Project: Power Aware Video Codec-Phase 1
PI: Dr. Nadeem Ahmad Khan
Co-PI: Dr. Jahangir Ikram

Power Consumption Analysis for Mobile Devices and Power Consumption Mathematical Models
Version 1.1

Doc Number: LUMS-MRL-PAV-004-ICT

Prepared by: Zahaib Akhtar,
Tayyab Naseer
Contents

1.0 System Level Power Consumption Analysis of Mobile Devices .......................................... 4
  1.1 Overview .......................................................................................................................... 4
  1.2 Sub-block Power Consumption ..................................................................................... 4
    Figure 1 Structure of modern mobile phones .................................................................... 5
    Table 1 Typical energy distribution in multimedia mobile phone .................................... 6
    Table 2 Encoding time vs. Memory Access ....................................................................... 7
    Figure 2 Energy consumption of three applications in video streaming ....................... 8
  1.2 Conclusion ....................................................................................................................... 8

2.0 Monitoring the Battery Status in Cell Phones and other Embedded Devices ............... 8
  2.1 Using the Ctelephony .................................................................................................... 8
  2.2 What is the S60 Development Platform? ..................................................................... 10

3.0 Power Measurement and Modeling Techniques ............................................................ 10
  3.1 Introduction .................................................................................................................. 10
  3.2 Model Proposed by Tiwari et al. .................................................................................. 10
    3.2.1 Issues and Problems in the Scheme ...................................................................... 11
  3.3 Model Proposed by Russell et al. ................................................................................ 11
    3.3.1 Evaluation of the Model ........................................................................................ 12
  3.4 Model Proposed by Gebotys et al. .............................................................................. 12
    3.4.1 Evaluation of the Model ........................................................................................ 13
  3.5 Model Proposed by Rizo-Morente et al. ...................................................................... 13
    3.5.1 Evaluations of the Model ....................................................................................... 14
  3.6 Recommendation .......................................................................................................... 14

4.0 References ...................................................................................................................... 15
1.0 System Level Power Consumption Analysis of Mobile Devices

1.1 Overview

In wireless video streaming- multimedia applications, memory transfers, modem operations and displaying peripherals operate on the mobile devices with limited energy. For this purpose, it is essential to model (i.e., estimate) the energy cost of the streaming system. A primary factor in determining the utility or operational lifetime of the mobile communication device is how efficiently it manages its energy consumption. The problem becomes even more vital when encoder with high computational complexity is integrated into the system. The device size and the small battery lifetime keep limiting the available power resources. Heat dissipation is also another important factor. Excessive heat dissipation makes the handheld device too hot to handle. This report provides some insight to the power consumption characteristics of the subsystems especially encoder as compared to the whole system. Arguments are supported with materials extracted from research papers and survey reports.

1.2 Sub-block Power Consumption

From a system level point of view the video streaming architecture can be divide into following sub-blocks

1. Cellular Subsystem
2. Application
3. Hardware Accelerators
4. Power Management
5. Video Peripherals
The **cellular subsystem** is the part of the device responsible for all phone related operations of the mobile. It is a digital baseband unit managing 3G modems (GSM, GPRS UMTS), network access, signal transmission and reception.

The **application subsystem** deals with all the multimedia processing algorithms and the user interfaces. The audio and video codecs, the image compression and processing, the communication, with all the peripherals and the connection to various networks (WLAN, Bluetooth and GPS) are tasks of this engine.

**Video Peripherals**: Apart the processing units we described above, mobile phones have various peripherals, which are connected with the different modules through digital serial communication links. These links are implemented and controlled once again by the master control unit, i.e. the application engine. Among the peripherals we will focus only on the **display**, due to its important contribution in the overall system’s power consumption.

*Figure 1 Structure of modern mobile phones*

*Source: References [9]*
Emphasis of this report is to figure out what percentage of power is consumed by the encoder as compared to the rest of the working blocks. This activity will provide us lucid idea about how critical the encoding section is under energy constraint environments.

According to a research paper [9] the more power consuming subsystem is the application engine. Energy distributions of all other subsystems are also listed in the following table.

<table>
<thead>
<tr>
<th>Subsystem Application</th>
<th>Application Energy Distribution</th>
<th>Subsystem</th>
<th>Subsystem Energy Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory subsystem</td>
<td>4.4%</td>
<td>Multimedia</td>
<td>39.5%</td>
</tr>
<tr>
<td>Instruction memory</td>
<td>9.9%</td>
<td>Memory</td>
<td>19.4%</td>
</tr>
<tr>
<td>Instruction memory</td>
<td>15.5%</td>
<td>Modem</td>
<td>21.5%</td>
</tr>
<tr>
<td>Instruction memory</td>
<td>7.8%</td>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>5.0%</td>
<td>LCD</td>
<td>17.6%</td>
</tr>
<tr>
<td>Instruction memory</td>
<td>3.7%</td>
<td>Others</td>
<td>2%</td>
</tr>
<tr>
<td>Instruction memory</td>
<td>13.9%</td>
<td>Others</td>
<td>2%</td>
</tr>
</tbody>
</table>

Table 1 Typical energy distribution in multimedia mobile phone

Source: See References[9]

**Memory subsystem** is the critical source of power consumption consuming 20% of the available resources. Instruction memory accesses and data memory accesses are of primary importance. Pipelining architectures are available to reduce the memory accesses for instruction fetching. Data compression techniques that minimize data and hence, the frequency of memory accesses, can reduce the memory power consumption. This stays true, however, as long as the additional computations do not add more power loss than the actual obtained power gain.

**Display:** The display and the display backlights have a significant impact on the device’s power consumption. How much is this impact depends on the size of the display, since the bigger the display the more the power consumed, the manufacturer and whether it is colorful or not. Also,
the architecture of the display can determine the energy loss. Displays with internal memory, for example, seem to spend less battery resources [10]. This can be explained by the fact that display content can be saved temporarily to the display’s internal memory, resulting in fewer accesses to external memories and thus, to less power consumption. Finally, the display backlights accounts for important energy overheads in a battery-operated device [11]. As the brightness of the backlights increases, more energy is consumed.

With reference to a survey [12] on the video encoding power consumption characteristics it is shown that encoding consumes most of the power as compared to other blocks such as transmission and video security. In this research paper they have divided the power consumed by the system into two parts (1) Memory access and (2) Encoding. Their results show that encoding consumes almost double the power memory access uses. Following table validates the argument.

<table>
<thead>
<tr>
<th>Test Sequence</th>
<th>Energy(memory access) (mJ)</th>
<th>Energy(encoding)(mJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>7.19</td>
<td>18.95</td>
</tr>
<tr>
<td>Bus</td>
<td>6.55</td>
<td>16.37</td>
</tr>
<tr>
<td>Harbor</td>
<td>7.05</td>
<td>17.55</td>
</tr>
<tr>
<td>Mother and Daughter</td>
<td>5.42</td>
<td>12.90</td>
</tr>
</tbody>
</table>

*Table 2 Encoding time vs. Memory Access*

*Source: References [12]*

Now this table clearly shows the amount of energy consumed by an encoder. It almost consumes 70% of the total power. So development of efficient methods of encoding videos will be an interesting and vital area of research for video streaming system.

Another research paper [13] which focuses on the power consumption by the video encryption also discusses the power consumption of encoding, decoding and encryption separately. Following graph extracted from its results validates the argument of excessive power consumption of an encoder.
1.2 Conclusion
All of the above documentation explains the vital role of encoding videos efficiently to be able to save the power consumption of the battery. Development of optimized and scalable algorithms, hardware dynamic and adaptive voltage and frequency scaling will be crucial for the power aware video encoding.

2.0 Monitoring the Battery Status in Cell Phones and other Embedded Devices

2.1 Using the Ctelephony

The Ctelephony is an API that is provided by the Symbian OS itself, the Ctelephony itself is developed in C++ and provides a list of functions to carry out specialized tasks, some of the functions that are implemented in Ctelephony are the following:

- **AnswerIncomingCall(TRequestStatus &,TCallId &,const TPhoneLine)const**
  - Answers an incoming new voice call. Fax and data calls cannot be answered.
- **EGetSignalStrength**
  - Measures the current Signal Strength available.
- **GetBatteryInfo(TRequestStatus &,TDes8 &)const**
  - Retrieves the status and charge level of the phone battery.

This class provides a simple interface to the phone's telephony system. It provides two services:
1. You can find out information about the phone. This class provides support for retrieving Phone Settings, Line Information, Call Functionality, Network Information and (basic) Supplementary Service Settings.

2. You can dial, answer and control voice calls. You cannot make fax or data calls.

Architecturally, CTelephony provides an interface to the Telephony Multimode API. This interface exposes only a subset of the complete Multimode functionality.

Note that a CActiveScheduler is required to be installed for all asynchronous CTelephony calls. Following is a broad categorization of the tasks that can be carried out with Ctelephony. As stated above, its basic job is to provide information about the phone.

- **Phone, battery and subscriber information**

  Find out about the phone - its make, model and serial number. Also find out about the battery, locks and the flight mode status.

- **Line and call information**

  Find out about calls in progress. Limited information is available for fax and data calls, but detailed information can be read about voice calls.

- **Network information**

  Find out about the currently connected network, including its name, status and signal strength.

- **Supplementary services setting information**

  Find out about the call forwarding, call barring, call waiting and caller identification status of the phone.

You can access this information in two ways:

1. You can get the current value, such as the current battery level, current signal strength and whether a call is currently being made. See the description of each item of information for an example.

2. You can request notification when information changes. For instance, you can be notified when the battery level changes, when the signal strength changes and when there is an incoming call to be answered.
Although for advanced development, the Ctelephony is not used as other licensed tools are available, these tools are mostly used by the manufacturers themselves, whereas on the other hand the Ctelephony is open for use by any third party developer. The Ctelephony and the complete package of a lot of other APIs are part of the S60 5th Edition C++ Developers Library. The S60 development platform can be downloaded for free of cost from forum.nokia.com

2.2 What is the S60 Development Platform?

The S60 Platform is a software platform for mobile phones that runs on Symbian OS. S60 is currently one of the best smartphone platforms in the world and currently under the ownership of Nokia, which is also in the process of making the source open for both the S60 user-interface which Nokia developed and licensed and the newly acquired underlying Symbian OS. Licensed users of S60 include Lenovo, LG Electronics, Panasonic and Samsung.

3.0 Power Measurement and Modeling Techniques

3.1 Introduction

A total of six research papers for the purpose of getting an in depth overview of existing power measurement techniques for DSP and embedded processors were studied. The papers covered a broad timeline ranging back to 1995 and as early as 2006. As many as four different techniques for power estimation in DSP processors have been identified from the papers. A broad categorization of these can be made as methods modeling the estimation at the Instruction Level and on the other hand, methods modeling the estimation at Average Power per Cycle.

3.2 Model Proposed by Tiwari et al.

The model proposed by Tiwari et al. has followed a systematic method of calculating the cost of each instruction by taking into account the Base Cost and the Circuit Overhead for each instruction, base cost is defined as “the average current drawn by the processor during the repeated execution of the instruction” and the overhead cost is defined as the cost ”needed to account for the effect of circuit state change for an instruction sequence consisting of different instructions” (Tiwari et al.). Once the current is known the value can be used to calculate the energy by using the following formula.

\[ E = (I \times V_{dd}) \times (n \times \tau) \]

Where:
- \( E \) is the energy
- \( I \) is current
- \( V_{dd} \) is supplied voltage
- \( n \) is the number of cycles
- \( \tau \) is cycle period

For the measurement of the current no special equipment has been used, and current is measured using a simple off-the-shelf oscilloscope. The instruction for which the base current consumption is to be measured is placed in an infinite loop and the current measured. However if two instruction say \( i \) and \( j \) are put into an infinite loop for measurement their current is always greater than the sum of their individual base currents. This difference is called the overhead cost. Hence the total energy consumed by a program is the sum of total base costs and the total overhead costs over all the instructions used.
In addition to these two types of cost another cost may be incurred by the operations involving the Booth Multiplier which is defined as the following in the paper “this is due to a special design at the inputs of the multiplier. A latch for each input operand is put between the multiplier and the operand bus to retain the old values until the next multiply instruction is executed, the state change at these input latches” accounts for an overhead that is separate from the simple state change overhead and must be added to the overall cost.

### 3.2.1 Issues and Problems in the Scheme

The instruction-level power modeling technique described above suggests that accurate results may be obtained if a table that gives the base cost for each instruction can be obtained and similarly another table for the overhead costs of the instructions can be achieved. Such tables can be empirically constructed through experiments. However, practically the power cost of instructions may vary depending on the operand value and hence would require extensive experimentation to build up an average cost for these instructions.

The other issue is that of the table size. For processors with “rich instructions sets” assigning power costs to all instructions and instructions pairs can lead to large tables. The way around this problem is to arrange the instructions into different classes so as to reduce the size of the table, however that comes with a loss in accuracy.

### 3.3 Model Proposed by Russell et al.

The model proposed by them uses an i960 processor as the target processor, two implementation of the i960 architecture namely, 80960JF and 80960HD were considered for the experimentation. The basic idea used by Russell et al. is to calculate the processors’s average power per cycle along with the execution time of the program (in cycles), the product of these two quantities will give the estimate of the Power consumed by the application for the duration it has run.

The following is the energy model for the processor running the program

$$E = \int_{t_0}^{t_n} P(t) \, dt$$

Where $T$ is the software execution time and $P(t)$ is the instantaneous power. Average power, $P_{ave}$ is defined as

$$P_{ave} = \frac{1}{T} \int_{t_0}^{t_n} P(t) \, dt$$

which is further written as

$$E = T \cdot P_{ave}$$

Hence the total energy consumed by a process is the product of the time it takes to execute and the average power consumption of the current for that processor. To compute the $P_{ave}$ a resistor was placed in series with the battery in between the processor and the battery. A digitizing oscilloscope namely LeCroy LC 534 with a sample rate of 500 Mbps was used to measure the instantaneous power. Channel 1 of the scope was used to measure $V_1$, the voltage from the battery and channel 2 was used to measure the differential voltage across the resistor $(V_1-V_2)$ and then scaled according to the resistance using the following expression.
The average power $P_{ave}$ was then computed for various iterations of the loop and the values averaged to give the average power.

### 3.3.1 Evaluation of the Model

The model gives a much better perspective than the model proposed by Tiwari et al. instead of modeling the estimation based on instruction level this model takes into consideration the average power per cycle. As an instruction may take more than one cycle to execute it makes more sense to have the estimate based on the average power per cycle.

The model is much simpler to implement than the model proposed by Tiwari et al. and does not require lengthy experimental procedures. The model uses a constant parameter i.e. the voltage which is easier to measure. The results of the test show a good accuracy and it states with 99% accuracy that the constant parameter model predicts power consumption with less than 8% error.

### 3.4 Model Proposed by Gebotys et al.

This model follows the principal of finding out the power consumption characteristics of the instruction set architecture and all their equivalent implementations in order to proceed with low power software designs of applications. In their own words this is described as following “if program implementations are based upon low power consumption constructs the final program power consumption will be low and the energy $W_{program}$ consumed by the processors executing the program $P$ will be minimized.”

The experimentation in the paper has been performed on a Starcore DSP Processor, SC140. When dealing with DSP processors the measurement of the current dynamic at instruction or program level is a difficult task. The difficulty lies in the type of the oscilloscope used and on the method of the synchronizing the software with the measuring equipment. The main ideas of this three-fold problem are as follows:

1. How can the software program be synchronized with the measuring equipment.
2. How the exact position of an instruction or of a set of instructions in the current waveform can be determined.
3. How does the current measurement take place for a program which does not execute in an infinite loop.

To solve the above three problems the team has come up with the following equipment:

1. StarCore 140 Software Development Platform (SC 140 SDP)
2. A Signal Pattern generator (HP 8113A)
3. A sampling type oscilloscope (TDS8000) equipped with high frequency sampling heads, high frequency current probes and high frequency voltage probes.

And the following methodology has been followed to solve the synchronization problem:

- The processor’s clock is externally provided at the processor speed by the pattern generator.
- The processor is kept in the wait state (a state that is equivalent with a standby state), were the state of the processor is preserved and the internal processor’s clock is disabled. This state is a very low current consumption state for the SC140 processor [1].

$$P(t) = I(t)V(t) = \frac{V_1(t) - V_2(t)}{R}V_2(t)$$
A non-maskable (or maskable) interrupt signal synchronous with the processor’s clock is generated by the pattern generator at the NMI pin of the processor.

The program or the instructions under measurement are boarded with a series of NOP instructions [1] before and after entering wait state.

The NMI signal triggers the oscilloscope and causes the SC140 processor to come out of the wait state. As the processor comes out of the wait state the program execution is directed towards the program under measurement.

At the end of the program execution the processor is introduced again in the wait state.

Based on the synchronization principle presented the program is incorporated within a SC140 assembly language template. The template is designed as an interrupt service routine. Normally the processor enters into a wait state after each execution of the interrupt service routine until a new interrupt request comes.

### 3.4.1 Evaluation of the Model

The difference between this method and the looping method followed by Tiwari et al. is that to the beginning of the program. This method simulates more accurately the real dynamic behavior of the one run type DSP applications.

Synchronizing the processor’s current draw measurement with the NMI signal provides an accurate data acquisition for the TDS8000 sampling oscilloscope. As a result the current waveforms display the true current dynamics within the processor while executing the program under measurement and give a stable waveform with true instantaneous values at each sample point in the $T_{NMI}$ interval.

However, the implementation of the system and the work station is much more complex than the system proposed by Russell et al. But this complexity does not come without a benefit, it provides a much realistic and accurate power estimation of the instructions than both the methods discussed above.

the processor enters the wait state after each time the program is executed instead of looping

### 3.5 Model Proposed by Rizo-Morente et al.

This paper proposes a novel method for estimating the dynamic current consumption of a processor. The method models dynamic current as the output of a linear system excited by a signal comprised of the total current due to each instruction.

The dynamic current behaviour is determined by the power supply system including the processor, regulator, decoupling components and printed board circuit. They model the dynamic current consumption of a processor power supply system as the output of a linear system excited by an input signal consisting of the total current consumption due to each instruction including the inter-instruction effect. Thus the estimated instantaneous current $y_c[n]$ can be calculated as the convolution of the discrete input signal, $x_d[n]$, with the system impulse response $h_i[n]$:

$$y_c[n] = \sum_{k=0}^{N} h_i[k] x_d[n - k]$$

The input signal $x_d[n]$ can be derived by applying the conventional static Tiwari model to each
individual instruction in the execution trace. For this work, a previously published static model for the TMS320VC5510 DSP processor was employed. The measurement framework is summarized as follows:

The target DSP processor used for the study is a Texas Instruments TMS320VC5510, with variable core voltage (0.9-1.6 V) and operating frequency of up to 200 MHz. It incorporates several special architectural features pertinent to the study. Among them, there are low power capabilities, parallel features, on-chip Single Access RAM (SARAM) and Double Access RAM (DARAM), two independent 40 bit MAC units and one 40 bit ALU.

The physical measurement methodology was applied using the C5510 Development Software Kit, with a 1.6 V core voltage and 24 MHz operating frequency connected to a PC running TI Code Composer Studio (CCS) version 2.56. CCS was used to download and run the test programs. External software routines were used to trigger measurements using a digital storage scope. The current drawn was measured with a non intrusive 0.1 mA resolution current probe. The probe bandwidth is 50 MHz, enough for these experiments. In order to avoid noise, each measurement was averaged over 128 instances. The measurements are completely repeatable.

The experiments results showed square correlation coefficients of 98% and 91% when compared with the impulse response. Almost all current estimates provide a Squared Cross Correlation in excess of 93.12%, however, some variations in the current consumption are not well captured. The worst case single error was 15.5%.

3.5.1 Evaluations of the Model

The static model does not contain characterized current values for every instruction of the instruction set. When an instruction is not characterized, errors may be introduced in the signal $x_d[k]$.

The ILPA (Instruction Level Power Analysis) model as introduced by Tiwari was not intended to model the dynamic variations of the processor current consumption. The ILPA models attribute a total current consumption to each instruction or instruction parameter and do not model the current as it is actually drawn. Therefore, these methods may be effective in measuring the long term mean current but do not attempt to estimate the cycle-to-cycle current variations. This model hence scored in that regard.

3.6 Recommendation

- In the light of the above studied models, the model that gives the most detailed and accurate measurement of the power usage of the DSP processor is the one introduced by Gebotys et al. the methodology has been quite clearly defined and the details on the used equipment is also sufficient. Therefore it will be easy to replicate the procedure for the estimation in our project and hence is the recommended model, however, the procedure outlines by the team involves heavy expenditure for the procurement of equipment.

- Second preference should be given to the model introduced by Russell et al. this model is very simple to implement and also gives an accurate estimate of the power consumption. The difference between this model and that introduced by Gebotys is the basic methodology that has been followed. Russell et al have computed the average power per cycle instead of modeling the estimation in terms of power dissipation per instruction. Hence this model can
give another perspective on the findings, the equipment to be purchased for the replication of this method is not as costly as preference no. 1.

4.0 References
The following is the list of the publications used as reference for this report.


