

# High Performance Parallel Processing Algorithms for Mobile Communication

Ravi Palaniappan, Thomas Clarke, Mohammad Z. Ahmad

*Institute for Simulation & Training, University of Central Florida, Orlando, FL 32826, USA*

rpalania@ist.ucf.edu

**Abstract-** The goal of this paper is to demonstrate the development of a dynamically reconfigurable scalable High Performance Signal Processor architecture for hand held radio applications that significantly reduce size, weight, and power over conventional technologies. The proposed software algorithms will be very useful for next generation radio and communication platforms such as Joint Tactical Radio Systems (JTRS) and other commercial software defined radios.

## I. INTRODUCTION

The software development kit for this project involved the use of multiple parallel processors connected by high-speed Infiniband connection to process waveforms in parallel and demonstrate the capability for speedups of approximately 10 times compared to conventional single processors. The Institute for Simulation and Training, University of Central Florida has acquired a High Performance computing cluster called STOKES that consists of 600 core processors with over 40 TB of storage, high speed inter-processor interconnects using Infiniband and the capability to perform 6.6 Trillion Floating Point Operations per Second (TFLOPS). The Infiniband interconnects are capable of data transfers at a speed of 40 Giga bits/sec. Using this high performance system the researchers at the Institute developed parallel algorithms to demonstrate advanced communication algorithm processing in parallel mode. As a proof-of-concept demonstration the researchers used wireless LAN 802.11a waveforms for benchmarking the parallel optimization algorithm in terms of Millions of Operations per second, processing latency, communication overhead and processing time compared to single processor processing of the waveform.

## II. PARALLEL ALGORITHM

In general 802.11a has been designed to run sequentially, hence there are sections which can and cannot be parallelized [1]. Trying to parallelize the entire algorithm will not result in an efficient parallel program. The main goal is to have a parallel implementation of the 802.11a mechanism in sections that are amenable to be parallelized.

The algorithmic approach towards writing and optimizing parallel code as applied to the problem were:

- A. Identify computational hotspots.
- B. Partition the problem into smaller independent tasks.
- C. Identify communication requirements between such tasks.
- D. Agglomerate smaller tasks into larger tasks.
- E. Map tasks to processors.

### A. Computational Hotspots

Computational hotspots are those parts of the program which consume the most run time. The main idea behind this is to identify these hotspots and partition them to be executed on different processing cores. We use MATLAB to program and identify the execution times for the different processes in the 802.11a protocol. MATLAB uses a built-in function called "Profiler" which profiles execution times for functions. It helps to debug and optimize M-files by tracking their execution times. For each function in the M-file, "Profile" records information about execution time, number of calls, parent functions, child functions, code line hit count, and code line execution time [2].

The MATLAB profiling tool identified that there are three hotspots in the 802.11a that can be parallelized. They were the OFDM modulation, IFFT and Multiplexing OFDM frames.

From the below MATLAB profiling results in Table 1 we observe that the modulator takes maximum execution times as shown in the following figures. We see that the modulator comprises almost 51% of net execution time followed by the IFFT.

### B. Partition the Problem into Smaller Independent Tasks

The identified hotspot code can now be partitioned into smaller tasks which can execute in parallel with each other. Partitioning is carried out in the following ways: **Data decomposition:** First partition the data and then partition the computation based on the data. **Functional decomposition:** Partition computation into smaller tasks.

The OFDM modulation involves quadrature phase-shift keying (QPSK) where the carrier undergoes four changes in the phase (four symbols) and thus represents 2 binary bits of data per symbol. It involves changing the phase of the transmitted waveform instead of the frequency, these finite phase changes representing digital data. A phase-modulated waveform can be generated by using the digital data to switch between two signals of equal frequency but opposing phase. If the resultant waveform is multiplied by a sine wave of equal frequency, two components are generated: one cosine waveform of double the received frequency and one frequency-independent term whose amplitude is proportional to the cosine of the phase shift. Thus phase shifts involve multiplying the input by sin and cosine values of specific ranges of parameters [1], [3].

TABLE 1  
MATLAB PROFILE SUMMARY

Function Name	Calls	Self Time (in sec)	Total Time (in sec)
Modulator	3252	31.814	31.814
IFFT	17	5.492	16.371
Multiplexer	1005	3.579	7.285
Other functions	3661	4.355	6.289

**Calls** represent the total number of function calls.

**Self time** is described as the time spent in the top-level function than in child-function calls

**Total time** is the net time spent in executing the function.

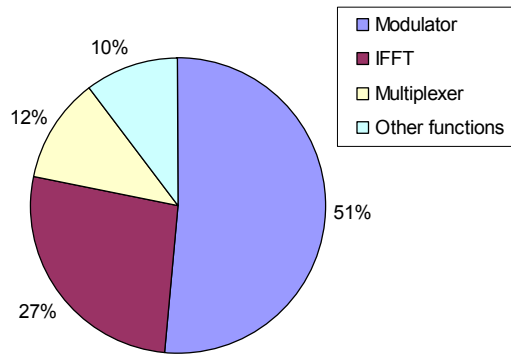


Fig. 1 Percentage of total time by each task as categorized by the profiling tool in MATLAB

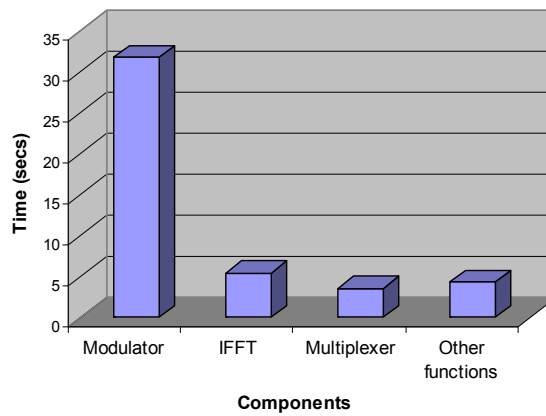


Fig. 2 Self-time taken by each component based on MATLAB profiling tool

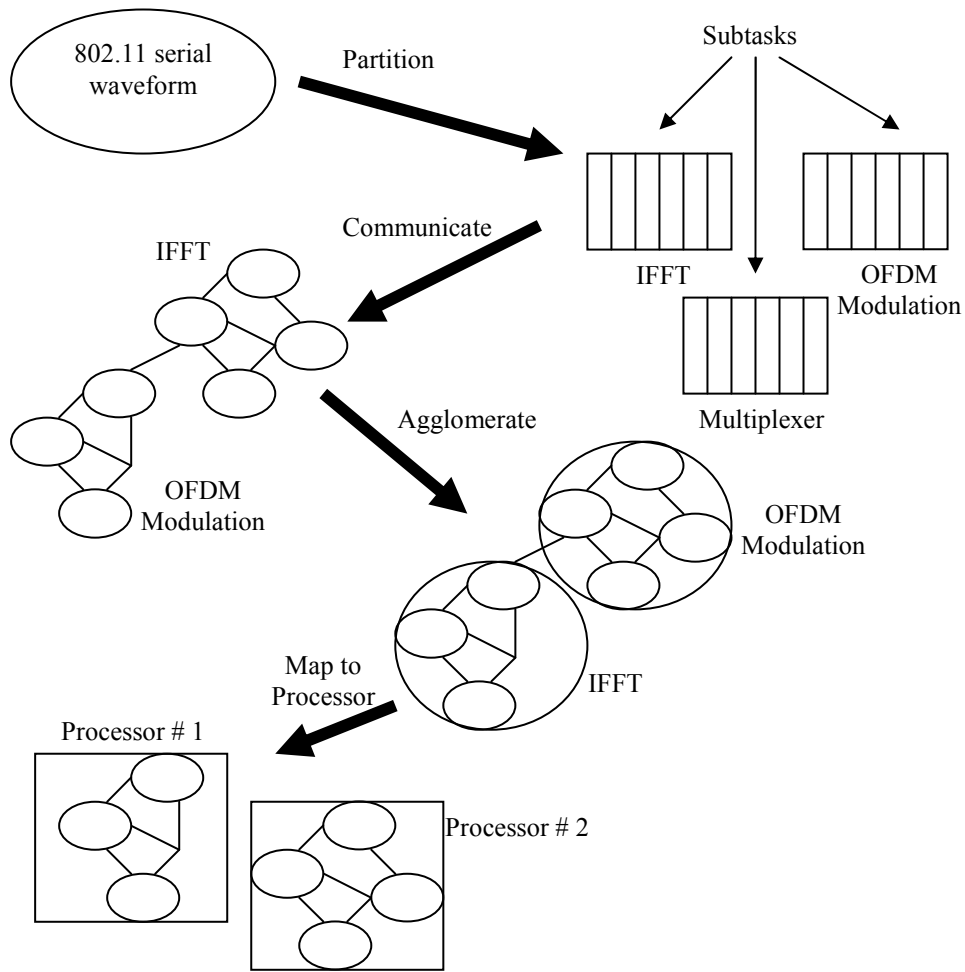


Fig. 3 Steps involved in the parallel algorithm development of the 802.11a waveform

TABLE 2  
SPEEDUPS FROM USING 4 MATLAB WORKERS

Loop Size	Sequential (secs)	Parallel (secs)	Speed-up
100	8.3422	3.8992	2.139464506
200	19.7311	7.2113	2.736136342
400	31.5542	12.8517	2.455254947
600	56.2179	24.0734	2.335270464

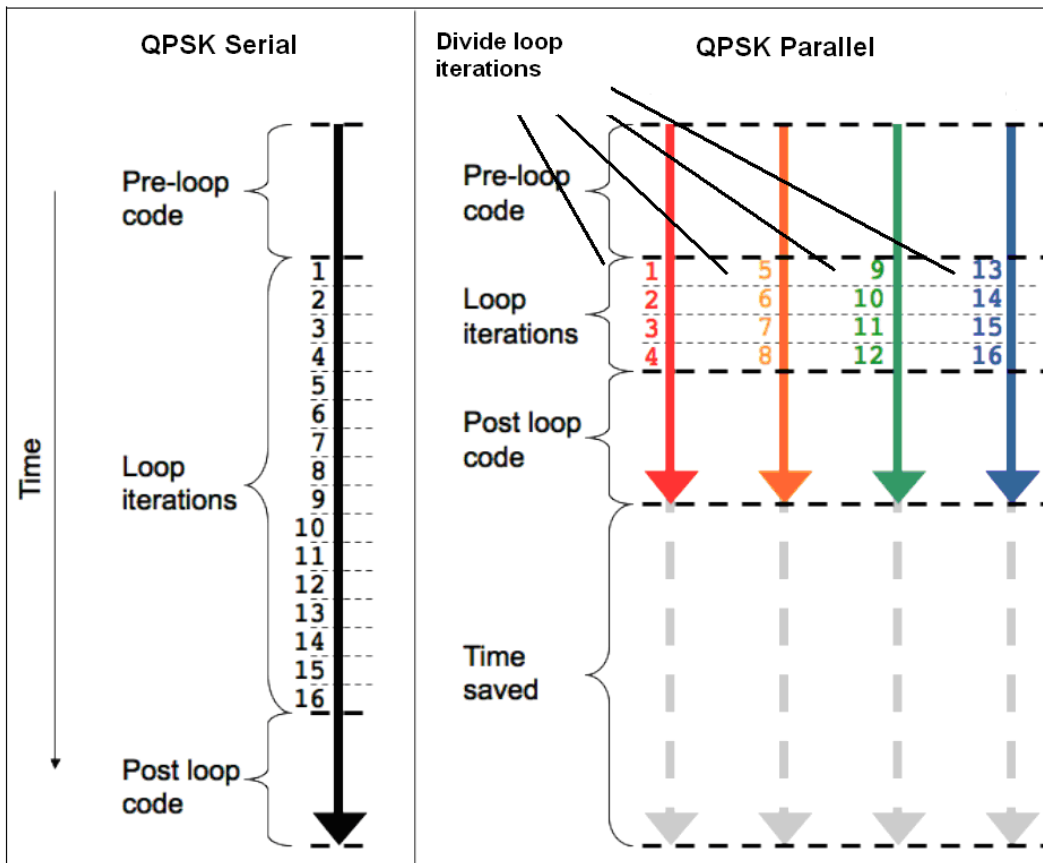


Fig. 4 QPSK Serial to Parallel Conversion

We thus have divided the QPSK serial modulation into a parallelized version that can be executed on multiple processors simultaneously. We similarly partition the IFFT and multiplexing into several different sub tasks which can be executed in parallel.

### C. Communication between Tasks

Once the sub-tasks above have been identified in the above steps communication parameters between tasks need to be set since some tasks would need data from another task [5]. For this communication between tasks is necessary. The intermediate goal is to identify tasks which most frequently communicate with another. In the next step, those tasks will be agglomerated into larger tasks. A small number of communication patterns typically arise:

**Local Communication:** tasks communicate with a small number of local neighbor tasks.

**Global Communication:** tasks communicate with potentially many, non-local tasks.

In the 802.11a model, communication between sub tasks is minimal [6]. With parameter sweeps the results obtained from each operation is independent of each other and actually requires greater agglomeration which we discuss next.

### D. Agglomeration

At this point, data partitions have been created and many tasks which can be executed in parallel have also been identified. Tasks can now be divided randomly and distributed among processors.

Based on communication patterns which would be most applicable in the QPSK modulation (where we use parameter sweeps extensively), agglomeration works best in this component. As a result, a dedicated co-processor could also be kept aside for only this purpose. Being a computational hotspot by far, having a dedicated processor core for modulation results in greater processing efficiency and faster overall speedup [7]. Communication is only required to combine the final results.

### E. Mapping Tasks to Processors

With individual tasks being identified standard task-scheduling algorithms are now used to dynamically map tasks to processors as computation proceeds. These work best when each task has rather small communication requirements. A dedicated processor for the QPSK modulation would entail a greater speedup because the subtasks performed in it are similar and have a degree of communication requirements which can be best optimized if executed on a single processor.

### III. EXPERIMENTAL RESULTS

We used MATLAB parallel workers to demonstrate the speedups obtained from implementing the 802.11a algorithm using 2, 3 and 4 MATLAB workers in parallel.

### IV. CONCLUSIONS

Through this work we have demonstrated the capability to utilize multi-core processors to increase the speed of wireless communication systems. These proof-of-concept experiments can be expanded to large scale systems such as Joint Tactical Radio Systems.

### ACKNOWLEDGEMENT

The authors would like to acknowledge Dr. Brian Goldiez, Deputy Director, Institute for Simulation and Training and Program Manager Troy Dere, RDECOM, Orlando for their support to conduct this research.

### REFERENCES

- [1] Bernard Sklar, *Digital Communications: Fundamentals and Applications*, 2<sup>nd</sup> Ed. Prentice Hall Communications Engineering and Emerging Technologies Series
- [2] MATLAB Profiling tool, MathWorks
- [3] Bondhugula, U.; Devulapalli, A.; Fernando, J.; Wyckoff, P.; Sadayappan, P., "Parallel FPGA-based all-pairs shortest-paths in a directed graph," *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006*.
- [4] Schiavone, G.A.; Tracy, J.; Palaniappan, R., "Preliminary investigations into distributed computing applications on a Beowulf cluster" *Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, pp: 13 – 17, 24-26 Aug. 2000
- [5] Bernd Freisleben, Ralph Jansen. "Analysis of routing protocols for ad hoc networks of mobile computer", *Proceedings of the 15th IASTED International Conference on Applied Informatics, Innsbruck, Austria, February 1997*
- [6] Boukerche, A., & Fabbri, A. (2000) "Partitioning parallel simulation of wireless networks" *In Proceedings of the 2000 Winter Simulation Conference (pp. 1449–1457)*
- [7] Boukerche, A., & Tropper, C. (1994). "A static partitioning and mapping algorithm for conservative parallel simulations" *ACM SIGSIM Simulation Digest 24(1)*, 164–172