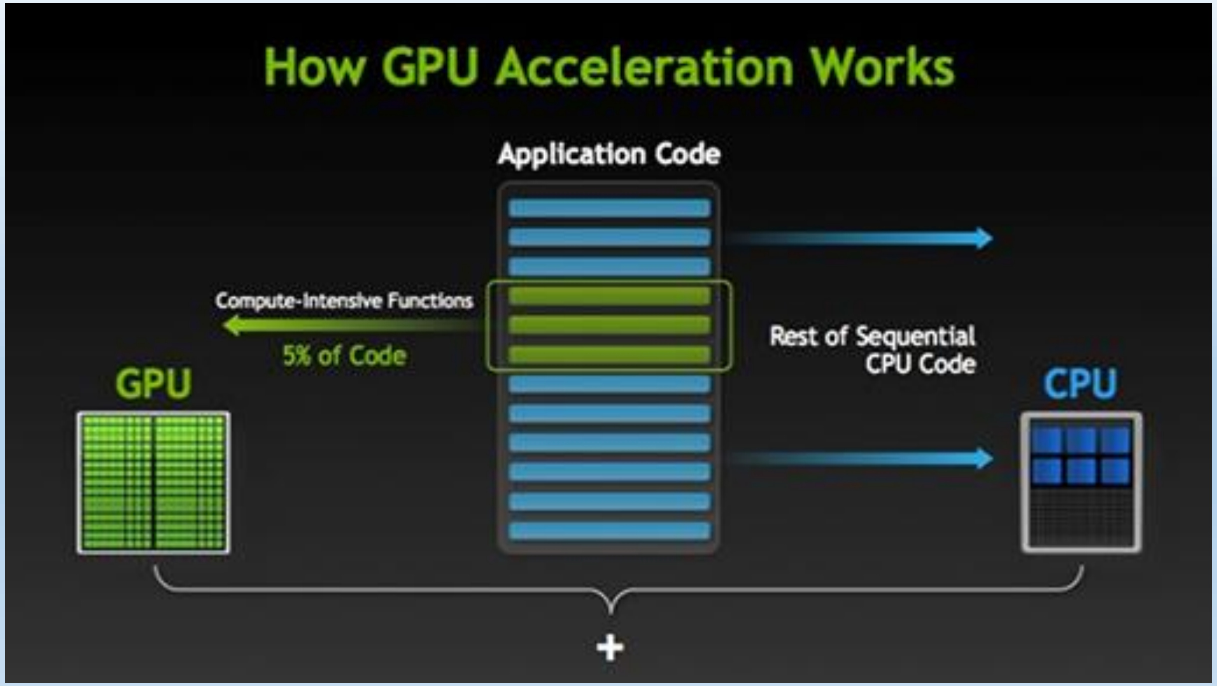
Manycore and GPU Channelisers

Seth Hall
High Performance Computing Lab, AUT

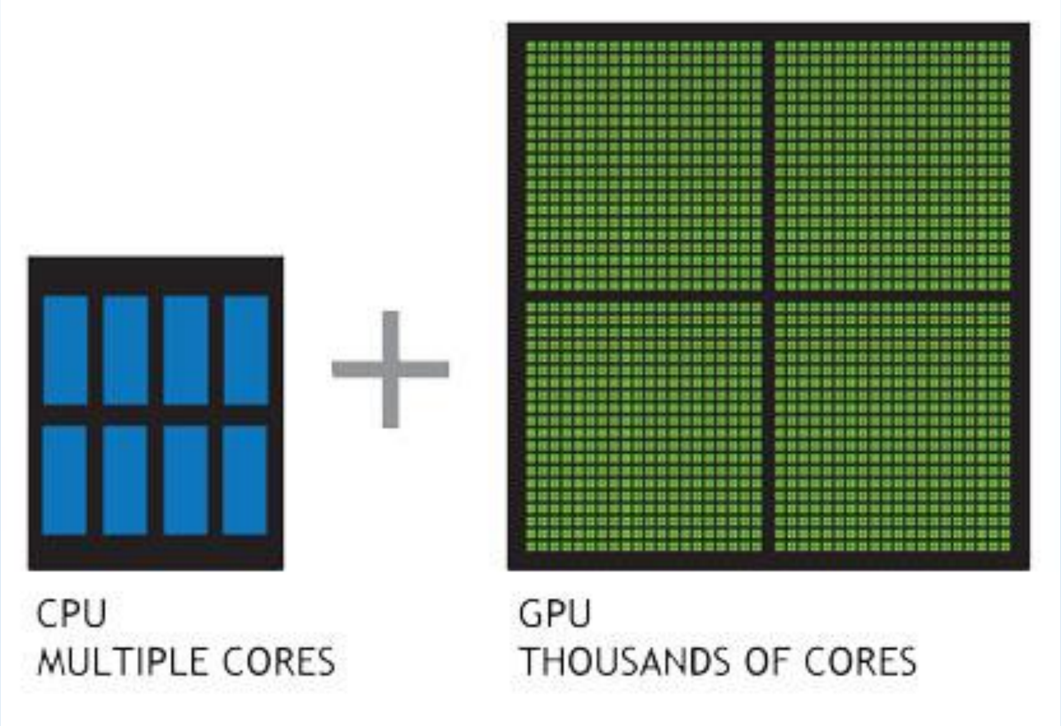
GPU Accelerated Computing

- GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate applications
- GPU-accelerated computing offers unprecedented application performance by offloading compute-intensive portions of the application to the GPU, while the remainder of the code still runs on the CPU


How GPU Acceleration Works
Application Code
Compute-Intensive Functions
5% of Code
GPU
Rest of Sequential CPU Code
CPU
+

CPU vs GPU Computing

- A simple way to understand the difference between a CPU and GPU is to compare how they process tasks
- A CPU consists of a few cores optimized for sequential serial processing while a GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously
- GPUs have thousands of cores to process parallel workloads efficiently



Hardware being Tested

Many-core Architectures

- Adaptiva Epiphany Parallella
- Kalray MPPA

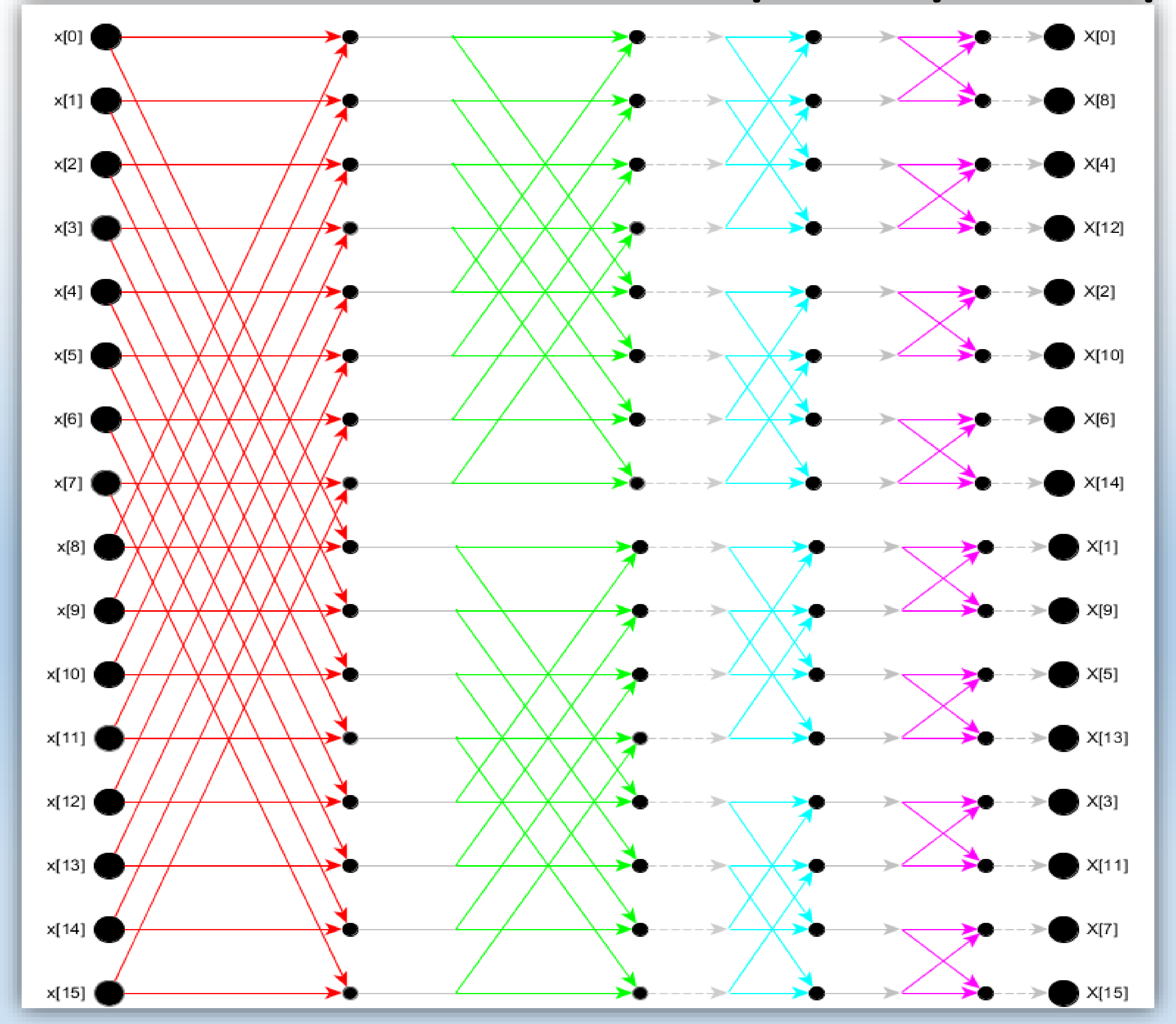
GPU based Architectures

- Nvidia Tegra K1
- Nvidia Tesla K40

Parallelising and implementing FFTs

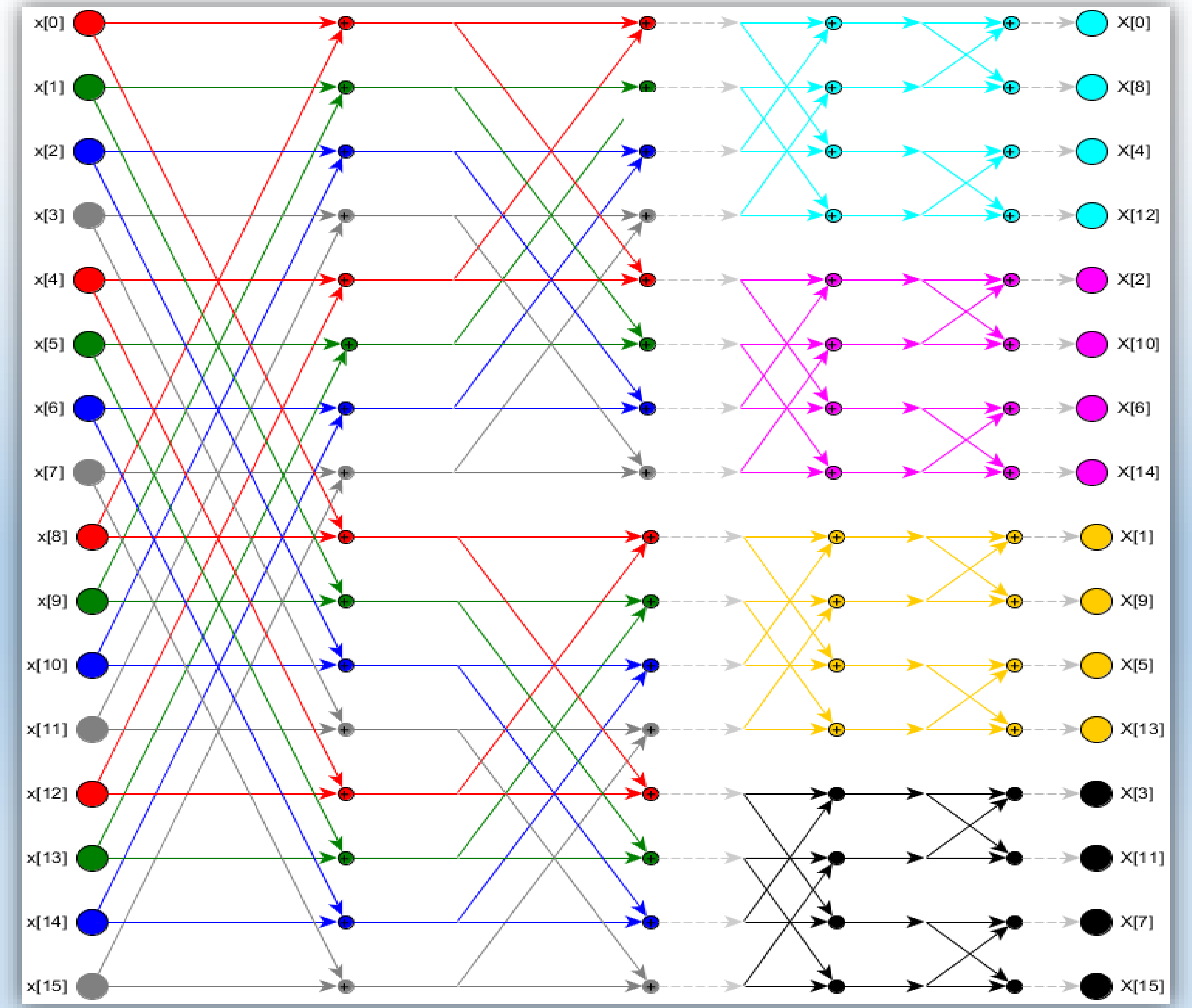
- What we have been focusing on in the HPC Lab is how to efficiently parallelize large FFTs on the Many-core and GPU based architectures.
- Concern not just about timings, but particularly power efficiency of the boards.
- Approach being used is a six-step FFT
- For an *N* size FFT we can break it up into two sets of *m* x *n* number of FFTs
- Example 16 point FFT can be broken up into two series of 4x4 FFTs run in parallel.
- We can choose *m* and *n* depending on the architecture.
- Example how much memory is near the processing cores

Radix-2 Decimation In Frequency 6 Step FFT



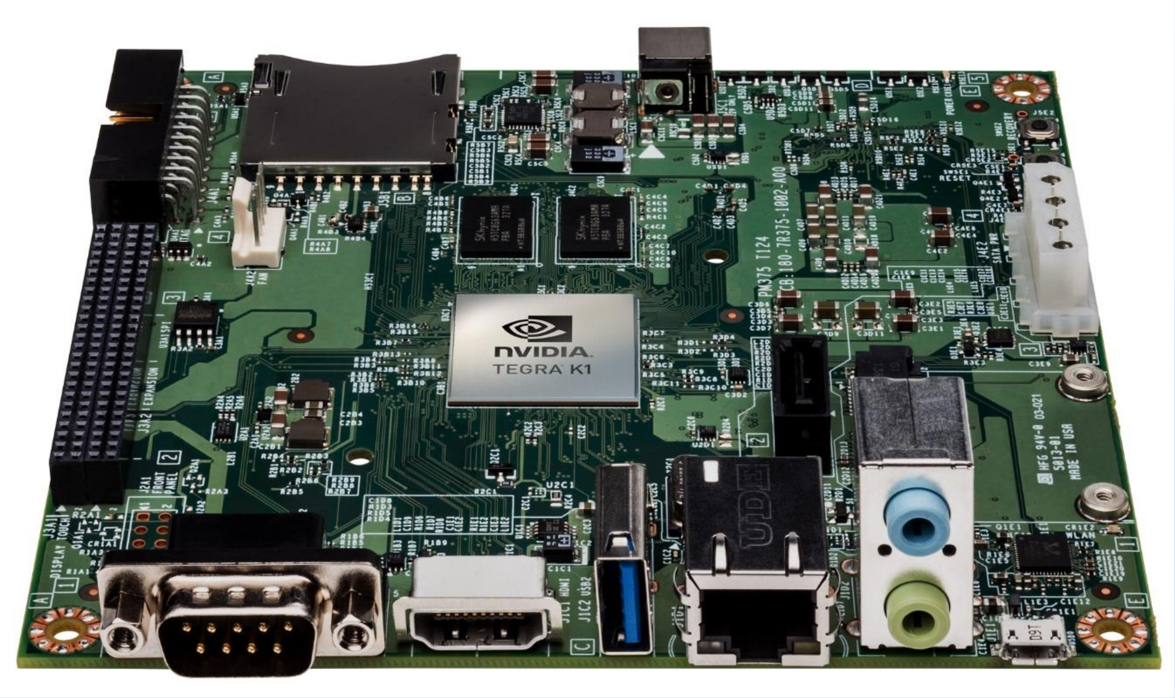
Parallelized Radix-2 Decimation In Frequency 6 Step FFT

1. Rearrange input data (group same colours together)
2. Perform m (number of coloured groups) lots of n (number of points in the group) FFTs in parallel (depending on number of cores available)
3. Multiplied each by twiddle correction
4. Rearrange the data again
5. Perform n lots of m point FFTs all in parallel
6. Rearrange data (bit reversal).



Jetson TK1 Specs

- Tegra K1 SOC
- NVIDIA Kepler GPU with 192 CUDA cores
- NVIDIA quad-core ARM Cortex-A15 CPU + low power companion core
- 2 GB RAM
- Power consumption: 5 watts
- About 50 times faster than Tegra 2
- OpenGL ES 3.1 & CUDA 6.5 support



Tegra K1 vs Tesla K40

- The reason we are mostly looking at more low power GPU (K1) over the more powerful K40 is for power efficiency.
- K40’s board power (235 watts) vs K1’s (5 Watts)
- K40 5 TFLOPS (single precision) K1 0.36 TFLOPS
- Actually K1 better performance in terms of FLOPS per watt of power

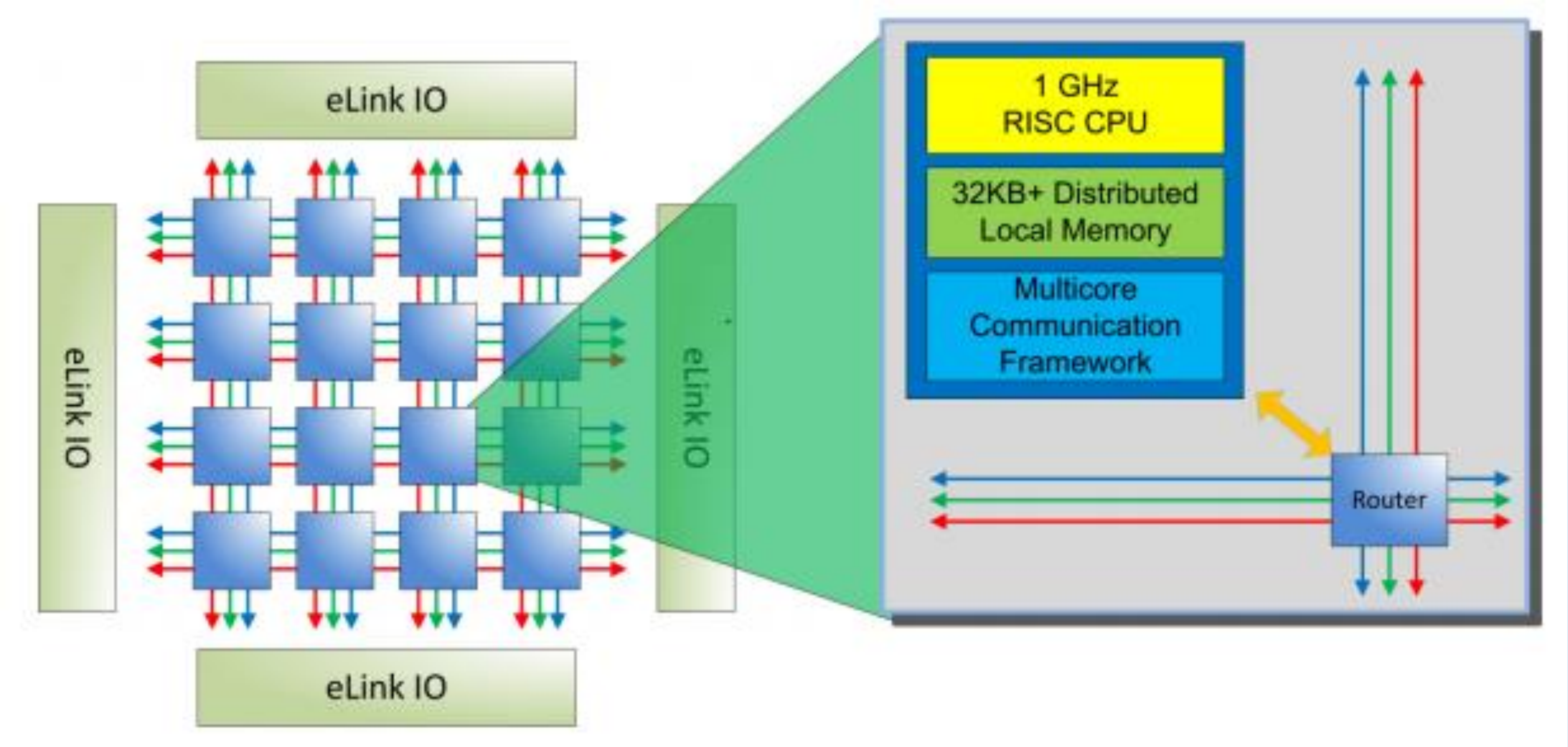
Implementation of FFTs on TK1

- Implementation of FFT on TK1 using CUDA and CuFFT library.
- CuFFT is a closed source library for an FFT implementation using the GPU.
- Support for combining multiple GPU to perform FFTs and also allows batch processing of multiple FFTs in parallel.
- Timings done and discovered can do a 2^{18} FFT in mere milliseconds. Most of the time delay is the I/O data transfers from CPU to GPU and creation of a plan.
- Planning a Open GL ES programmable shader implementation of 6 step FFT and seeing how it compares to CUDA the version.

Epiphany Parallella Specs

- Zynq-Z7010 or Z7020 Dual-core ARM A9 CPU
- 16-core Epiphany co-processor
- 1 GB RAM
- 5 Watts power

Epiphany Architecture

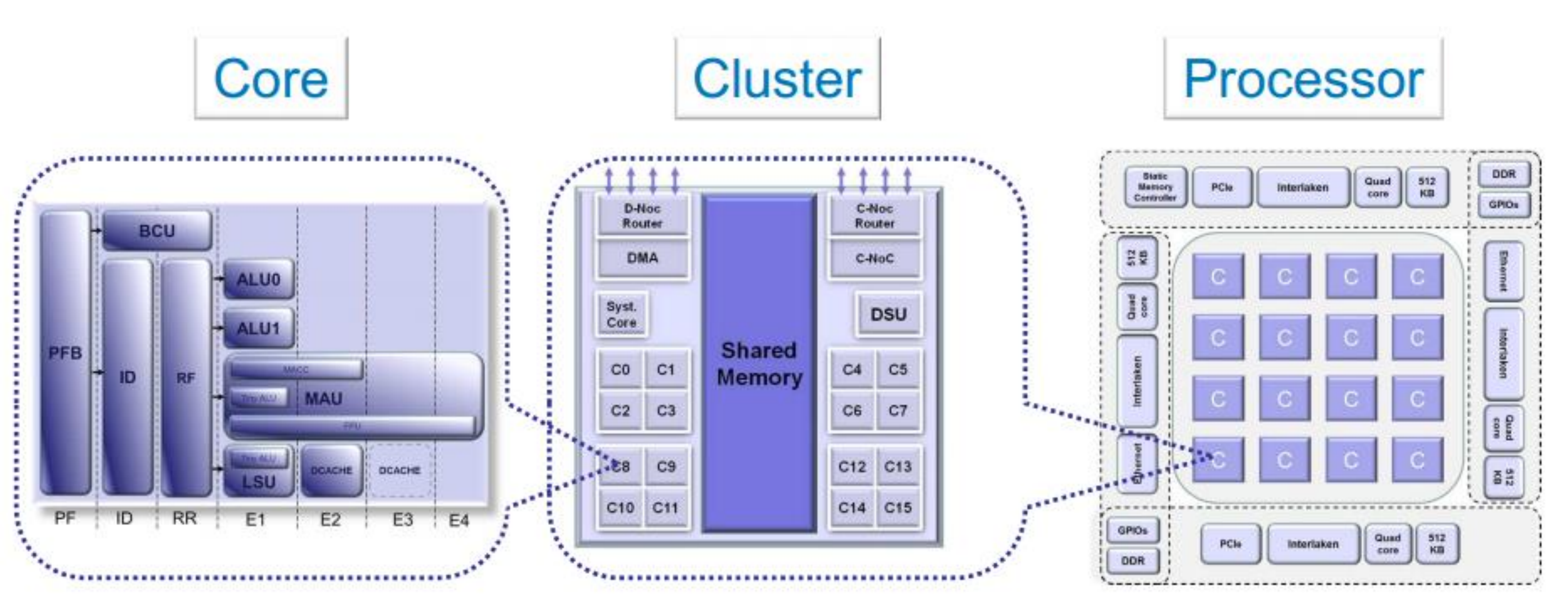


Company claims 4096 RISC cores planned by 2016!

Kalray MPPA Specs

- Massively Parallel Processing Array.
- 5 watts power consumption but can reach up to 10 watts when high processing power is needed.
- 256 cores.
- 2 x PCI Express bus.

Kalray MPPA 256 Architecture



FFTs on Kalray MPPA

- French student Julien Hascoet worked on implementing FFTs efficiently on the Kalray MPPA.
- 2^{18} single precision point FFT was broken up into 512 FFTs of 512 points.
- Using only a single core on a cluster was able to execute an FFT in 110.38mS and using all 16 cores in a cluster 7.94mS (speedup of 13.9)
- Times do not include transfer speeds to memory
- Kalray has another 15 clusters that could be used to perform more FFTs

AFTERNOON TEA TIME!!!!