

# Power - Time Efficient Algorithm for Computing FFT in Sensor Networks

## (Extended Abstract)

Turkmen Canli, Mark Terwilliger<sup>S</sup>, Ajay Gupta<sup>S</sup>, and Ashfaq Khokhar

University of Illinois at Chicago and <sup>S</sup>Western Michigan University

Deployment of a vast array of tiny smart sensors (sensor devices with processors) interconnected over wireless channels are enabling their pervasive use in a variety of defense and commercial applications, such as environmental monitoring (e.g. traffic, habitat, security), industrial sensing and diagnostics (e.g. factory, appliances), infrastructures (e.g. power grid, water distributions, waste disposal), and battlefield awareness (e.g. multi-target tracking). While the task of developing and implementing pervasive applications for such a scenario is exciting, it poses tremendous challenges. Sensor networks are different from existing wireless communication networks in the sense that sensor nodes are battery powered and recharging is usually unavailable, so energy is an extremely expensive resource. In this context, design of algorithms for processing information in sensor networks is an emerging research area. The objective is to design algorithms that are energy efficient but also exploit the computing resources available in sensor nodes. In this paper, we present a power-time efficient algorithm for computing Fast Fourier Transform over data distributed across smart sensors. The Fast Fourier Transform (FFT) [5] has been studied extensively as a frequency analysis tool in diverse application areas such as audio, signal, and image processing [9], and several other real time data applications [6,8].

### I. BACKGROUND

A generic distributed algorithm for 1-d FFT on an input of  $N$  data points requires  $\Theta(N \log N)$  complex multiplication operations [7] which is the most time consuming computation for large data sets. This is particularly true in the case of sensor nodes having very simple processors. Thus, in order to develop power efficient distributed algorithms for FFT in a sensor network, these complex multiplications must be load balanced so that the energy drain at every sensor is optimized. For the time being, we assume a simplified model of communication where all the sensors are within communication range of each other.

Assuming one data sample per sensor, a straightforward approach to realizing the  $N$ -point 1-d FFT collaboratively on an  $N$ -sensor network would be as follows. The Fast Fourier transform (FFT) computation has  $\log N$  stages. A butterfly communication structure is followed among the  $\log N$  stages as the computation proceeds from one stage to next. (See Figure 1 which depicts the communication pattern for a 8-point FFT

over a 8-sensor network in 4 stages.) Between any two consecutive stages, the two sensors forming the butterfly communication pattern compute the partial FFT. The computation and communication at any stage between butterfly partners  $p_1$  and  $p_2$  is: (a) sensor  $p_1$  sends a copy of  $x_1$  to  $p_2$ , (b)  $p_2$  sends a copy of  $x_2$  to  $p_1$ , (c)  $p_1$  computes  $x_1 = x_1 + w * x_2$ , and (d)  $p_2$  computes  $x_2 = x_1 - w * x_2$ . Therefore, at every stage each sensor is involved in transmitting and receiving one packet over the radio channel, one complex multiplication and one addition/subtraction. Note that this results in a load balanced FFT computation further implying that both the sensors utilize their batteries equally and evenly. We refer to this algorithm as the Conventional Load Balanced Distributed FFT algorithm.

In particular, as observed in [5] and [10], both the butterfly partner sensors at every stage compute the term  $w*x_2$  which is redundant and unnecessarily consumes sensor's available power. The authors in [5] further suggest an approach that avoids this redundancy at the expense of load balancing.

The approach proposed in [5] is shown in Figure 2 and the procedure at the butterfly partners,  $p_1$  and  $p_2$ , at any stage is as follows: (a)  $p_2$  computes  $w*x_2$ , (b)  $p_1$  sends a copy of  $x_1$  to  $p_2$ , (c)  $p_2$  sends a copy of  $w*x_2$  to  $p_1$ , (d)  $p_1$  computes  $x_1 = x_1 + w * x_2$ , and (e)  $p_2$  computes  $x_2 = x_1 - w * x_2$ . We again have transmission of two packets and both the sensors are involved in addition/subtraction but only one sensor ("lower" partner) is involved in complex multiplication.

It can be easily shown that the above algorithm, which we call Unbalanced Power-Aware Distributed FFT algorithm, uses  $N \log N$  transmissions,  $N \log N$  additions/ subtractions and  $N/2 \log N$  complex multiplications. This algorithm removes the redundancy of multiplications but at the expense of load balancing with respect to energy usage at sensors. For example, sensor  $p_0$  is not involved at all in complex multiplications whereas sensor  $p_7$  performs complex multiplications in all the  $\log N$  stages of Figure 2. This would imply that  $p_7$  would drain its battery quickly and would go out of service much sooner.

We next present a balanced power and time efficient algorithm. The proposed algorithm is completely different from our earlier work [10] which was only efficient in terms of power but was a slower algorithm. In this paper we introduce the notion of power-time efficiency similar to area-time efficiency in VLSI algorithms proposed in the literature.

## II. PROPOSED POWER-TIME EFFICIENT FFT ALGORITHM

In this section we present an algorithm for distributed FFT which is load balanced not only in its computations and communications but also energy usage in the sensor network. Furthermore, our proposed solution removes the redundant multiplication computation from the Conventional Load Balanced Distributed FFT and load balances the  $N/2 \log N$  multiplications of the Unbalanced Power-Aware Distributed FFT algorithm. We know that there are  $N/2$  complex multiplications at every one of the  $\log N$  stages in the Unbalanced Power-Aware FFT algorithm. The basic idea behind balancing the multiplication computation among the sensors is to perform the Unbalanced Power-Aware algorithm for the first  $(\log N)/2$  stages and then permute the data among sensors such that the sensors that have performed more multiplication in the previous iteration exchange their data with those that have performed fewer multiplications. We have identified this permutation as the shuffle-complement permutation explained in the following.

- Given the binary representation of the sensor id  $\langle b_{n-1} b_{n-2} \dots b_1 b_0 \rangle$ , the complex multiplication execution pattern of any sensor in the Unbalanced Power-Aware algorithm is  $b_0 b_1 \dots b_{n-2} b_{n-1}$ , where a '1' in bit position  $i$  indicates that the sensor is performing the multiplication during the  $i^{\text{th}}$  stage of the FFT, for  $0 \leq i \leq n-1$ . **For example in the figure 2, sensor 4,  $\langle 100 \rangle$ , has multiplication pattern of  $\langle 001 \rangle$ , it performs multiplication at stage 2, does not perform multiplication at stages 0 and 1.**
- Sensor  $\langle b_{n-1} b_{n-2} \dots b_1 b_0 \rangle$  follows the multiplication pattern  $b_0 b_1 \dots b_{n/2-1}$  for the first  $(\log N)/2$  stages.
- Shuffle-complement Permutation: After  $\log N/2$  stages, sensor  $\langle b_{n-1} b_{n-2} \dots b_1 b_0 \rangle$  exchanges its data with the sensor  $\langle b_0' b_1' \dots b_{n-2}' b_{n-1}' \rangle$ .
- FFT computation proceeds as if the butterfly structure of the first  $\log N/2$  stages has been flipped.

Due to the space limitations we omit the correctness proof but intuitive idea is the following: If a sensor has a multiplication pattern of say 1111, it should be changed to 1100 (so that the sensor performs complex multiplications only for  $\log N/2$  stages). On the hand if a sensor has the multiplication pattern of 0000, it should be changed to 0011. For this example case, sensors  $\langle 0000 \rangle$  and  $\langle 1111 \rangle$  exchange their data after  $(\log N)/2$  stages and the distributed FFT computation proceeds as if the butterfly in the first  $\log N/2$  stage has been flipped. The pictorial representation of the overall algorithm is given in Figure 3 for a 16-point FFT on a 16-node sensor network.

For our experiments, sensor networks were considered spread over a 10-meter x 10-meter region with the number of nodes covering that region varying between 8 and 4096. It is assumed that each sensor is equipped with a StrongARM SA-1100 microprocessor. This low-power embedded processor can operate at many clock frequencies, but our results are based on a frequency of 59 MHz. When calculating the communication time and energy cost for the three algorithms, the JouleTrack [6] software was used. This web-based tool provides both energy consumption as well as execution time data based on the individual FFT

sensor programs. Our simulation results show that the proposed algorithm eliminates typical redundant computations in a distributed FFT algorithm and uniformly maps complex multiplications over all the sensors nodes using a shuffle-complement permutation in between the application of an unbalanced algorithm. We show that the proposed algorithm improves energy consumption by **36 to 40%** compared to the conventional FFT, and compared to the power efficient unbalanced algorithms of [5], it significantly improves the network life in term of number of  $N$ -point FFTs that can be computed. This all is achieved in the presence of execution time which is equivalent to the standard parallel algorithm

## REFERENCES

- [1] Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks", IEEE Communication Magazine, Aug. 2002 p.p. 102-114
- [2] J.W. Cooley, P.A. Lewis, and P.D. Welch, "The Fast Fourier Transform and its Application to Time Series Analysis," Wiley, New York, 1977. In statistical Methods for Digital Computers.
- [3] W.M. Gentleman, G. Sande, "Fast Fourier Transforms for Fun and Profit," In Proc. 1966 Fall Joint Computer Conference AFIPS 29, pages 563-578, 1966.
- [4] A.V. Oppenheim and A.S. Willsky. "Signals and Systems," Prentice Hall, Englewood Cliffs, New Jersey, 1983.
- [5] C. Chiasserini and R. Rao, "On the Concept of Distributed Digital Signal Processing in Wireless Sensor Networks," IEEE MILCOM, 2002.
- [6] A. Sinha and A.P. Chandrakasan, "JouleTrack – A Web based Tool for Software Energy Profiling," Proceedings of the 38<sup>th</sup> Design Automation Conference, June 2001, <http://www-mtl.mit.edu/research/icsystem/uamps/>.
- [7] V. Kumar, A. Grama, A. Gupta, G. Karypis. Parallel Computing: Design and Analysis of Algorithms. Benjamin-Cummings Publishing Company, 2<sup>nd</sup> edition. 2003.
- [8] S.-H. Cho and A. P. Chandrakasan, "Energy Efficient Protocols for Low Duty Cycle Wireless Microsensor Networks", ICASSP 2001, May 2001.
- [9] W. Heinzelman, A. Sinha, A. Wang, and A. P. Chandrakasan, "Energy-Scalable Algorithms and Protocols for Wireless Microsensor Networks," *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP '00)*, June 2000, <http://www-mtl.mit.edu/research/icsystems/uamps/>.
- [10] T. Canli, M. Terwilliger, A. Gupta, and A. Khokhar, Power Efficient Algorithms for Computing Fast Fourier Transform Over Sensor Networks, First IEEE International Conference on Sensor and Ad Hoc Communications and Networks, 2004, under submission.

# Power - Time Efficient Algorithm for Computing FFT over Wireless Sensor Networks

Turkmen Canli, Student Member, Mark Terwilliger, Student Member, Ajay Gupta, Member, and Ashfaq Khokhar, Senior Member

## Problem Statement

- Technology advances enable pervasive use of sensor networks in a variety of defense and commercial applications
- Sensor nodes are battery powered and recharging is usually unavailable, so energy is an extremely expensive resource
- Design of algorithms for processing information in sensor networks is an emerging research area
- 1-d distributed FFT can be realized with sensor networks through butterfly communication structure
- Redundant calculations in 1-d distributed FFT can be eliminated via data shuffling in a balanced way

## Unbalanced Redundancy-Free Power-Aware FFT [5]

- $(N \log_2 N)/2$  complex multiplications
- $N \log_2 N$  additions / subtractions
- $N \log_2 N$  transmissions
- Given the binary representation of the sensor id  $\langle b_{n-1} b_{n-2} \dots b_1 b_0 \rangle$ , the complex multiplication pattern of any sensor is  $\langle b_0 b_1 \dots b_{n-2} b_{n-1} \rangle$ , where a '1' at bit position  $i$  indicates that the sensor is performing the multiplication during the  $i^{\text{th}}$  stage of the FFT, for  $0 \leq i \leq n-1$ .
  - For example in Figure 2, Sensor 4  $\langle 100 \rangle$  has a complex multiplication pattern  $\langle 001 \rangle$ . It does not perform multiplication at stage 0 and 1 but it does at stage 2.
- Some sensors will drain their energy much faster than others => decreased sensor network lifetime

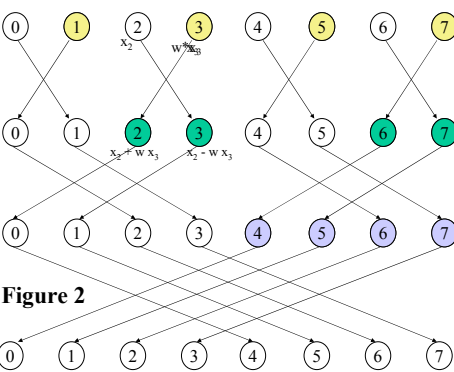


Figure 2

## Conventional Distributed FFT [7]

- $N$  point-FFT
  - 1 input point per sensor
  - 1 output point per sensor
- $N \log_2 N$  complex multiplications (twice the number of multiplications in a serial algorithm, example shown for an 8-point FFT in Figure 1.
- Colored nodes represent the nodes computing complex multiplications
- $N \log_2 N$  additions/subtractions
- $N \log_2 N$  transmissions

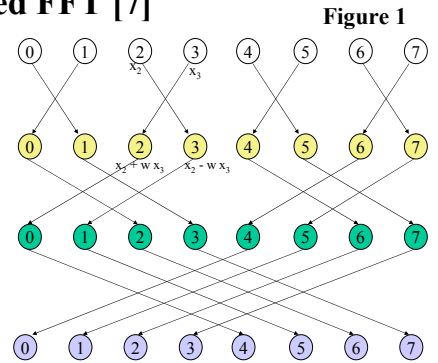


Figure 1

## Proposed Power-Time Efficient FFT

- Two concatenated stages of the unbalanced algorithm in Figure 2, each stage is inverse of the other in terms of multiplication pattern. Example is shown in Figure 3 for a 16 node network.
- Sensor data values are permuted at the end of the first stage
- $(N \log_2 N)/2$  complex multiplications
- $N+N \log_2 N$  transmissions
- $N \log_2 N$  additions / subtractions
- Permutation formula:
  - Inverse-shuffle  $n-1$  times and complement
  - Every Sensor  $\langle b_2 b_1 b_0 \rangle$  sends its data to Sensor  $\langle b'_0 b'_1 b'_2 b'_3 \rangle$
- Correctness:
  - $\langle b'_0 b'_1 b'_2 b'_3 \rangle$  sends its data to  $\langle (b_3)' (b_2)' (b_1)' (b_0)' \rangle = \langle b_3 b_2 b_1 b_0 \rangle$ .
  - If they are not same,  $\langle b_3 b_2 b_1 b_0 \rangle$  and  $\langle b'_0 b'_1 b'_2 b'_3 \rangle$  exchange their data.

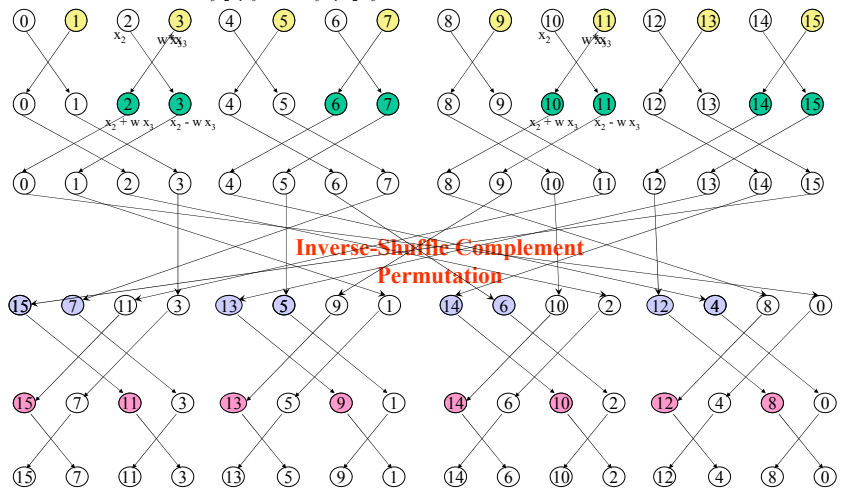
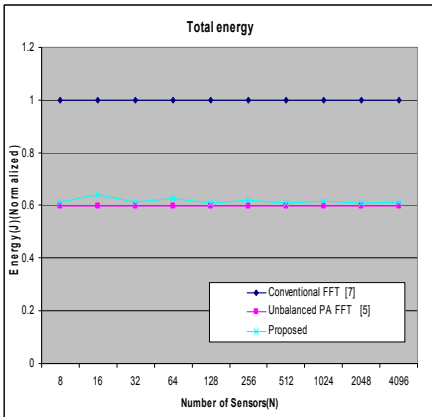
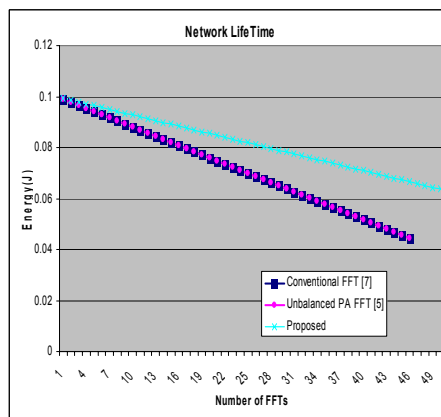


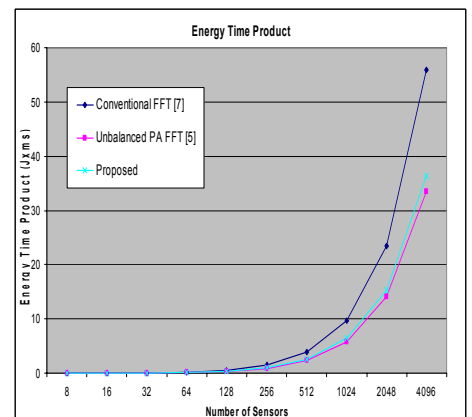
Figure 3



36-40% improvement in power utilization



Enhanced Network Life



Power-Time Efficiency