임베디드 시스템에서 DFS 기법을 이용한 동적 전력 관리

권기현*, 김남용**, 변형기***

요 약

XScale PXA255 기반 Embedded Linux 환경에서 전력 소비를 줄이기 위해 DFS(Dynamic Frequency Scaling) 기법의 디바이스 드라이버를 제작하고 이 드라이버가 포팅되어 있는 임베디드 타겟보드의 전력을 관리하기 위한 미들웨어 DFM(Dynamic Frequency Management)를 설계하고 구현하여 임베디드 시스템의 전력 소비를 감소하는 방법을 제시한다.

Dynamic Power Management using Dynamic Frequency Scaling in Embedded System

Ki Hyeon Kwon*, Namyong Kim**, Hyung-Gi Byun***

Abstract

In order to decrease the power consumption in Embedded Linux environment based on XScale PXA255, We produce the device driver of DFS(Dynamic Frequency Scaling) technique, design and implement the middleware DFM(Dynamic Frequency Management) to scale the power of embedded target board with porting this device driver, suggest the method to reduce the Embedded system’s power consumption.

Keywords : Dynamic Frequency Management, Dynamic Frequency Scaling

1. Introduction

Embedded system is built in different devices from cellular phone, PDA, digital TV to cutting-edge weapons, and a key technology deciding the value added of production, thus it is a paining as a new industry field in which the growth possibility is very high in post pc times, but at the same time the problem for power consumption of processor is on the rise[1].

As a study to reduce the power usage of embedded system, low power WPEG decoder design[2], voltage scheduling in the low power task using the analysis of static time[3], such several researches to decrease the power consumption in embedded system are fulfilled[4].

Due to the electric specific of processor, the more its performance is enhanced, the more spending of power is increased. However, the consumption quantity of processor is variable according to the achieved works; the maximum quantity is not always needed. Thus by controlling the voltage and operating frequency supplied to the processor according to the amount used of processor, the power consumption may be reduced. This method is named as DVS(Dynamic Voltage Scaling).

But the embedded system is operated on the basis of embedded processor to treat a specific purpose, and in order to meet this, it is necessary to plan the program not only to maximize the efficiency of target application but to decrease the power consumption. Thus this study suggests the program to induce the lower power
r with maintaining the performance of system by applying DFS(Dynamic Frequency Scaling) technique on the basis of benchmark for through of embedded system according to variable applications.

2. Related Study

2.1 DVS(Dynamic Voltage Scaling)
Dynamic voltage scaling technique is a method to reduce the mount used of energy by controlling the supply voltage of processor composed of CMOS circuit. In general because the power(P) consumption of CMOS circuit has a relation of square to the supply voltage, the reduction of supply voltage is very efficient method to decrease the consumption of energy. However, when the supply voltage is decreased, the maximum operating frequency of circuit also slows, thus the throughput of processor is reduced. The technique establishing the speed of clock and voltage in the non-damage limit of service quality demanded for given conduct by using the relations of energy consumption and the processing speed of such processor is the dynamic voltage scaling technique.[5][6][7].

2.2 DPM(Dynamic Power Management)
Dynamic power management is a technique to reduce the consumption of power by transferring to pertinent power state according to the demands of given performance among the separate power states supplied with applying device for inactive condition on the basis of use pattern by composition element of system. According to the specifics of several devices organizing system, detailed implementations are varied but basically the exact estimation for the use pattern in which applying device is not used is necessary and when the transition to the state of low power is determined, the time and energy incidental expenses by state transition should be considered.

Next is power induction formula.
\[ P = V \cdot I \] (1)
\[ I = \frac{Q}{s} \text{ [Coulomb/second]} = Q \cdot f \left( : f = \frac{1}{s} \right) \] (2)
\[ Q = C \cdot V \text{ [Farad \cdot Voltage]} \] (3)
\[ : P = V \cdot I = V \cdot \frac{Q}{s} = V \cdot Q \cdot f = V \cdot C \cdot V \cdot f \] (4)
\[ = V^2 \cdot C \cdot f \text{ [V}^2\text{F/s]} \]

2.3 DVFS (Dynamic Voltage Frequency Scaling)
The most efficient low power method is DVFS. This is a technique to reduce the power consumption by converting the voltage and frequency of IP(Intellectual Property) including variable operating frequencies in the system according to the regular task schedule, and these system structures need the sync signal in the chip and chip communication and due to the IP cores including different operating frequencies, it is consisted of various IO interfaces, thus PPLL is needed to occur multi clock for the communication among them.

2.4 DFS (Dynamic Frequency Scaling) of PXA255
Next is dynamic frequency scaling of PXA255. DFS is a technique to change the operating frequency variably according to the processing to be treated.

2.4.1 PXA255 Clock Operator
When the clock of PXA255 is inputted, 3.6864MHz crystal is connected to the pin of PXTAL and PEXTAL, and 32.768KHz crystal is connected to the pin of TXTAL and TEXTAL. For this, oscillators and PLLs are assembled in the inside of PXA255, so any exterior device
s are not demanded except for the crystal to occur the clock.

2.4.2 Core PLL (Phase Locked Loop)

Core PLL supply the clock by using the CPU core, memory controller, LCD controller, DMA controller. These are amplified according to the following variables: Table 1.

<table>
<thead>
<tr>
<th>L</th>
<th>Crystal Frequency to Memory Frequency Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Memory frequency is determined as [memory clock frequency=crystal frequency(3.6864MHz) × L], the value of L may be 27, 32, 36, 40, 45.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th>Memory Frequency to Run Mode Frequency Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RUN mode frequency is determined as [RUN mode frequency = memory frequency × M] the value of M may be 1, 2 or 4.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>Run Mode Frequency to Turbo Mode Frequency Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TURBO mode frequency is determined as [TURBO mode frequency = RUN mode frequency × N], the value of N may be 1.0, 1.5, 2.0, 3.0.</td>
</tr>
</tbody>
</table>

2.5 Clock Configuration-Related Registers.

2.5.1 Core Clock Configuration Register (CCCR)

Thorough the CCCR register, memory frequency multiple(L) related with Core Clock frequency, RUN mode frequency multiple(M), TURBO mode frequency multiple(N) are configured.

2.5.2 CCLKCFG Register of Coprocessor 14

2.5.1 Core Clock Configuration Register (CCCR) is used for beginning of the TURBO mode and frequency variation sequence. When it begins as frequency variation sequence, in order to keep constant the Turbo Bit, Read-Modify-Write process, that is, the stage of reading this register then modifying only applying bit and writing it again should be employed.

2.6 Frequency Variation Process

The frequency variation process is used when clock frequency of Core PLL is converted and in this process, CPU, memory controller, LCD controller and DRAM clocks related with Core PLL are stopped and modules except for them are operated. This process is used to change the frequency configured as default at an early stage of booting, or may be employed to operate with lower frequency to spare the power by processor.

3. DFM Middleware Design

3.1 System Structure Design

Among some elements of this system, monitoring system regularly checks the amount of power and transmits a needed frequency scaling parameter value to DFM sever and makes it possible to scale the power of target board (Fig. 1).

(Fig. 1) System Structure
3.2 State Transition Diagram

When monitoring system asks DFM server to send a frequency of specific target board, the DRM server including all data of target system gets to ask the target board to send frequency data. Reading the current CCCR of it the target board transmits the data to DFM server, after receiving this, the DFM server transmits frequency information to the monitoring system(Fig. 2).

(Fig. 2) The Protocol Flow of Frequency Information Request

In adverse, during the frequency setting, the number of target system, frequency values are transmitted in the monitoring system. After receiving this, DFM server sends the frequency and device number to the target system and adjusts the setting. Following figure is protocol flow during the frequency setting of target board(Fig. 3).

(Fig. 3) The Protocol Flow of Frequency Setting

3.3 DFS Device Driver Implementation

The module for fulfillment of DFS is implemented in the device driver. The sort of device driver suited to implement DFS is character device driver, which is located in kernel level and scales the frequency by applying the data transmitted from DFM server to PXA255 processor through the device driver(Fig. 4).

(Fig. 4) Data Flow of Target System

Next the source is the part of Device Driver for frequency scaling and control the CCCR register by ioctl function(Fig. 5).

The number 2, 3 lines configure CCCR register and change the CPU frequency. The number 4, 5 lines change the coprocessor register to change the operation speed. The number 6, 7, 8 verify the existence of turbo mode. The number 9, 10 lines activate the turbo mode if the turbo mode exists.

(Fig. 5) The Part of the Device Driver
4. Implementation & Evaluation

This paper explains the model of actual system and investigates the results for test. And next is the implementation and evaluation of the system.

4.1 Test Environment for DFM

Laptop, PC which we usually use become the monitoring system and window, windows CE are porting as operating system. DFM server is implemented in general PC and the operating system is Linux, and the target system equips the embedded Linux.

<Table 3> DFM server Specification

<table>
<thead>
<tr>
<th>MODEL</th>
<th>Samsung Magic Station m2761</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Linux 2.2.17</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel Pentium III 800MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>448 MB</td>
</tr>
</tbody>
</table>

<Table 4> Embedded System Specification

<table>
<thead>
<tr>
<th>MODEL</th>
<th>X-Hyper255B</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Embedded Linux 2.4.18</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel PXA255 400MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>32MB Flash ROM, 64MB SDRAM</td>
</tr>
</tbody>
</table>

The specification of PDA, DFM server and embedded System is in Table 3 and Table 4 separately.

4.2 Performance Evaluation

The file used in test is the video supplied basically by embedded device-manufacture enterprise its spec is Table 5.

<Table 5> Testing File Specification

<table>
<thead>
<tr>
<th>File name</th>
<th>holeman_cf.mpg</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Size</td>
<td>3.63MB</td>
</tr>
<tr>
<td>Average Data Rate</td>
<td>170.19KB per second</td>
</tr>
<tr>
<td>Image Size</td>
<td>352 x 240</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>29.97 fps</td>
</tr>
</tbody>
</table>

The frequencies for each modes are as follows: mode1:99.5MHz, mode2: 199.1MHz, mode3: 298.6MHz, mode4: 132.7MHz, mode5: 199.1MHz, mode6: 298.6MHz, mode7: 398.1MHz, mode8: 265.4MHz, mode9: 331.8MHz, and mode10: 398.1MHz, 400MHz. The Frame rates when the files are running for each CPU frequency scaling are as (fig.6).

(Fig. 6) Frame Rates for Frequency Scaling

From the [fig. 6], it is noted that the frame rate of video file implemented for clock frequency, 99.5MHz in mode 1 is lower than the frame rate of the frequency, 398.1MHz in mode 10. It shows that mode 4, 8 and 10 has lower frame rate compared with mode 3, 7, 9 respectively.

It is found that in mode 1 the video file processing time needs 152 seconds but in model 0, the processing time needs only 49 seconds (Fig. 7). It shows that mode 4, 8 and 10 has lower time spent compared with mode 3, 7, 9 respectively.
In general the power consumption may be significantly fluctuated according to the component installed in the embedded device, the service of the device, and application softwares used. Suggested low power system mode 1 is a software based on middleware, and the reducti on plan was suggested to change the frequency according to the processor spec then decrease the power consumption by controlling the microprocessor, and on the basis of this, not only in middleware field but also in operating system field, the reduction of power consumption should be studied.

5. Conclusion

This paper studied the design and implementation of dynamic power scaling middleware using the DPS technique for reducing the power consumption, with focusing on microprocessor.
권 기현

1995년: 강원대학교 대학원 컴퓨터 과학과 졸업(이학석사)
2000년: 강원대학교 대학원 컴퓨터 과학과 졸업(이학박사)

1996년~2002년: 동원대학교 인터넷정보과 교수
2002년~현 재: 강원대학교 공학대학 전자정보통신공학부 교수
관심분야: 마디웨어, 임베디드소프트웨어, 무선센서 네트워크, U-Healthcare

김 남용

1988년 연세대학교 전자공학과 (공학석사)
1991년 연세대학교 전자공학과 (공학박사)

1992년~1998년: 관동대학교 전자통신공학과 부교수
1998년~현 재: 강원대학교 공학대학 전자정보통신공학부 정보통신공학전공 교수
관심분야: Adaptive equalization, RBFN algorithm, odor sensing systems.

변 형기

1990년: Manchester University 전자기공학 과 MSc.
1995년: Manchester University 계측공학과 Ph.D.

1996년~현 재: 강원대학교 공학대학 전자정보통신공학부 정보통신공학전공 교수
관심분야: 전자 후미각 시스템 및 패턴인식