BATTERY – FRIENDLY DESIGN OF SIGNAL PROCESSING ALGORITHMS

Praveen Raghavan & Chaitali Chakrabarti
Department of Electrical Engineering
Arizona State University, Tempe, AZ – 85287
{praveenr, chaitali}@asu.edu

ABSTRACT

Portable systems today are designed with lowering the energy consumption as the primary design metric. This is unfortunate since maximizing battery lifetime is a more appropriate metric, and lowering energy does not necessarily mean improving battery lifetime. In this paper we first show how to design battery-friendly implementations of common signal processing kernels such as FIR filters and FFT. The basic idea is to generate a load profile that results in better battery behavior. Next, we demonstrate how frequency scaling can be used effectively to improve the battery behavior of an application such as MPEG2.

1. INTRODUCTION

Use of portable devices in many signal processing and communication applications is on the rise. These are designed with reduction in energy consumption as the prime design metric. The assumption is that reducing the energy consumption of the system will increase the lifetime of the battery. This is not necessarily true, since two implementations that have comparable energy consumption can have very different battery performance. In this paper we present an approach to designing battery-friendly versions of signal processing algorithms so as to maximize the battery lifetime.

In recent years, there has been some work in analyzing the battery characteristics [1, 2], and using these characteristics to design battery-conscious task scheduling algorithms (in the context of operating system) [3, 4, 5]. There has been no work done in the area of design of battery-efficient signal processing algorithms.

In this paper we first look at how common signal processing kernels like FIR, FFT can be modified so as to make the kernels more battery-friendly. While many of the techniques are borrowed from energy-efficient algorithm design, some of the techniques capitalize on the non-linearities of the battery and are unique to battery-operated systems. For instance, while changing the order of the computation mildly changes the energy consumption, it has a significant effect on the battery performance. Next, we demonstrate how frequency scaling can be used to prolong battery life. We show the effect of frequency scaling on different slack distribution schemes for a complex algorithm such as MPEG2. A slack distribution scheme that generates a load profile with decreasing order of load currents does substantially better than other orderings.

This paper is organized as follows: Section 2 motivates the approach and the setup used for the experiments. Section 3 discusses the battery model and its properties. Battery friendly FIR filters are described in section 4. Section 5 gives details on battery efficient FFT decomposition. Section 6 explains how slack utilization can be done for MPEG2. Finally section 7 concludes the paper.

2. BACKGROUND

2.1 Motivational Example

In the design of portable systems, the assumption is that reducing the energy consumption will increase the lifetime of the portable device. This is not necessarily true, since two implementations that have comparable energy consumption can have very different battery performance. To illustrate this phenomenon consider the example shown in table 1.

Case	Avg load Current (mA)	Avg Exec. Time (min)	Charge in mA-min	DualFoil Voltage in V	% Redu. in Volage
I	400	6	2400	3.97	30.90
II	300	8	2400	4.01	26.90
III	200	12	2400	4.05	22.70
IV	100	24	2400	4.10	18.70
V	250	9	2250	4.04	24.00

Table 1: Same energy consumption is not equivalent to same battery behavior

0-7803-7795-8/03/ $17.00 © 2003 IEEE

The first four cases (I to IV) have different load profiles (current vs. time) but have the same energy consumption. Simulations on DualFoil, a low-end battery simulator, show that case IV that has the lowest average load current has the lowest reduction in battery voltage, and thus the best battery performance.

Now consider cases IV and V, which have different energy consumption. Note that case V has a lower energy consumption as compared to case IV. However, the drop in battery voltage in case V is 24.00% as compared to 18.70% for case IV. The lower drop in battery voltage is because case IV has a lower load current than case V. This demonstrates that lower energy consumption does not necessarily mean that the implementation is battery efficient.

2.2 Simulation Environment

The battery simulator used in this paper to estimate the reduction in battery voltage is DUALFOIL a low-level Li-ion simulator [6]. The estimation error is within 5%. The energy and current estimation for our programs was done using Joule Track [7], a web-based tool for software energy profiling of a StrongARM based processor. Energy prediction by JouleTrack is within 3% of error. The frequency of the StrongARM processor can be varied in 11 discrete steps from 206MHz to 59MHz.

3. BATTERY PROPERTIES

The two most important properties of a battery are the rate capacity effect and recovery. The rate capacity states that the charge capacity of battery is dependent on the rate of discharge; the capacity increases as the load current decreases. The recovery effect is due to self-recharging which happens when the battery is allowed to rest for sometime. Fig. 1 illustrates the rate capacity effect by plotting the battery voltage as a function of discharge time for two different loads. Note that decreasing the load by a factor of 2 results in the battery lifetime increasing by more than a factor of 2. An effective way of decreasing the load current is by lowering the frequency of operation. Lowering the frequency by a factor of s results in lowering the voltage by a factor of $s_1 = f(s) = 3.33s^3 - 0.24s^2 + 0.49s + 0.42$, (obtained empirically) and lowering the current by a factor of $s \times s_1$. The significant drop in the load current results in enhanced battery performance – a feature that should be exploited in systems that have slack.

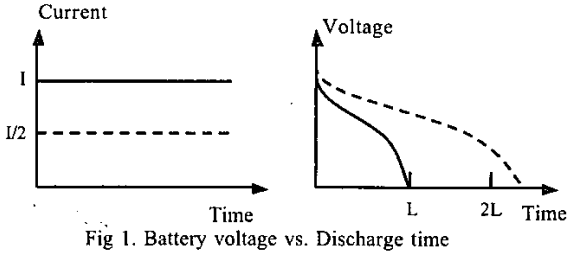


Fig 1. Battery voltage vs. Discharge time

In the design of battery-friendly algorithms, it is important to consider the load profile (current vs. time) generated and choose the one that results in better battery performance. In our earlier work in [3], we have demonstrated that the performance of a battery is related to the order of the loads that it is subjected to. We illustrate this with the help of the example in Fig. 2 where three tasks have been ordered in three different ways. Case I that corresponds to increasing order of loads has the highest drop in battery voltage while case III that corresponds to tasks ordered in decreasing order of loads has the lowest drop in battery voltage and thus the best performance. Thus, tasks should be ordered in decreasing order of load currents.

4. BATTERY-FRIENDLY FIR

FIR filtering is one of the most commonly used signal processing operations. An N-tap FIR filter is defined by:

$$y[n] = \sum_{k=0}^{N-1} x[n-k]h[k]$$

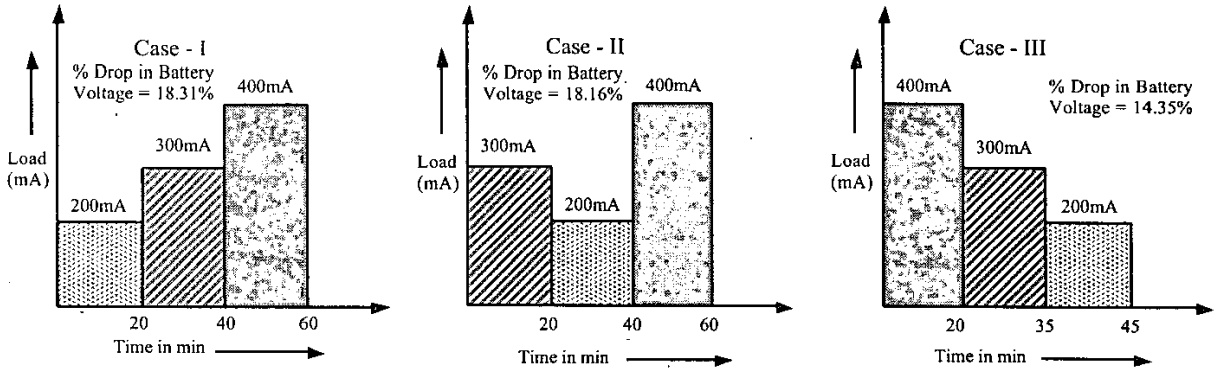


Figure 2: Battery behavior for different load profiles

FIR filtering is essentially a set of MAC (Multiply and Accumulate) operations. Efforts to reduce energy consumption in FIR filters include choice of coefficients so that the Hamming distance between successive coefficients is reduced, use of differential coefficients [8], use of canonical sign digit representation [9], etc. Also Energy-Quality tradeoffs have been demonstrated for such filters in [10]. While these techniques can be used for lowering the energy consumption, additional gains in battery lifetime can be obtained by shaping its current profile.

Consider an FIR filter with the variation in coefficients as shown in Fig. 3. The corresponding current profile is shown in dashed lines. It can be seen that for larger coefficient values, the current drawn is larger. Hence if we can reorder the coefficients as shown in Fig. 4, we can have a non-increasing current profile, which would give rise to a better battery performance. The accuracy of the result is also plotted along side the current profile with a dotted line. A battery-friendly FIR filter computation would thus consist of reordering the coefficients in decreasing order and computing the larger-valued coefficients first.

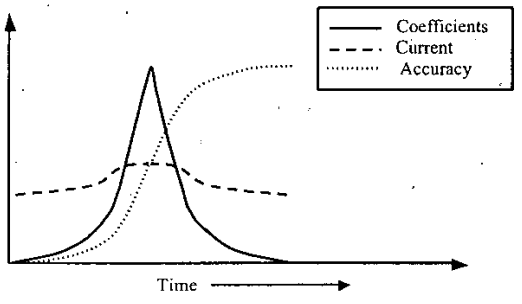


Figure 3: FIR Filter (Initial Arrangement)

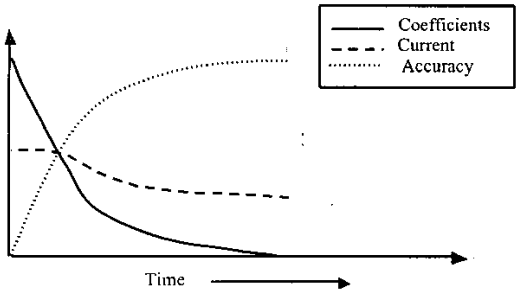


Figure 4: FIR Filter (New Arrangement)

We illustrate this with the following three filter examples: The first example is a 128 tap complex filters while second and the third is a 64 tap complex filter. All the filters have coefficient values ranging from 0 to $1x10^6$; about twenty coefficients have values that are much higher than the remaining coefficients. The inputs to the FIR filter are correlated. Table 2 shows the energy consumption in the original filter and the reordered filter, and the percentage drop in battery voltages for these examples. These values are obtained after repeating the filter computation 6 x 10^6 times, which corresponds to few tens of minutes. While the energy consumption is almost identical, the drop in battery voltage for the reordered filter is about 1.5 – 2% less.

Filter	Old Arrangement		New Arrangement	
	Energy(J)	% Drop In Battery Voltage	Energy(J)	% Drop In Battery Voltage
1	128.34	13.82%	129.84	13.10%
2	64.26	9.73%	65.48	9.46%
3	59.22	8.30%	61.26	8.60%

Table 2: FIR Filter Results at f = 206MHz (when program was executed 6 x 10^6 times)

Now consider the case where the filters are approximated by using fewer taps at the expense of quality degradation as in [10]. Fig. 5 shows the battery voltage drops for the original, re-ordered filter with all taps and re-ordered filter with twenty taps (that causes a small quality degradation of 3.21%) for Filter 1 in Table 2. Thus, reducing the number of taps results in a smaller drop in battery voltage (7.24% instead of 13.1%) with very little quality degradation. Note that as the frequency of operation reduces, the difference in the drop in battery voltage between the original and reordered filter is smaller. This is because the variation in the current in the original and the reordered arrangements reduces with frequency.

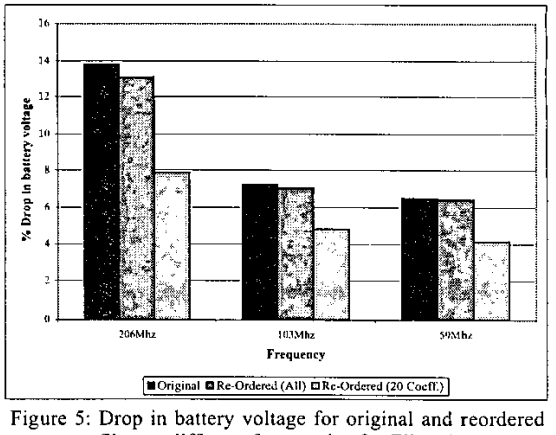


Figure 5: Drop in battery voltage for original and reordered filter at different frequencies for Filter 1

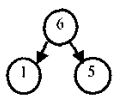
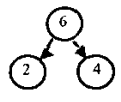
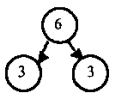
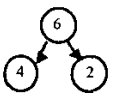
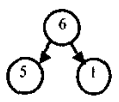
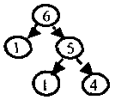
Decomposition						
Time (in min)	4.905	5.934	6.113	6.035	5.435	7.699
Energy (in Joules)	10.242	12.36	12.72	12.60	11.36	16.08
% Drop in battery voltage	3.219	3.597	3.622	3.644	3.41	4.18

Table 3: Results for 2^6 point FFT (when executed 6 x 10^3 times)

5. BATTERY-FRIENDLY FFT

In many areas of mobile computing, there is a great demand for signal-processing algorithms based on the Discrete Fourier Transform (DFT). An efficient way to implement DFT is the Cooley-Tukey algorithm referred to as the Fast Fourier Transform (FFT) [11]. This reduces the arithmetic cost of the computation of DFT of size N from $O(N^2)$ to $O(N \log(N))$. There are many ways in which a FFT can be decomposed. All decompositions have the same arithmetic cost but depending on the data access pattern, the time required for computation differs dramatically [12]. Interestingly, the different decompositions have very different energy consumption and very different battery performance.

5.1. Mathematical Framework

Consider a 2^n point FFT decomposed into two FFTs of sizes 2^l and 2^m where $n = l + m$
$L = 2^l$, $M = 2^m$, $N = 2^n$ and $N = LM$.

This FFT can be expressed as a double sum over the elements of the rectangular array multiplied by the corresponding phase factors. Hence we have:

$$X(p,q) = \sum_{k=0}^{M-1}\sum_{j=0}^{L-1} x(j,k)W_N^{(Mp+q)(kL+j)}$$

$$= \sum_{j=0}^{L-1}\left\{W_N^{jq}\left[\sum_{k=0}^{M-1} x(j,k)W_M^{kq}\right]\right\}W_L^{jp} \quad \ldots (1)$$

The calculation of equation 1 can be subdivided into three steps:

1. Computing M-Point DFTs for each of the L rows.
2. Scaling with the twiddle factor.
3. Finally, computing L-point DFTs for each of the M Columns

5.2. Energy and Battery Voltage Results

The results for 2^6 point FFT for various decompositions are shown in Table 3. All the results are for the case when the code is executed 6 x 10^3 times. Note that there is a significant difference in both the execution time and energy consumption for even such a small example. The drop in battery voltage is not an eye-catcher since the execution time is of the order of few minutes and the drop is very small in the first few minutes of operation. From this simple example we see that splitting the subtrees equally is not the best strategy. In fact, for the StrongARM platform, it is preferable to do a right hand tree decomposition as opposed to left hand tree decomposition. However, the optimal decomposition would differ from machine to machine.

Table 4 shows the time, energy consumption and percentage drop in battery voltage for the top five decompositions for various point FFTs mentioned in [12]. It can be clearly seen that for the top five cases, there is a wide variation in the time required to compute the FFT, energy consumed for the computation as well as the battery voltage. For the 1024-point FFT, the time taken for a $(2^5x(2^2x2^3))$ decomposition is 13.19% higher than that of a $(2^4x(2^1x2^5)$ decomposition resulting in a 20% difference in drop of battery voltage when executed 6 x 10^3 times.

Note that the decomposition that results in minimum time is the same as the decomposition that results in minimum energy, which is the same as the decomposition that results in higher battery performance. This is because for the StrongARM processor, the variation in current as it executes the different algorithms is negligible. Since the load profiles of different algorithm depend only on the execution time, as the execution time increases, energy increases and the battery voltage decreases.

FFT Type		Time(in min)	Energy(in Joules)	% Drop in Battery Voltage
Point	Impl.			
128	2^1x2^6	10.17	21.33	4.83
128	2^2x2^5	12.32	25.80	5.28
128	2^3x2^4	12.85	26.93	5.38
128	2^4x2^3	12.96	27.14	5.40
128	$2^2x(2^1x2^4)$	17.95	37.56	6.11
256	2^2x2^6	25.60	53.79	6.89
256	2^1x2^5	26.86	56.38	7.00
256	2^5x2^3	27.02	56.74	7.02
256	$2^1x(2^1x2^6)$	31.01	64.80	7.35
256	$2^1x(2^2x2^5)$	34.00	71.16	7.63
512	2^3x2^6	55.92	117.48	10.26
512	2^1x2^5	57.08	119.94	10.42
512	2^5x2^4	57.23	123.12	10.44
512	$2^1x(2^3x2^5)$	71.74	150.40	12.53
512	$2^3x(2^1x2^5)$	77.31	161.79	13.43
1024	2^4x2^6	119.23	250.86	21.46
1024	2^5x2^5	119.69	251.76	21.55
1024	$2^4x(2^1x2^5)$	162.25	339.75	34.08
1024	$2^5x(2^1x2^4)$	167.03	349.86	34.26
1024	$2^4x(2^2x2^4)$	177.60	372.12	37.90
1024	$2^5x(2^2x2^3)$	183.66	372.12	41.07

Table 4: Results for 5 best decompositions in [12] for $n = 2^7$ to $n = 2^{10}$ point FFT repeated 6×10^3 times

Next we analyze the drop in battery voltage for different frequency settings of the StrongARM processor. As the frequency reduces, the time increases, but the current reduces significantly. The large drop in the load current results in significantly better battery performance. Fig. 6 shows the drop in battery voltage when the best decomposition (from Table 4) is executed at 3 different frequencies. Clearly, if there is sufficient timing slack, the processor should be operated at a lower frequency.

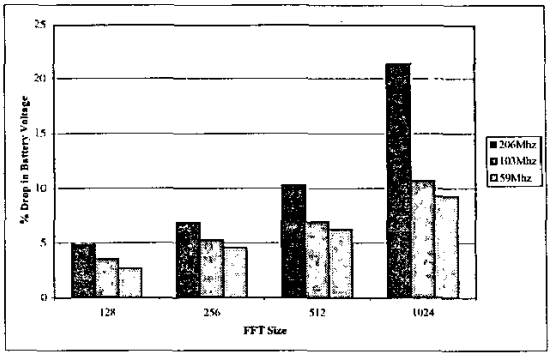


Figure 6: Battery performance at different frequencies for FFTs of size $n = 2^7$ to $n = 2^{10}$

6. BATTERY – FRIENDLY MPEG

Video compression is most vital for storage and transmission of video data. MPEG2 offers one of the most efficient ways to compress video data. The block diagram for MPEG2 is show in Fig. 7. We have divided the MPEG2 kernel into 3 main parts as shown by the dotted blocks, viz. DCT, Motion Estimation (ME) and VLC Encoding. Table 5 describes the time, current and energy for these blocks while processing the I frame at 206 MHz. The overhead due to the other parts is ignored compared to these three blocks.

	DCT	Motion Estimation	VLC
Time (in μsec)	46.7	584.9	242.5
Current (in A)	0.2669	0.2565	0.2680
Energy (in μJ)	18.68	225.50	99.00

Table 5: Time, Current & Energy at f = 206 MHz for different blocks of MPEG2

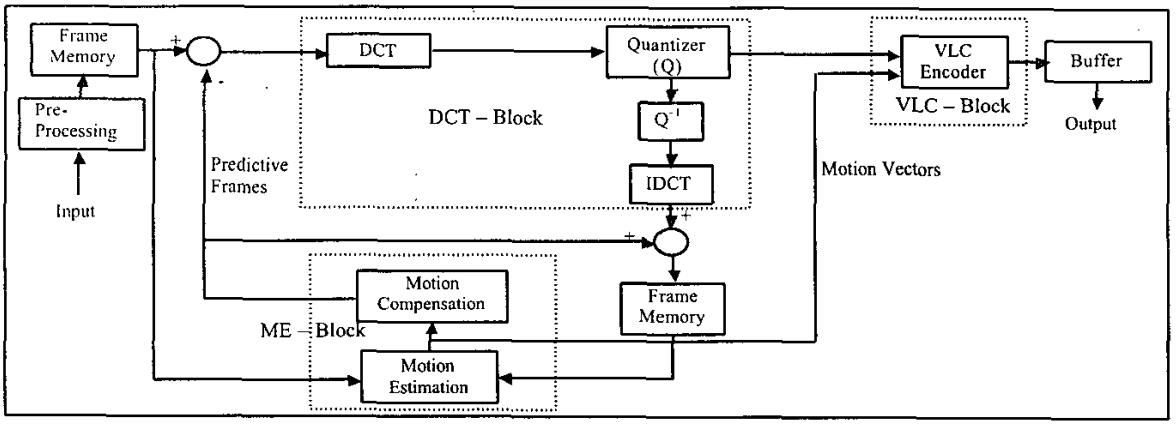


Figure 7: MPEG Block Diagram

The DCT has been implemented using the Chen-Wang algorithm and motion estimation has been implemented using the full search method. The block size is 8 x 8 and the frame size is 352 x 288 (CIF). Clearly motion estimation is the most energy-consuming block and approximations in it will lead to a longer battery life.

The time required to run all three blocks at full speed (206 MHz) is 864.1μsec. Consider the case when the I frames are run at full speed and the P and B frames are run at lower frequency based on available slack. Assume that it is possible to operate the different blocks (DCT, Motion Estimation, VLC) at different frequencies. Next we show various ways to distribute the available slack and their effect on battery performance. Table 6 shows the drop in battery voltage for different distributions of available slack. The slack, time and energy are listed for one iteration but the drop in battery voltage is quoted for 6 x 10^3 iterations, which corresponds to few tens of minutes.

For instance when the slack available is 309μsec, the first arrangement has an energy consumption of 235.14μJ compared to 252.38μJ for the second arrangement. Hence we would expect the battery voltage for the second arrangement to drop more. But this is not the case; the drop in voltage in the first arrangement is larger than the second arrangement. The main reason for this is that the currents are arranged in increasing order in the first arrangement and in decreasing order in the second arrangement.

Slack (in μsec)	DCT (in MHz)	Mot. Est. (in MHz)	VLC (in MHz)	Time (in μsec)	Energy (in μJ)	%Drop in Battery Voltage
78.9	148	192	206	943.3	304.16	17.85
	206	206	162	943.8	306.58	15.77
309	89	148	206	1165	235.14	15.31
	206	192	103	1173	252.38	12.33
660	59	103	206	1520.5	194.95	13.66
	206	133	89	1520.8	173.12	9.98

Table 6: MPEG2. Distribution of slack

In fact the slack distribution algorithm should be such that the later tasks get a larger share of slack i.e., operate at a lower frequency. Table 7 shows two configurations with decreasing load profiles, equal energy consumption, and comparable execution times that have a 1% difference in battery voltage drop even when executed for tens of minutes.

Slack (in μsec)	DCT (in MHz)	Mot. Est. (in MHz)	VLC (in MHz)	Time (in μsec)	Energy (in μJ)	%Drop in Battery Voltage
567.9	206	133	118	1435	183.9	11.27
	206	148	89	1429	183.5	10.27

Table 7: MPEG2. Steeper current profile does better

Configuration 2 does better than configuration 1 since the 3rd task (VLC) is allocated a larger chunk of the slack resulting in the current profile being steeper.

7. CONCLUSION

In this paper we have looked at ways in which signal processing algorithms can be made more battery-efficient. We first looked at how reordering and frequency scaling affects battery lifetime in case of FIR filters. Next we looked at how various decompositions for FFT affect the battery life. Finally, examined how slack can be utilized in a battery-friendly way, in case of larger signal processing algorithms, like MPEG2.

8. REFERENCES

[1] T. Martin, "Balancing Batteries, Power and Performance: System Issues in CPU Speed Setting for Mobile Computing," *Ph.D. Dissertation*, Aug. 1999.

[2] M. Pedram and Q. Wu, " Design Considerations for Battery Powered Electronics," *Proc. of DAC*, 1999.

[3] P. Chowdhury, C. Chakrabarti, "Battery-aware Task Scheduling for System-on-Chip Using Voltage/Clock Scaling," *Signal Processing Systems, IEEE Workshop on*, pp. 201-206, 2002.

[4] D. Rakhmatov, S. Vrudhula and C. Chakrabarti, " Battery-Conscious Task Sequencing for Portable Devices including Voltage/Clock Scaling," *Proc. of DAC*, pp – 189-194, 2002.

[5] J. Luo and N. Jha, "Battery Aware Static Scheduling for Distributed Real Time Embedded Systems", *Proc. of DAC*, 2001.

[6] T. Fuller, M. Doyle, and J. Newman, "Simulation and Optimization of the Dual Li-ion Insertion Cell," *Jour. Electrochemical Society*, 14(1), 1994.

[7] A. Sinha, A. Chandrakasan, "JouleTrack – A Web Based Tool for Software Energy Profiling," *Proc. of DAC*, pp – 220-225, 2001.

[8] N. Sankarayya, K. Roy and D. Bhattacharya, "Algorithms for Low Power and High Speed FIR Filter Realization Using Differential Coefficients," *IEEE Trans on Circuits and Systems*. Vol 44, No. 6, pp – 488-497, July 1997.

[9] R. Hartley, "Optimization of CSD Multiplier for Filter Design," *IEEE Intl. Symposium on Circuits and Systems*, Vol. 4, pp – 1992-1995, 1991.

[10] A. Sinha, A.Wang and A. Chandrakasan, "Algorithmic Transformations for Efficient Energy Scalable Computation," *Proc. of ISLPED*, 2000.

[11] J.W. Cooley, J.W. Tukey, "An Algorithm for the machine calculation of complex Fourier Series," *Mathematical Computation*, Vol. 29, pp – 297-301, 1965.

[12] Gavin Haentjens, "An Investigation on Cooley-Tukey Decompositions for the FFT", *Master's Thesis*, Carnegie Melon University, 2002