Research Journal of Applied Sciences, Engineering and Technology 2(2): 102-106, 2010
ISSN: 2040-7467

Submitted Date: April 29, 2009 Accepted Date: January 10, 2010 Published Date: March 10, 2010

DVS 926 CPU for Mobile Handheld Devices

[1]A. Rajalingam, [1]B. Kokila and [2]S.K. Srivatsa
[1]Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering College, Chennai, India
[2]Anna University, Chitlapakkam, Chennai, India

Abstract: The DVS 926 CPU provides an integrated solution for addressing power management of SoC devices. The SoC includes the main components of mobile handheld devices along with multiple voltage domains. This new methodology is used to EDA Tools for using soft-IP in multiple frequency- and voltage-domain designs. This is the standard interface to on- and off-chip power supplies and controls the processor's performance levels. The SoC software algorithms are derived the minimum sufficient performance for running dynamic workloads.

Key words: DVS 926 (Dynamic Voltage Scaling), EDA tools, low power, mobile devices, SoC

INTRODUCTION

Low power consumption is arguably the most important feature of embedded processors with significant impact on the cost and physical size of the end device. Even though the processor may not be the most power-hungry component of a system, it is essential to manage processor power in order to reduce overall system power consumption. Better processor power efficiency can increase the available power budget for features such as color screens and backlights, which are growing in popularity on portable devices.

Historically, low power consumption in embedded processors has been achieved through simple designs, limited use of speculation, and employing a number of low-power sleep modes that reduce idle-mode power consumption. Embedded processors are now performing more sophisticated tasks, which require ever-higher performance levels. As a result, new processor designs are more dependent on sophisticated architectural techniques (such as prediction and speculation) to achieve high performance. Unfortunately, such techniques can also significantly increase the processor's power consumption.

Process technology trends are also complicating the power story. Until recently, CMOS transistors consumed negligible amounts of power under static conditions. However, as process geometries shrink to provide increasing speed and density, their static (leakage) power consumption has also increased. Current estimates suggest that static power accounts for about 15-20% of the total power on chips implemented in 0.13 μm high-speed processes. Moreover, as process technology moves below 0.1μm, static power consumption is set to increase exponentially, and will soon dominate the total power consumed by the processor.

Figure 1 shows projections for leakage power increase in future process technologies. There is a strong correlation between the operating temperature and the amount of leakage power. However, regardless of the temperature, all lines exhibit exponential trends. In embedded processors, where the majority of transistors are usually dedicated to memory structures (such as caches), leakage is a particularly important problem to attack, since the static power consumption of these structures can dominant overall power consumption (Shin and Kim, 2001).

MATERIALS AND METHODS

Power saving opportunities: A way to bridge the gap between high performance and low power is to allow the processor to run at different performance levels depending on the current workload. An MPEG video player, for example, requires about an order of magnitude higher performance than an MP3 audio player. Even greater savings can be achieved by reducing the processor's supply voltage as the clock frequency is reduced. Dynamic Voltage Scaling (DVS) exploits the fact that the peak frequency of a processor implemented in CMOS is proportional to the supply voltage, while the amount of dynamic energy required for a given workload is proportional to the square of the processor's supply voltage (Weiser *et al.*, 1994). Reducing the supply voltage while slowing the processor's clock frequency yields a quadratic reduction in energy consumption, at the cost of increased run time.

Often, the processor is running too fast. For example, it is pointless from a quality-of-service perspective to decode the 30 frames of a video in half a second, when

Corresponding Author: A. Rajalingam, Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College, Chennai, India

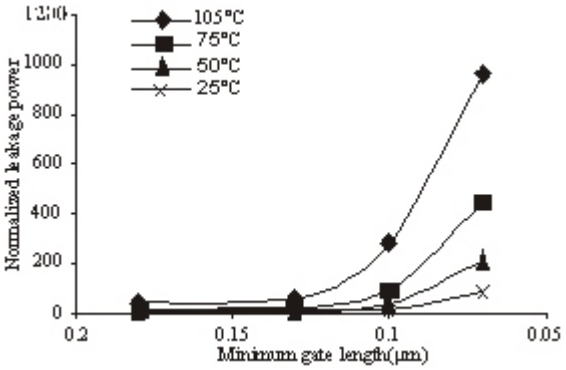

Fig. 1: Normalized leakage power through an inverter - The circuit simulation parameters including threshold voltage were obtained from the Berkeley Predictive Spice Models (Anonymous, 2009). The leakage power numbers were obtained by HSPICE simulations

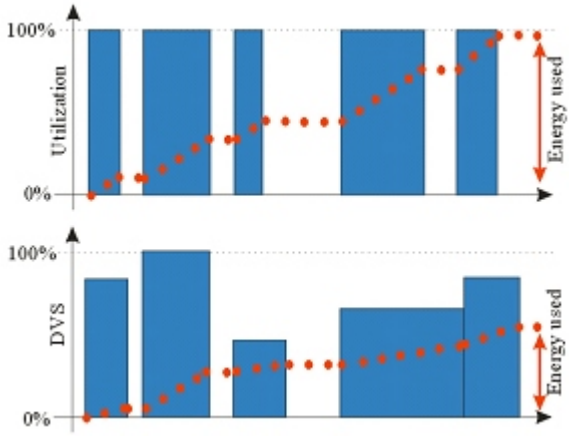

Fig. 2: Energy consumption using traditional power management vs. dynamic voltage scaling

the software is only required to display those frames during a one second interval. Completing a task before its deadline is an inefficient use of energy (Rajalingam and Srivatsa, 2008). The key to taking advantage of this trade-off is the use of performance-setting algorithms that aim to reduce the processor's performance level (clock frequency) only when it is not critical to meet the application's deadlines. Figure 2 shows the significantly lower total energy consumption that is achievable using dynamic voltage scaling compared with traditional gated-clock power management, for the same workload (Pering *et al.*, 1998). Note that with DVS, the lower supply voltage reduces static power even when the clock is gated off.

Static leakage power can also be substantially reduced if the processor does not always have to operate at its peak performance level. One technique for accomplishing this is adaptive reverse body biasing (ABB). Combined with dynamic voltage scaling, this can yield substantial reductions in both leakage and dynamic power consumption (Mosse *et al.*, 2004). The key enabler for controlling both DVS and ABB is knowledge about how fast a given workload needs to run. This information can be provided by performance-setting algorithms that take various operating system and optional application-specific information into account to provide an estimate for the necessary performance level of the processor (Rajalingam and Kumar, 2009).

While DVS and ABB are effective ways of managing the processor's power consumption, integrating these ideas into SoC designs has proven to be a significant challenge. The key issue is that not all parts of the SoC can be scaled in equal measure. Consequently, multiple voltage and frequency domains with asynchronous interfaces are required (Lee and Krishna, 1998). Moreover, these extra parameters complicate testing and validation processes and requires special support from synthesis tools.

DVS 926: Completing a task before its deadline, and then idling, is significantly less energy efficient than running the task more slowly so that the deadline is met exactly. The goal is to reduce the performance level of the processor without allowing applications to miss their deadlines. The central issue is how the right level of performance can be predicted for the application.

DVS 926 provides a hardware and software mechanism for achieving these goals: it standardizes the interface for setting the processor's performance level, specifies counters for measuring the amount of work that is being accomplished, and includes operating system and application-level algorithms for predicting future behavior.

The DVS 926 software layer has the ability to combine the results of multiple algorithms and arrive at a single global decision. The policy stack illustrated in Fig. 4 supports multiple independent performance-setting policies in a unified manner (Ishihara and Yasuura, 1998). The primary reason for having multiple policies is to allow the specialization of performance-setting algorithms to specific situations, instead of having to make a single algorithm perform well under all conditions. The policy stack keeps track of commands and performance-level requests from each policy and uses this information to combine them into a single global performance-level decision when needed.

The different policies are not aware of their positions in the hierarchy and can base their performance decisions on any event in the system. When a policy requests a performance level, it submits a command along with its desired performance to the policy stack (Chung, 2002; Martin, 2002). The command specifies how the requested performance should be combined with requests from lower levels on the stack: it can specify to ignore (IGNORE) the request at the current level, to force (SET) a performance level without regard to any requests from below, or set a performance level only if the request is greater than anything below (SET_IFGT). When a new

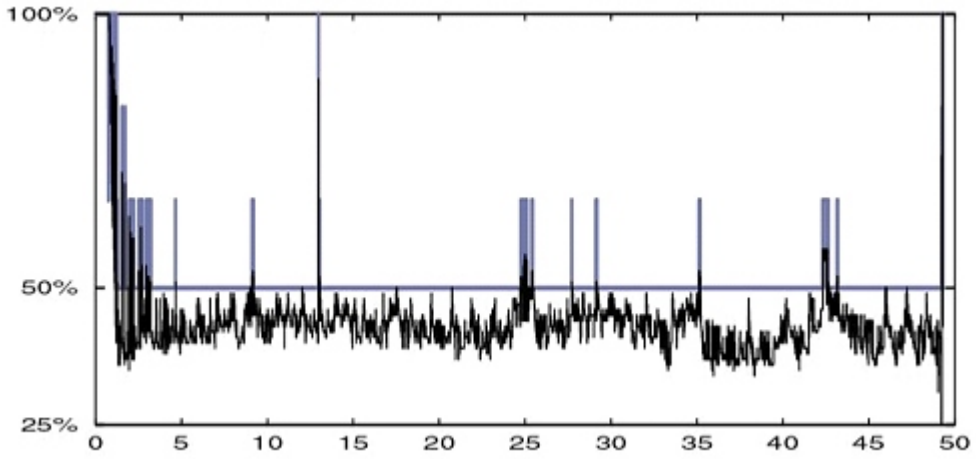

Fig. 3: Performance policy stack

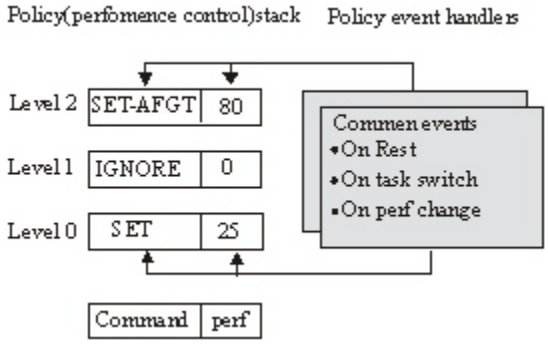

Fig. 4: Performance-setting during MPEG video playback of Red's Nightmare – The graph shows both the internal state of the performance predictor algorithm as well as the quantization to the four available performance levels on the test platform

performance level request arrives, then the commands on the stack are evaluated bottom-up to compute the new global performance level. In Fig. 3, the evaluation would yield the following: at level 0 the global prediction is set to 25, at level 1 it remains at 25 and level 2 changes the prediction to 80.

Using this system, performance requests can be submitted any time and a new result computed without explicitly having to invoke all the performance-setting policies. While policies can be triggered by any event in the system and they may submit a new performance request at any time, there are sets of common events of interest to all. On these events, instead of re-computing the global performance level each time a policy modifies its request, the performance level is computed only once after all interested policies' event handlers have been invoked. Currently the set of common events are: reset, task switch, task creates, and performance change. The performance change event is a notification, which is sent to each policy and does not usually cause any changes to the performance requests on the stack.

To achieve the most effective energy reduction with minimal intrusion, application monitoring and performance-setting decisions need operating system involvement. Figure 4 shows qualitatively the successive performance-setting decisions during a run of an MPEG video playback benchmark.

The test platform had four available evenly spaced performance levels corresponding to 100, 83, 66 and 50% of peak performance. As can be seen, in some cases the algorithm would have selected a performance level even lower than the minimum of the machine.

SoC design flow issues: Dynamic Voltage Scaling currently has only been commercially exploited in stand-alone CPU integrated circuits. To support voltage scaling of processing sub-systems within a system-on-chip design is a challenge in a number of areas for both the EDA tools and design methodology:

- Multiple physical power domains
- Synchronous clock relationships across boundaries
- Standard-cell library and RAM compiler design views
- Static timing verification
- Manufacturing test

Multiple power domains require careful handling at interfaces where some form of analog level shifting is required between different voltages (Chung-Hsing *et al*., 2001; Pering *et al*., 1998; Shin and Kim, 2001). Also many EDA tools treat voltage rails as special global resources, which are implicitly connected, which make separating voltage domains a manually intensive design step.

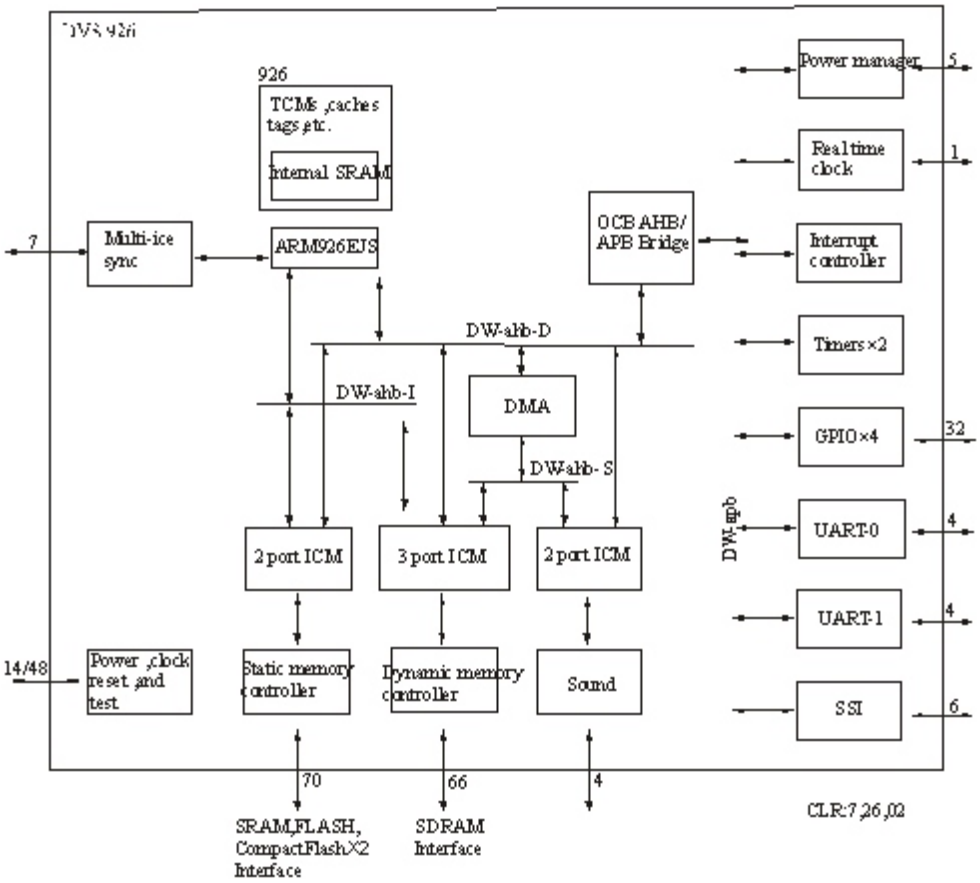

Fig. 5: The main components of the DVS926 test chip, which includes multiple voltage domains for low-power operation.

Best-practice SoC design flows typically assume synchronous clocking relationships between subsystems in order to allow top-level static timing-closure and analysis, automatic test structure insertion and test pattern generation. Ideally, multiple voltage domains should be treated as asynchronous because the tolerance of buffered clocks across the top-level system becomes near-impossible where subsystems can have variable voltage with respect to each other. Difference subsystems inherently have variable clock buffer latencies.

Cell libraries and memory compilers are normally characterized and modeled for a process and temperature across a tightly toleranced (±5 to ±10%) supply voltage range. For voltage scaling design integrity more comprehensive timing models are required. The design tools make this harder because in order to design for multiple performance levels, one has to first specify the target subsystem frequency and then determine, statically or adaptively, the power supply requirements for sufficient voltage to maintain operation. However, from a design-flow perspective, one has to work the other way around: start with a defined voltage and then calculate the achievable performance from the static timing analysis at this precise voltage. Characterizing RAMs at low voltage is complicated by the fact that sense amplifier performance degrades non-linearly with respect to logic gate speeds. Verification of static timing and functional test are complicated with voltage scaling of parts of the system-on-chip design. The EDA tools need to be guided.

RESULTS AND DISCUSSION

Front-end design implications: Front-end design tools typically read in RTL descriptions in Verilog or VHDL of the hardware design and for both simulation and synthesis. Such HDL descriptions have no concept of multiple power rails; a global view of power and ground are assumed. Similarly, clocks and resets are treated as ideal signals that are later buffered as high fan-out carefully balanced buffer-tree networks.

The boundaries between voltage domains must be handled with detailed management of hierarchy and the instantiation of explicit voltage level shifter cells between different voltage rails. The onus is on the designer to carefully abstract out the top-level management of clocks, resets, and test scan chains and power management such that individual subsystems can be synthesized and even hardened independently using standard ASIC design flows.

Back-end design implications: The layout tools need to understand separate voltage rails and this may require manual intervention and careful inspection and review of conversion from the front-end logical design flow to the place and route implementation phase. In the worst-case the cell library may need to be replicated with special cell and power-rail naming schemes to ensure that optimization, setup and hold timing fixes applied to the post-routed top-level design do not accidentally stray over voltage domains or level-shifter boundaries. Design verification needs to be extended beyond standard ASIC design flows to cover the extra complication of analog level shifter integrity especially with respect to power domains that can be powered off completely and which must not draw static currents from driven inputs or need outputs clamped during power down and power up (i.e. operating outside valid logic state operation). The resulting DVS926 test chip is shown in Fig. 5.

CONCLUSION

An ARM926EJ-S based design with independent voltage scaling of the cached CPU, which tackles all these design tools has been completed by ARM in collaboration with Synopsys, TSMC, and National Semiconductor. The SoC includes the main components of mobile handheld devices along with multiple voltage domains. The aim of the collaboration has been to refine the design and methodology techniques and to understand their implications on the EDA tools.

REFERENCES

Anonymous, 2009. http://www-device.eecs.berkeley.edu

Chung, E., L. Benini and G. Micheli, 2002. Contents provider-assisted dynamic voltage scaling for low energy multimedia applications. 2002 Int'l Symposium on low power electronics and design, August.

Chung-Hsing, H., K. Ulrich and H. Michael, 2001. Compiler-directed dynamic voltage and frequency scheduling for energy reduction in microprocessors. ISLPED, Aug., pp: 275-278, DOI: 10.1109/LPE. 2001.945416.

Ishihara, T. and H. Yasuura, 1998. Voltage scheduling problem for dynamically variable voltage processors. ISLPED, August., pp: 197-202.

Lee Y.H. and C.M. Krishna, 1998. Voltage clock scaling for low energy consumption in realtime embedded systems. Proceedings of the 6th International Conferance on Real Time Computing Systems and Applications.

Martin, S., K. Flautner, D. Blaauw and T. Mudge, 2002. Combined dynamic voltage scaling and adaptive body biasing for optimal power consumption in microprocessors under dynamic workloads. Proceedings of the International Conference on Computer Aided Design (ICCAD 2002), San Jose, CA, November.

Mosse, D., H. Aydin, B. Childers and R. Melhem, 2000. Compiler-assisted dynamic power-aware scheduling for real-time applications. COLP (Workshop on Compiler and OS for Low Power), Defense advanced research project agency through the PARTS (Power-Aware-Real-Time-System). Philadelphia, 19 October.

Pering, T., T. Burd and R. Brodersen, 1998. The simulation and evaluation of dynamic voltage scaling algorithms. ISLPED, Aug., pp: 76-81.

Rajalingam A. and S.K. Srivatsa, 2008. Processor power consumption using DVS scheduling. National conference on emerging techniques in communication engineering, Feb., pp: 273-277.

Rajalingam, A. and N. Kumar, 2009. Low power multimedia applications using buffers for DVS algorithm. National Conference on Emerging Trends in computing and Information Technology, Mar., pp: 28.

Shin D. and J. Kim, 2001. A profile-based energy-efficient intra-task voltage scheduling algorithm for hard realtime appli-cations. ISLPED.

Weiser, M., B. Welch, A. Demers and S. Shenker, 1994. Scheduling for reduced CPU energy. First Symposium on Operating System Design and Implementation OSDI '94, Nov., pp: 13-23.