# Implementation of 2DFFT Processing and Large Dense Matrix Multiply on the TMS320C6678 Processor

Kerry Barnes (kbarnes@gedae.com), William Lundgren (wlundgren@gedae.com), James Steed (jsteed@gedae.com), Amon Arnon Friedmann, Ph.D. (TI, arnon@ti.com), Hector Rivera (TI, hrivera@ti.com)
Gedae, Inc., 1247 N. Church St., Suite 5, Moorestown, NJ 08057, TI, 968 Fairfield Lane, Allen, TX 75013

## Introduction

TI's TMS320C6678 chip is an emerging modern multicore processor. This paper analyzes the implementation of 2 algorithms on this processor. The choice of the 2DFFT and large dense matrix includes algorithms that stress the computation and the movement of data. The data sizes for both algorithms are chosen to require the use of system memory. The 2DFFT was chosen over the SAR because it stresses the memory bandwidth a bit more but still has the necessary large matrix transpose. The 2DFFT processing is limited by the efficiency of data movement to and from system memory and the overlap between data movement and processing. The large dense matrix multiply is an O(N^3) algorithm and should be limited by the computation. Having said that, care must be taken to design the algorithm and avoid inefficiencies associated with long memory strides and narrow memory transfers. In particular, the movement of data through hierarchal memory must be structured to insure efficient use of cache. The throughput achieved on the C6678 processor will be compared with the previous benchmarks on the Cell/B.E. and x86 multicore processors.

The benchmarking of these algorithms will be performed with the use of latest version of Gedae. Gedae is a technology and a set of tools that enables implementation of sophisticated algorithms to optimize the memory footprint and the movement of data among memories. As a result the strategies implemented are more sophisticated than expected yet completed with modest engineering effort.

In this paper we describe the TMS320C6678 architecture. Further, we report the processing throughput achieved for 2DFFT and matrix multiply algorithms on the C6678 processor and compare those numbers with the same algorithms on the Cell/B.E. and x86 architectures. We describe the parallelization and efficiency strategies in detail.

## TMS320C6678 and its Transfer Limitations

The C6678 processor has 8 identical processing cores. See figure 1. Each core has its own 512 kB L2 cache. The data is transferred among all elements of the system via the TeraNet bus. The data transfers are managed by the Multicore Navigator. There is also a shared memory available in the memory subsystem. External memory of up to 8 GB is available via a memory controller with a bandwidth of 12.8 GB/s – much less than the high speed TeraNet bus with its bandwidth of 256 GB/s. Each C6678 core is capable of processing 20 GFLOPS for an aggregate of 160 GFLOPS. The processing core is also fixed point enabled at 40 GMAC/s. When processing large data sets, data must be stripmined from the system memory. In these situations the bandwidth over the memory controller is a bottleneck to the processing speed.

While the C6678 cores can directly access system memory, the hardware cache management for a portion of the memory can be disabled and manually managed from the Multicore Navigator and the memory subsystem using a direct memory access (DMA) API. The MultiCore Software Development Kit (MCSDK) supports transfers for both contiguous and tiled memory transfers.

The TeraNet has an aggregate transfer rate of 250 GM/sec but efficiency is determined by how well the transfers match the characteristics of the TeraNet and memory subsystem. To efficiently use these DMA transfers, several design considerations must be taken into account. In particular, the design must take into account the optimal transfer size and memory alignment. Experiments with these characteristics will be reported in the paper in diagrams as illustrated in Figure 2.

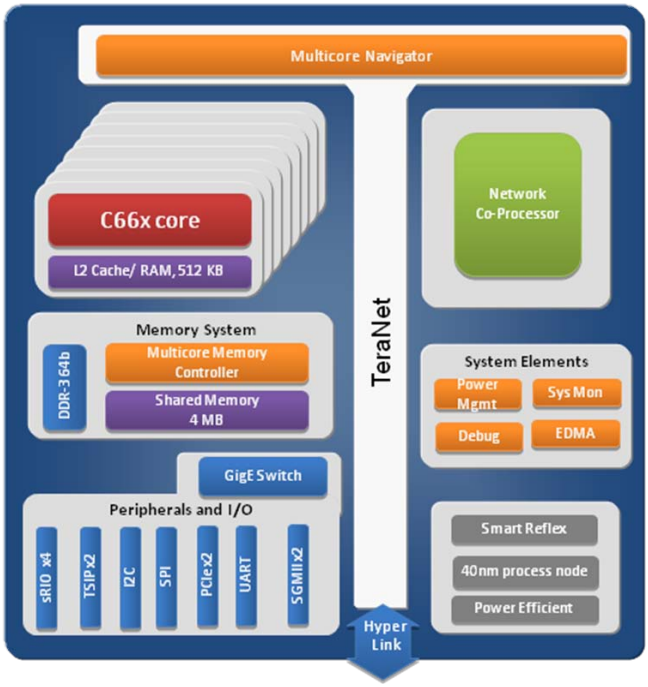


**Figure 1 – The TMS320 processor has 8 Processing Cores. The bandwidth to memory often is a bottleneck.**

## Algorithms

The 2-D FFT is implemented on the Cell C6678 DSP architecture. We consider the data size 512x512 and larger – the data is too large to fit in a single SPE's local storage, and it is too large to fit in the aggregate local storage. Data must be stripmined from system memory, including

intermediate stages (transposes) in the algorithm. We present two algorithms that can accommodate these sizes – a three phase algorithm and a four phase algorithm. The three phase algorithm accommodates larger matrices at the cost of more transfers to and from system memory.

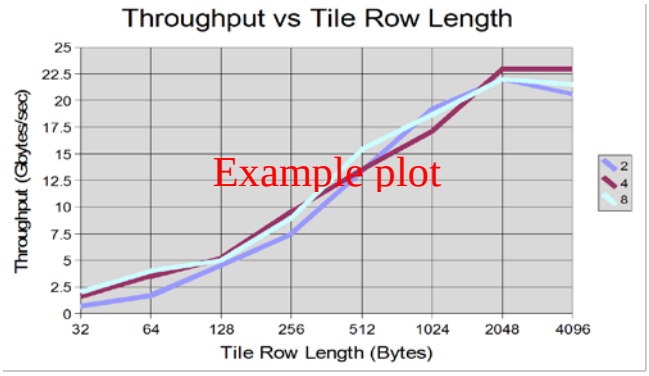


**Figure 2 – Maximum bandwidth to memory is achieved when DMA … .**

The 3 phase algorithm requires 3 transfers of data into and 3 out of system memory. The 4 phase algorithm does the transpose on the fly eliminating 1 transfer into and 1 out of system memory.

The 3 phase algorithm:

- FFT rows into and out of system memory
- Transpose tiles into and out of system memory
- FFT columns into and out of system memory.

The 4 phase algorithm:

- FFT a block of rows moved from system memory
- Transpose a stream of tiles from data block and move into system memory
- Stream tiles into memory and copy into tile into a block of contiguous columns
- FFT the block of columns and stream back into system memory

With both algorithms, the DMA transfers are double buffered to/from system memory to overlap the data transfers with the transposes, memory copies and FFTs.

The matrix multiply is implemented using tiled matrices. The matrix multiply is computed on the submatrices and then accumulated into a final result. The tiles are streamed in and out of system memory using the most efficient transfers available.

## Performance and Conclusions

The Gedae language and compiler are used to achieve high performance with minimal development time. The language allows for direct specification of the tiling of large matrices. The compiler automates a bare metal implementation of the DMA and DMA list accesses. The runtime components are autocoded to introduce zero overhead to the execution time of the algorithms. Additionally, C6678-specific optimization strategies are automatically incorporated.

The upper bound on performance is dependent on the transfer rate over the system memory controller. The theoretical bandwidth limit is 12.8 GB/s. As shown in Figure 2, the maximum we have achieved in the laboratory while simultaneously using all 8 SPEs is xxx GB/s. We use the equation $10*N^2*\log_2 N$ to compute the number of FLOPS for the 2-D FFT algorithm. Assuming a 12.8 GB/s bandwidth, the upper bound is 36 GFLOPS. Assuming a xxx GB/s bandwidth, the upper bound is yyy GFLOPS. For the current implementation we are measuring zz.z GFLOPS for the four phase algorithm, which equates to an aa.a GB/s sustained bandwidth over the memory controller. An example Trace Table of this execution is shown in Figure 3.

Modern chips add more coprocessors with wider vector ALUs offering great promise of high throughput. As the chips provide more FLOPS, the programming challenge moves towards keeping the pipelines sated with data. Gedae provides a powerful platform for programming and debugging these bandwidth issues.

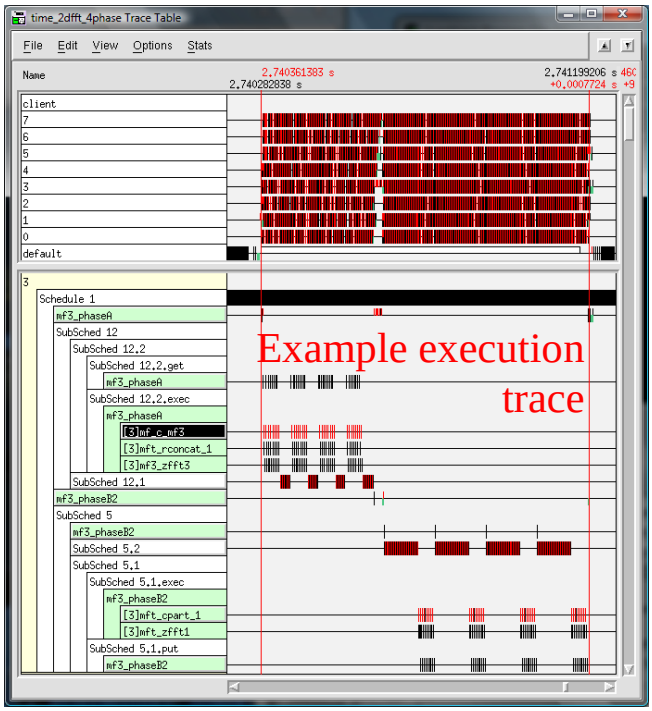


**Figure 3 – Execution trace in the Gedae trace table shows the loading of the 8 processors and the … .**

## References


[1] "TMS320C6678, Multicore Fixed and Floating-Point Digital Signal Processor", Literature Number: SPRS691, November 2010

[2] Lundgren, W. et al. "Implementation of 2-D FFT on the Cell Broadband Engine Architecture," HPEC, 2009

[3] Lundgren, W. et al. "Simple, Efficient, Portable Decomposition of Large Data Sets," HPEC, 2008.