The Development and Performance Analysis of a Distributed Corner Turn in Multicore Embedded Systems using the AXIS Graphical Software System

Tom Litrenta, Application Engineer
Radstone Embedded Systems, Part of GE Fanuc Embedded Systems
tom.litrenta@radstone.com

Modern Radar System Requirements

Modern day radar systems need to perform sophisticated computationally intensive algorithms and these systems must be designed with space constraints[1].. Even single channel radar systems require many processors to keep up with today's demanding real time I/O requirements. Thus multicore processors are an extremely attractive alternative to this marketplace.

Processing Radar Data with Multiple Processors or Processor Cores

Typically radar data is digitized, and placed in a single processor. Next the data is scattered to a collection of distributed processors. Each of the processors performs one or more FFT's on each row of data. This row data is referred to as the fast time dimension. This range compression step can be performed optimally as each row of range data is contiguous in memory.

Then it is desired to perform FFT's on each column of data. This column data is often referred to as the slow time dimension. However each column of data is now distributed across many processors and is therefore NOT contiguous in memory. Any FFT on noncontiguous data is woefully slow!

Therefore data is redistributed so that each column of data is placed as a row of data on a single processor. This data is now contiguous in memory and can be processed optimally. This redistribution of data has been historically referred to as a corner turn. In many systems the corner turn operation is the most time consuming operation and may be the limiting factor for scalability.

The Distributed Corner Turn Algorithm and Performance

Since the distributed corner turn can often become the most time consuming segment of a radar application, it is imperative to implement it with a highly efficient algorithm.. Two considerations must be especially kept in mind:

- Processor utilization
- Memory and cache usage

We have implemented an algorithm which is efficient in both regards. For the two dimensional data matrix, diagonal tiles are transposed in place to assure optimal memory usage. Non-diagonal tiles are transposed and swapped across processors as shown in figures 1 and 2.

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)

Figure 1 – Two Dimensional Matrix BEFORE corner turn and viewed as tiled for a 4 processor system

T(1,1)	**T** (2,1)	**T** (3,1)	**T** (4,1)
T (1,2)	**T** (2,2)	**T** (3,2)	**T** (4,2)
T (1,3)	**T** (2,3)	**T** (3,3)	**T** (4,3)
T (1,4)	**T** (2,4)	**T** (3,4)	**T** (4,4)

Figure 2 – Two Dimensional Matrix AFTER corner turn and viewed as tiled for a 4 processor system

Note that **T** is the transpose operator so **T**(2,1) in figure 2 is the transpose of input tile (2,1) in figure 1.

A summary of the results is shown below for 4kx4k complex input data.

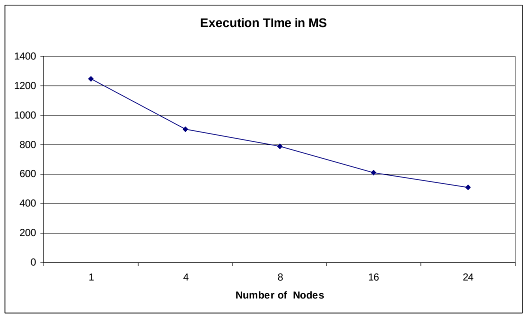


Figure 3– Performance Improvement of Corner Turn as the Number of Processors Increases for a PowerPC 7447 System

The role of the AXIS graphical software system will be examined in the development of this distributed corner turn.

The AXIS Graphical Software System

AXIS is an unique integrated graphical toolset for multiprocessor software development and performance analysis.[2] AXIS is ideally suited to multicore processors as they must be treated as a multiprocessor system for application development purposes.. Some of the components of AXIS are described below.

HardwareView is used to identify various heterogeneous system components and I/O connections and to make sure that the components needed to perform the corner turn are properly connected.

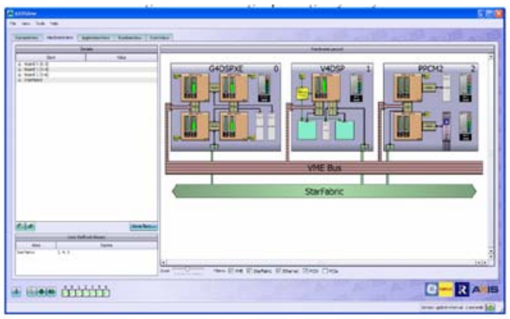


Figure 4 – HardwareView showing system connections

ApplicationView is a graphical tool, for building multiprocessor applications. I/O channels between tasks are graphically defined. The file generated by ApplicationView is used by AXISFlow, which is similar to MPI, and performs the data distribution for the corner turn

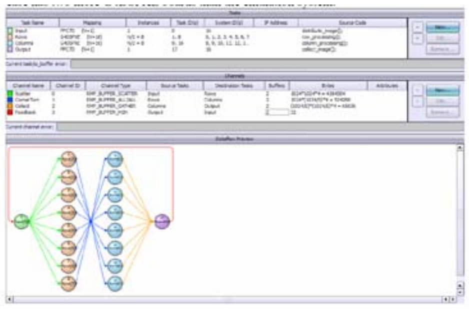
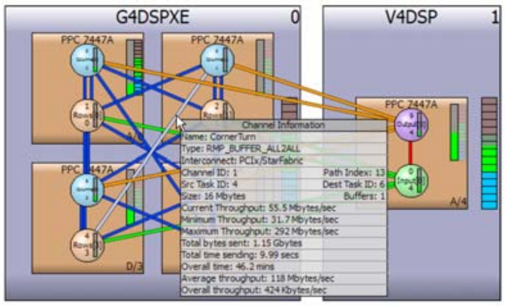


Figure 5– ApplicationView sets up tasks and channels

The FFT's needed in this radar application are highly optimized and are part of AXISLib.

RuntimeView provided corner turn real time performance analysis such as CPU usage and I/O performance. This information is available on a task by task basis.



Channel Information
Name: CornerTurn
Type: RMP_BUFFER_ALL2ALL
Interconnect: PCIx/StarFabric
Channel ID: 1 Path Index: 13
Src Task ID: 4 Dest Task ID: 6
Size: 16 Mbytes Buffers: 1
Current Throughput: 55.5 Mbytes/sec
Minimum Throughput: 31.7 Mbytes/sec
Maximum Throughput: 292 Mbytes/sec
Total bytes sent: 1.15 Gbytes
Total time sending: 9.99 secs
Overall time: 46.2 mins
Average throughput: 118 Mbytes/sec
Overall throughput: 424 Kbytes/sec

Figure 6 – RuntimeView shows individual tasks and channel data performance

EventView allows instrumentation of code for performance analysis of the corner turn Relative timing information among all tasks is available to the user for further analysis.

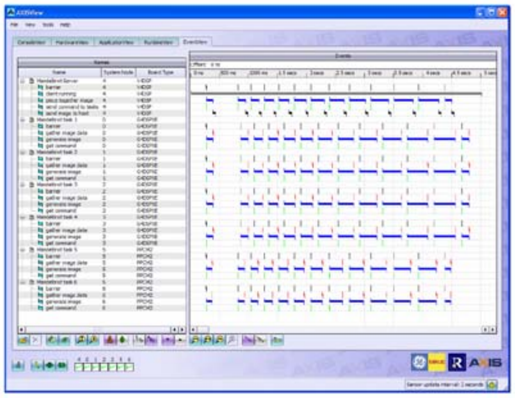


Figure 7 – EventView illustrates timing of events among tasks and channels

References

[1] J. Meyer, Multi-function radar for the deployed warrior using VPX-REDI and RapidIO , Military Embedded Systems, Fall, 2006
[2] D. Tetley, The role of software tools in developing and deploying multiprocessor systems, VMEbus Systems, Volume 24, Number 2, April 2006