

Resource-aware Distributed Split Radix FFT on Wireless Sensor Networks

Sherine Abdelhak, Jared Tessier, Soumik Ghosh, Magdy Bayoumi, and Nian-Feng Tzeng
 The Center for Advanced Computer Studies, University of Louisiana at Lafayette
 {spa9242, jst2777, sxg5317, mab} @ cacs.louisiana.edu

Introduction

In this work we are proposing a light-weight resource-aware distributed Fast Fourier Transform on energy-constrained wireless sensor networks (WSN). The distribution of the FFT, not only increases computational capabilities of WSN, introduces flexibility and reliability, but also increases the network lifetime. This work has two folds: (a) an efficient implementation of the real split radix FFT based on [7], with improved bit reversal and a lookup table (LUT) for the twiddle factors, (b) energy and channel aware scheduling and task allocation. The proposed scheme achieves $4\times$ speedup in computation and increases the network lifetime by a factor of 3.8. The experiments were done on a test bed of Telosb [1] sensor nodes featuring TI's 16-bit MSP430 microcontroller, and Chipcon's CC2420 radio module. The power consumption of the nodes was determined using the shunt resistor method [3]. The code was written on TinyOS [2], and a java GUI was developed to provide a user friendly interface.

The fast Fourier transform has a diverse set of applications such as spectrum analysis, digital filtering, signal and image processing. The most widely used FFTs are those whose lengths are power-of-two, in particular, radix-2 and radix-4. On the other hand split radix algorithm, derived by Duhamel and Hollmann [5] is a combination of radix-2 and radix-4 algorithms, and achieves fewer arithmetic operations at the expense of a less regular structure. Since the main concern in battery-operated WSNs is the energy efficiency, the Split-radix was chosen.

Split-radix FFT partitions a N -point discrete Fourier transform (DFT) into 3 parts: a DFT of length $N/2$ over the even-numbered indices, a DFT of length $N/4$ over the odd indices which satisfy Eq.1, and a DFT of length $N/4$ over the odd indices which satisfy Eq.2.

$$(index \text{ Mod } 4) = 1 \quad (1)$$

$$(index \text{ Mod } 4) = 3 \quad (2)$$

A split radix can be broken down into 3 partitions as follows: let Y_r , Z_r , and H_r be, respectively, the split radix FFT of the sequences x_{2k} , $0 \leq k \leq N/2-1$, x_{4k+1} , $0 \leq k \leq N/4-1$, and x_{4k+3} , $0 \leq k \leq N/4-1$, then the original FFT has the following solution [8]:

$$X_r = Y_r + (w_N^r Z_r + w_N^{3r} H_r), X_{r+N/4} = Y_{r+N/4} - j(w_N^r Z_r - w_N^{3r} H_r) \quad (3)$$

$$X_{r+N/2} = Y_r - (w_N^r Z_r + w_N^{3r} H_r), X_{r+3N/4} = Y_{r+N/4} + j(w_N^r Z_r - w_N^{3r} H_r)$$

where $0 \leq r \leq N/4-1$

This partitioning can be done recursively, and this is the basis of our distributing algorithm.

Several optimizations were implemented in order to make the FFT implementation viable on the sensor nodes. The main optimization, however, is implementing an efficient bit reversal which computes only $N/8$ reverse operations

based on [4]. Figure 1a shows the energy consumption of split radix FFT versus the radix 2, where the savings of the former are up to $2x$. Figure 1b reports the energy savings in of the improved bit reversal versus the conventional where the savings are up to $2.7x$.

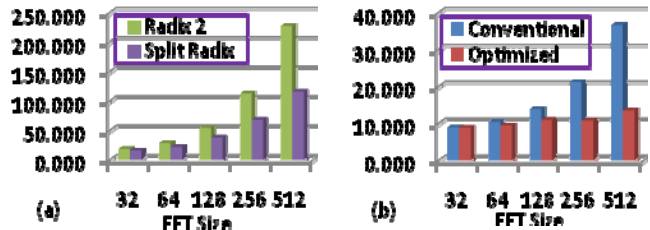


Figure 1: Energy consumption in μJ for (a) Split radix vs. Radix 2 FFT, (b) Improved vs. Conventional bit reversal

Resource-aware Task Allocation

To achieve a *resource-aware* task allocation, the nodes are required to gather some status information: (a) **Available storage (S)** at the nodes: indicates the maximum FFT size that can be assigned to this node, (b) **Energy status** of the node: also gives an upper bound on the FFT size that can be processed, based on predetermined energy consumption per FFT size (given in Figure 1a). Telosb nodes provide the **residual voltage (RV)** as the only physical layer parameter that can reflect the energy status of the node. (c) **Link Quality** to reach the node: is required to estimate the cost associated with assigning tasks to this node. A bad-quality link incurs the overhead of retransmission, and lost packets. Abiding by the IEEE 802.15.4 [6] specifications, the CC2420 radio chip provides the Link Quality Indicator (LQI) value. LQI can accurately reflect the Packet Error Rate (PER) of a channel, if measurements are taken over hundreds of packets which is expensive in WSN. Therefore, each node saves LQ and **Confidence** for each neighbor where LQ is the sum of the LQI values of the packets received from a particular neighbor, and **Confidence** is the number of packets (normalized by 100) that were considered in the LQI calculation. This parameter reflects the confidence level of the estimate. To the best of our knowledge, this is the first work in which LQI is used for scheduling purposes.

Based on S and RV , the maximum FFT size, a node can handle, is determined. Based on $LQ \times Confidence$, the node can determine the PER and estimate the number of retransmissions incurred. As a result, the sensor nodes are modeled as a set $U = \{u_1, u_2, \dots, u_y\}$. Each node u_i has the following attributes: (1) Maximum FFT size (F_i), (2) packet error rate (P_i).

Proposed Distribution

The proposed resource-aware distribution depends on three parameters imposed by the underlying platform: (a)

LMem (Memory Limit): Indicates the maximum FFT size that a node can store. **(b) LComp (Computation Limit):** Determines the maximum FFT points that can process by the node at a certain period of time. **(c)LComm (Communication Limit):** Indicates the payload size in bytes, per message. It is imposed by the Zigbee alliance to be at most 127B. *LComm* is essential for the distribution algorithm such that the biggest partition size should not exceed the size of one message. The computation can, then, start upon the reception of the *first* data message. For Telosb, *LMem* is 4096 1-Byte-points, *LComp* is 512, and effective *LComm* is 115B only

Consider the computation of one large N -point FFT whose inputs are aggregated from the samples of several nodes. Let node C be the distributing node which holds the N samples where $N > LComp_C$. The proposed distribution algorithm has the following steps: **Step1 'DFT partitioning'**: The original DFT is, recursively, partitioned into n smaller p DFTs (partitioned DFT) whose size $N_p \leq LMem$. The

partition is as follows: (a) $N/2$ -point DFT formed of the even-indexed inputs. This partition is identified by the suffix 'e' for even, (b) $N/4$ -point DFT formed of the odd-indexed inputs whose indices satisfy Eq.1. It is identified by the suffix 'o1', (c) $N/4$ -point DFT formed of the odd-indexed inputs whose indices satisfy Eq.2 and is identified by the suffix 'o3'. Note that, based on the Available Storage of a node, the last two partitions can be grouped and sent to one node; in this case the partition has the suffix 'o'. Yet, their computation is handled each at a time. Figure 2a shows the DFT partitioning until $N_p \leq LMem$ for $N=16384$,

LMem=4096. Figure 2b shows the final lists ready to be distributed. **Step2 'Vice Nodes Selection'**: Node C selects n Vice nodes, of its neighbors, which will handle the distribution of a p DFT. In this manner, Node C can quickly resume other functionalities while the Vice nodes carry on the distribution. Following the previous example, the p DFT size is 4096-point, and four Vice nodes need to be selected. Node C specifies the p DFT length for each Vice node, the p DFT type ('e','o','o1','o3') and the Vice node's transmission time slot. **Step3 'Computing-cluster Formation'**: During its assigned time slot, each Vice node broadcasts a *REQUEST* message. Based on the RSSI of the *REQUEST* messages, a neighboring node i selects *one* Vice node, sends its *status* information (F_i, P_i), and ignores any messages received from other Vice nodes. **Step4 'Distributing / computing the pDFT'**: handled by the Vice node and its cluster. It involves DFT data distribution, task allocation, and scheduling. Each Vice node, further partitions the p DFT into m N_q -DFTs where $N_q = LComm$.

These DFTs are the tasks which can be modeled as a set $T = \{t_1, t_2, \dots, t_x\}$, where each task has a 'History' attribute. *History* is the combination of the task's type and the types of its parents. A *History* $h_i = \{e, e, o1\}$ indicates that the task is *o1* and has 2 parents of type *e*. Each type is assigned a different bit sequence: 'e' = 0, 'o' = 1, 'o1' = 10, and 'o3' = 11. As such, the *History* attribute of a task of size $LComm = 2^x$, corresponding to a p DFT of size $LMem = 2^y$ has $2^{(y-x)}$ bits. Each task involves the computation of a DFT of size $LComm$, and a set of twiddle factors of different sizes, depending on the *History* attribute. The size

of twiddle factor W_V^r ($0 \leq r \leq V/4 - 1$) is $2^{(x+index)}$ where *index* is the index of the second one from the right in a set of 2 consecutive ones in the *History* attribute, or the index of the first 1 from the right, if there is no consecutive 1. Note that although the tasks are of different *History*, the energy consumption for the computation and data reception is almost even, ensuring load balancing. The transmission energy, on the other hand is slightly different, since the tasks of type 'e' broadcasts extra messages holding $W_{V/4}^r$ ($0 \leq r \leq V/16 - 1$). This is best illustrated in Figure 3. Mapping the tasks to the nodes is done in a simple, lightweight procedure. The nodes are sorted, primarily, in decreasing order of F , then in increasing order of P . The number of tasks ($nTasks$) assigned to a node i can be 1 or an even number $nTasks < F_i < LComp$. If $nTasks > 1$ then the tasks are merged into 1 FFT computation and the *History* of the first task is adopted by the node. **Step5 'Result Gathering'**: The nodes in each computing cluster send their results and the *History* attribute to the corresponding Vice node. The Vice node simply stacks the values obtained in the order given by the *History* attribute, and a recursive derivation of the formula in Eq. 3 is applied which only involves addition/subtraction operations. The Vice nodes, then, send their results to the distributing node which does the final computation as in Eq. 3.

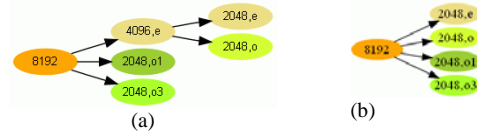
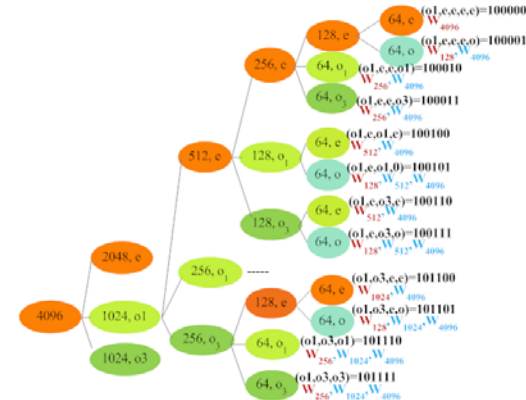


Figure 2: (a) Partitioning a 16384-point split-radix DFT, (b) Final result of the partitioning.



References

- [1] J. Polastre, R. Szewczyk and D. Culler, Telos: "Enabling ultra-low power wireless research," In Proc. 4th Int'l. Conference on Information Processing in Sensor Networks, 2005
- [2] Philip Levis, TinyOS programming, <http://csl.stanford.edu/pal/pubs/tinyos-programming.pdf>, last visited: 02-2009.
- [3] A. Milenkovic, M. Milenkovic, E. Jovanov, D. Hite and D. Raskovic, "An environment for runtime power monitoring of wireless sensor network platforms," 37th Southeastern Symp. on System Theory, 2005.
- [4] D. Sundararajan, M. Omair Ahmad, M.N.S. Swamy, "A Fast FFT bit-reversal algorithm", IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 41, pp. 701-703, 1994
- [5] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," *Electron Lett.*, vol. 20, pp. 14-16, 1984.
- [6] IEEE std. 802.15.4: Wireless Medium Access Control and Physical Layer specific. for Low Rate Wireless Personal Area Networks, 2003
- [7] H. Sorensen, D. Jones, M. Heideman, C. Burrus "Real-valued fast Fourier transform algorithms," IEEE Transactions on Acoustics, Speech and Signal Processing, vol.35, pp.849-863, 1987
- [8] E. Chu and A. George, *Inside the FFT Black Box*, CRC, 2000.