

Intelligent Energy Manager (IEM) Benchmarking on a Freescale's iMX31 Multimedia Processor

Suji Velupillai & Ken Tough
(Intrinsyc Software)



September 2006

Introduction

Increased functionality and performance demands are challenging System-on-Chip (SoC) designers to seek better methods for optimizing available battery life in portable applications such as Smart-phones, feature phones, media players and other portable devices. Conventional power management techniques have improved battery life in embedded systems through the use of low-power idle and sleep modes; and looking forward these techniques will continue to play a significant role particularly on aggressive processor geometries such as 65nm and 45nm. However, this does not help during active processing time, i.e. when the processor is running an application.

Moreover, as phones are capable of running very different types of applications, requiring different levels of processor performance, the ability to reduce/increase the processor performance to match the current processor workload is very desirable. The hardware technique used today by SoC manufactures to allow varying processor workloads is called Dynamic Voltage and Frequency Scaling (DVFS).

This paper is designed to show the how ARM's Intelligent Energy Manager (IEM)¹ can be used to control DVFS hardware under the Windows CE operating system. IEM was integrated on Freescale's iMX31 Multimedia Processor.

The integration of IEM for Windows CE was carried out by Intrinsyc Software², as part of Intrinsyc's power management offering for Windows CE.

Intrinsyc Software is a mobility software and services company which adapts Windows CE and Windows Mobile for new hardware (Board Support Packages), including developing or adapting DVFS drivers and other power management solutions for new silicon. Intrinsyc can select, integrate, and customize IEM policies for specific systems and applications, integrating with the Windows Advanced Power Management mechanisms for on and off-chip devices.

Scope

Because the iMX31 development platform (ADS) is not an optimized mobile platform, it was more appropriate to limit measurement of power consumption to the CPU core only. For further details of the benefits IEM can bring to a complete mobile or consumer device please contact ARM directly.

¹ http://www.arm.com/products/esd/iem_home.html

² <http://www.intrinsyc.com>

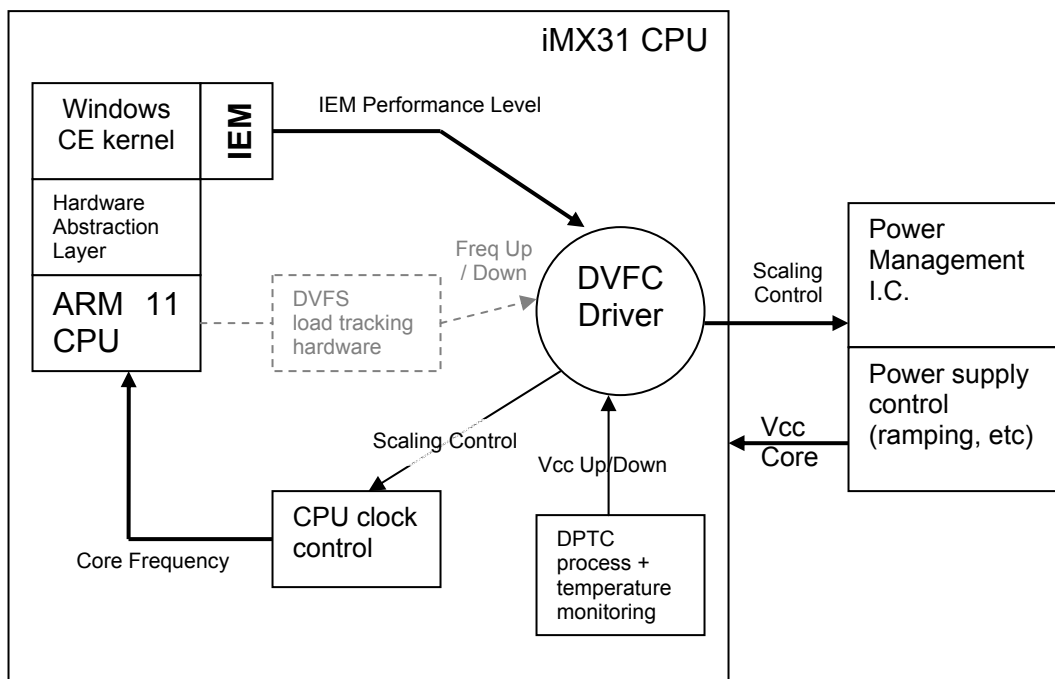
System Utilized For Testing

The hardware used for benchmarking was Freescale's i.MX31³ Application Development System (ADS). The development system included:

- CPU board with i.MX31 ARM-11 MCU
- Power management board with MC13783 Atlas chip
- The supplied software was the RTM9 Board Support Package (BSP).

Intrinsyc Software integrated the IEM code onto Freescale's RTM9 Windows CE BSP. The iMX31 has a combined DVFC ("Dynamic Voltage/Frequency scaling and Process/Temperature Compensation") driver, and this was modified to take performance change requests from IEM, instead of handling the frequency up/down requests ("fsvai") issued by the iMX31 hardware load tracking registers.

To isolate the effects of IEM, the "Dynamic Process and Temperature Monitoring" hardware and its requests for altering Vcc were disabled in the tests. These requests could be combined with performance data from IEM as input to DVFC.



For testing, video playback was used to load the system, with a variety of video frame and bit rates to generate different load points.

³ <http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=01J4Fs29733642>

The final image was compiled together with the debug-enabled media player dll (wmvmod.dll), which outputs diagnostic messages to the console if any video frames are dropped. This confirms that IEM scaling does not unduly impact system performance. (Where applications or drivers detect such degradation, it could be possible to feed back events to IEM, to adapt IEM's performance level.

The iMX31 supports two DVFS performance points:

Performance Point	Voltage (V)	Frequency (MHz)
High	1.60	532
Low	1.35	266

These points are selected since the minimum core voltage (1.35V) can be achieved at core frequency 266 MHz. Since power consumption varies by square of voltage and only linearly with frequency, additional performance points for reduced core frequency were not used. These could easily be added.

Core power consumption was measured through voltage and current readings at measurement points on the ADS CPU (JP8 [Core], JP12 pins 2-3 and JP33 [on-chip peripherals]). A fault in this version of CPU layout means the small resistance across JP8 is unknown, though relative change in measured voltage at this point indicates relative change in core current.

Video Files Information Used For Testing

The table below lists the video clips that are used to load the device during the test.

Filename	Frames per Seconds (fps)	Bandwidth (bytes)
kelsyville_av1400.wmv ⁴	29.5	1408255
kelsyville_av350.wmv	29.5	358187
kelsyville_av32.wmv	13.7	36196

Duration of the video data is two minutes and twenty seconds. During the test the video clip is being played in "Repeat" mode.

DVFS Power Savings Concepts

Without dynamic voltage/frequency scaling, systems normally implement power savings through low power consumption modes while the processor is idle (awaiting interrupt/event). The concept of IEM/DVFS is to scale the core frequency and voltage, thus scaling the processor performance appropriate to the current processor load. With reduced performance the processor will consume less power while spending less of its time "idle". With IEM/DVFS, the processing overhead consumed by switching in and out of idle mode is reduced, as is any power consumed by the core while in idle mode (since the core voltage at this time is now reduced).

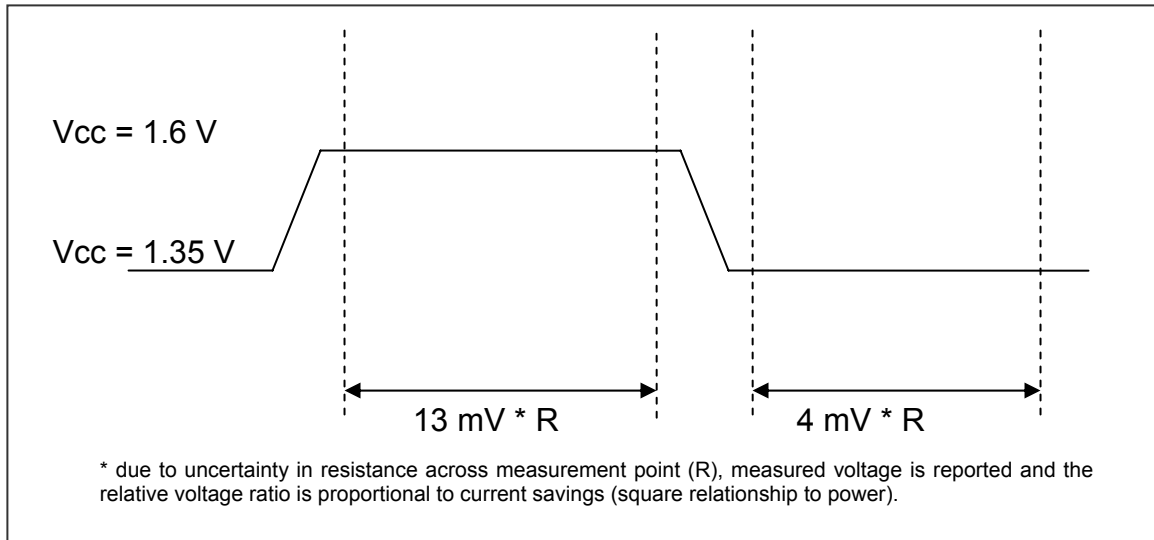
The time when core voltage/freq are scaled low adds to savings, while the decreased time during which the core is idle, reduces savings.

Calculations of power saved must thus take into account both the ratio of core power consumed at high performance scaling versus low performance scaling, as well as the ratio of core power consumed in idle versus non-idle modes.

The key benefit of IEM is that its performance-evaluating "policy" algorithms can be adapted for various system loading patterns (in a variety of use cases). Being software based, it does not require specific hardware implemented in processor silicon, and can be tailored for device/system events per-use case.

⁴ Insert video location if possible

Power Consumption When Scaled High / Low



The figure above depicts the core voltage switching when processor loading is high and when the processor loading is low. The current (voltage) measurement is averaged over the time as shown by the dash lines. Thus, the power consumed at both loads can be calculated as follows:

- Power when scaled high = $(13 \text{ mV} * R) (1.6 \text{ V})$
- Power when scaled low = $(4 \text{ mV} * R) (1.35 \text{ V})$

$$\text{Scaled Coefficient} = (5.4 \text{ R} / 20.8 \text{ R}) = 0.25$$

This coefficient is later used to calculate the energy saving due to integrated IEM.

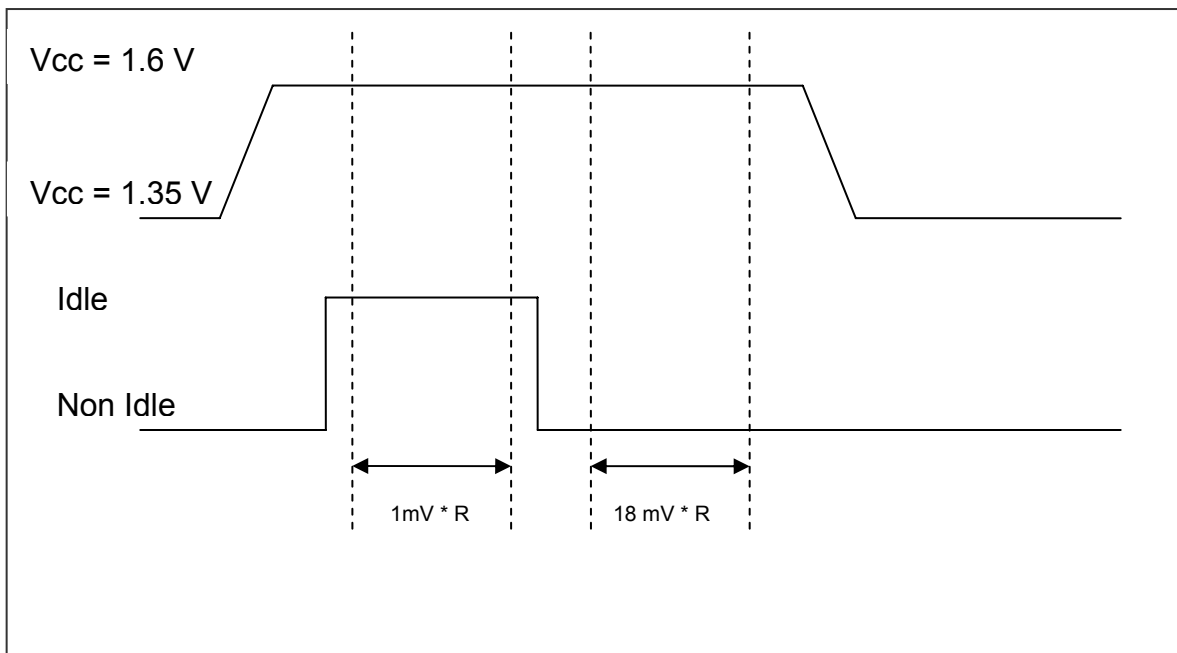
The absolute value of core current measured at high performance (maximum processing, minimum time in idle) is 270 mA.

The time during which the core supply is scaled high, and scaled low, is monitored via a diagnostic thread which determines the supply status from the DVFC driver.

Idle vs Non-Idle Time

A GPIO was used to indicate when the kernel moved the CPU to idle (Wait-For-Interrupt) mode. This period includes the portion of processing used in switching the kernel in and out of idle (overhead).

The figure below depicts the current measurement of the system switching between idle mode and non-idle mode while the core voltage and frequency are scaled high. As expected the current (voltage * R) measurement shows that during non-idle more power is consumed versus during idle. From this we can say that at high core voltage, approx 1/20 of the power is consumed during idle time.



There is some current consumption during system idle, and at high core supply this was measured at an average (again, including idle switching overhead) absolute value of about 7mA.

We assume that the ratio of power consumed while idle vs non-idle will be approximately the same at high core voltage as the idle:non-idle ratio at low core voltage.

Thus, Idle Coefficient = $1 / 18 \approx 0.05$

This coefficient is later used to calculate the energy saving due to integrated IEM. Scaling down the core frequency will result in decreased idle time, offsetting the

savings due to lower core voltage, and so this correction factor must be applied to calculate the energy savings.

The total time during which the processor is idle is measured through software, using the Windows kernel GetIdleTime() functionality. A thread monitors and averages the idle time a few times per second, and the impact of this measurement is assumed to be minimal. The time includes non-productive overhead in the kernel used for switching in and out of idle.

Benchmarking Results

In order to collect the data effectively and accurately as possible, a diagnostic program is executed while loading the system. This program outputs averaged data every five seconds. The data includes the portion of the time that the voltage is low (scaled %) and the portion of the time that the system is idle (idle %). The idle time is calculated using the API call GetIdleTime (), which returns the amount of time (milliseconds) that the system has been idle.

Before collecting data the video clip is played repeatedly until the scaled % is stable. Additionally, to maintain the consistency of data used for the calculation, data is acquired after video clip passed one minute mark. Finally, averaged the data is collected for a duration of one minute.

All the data presented in this document collected using IEM's "Mean" policy. The table below depicts the average of the data obtained from the console.

When DMA is enabled:

Filename	IEM Enabled		IEM Disabled	
	Idle (%)	Scaled (%)	Idle (%)	Scaled (%)
kelsyville_av1400.wmv	25	19	30	0
kelsyville_av350.wmv	28	53	45	0
kelsyville_av32.wmv	67	98	81	0

When DMA is disabled:

Filename	IEM Enabled		IEM Disabled	
	Idle (%)	Scaled (%)	Idle (%)	Scaled (%)
kelsyville_av1400.wmv	21	20	23	0
kelsyville_av350.wmv	28	49	43	0
kelsyville_av32.wmv	66	80	80	0

NOTE: During the test, executing IEM did **not** cause the media player to drop any frames. This clearly shows that IEM does not degrade the performance of the system.

Comments On The Mean Policy

The mean policy is a simple averaging algorithm provided by ARM. The policy was used as this was the only policy available to Intrinsic Software at the time of the evaluation.

The mean policy looks to average the workload over a period of time, using a fixed window period to estimate the workload. The policy compares the work/idle time of the CPU over a fixed period. It does not vary the window size and does not look at full scale performance versus relative performance.

For the single use case applications, i.e. only one application running at time, the mean policy provides good energy savings. However, for a real system such as a mobile phone with many tasks scheduled in the OS, more complex algorithms are required. ARM has a range of policies that use dynamic varying window size, understand relative workload compared to full scale workload, and are able to take into account task deadlines.

Why Enable/Disable The DMA?

At the early stage of IEM integration and testing, we noticed that both IEM and DMA locked the system. Thus, we disabled the DMA and continued the IEM integrating and testing. After completing the IEM integration, DMA was enabled and tested. Extended hours of tests showed that this problem can not be reproduced easily, it only occurred twice so far. With DMA enabled the system passed over forty eight hours of high load (kelsyville_av1400.wmv) test. Now, all the tests are being conducted with DMA enabled and there have been no system locking.

We presented both data because there are some noticeable differences in the power savings, see next section. This is mainly because media player uses DMA.

Power Saved With Different Processor Loading

The energy when the CPU is running and system supply (voltage and frequency) are scaled high is E_{Max} .

The proportion of the time that the CPU is idle is $Idle\%$ and the proportion of time that the CPU is working is:

$$Work\% = 100 - Idle\%$$

Due to less energy being consumed while the system is idle, the energy for a period of time when the system supply is always scaled high (i.e. without IEM) is:

$$E_{non-IEM} = E_{Max} \times Idle_{non-IEM}$$

Where

$$\begin{aligned} Idle_{non-IEM} &= Work\%_{non-IEM} / 100 + (E \times IdleCoef) \times Idle\%_{non-IEM} / 100 \\ &= (1 - Idle\%_{non-IEM} / 100) + (IdleCoef \times Idle\%_{non-IEM} / 100) \end{aligned}$$

And the energy for a period of time when the system scaled high and scaled low (i.e. with IEM) is:

$$E_{IEM} = E_{Max} \times Scaled_{IEM} \times Idle_{IEM}$$

where

$$Idle_{IEM} = (1 - Idle\%_{IEM} / 100) + (IdleCoef \times Idle\%_{IEM} / 100)$$

$$Scaled_{IEM} = 1 - Scaled\%_{IEM} / 100 + ScaledCoef \times Scaled\%_{IEM} / 100$$

The energy saved using IEM controlled DVFS is

$$E\%_{Saved} = \frac{(E_{non-IEM} - E_{IEM})}{E_{non-IEM}} \times 100 = \left(1 - \frac{E_{IEM}}{E_{non-IEM}}\right) \times 100$$

$$E\%_{Saved} = \left(1 - \frac{Scaled_{IEM} \times Idle_{IEM}}{Idle_{non-IEM}}\right) \times 100$$

The $IdleCoef$ and $ScaledCoef$ are as calculated using the measurements in the previous sections.

Results

When DMA is disabled:

Filename	IEM Enabled		IEM Disabled		Power Saved %
	Idle (%)	Scaled (%)	Idle (%)	Scaled (%)	
kelsyville_av1400.wmv	21	20	23	0	13
kelsyville_av350.wmv	28	49	43	0	22
kelsyville_av32.wmv	66	80	80	0	38

When DMA is enabled:

Filename	IEM Enabled		IEM Disabled		Power Saved %
	Idle (%)	Scaled (%)	Idle (%)	Scaled (%)	
kelsyville_av1400.wmv	25	19	30	0	9
kelsyville_av350.wmv	28	53	45	0	23
kelsyville_av32.wmv	67	98	81	0	58

Sample Raw Data For DMA Enabled Test

Kelsyville_av1400.wmv				
Duration	IEM Enabled		IEM Disabled	
	Idle %	Scaled %	Idle %	Scaled %
0:05	26	19	29	0
0:10	26	20	28	0
0:15	26	19	30	0
0:20	25	19	32	0
0:25	26	19	31	0
0:30	24	19	31	0
0:35	24	19	27	0
0:40	26	19	29	0
0:45	26	19	29	0
0:50	24	20	33	0
0:55	25	19	32	0
1:00	23	22	32	0
Average	25	19	30	0

Kelsyville_av350.wmv

Duration	IEM Enabled		IEM Disabled	
	Idle %	Scaled %	Idle %	Scaled %
0:05	29	52	46	0
0:10	25	53	46	0
0:15	29	53	46	0
0:20	25	53	46	0
0:25	29	54	45	0
0:30	29	53	44	0
0:35	29	52	44	0
0:40	25	53	45	0
0:45	27	53	45	0
0:50	29	53	44	0
0:55	26	53	43	0
1:00	28	52	46	0
Average	28	53	45	0

Kelsyville_av32.wmv

Duration	IEM Enabled		IEM Disabled	
	Idle %	Scaled %	Idle %	Scaled %
0:05	65	98	82	0
0:10	65	98	80	0
0:15	66	98	80	0
0:20	66	98	81	0
0:25	72	98	79	0
0:30	68	98	79	0
0:35	67	98	80	0
0:40	68	98	83	0
0:45	66	98	80	0
0:50	66	99	83	0
0:55	65	99	79	0
1:00	65	99	80	0
Average	67	98	81	0

Conclusions

The relatively simple Mean policy used in the tests show IEM provides a power saving of between 9% and 58% over the condition where no dynamic frequency/voltage scaling is enabled on the iMX31. Power saving is greatest under conditions of low load, and under heavily loaded conditions a saving of 10% to 25% would be expected. Since typical system use-cases for mobile devices have a great deal of time in near-idle condition, the saving expected in actual operating conditions would be much greater.

Savings would be increased using more advanced IEM policies tailored to system, and by combining iMX31 DPTC (process/temperature control) with IEM software-monitored performance level, to control the frequency/voltage performance point.