System-level Early Power Estimation for Memory Subsystem in Embedded Systems

Jinsong Ji\(^1\,2\)\(^+\) Chao Wang\(^1,2\) Xuehai Zhou\(^1,2\)

\(^1\)Dept. of Computer Science
University of Science and Technology of China
Hefei, Anhui, 230027, China
jsji@mail.ustc.edu.cn
saintwc@mail.ustc.edu.cn

\(^2\)Embedded System Laboratory
Suzhou Institute for Advanced Study, USTC
Suzhou, Jiangsu, 215223, China
xhzhou@ustc.edu.cn

Abstract

Early power estimation is important to guide architectural design, especially for embedded systems. Since the power consumption of memory subsystem dominates, while DRAM and NAND Flash are the two main storage mediums nowadays, we analyze the power model of DRAM and propose a power model for NAND Flash, considering its system-level behaviors. Experimental results show that the accuracy of model proposed can be up to 95%.

1. Introduction

Nowadays, power consumption is already one of the leading constraints in designing embedded systems[8]. And memory system cost is a major factor in the total power consumption[12, 7]. A share of up to 80% of the total power has been attributed to the memory subsystem in the signal processing domain [2]. So system designers are paying more and more attention to memory subsystem powers.

In order to create power efficient designs, accurate power budgets for the memory subsystem are essential, whether it is for calculating battery life, planning cooling system, or determining the power supply.

Though there have been several different power estimation techniques[6, 9, 15, 16, 3, 12, 11] and different related tools, such as SimplePower[14], Wattch[1], JouleTrack[13], VPR[10] et al; most of them focus on micro-architecture level power estimation. System designers usually could not utilize these techniques or tools in early design stage, because they may not have enough micro-architecture information, or enough time to run a whole simulation for the design. Another problem is that most of the tools focus on processors or controllers, memory subsystems are ignored, to say nothing about memory hierarchies.

2. System-level Power Models

In this paper, we propose a mathematical power model for typical memory subsystem in embedded systems, which can help designers gain accurate power budgets in early stage of architecture design, requiring only the knowledge of system level behaviors and related medium parameters. We first analyze the main components of modern memory hierarchy, then describe the power model of DRAM provided by Micron[5] and propose our model of NAND Flash in details; after that we show our experimental results which verified the models presented.
most of them are more or less working in the same mechanism; and the DRAMs are the most prevalent ones, so we decided to analyze the DRAM power model as a representative. As for Flash, many current designs are moving towards NAND Flash to take advantage of its higher density and lower cost, so we choose to model NAND Flash instead of NOR Flash.

2.1 DRAM Power Model

According to the working mechanisms of DRAM, its power consists of three key components: background power, activate power, read/write power.

2.1.1 Background Power

As shown in Fig. 2, CKE signal is the master on-off switch of DRAM. When CKE is LOW(CKEL), the DRAM goes into PowerDown state; and when it is HIGH(CKEH), the DRAM goes into Active state. The current changes during these transitions even without read/write access. So we can calculated these background powers easily:

\[
P_{\text{pre-}pdn} = I_{\text{pre-}pdn} \times V_{DD}
\]
\[
P_{\text{act-}pdn} = I_{\text{act-}pdn} \times V_{DD}
\]
\[
P_{\text{pre-}stby} = I_{\text{pre-}stby} \times V_{DD}
\]
\[
P_{\text{act-}stby} = I_{\text{act-}stby} \times V_{DD}
\]

Meanwhile, DRAM have to refresh in the background. We assume that the device is in precharge power-down state most time except when the actual REFRESH commands are executed. Thus, the average extra power for refresh command is:

\[
P_{\text{REF}} = (I_{\text{REF}} - I_{\text{pre-}pdn}) \times V_{DD}
\]

2.1.2 Activate Power

All DRAM banks have to be activated (ACT) first before read/write access. After the ACT command, a large amount of current is used to decode the command/address and transfer data from the DRAM array to sense amplifiers. Once this is complete, the DRAM is maintained in an active state and draws \(I_{\text{act-}pre}\) until a PRE command is issued. The PRE command restores the data from the sense amplifiers into the memory array and resets the bank for the next ACT command. Once this is complete, the device is returned to precharge state. This cycle is then repeated at \(t_{RC}\) intervals between ACT commands. Notice that CKE is always held HIGH during this period, we have to subtract the background current it draws. Therefore, the activate power is:

\[
P'_{\text{ACT}} = (I_{\text{act-}pre} - I_{\text{act-}stby}) \times V_{DD}
\]

But as DRAM may work in lower clock frequency or bank interleave mode, the interval between two ACT commands is not always \(t_{RC}\). Fortunately, it is easy to scale the ACT current for other modes of operation.

\[
P_{\text{ACT}} = (I_{\text{act-}pre} - I_{\text{act-}stby}) \times \frac{n_{\text{READ}}}{n_{\text{ACT}}} \times V_{DD}
\]

2.1.3 Read/Write Power

During read/write access, assume that the currents are \(I_{\text{read}}\) and \(I_{\text{write}}\) respectively. The read/write powers can be calculated as follows:

\[
P_{\text{READ}} = (I_{\text{READ}} - I_{\text{act-}stby}) \times \frac{n_{\text{READ}}}{n_{\text{ACT}}} \times V_{DD}
\]
\[
P_{\text{WRITE}} = (I_{\text{WRITE}} - I_{\text{act-}stby}) \times \frac{n_{\text{WRITE}}}{n_{\text{ACT}}} \times V_{DD}
\]

But the equations above are not the complete answer. To drive the outputs, additional DQ currents are required.

\[
P_{\text{perDQ}} = V_{OUT} \times I_{OUT}
\]
\[
P_{\text{DQ}} = P_{\text{perDQ}} \times (n_{\text{DQ}} + n_{\text{DQS}}) \times \frac{n_{\text{READ}}}{n_{\text{ACT}}}
\]

2.1.4 Total Power of DRAM

After showing the basic components above, it is now in place to calculate the total power. Considering different usage conditions, the parameters as in Table 1 are needed for
### Table 1. Parameters for DRAM Model

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Comments</th>
<th>Typical Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{pre}$</td>
<td>Percentage of all anks are precharged</td>
<td>40%</td>
</tr>
<tr>
<td>$p_{ckpre}$</td>
<td>Percentage of the PRE time when CKE is LOW</td>
<td>50%</td>
</tr>
<tr>
<td>$p_{ckact}$</td>
<td>Percentage of the ACT time when CKE is LOW</td>
<td>0%</td>
</tr>
<tr>
<td>$t_{ACT}$</td>
<td>The average time between ACT commands</td>
<td>120 ns</td>
</tr>
<tr>
<td>$P_{READ}$</td>
<td>Percentage of CK cycles that output read data</td>
<td>1%</td>
</tr>
<tr>
<td>$P_{WRITE}$</td>
<td>Percentage of CK cycles that input write data</td>
<td>24%</td>
</tr>
</tbody>
</table>

input. The total power of DRAM is:

$$P_{total} = P_{bg} + P_{ACT} + P_{R/W}$$

where:

$$P_{bg} = P_{pre-pdn} \times p_{pre} \times p_{ckpre}$$
$$+ P_{pre-stby} \times p_{pre} \times (1 - p_{ckpre})$$
$$+ P_{act-pdn} \times (1 - p_{pre}) \times p_{ckact}$$
$$+ P_{act-stby} \times (1 - p_{pre}) \times (1 - p_{ckact})$$
$$+ P_{REF}$$

$$P_{R/W} = (P_{READ} \times P_{DQ}) \times p_{READ}$$
$$+ P_{WRITE} \times p_{WRITE}$$

Most of the parameters in the equations can be found in DRAM’s datasheet, others such as read/write percentage can be easily estimated in early design stage.

### 2.2 NAND Flash Power Model

Compared to DRAM model, NAND Flash are simpler. In standby state, the Flash could not be accessed, the current $I_{standby}$ is usually small enough to be ignored. In operation state, as shown in Fig.3, a typical flash operation includes at least three stages: command, address, data. Notice that the command and address stages are almost the same for all operations (READ/PROGRAM/ERASE), except for different command codes, we denote them as cmd+addr stage.

#### 2.2.1 Read Power

The read operation consists of only two major stages, so the average power can be calculated according to their percentages:

$$P_{READ}^* = P_{READ_{cmd+addr}}^* \times p_{READ_{cmd+addr}}^*$$
$$+ P_{READ_{data}}^* \times p_{READ_{data}}^*$$

#### 2.2.2 Program Power

For PROGRAM and ERASE operations, we have to include the stage of status, when some specific bits of status registers are read to confirm the success of operation.

$$P_{PROG}^* = P_{PROG_{cmd+addr}}^* \times p_{PROG_{cmd+addr}}^*$$
$$+ P_{PROG_{data}}^* \times p_{PROG_{data}}^*$$
$$+ P_{PROG_{status}}^* \times p_{PROG_{status}}^*$$

where:

$$P_{PROG_{cmd+addr}}^* = I_{WRITE}^* \times V_{DD}^*$$
$$P_{PROG_{data}}^* = I_{WRITE}^* \times V_{DD}^*$$
$$+ P_{perDQ}^* \times (n_{DQ}^* + n_{DQS}^*)$$
$$P_{PROG_{status}}^* = I_{READ}^* \times V_{DD}^*$$
$$+ P_{READ}^* \times p_{polling}^*$$

Figure 3. Flash Operation Diagram

Here, $V_{DD}^*$ is the maximum voltage which can be found in the datasheet. $I_{WRITE}^*$, $I_{READ}^*$, $V_{OUT}^*$, $I_{OUT}^*$ are also provided by vendor. $p_{READ_{cmd+addr}}^*$ and $p_{READ_{data}}^*$ are the percentage of time consumed in each stage. They could be calculated easily from the timing diagrams, and are almost constant. $n_{DQ}^*$ and $n_{DQS}^*$ are the pin number of outputs and their strobes.
Table 2. Parameters for NAND Flash Model

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Comments</th>
<th>Typical Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p^*_{\text{READ}}$</td>
<td>Percentage of READ operation</td>
<td>1%</td>
</tr>
<tr>
<td>$p^*_{\text{PROG}}$</td>
<td>Percentage of PROGRAM operation</td>
<td>24%</td>
</tr>
<tr>
<td>$p^*_{\text{ERASE}}$</td>
<td>Percentage of ERASE operation</td>
<td>24%</td>
</tr>
</tbody>
</table>

Here all parameters have the same meanings with READ operation, except for $p^*_{\text{polling}}$, $p^*_{\text{polling}}$ is the ratio of data polling during the whole programming progress, it depends on the software implementation. If user uses R/B pins to determine the end of operation, then there would be only one polling operation; otherwise user has to read the status register time and time again to check whether the programming is still in progress. The maximum number of polling is $k = \lfloor t^*_{\text{status}}/t^*_{\text{polling}} \rfloor$, where $t^*_{\text{status}}$ is the total time of status checking and $t^*_{\text{polling}}$ is the time of one single data polling. So,

$$p^*_{\text{polling}} = \frac{t^*_{\text{polling}} \times k}{t^*_{\text{status}}} \quad \text{where} \quad k \in [1, \lfloor t^*_{\text{status}}/t^*_{\text{polling}} \rfloor].$$

2.2.3 Erase Power

ERASE operation is mostly the same as PROGRAM, except that ERASE doesn’t have data stage. So,

$$P^*_{\text{ERASE}} = P^*_{\text{ERASE cmd+addr}} \times P^*_{\text{ERASE cmd+addr}} + P^*_{\text{ERASE status}} \times P^*_{\text{ERASE status}},$$

where

$$P^*_{\text{ERASE cmd+addr}} = I^*_{\text{WRITE}} \times V^*_{DD}$$

$$P^*_{\text{ERASE status}} = I^*_{\text{READ}} \times V^*_{DD} + P^*_{\text{READ}} \times p^*_{\text{polling}}$$

2.2.4 Overall NAND Flash Power

Considering the parameters in Table 2 for input, we have:

$$P^*_{\text{total}} = P^*_{\text{READ}} \times P^*_{\text{READ}} + P^*_{\text{PROG}} \times P^*_{\text{PROG}} + P^*_{\text{ERASE}} \times P^*_{\text{ERASE}}$$

Now we can estimate the total power of NAND Flash with only the knowledge of operation percentages. What is more, with these two key component models, we could estimate the power of whole memory subsystem, requiring only the hierarchy and system-level information.

3 Experimental Results

For evaluation, we use Xilinx’s Vertex-II Pro Development Board for DRAM model and Altera’s Cyclone Development Board for NAND Flash model.

Table 3. Measured Power data of DDR DRAM

<table>
<thead>
<tr>
<th>Operations</th>
<th>Assembly Code</th>
<th>Binary Code</th>
<th>Current (A)</th>
<th>Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Bootloop</td>
<td></td>
<td></td>
<td>0.87</td>
<td>4.35</td>
</tr>
<tr>
<td>Write all &quot;0&quot;</td>
<td>ADDIK r3,0,0</td>
<td>B0000000</td>
<td>30600000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IMM 10240</td>
<td>B0002800</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SWI r3,0,0</td>
<td>F8600000</td>
<td>1.21</td>
<td>6.05</td>
</tr>
<tr>
<td></td>
<td>BRI -8</td>
<td>B800FF8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write all &quot;1&quot;</td>
<td>ADDIK r3,0,-1</td>
<td>3060FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IMM 10240</td>
<td>B0002800</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SWI r3,0,0</td>
<td>F8600000</td>
<td>1.16</td>
<td>5.8</td>
</tr>
<tr>
<td></td>
<td>BRI -8</td>
<td>B800FF8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>IMM 10240</td>
<td>B0002800</td>
<td>1.35</td>
<td>6.75</td>
</tr>
<tr>
<td></td>
<td>LWI r3,0,0</td>
<td>E8600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRI -8</td>
<td>B800FF8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Working Voltage of Development Board is 5V

3.1 DRAM Model

We developed benchmarks in assembly within Xilinx EDK, debugged them with XMD and measured the current. First, we downloaded the bootloop program into its embedded CPU, measured the current of board without DRAM module to gain the basic power for all other modules on board. Then we installed the DRAM module and downloaded a benchmark with one single loop, without accessing DRAM, to measure the background power of DRAM. After that, we downloaded benchmarks reading and writing DRAM to gain the related powers. Finally, we got the results in Table 3 and related powers calculated below:

$$P_{bg} = (4.5 - 4.35)/8 \times 18.75mW$$

$$P_{\text{READ}} = (6.75 - 4.5)/8 \times 281.25mW$$

$$P_{\text{WRITE}} = (6.05 + 5.8)/2 - 4.5)/8 \times 178.125mW$$

Meanwhile, we got 12.9mW, 229 mW and 187.6 mW respectively from the DRAM model. Fig. 4 shows the comparison of two data sets, indicating that the accuracy of DRAM model could reach 95%.

3.2 NAND Flash Model

In order to verify the NAND Flash power model, we also developed benchmarks in NIOS II IDE. We used the C APIs provided by Altera’s flash driver to access the flash, then collected the current data in Table 4, and calculated its powers below:

$$P^*_{\text{READ}} = (3.78 - 3.69) \times W = 90mW$$
Meanwhile, we got 142.3 mW, 217.3 mW and 217.3 mW from the NAND Flash model. The measured data is not so good as DRAM model, but the accuracy is still above 80%. After examining the data carefully, we found out that the accuracy is somehow affected by the precision of our measuring instruments and the working voltage of Development Board: the working voltage is 9 V, while the precision of current measuring instruments is 0.01 A under 9 V, which indicates that we can’t get power data between 180 mW and 270 mW. Another reason may be that the software driver in NIOS II IDE is too complicated, it may cause extra powers during execution.

4 Summary and Conclusions

This paper analyzes a mathematical power model of DRAM and presents another one for NAND Flash. These models estimate the total power of memory subsystem, requiring only the percentage of different operations and the parameters provided by memory medium datasheets. Experimental results show that the accuracy of them are 95% and 80%. The accuracy of Flash model was affected by the precision of our measure instruments. Compared against other power estimation techniques and tools, the models are simpler but accurate enough. With these two models, system designers can make an early power estimation easily and accurately without too much effort. Refining the model of NAND Flash by considering more about the microoperations is one of our future work.

ACKNOWLEDGEMENTS

This article was supported by 863 Research and Development Program. And we would like to thank Xilinx Inc. for providing the Development Board and Mr. Wang Hengcai for providing measure devices.

References