Journal of VLSI Signal Processing 39, 15–33, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Power-Aware 3D Computer Graphics Rendering

JEONGSEON EUH, JEEVAN CHITTAMURU* AND WAYNE BURLESON
Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01002, USA

Received March 20, 2003; Revised June 4, 2003; Accepted June 5, 2003

First online version published in September, 2004

Abstract. Real-time 3D Graphics rendering consumes significant power because of its very high computation and memory access rate. Due to variation in workload and perceptual tolerance, power-awareness can optimize this power consumption significantly, thus facilitating migration to future power-constrained devices such as personal digital assistants (PDAs), tablets, wearables, phones etc. This work proposes such a low power system based on Approximate Graphics Rendering (AGR). The AGR system supports various algorithms and incremental changes to the computational mechanism based on certain pre-specified parameters. The knowledge available apriori about the signal and noise models of graphic images and Human Visual Perception (HVP) are used to select the configuration that meets the quality needs at the lowest power consumption.

The power savings using the AGR system are examined for two power hungry stages of the 3D graphics rendering system, namely *shading* and *texture mapping*. Besides supporting various algorithms, two novel parameterizable computation schemes are proposed. First, iterative COordinate Rotation DIgital Computer (CORDIC) algorithm based units are incorporated for certain computations. Second, a scheme for dynamically enhancing the perceived image spatial correlation for reduced computations is presented.

A hardware synthesis and estimation methodology based on realistic graphics content from the well-known 3D graphics benchmarks and the game *Quake2* [1] is used for estimation of power savings. Significant power savings of 75.1%, 73.8% and 72% are demonstrated in the shading, texture mapping function blocks and CORDIC based 3D vector interpolator respectively.

Keywords: 3D Graphics, low-power, reconfigurable, shading, texture mapping

1. Introduction

3D graphics has been receiving great attention recently due to its use in various applications such as movie making [1], 3D games [2], virtual reality modeling [3] and 3D Graphical User Interface (GUI) development [4]. The rendering stage of the 3D graphics engine, one of its most power consuming stages, applies color to each of the pixels of the 3D image generated on the computer screen. The stage requires hundreds of millions of computations and memory accesses per second for real-time rendering, thus consuming significant power. The migration of 3D graphics to power-constrained post-PC systems such as PDAs, tablets, wearables, phones, etc., requires significant reduction in its power consumption. This necessitates the development of novel techniques which are aware of the application characteristics to supplement the power savings possible using the general low power techniques described in [5].

Recently, adaptive platforms supporting power- and energy-aware computations are emerging in multimedia and various signal processing applications [6, 7].

*Present address: Digital Design Engineer, NuLife Semiconductors, Chennai, India. Email: jeevan@nulifetech.com

The awareness is characterized by the ability to scale system resource consumption in accordance with the operating conditions. Typically, power awareness is achieved in these systems by providing scope for incremental refinement in quality with resources expended. Various parameters like filter coefficients, filter and transform lengths, search spaces, word lengths, number of iterations, etc., have been utilized for implementing incremental refinement. This can result in reduced computations and power either by compromising the quality or by expending only the amount of resources needed to satisfy perceptual tolerance (e.g., HVP) [8]. The appropriate configuration is determined based on input signal statistics and system resources.

This work proposes an AGR system that can dynamically choose among a variety of algorithms and parameter values that determine incremental changes to the computations. Two novel features are incorporated to extend scope for parameterized computations: Dynamically altering the spatial correlation of the image by varying the word length of image pixels compared for identifying the correlation; Utilizing CORDIC based vector interpolator to facilitate adaptive precision control with parameterized number of iterations.

The content variation exhibited by 3D graphics due to motion and scene change and the low sensitivity of visual system to moving objects and after-image phenomenon are leveraged to achieve reduced power in an AGR system without noticeable quality degradation. As a graphics image is created artificially, the information regarding motion and scene-change are readily available prior to rendering. The human visual perception characteristics are also thoroughly understood. Hence, power-awareness can be incorporated easily in a graphics system by leveraging the above characteristics. Note that it is comparatively difficult to obtain accurate signal and noise information in video and other signal processing applications.

In this work, two main steps of the rendering stage namely, *shading* and *texture mapping*, are implemented on the AGR system. Power savings is measured using a hardware synthesis and estimation methodology with realistic graphics content from the well-known 3D graphics benchmarks and the game *Quake2*. Power savings of up to 72, 75.1, and 73.8% are obtained from the CORDIC normal vector interpolator, and by using algorithm level adaptations to shading and texture mapping respectively. Exploiting image correlation results in an orthogonal power savings of up to 31% from the texture mapping interpolation unit, without any bearing on content-variation and HVP.

The organization of this paper can be detailed as follows. Section 2 gives the background information regarding the graphics rendering units of interest and the mechanisms for power-aware computations. The theoretical and experimental procedures used for estimating power savings and error associated with the proposed techniques are discussed in Section 3. Section 4 explains the various algorithm level reconfigurations proposed for shading. The CORDIC algorithm based parameterized computations for shading are discussed in Section 5. Adaptive texture mapping using HVP control with support for dynamic voltage scaling is detailed in Section 6. The exploitation of texture mapping data characteristics to achieve parameterized power-quality trade-offs is studied in Section 7. Conclusions and future work are listed in Section 8.

2. Background

This section gives a brief introduction to the two units of 3D graphics rendering system focused on in this work: *shading* and *texture mapping*. Next, the concept behind the two vital mechanisms for power-awareness in 3D graphics, AGR and HVP is discussed. Ultimately the 3D CORDIC algorithm which facilitates parameterized power-quality trade-offs is introduced.

2.1. Shading

Shading belongs to the rasterization stage of 3D graphics. It is the process of performing lighting computations to determine the intensity or color of each pixel [9] within a graphics primitive (typically a triangle). The lighting equation for multiple light sources is:

$$I = k_a I_a + \sum_{i=1}^{n} I_{li}[k_d(\vec{N} \cdot \vec{L}_i) + k_s(\vec{R}_i \cdot \vec{V})^s] \quad (1)$$

where i represents the i_{th} light source, k_a is the ambient-reflection coefficient, k_d is the diffuse-reflection coefficient, k_s is the specular-reflection coefficient, $\vec{N}$ is the unit normal surface vector, $\vec{L}_i$ is the unit vector directed toward the i_{th} light source, $\vec{R}_i$ is the specular reflection unit vector of $\vec{L}_i$, $\vec{V}$ is the unit vector directed towards the viewer, and s is the specular reflection parameter. The first term is the ambient light component, the second term is the diffuse light component, and the last term is the specular light component.

Three types of shading are defined based on the procedure used for computing the overall intensity at various pixels within the primitive.

Flat shading, the simplest and least accurate technique, assumes that each triangle has a uniform color throughout its surface. It performs the intensity calculation only once per triangle, thus leading to conspicuous artifacts in certain situations.

Gouraud shading, a more precise technique computes Eq. (1) at each vertex of a triangle. A Discrete Difference Analyzer(DDA) algorithm is applied to interpolate the vertex intensities to obtain the values of the pixels within and on the edges of the triangle. However, it doesn't compute the specular term, thus failing to depict the shininess of the primitive.

Phong shading, the most comprehensive as well as expensive counterpart, uses the DDA algorithm to compute the normal vector $\vec{N}$ at each pixel within the primitive. The intensity of the pixel is then obtained as per Eq. (1). With the specular term, Phong shading gives a more realistic image for a shiny object than Gouraud shading.

2.2. Texture Mapping

Texture mapping is a process in which a 2D or 3D bit-map image called a texture is applied to an object in the 3D world which is to be mapped to a 2D screen. For example, a brick can be texture mapped to a wall, wood can be mapped to a floor and terrain mapping can be used in flight simulation. Due to the absence of an exact mapping between a texture pixel(texel) and a screen pixel, an interpolation calculation is often required for high quality texture mapping. The interpolation can be performed using a variety of algorithms like point sampling, bilinear interpolation, trilinear interpolation and anisotropic interpolation. The level of detail of an object decreases with an increase in the distance. Hence, the use of a single bit-map image leads to artifacts. To realistically depict varying detail, a set of pre-filtered images of progressively, but discretely decreasing levels of detail called a mip-map is used.

Figure 1 shows how a screen pixel $(x, y)_d$ is assigned a value using various interpolation algorithms in case of mip-mapped textures. When performing point sampling, the value of T_0 is used since it is the texel closest to the reflection of (x, y) in the nearest mip-map level l, i.e. $(x, y)_l$. In bilinear interpolation mode, the pixel value is a weighted average of the texels surrounding $(x, y)_l$. Numerically it can be represented as shown in

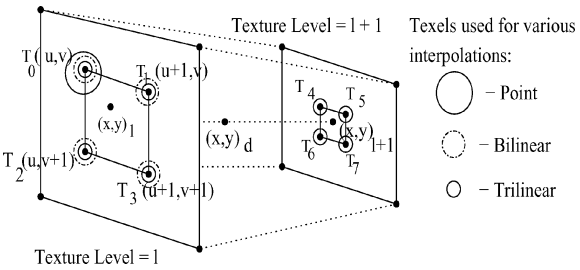

Figure 1. Texels used for various interpolations in case of mip-mapped textures. ($(x, y)_d$-Required texel, d = Required level ($1 < d < 1 + 1$)T_i($i = 0$–7) Actual texels.)

Eq. (2), where $I_{(x,y)}$ is the bilinearly interpolated texture color at pixel position (x, y), T_i and W_i are the color value, and the weight of a texel respectively, at the i_{th} neighboring position of (x, y).

$$I_{(x,y)} = \sum_{i=0}^{3} T_i \times W_i \tag{2}$$

When trilinear interpolation is used, the value of $(x, y)_d$ is the weighted average of the intensities corresponding to each of the 4 texels in the adjacent levels l and $l + 1$, respectively, as shown in Eq. (3). Here W_l and W_{l+1} are the weights corresponding to levels l and $l + 1$, respectively.

$$I_{(x,y)} = W_l I_{(x,y)_l} + W_{l+1} I_{(x,y)_{l+1}} \tag{3}$$

In this work, a conventional texture mapping scheme using mip-mapped textures is assumed. Trilinear interpolation is performed when the screen pixel (x, y) maps to a level d in between two successive mip-map levels l and $l + 1$ and bilinear interpolation is applied when it falls outside the largest or smallest levels of detail texture. Sections 6 and 7 propose techniques for obtaining significant power savings from the texture mapping system.

2.3. Approximate 3D Graphics Rendering

The AGR methodology aims to achieve energy-awareness by utilizing the *Approximate Signal Processing*(ASP) technique [10, 11] postulated for signal processing applications. The ASP concept advocates that the system provide a trade-off between the quality of processing results and the cost of data processing, such as energy consumption (E). For a system with time varying levels of energy, the ASP system facilitates incremental refinement in quality with the amount

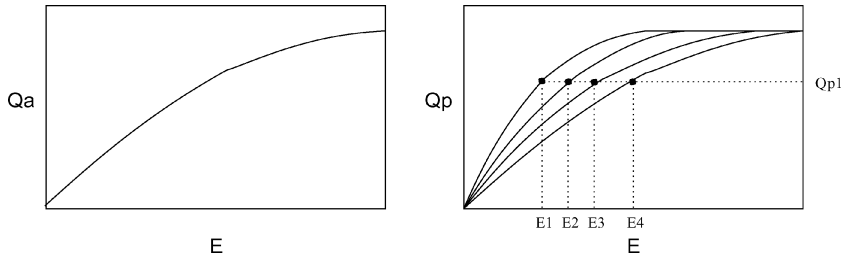

Figure 2. E-Q and E-Qp relationships.

of energy expended. Thus the ASP system provides a constant absolute quality (Qa) directly proportional to the energy supplied (E).

In case of multimedia systems such as video and graphics, the human visual system establishes an upper bound on the quality perceived, namely perceptual quality (Qp). Even for a given absolute quality Qa, as obtained by expending an energy E, the quality Qp varies along with the content to be perceived by the human visual system. Thus the E-Qp relationship is many-to-one in contrast to the one-to-one relationship existing between E-Qa, as shown in Fig. 2. The AGR system supports multiple qualities of diverse energy requirements and exploits the E-Qp relationship to utilize the energy efficiently. The image information readily available at the high level steps of graphics processing can be used to determine the configuration that provides the quality Qp based on the HVP model.

2.4. Human Visual System

Human Visual System is wonderful in that it can identify small differences in intensities and depths even from a great distance, and adjust to light differences across orders of magnitude. However it exhibits some limitations in that it compresses the received light signals and evinces non-uniformity in visual detail that it can perceive across various regions of the eye. These limitations can be exploited to produce images without any perceivable artifacts at less computational expense [12].

The amount of detail that the eye can recognize is characterized by a term called *spatial frequency*, measured in cycles/degree (*c/deg*). The amount of stimulus's intensity with respect to its surroundings is defined as *contrast*. A plot indicating the contrast at various spatial frequencies and velocities of the objects is called Spatio-temporal Contrast Sensitivity Function (STCSF) [13, 14], which can be represented as shown in Fig. 3. As it can be observed, the visual sensitivity for an object varies with the spatial frequency and the velocity in *degrees per second* (dps). Further the human visual system retains the image it sees, for a certain duration, resulting in after-image phenomenon.

HVP has been leveraged extensively in video and graphics. In the movie industry, the after image phenomenon has been leveraged to maintain a low frame

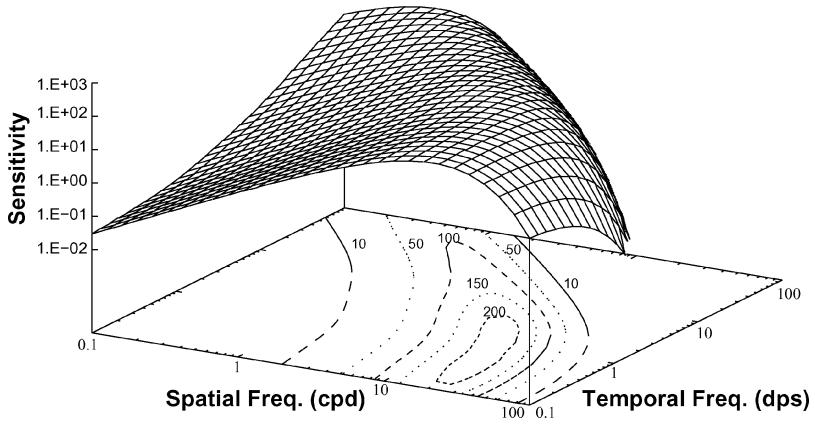

Figure 3. Spatio-temporal contrast sensitivity function (STCSF) plot.

rate of 24 frames/second. HVP has also been used for optimizing 3D Graphics computations such as Level of Detail (LOD) generation algorithms [15–18]. The LOD technique based on the distance between the object and the view point, has been explored to reduce the rendering computation, storage and transmission needs. Both HVP and visual attention concepts are adapted to accelerate rendering speed by reducing computations [19] and memory storage [20]. HVP has also been utilized to produce realistic graphic images by using special effect algorithms such as motion blur [21] and depth of field [22] to mimic the camera system.

In this work, an object's screen velocity and depth from a viewer are used as the criteria for adaptive shading [23] and adaptive texture mapping [24], based on the STCSF characteristics.

2.5. 3D CORDIC Algorithm

The CORDIC algorithm is an iterative algorithm that uses only shift-add functions for implementing various signal processing functions like sine, cosine, vector rotation, coordinate transformation, and even linear functions. The basic theory of the CORDIC algorithm can be explained with two-dimensional vector rotation example. Equation (5) shows the equations for the general vector rotation CORDIC algorithm. x and y are the cartesian components, z is used for the angle accumulation, K_i is the scale constant for i_{th} iteration, and σ_i is the sign of accumulated angle for i_{th} iteration. To rotate a vector $\mathbf{V}(x, y)$ by angle z, x_0, y_0 and z_0 are initialized with x, y and z, respectively. After n_{th} iteration, if z_n converges to 0, then x_n and y_n are taken as the components of the rotated vector. The rotation angle for each iteration is pre-computed as $\tan^{-1}(2^{-i})$ and stored in a small Read Only Memory (ROM).

$$\begin{aligned} x_{i+1} &= K_i(x_i - \sigma_i y_i 2^{-i}) \\ y_{i+1} &= K_i(y_i + \sigma_i x_i 2^{-i}) \\ z_{i+1} &= z_i - \sigma_i \tan^{-1}(2^{-i}) \\ K_i &= \frac{1}{\sqrt{1 + 2^{-2i}}} \\ \sigma_i &= \pm 1. \end{aligned} \tag{4}$$

3D vector rotation can be achieved by either cascading 2D CORDIC blocks [25], or introducing redundant variables to the CORDIC algorithm [26] to overcome the speed problem. Equation (5) introduced in [26] is analogous to 2D CORDIC equation except that three additional variables (u, v, w) are used and z is a coordinate component, rather than an angle value.

$$\begin{aligned} u_{i+1} &= K_i^2(u_i - x_i p_i 2^{-i} + v_i t_i 2^{-i} + y_i t_i p_i 2^{-2i}) \\ v_{i+1} &= K_i^2(v_i - y_i p_i 2^{-i} + u_i t_i 2^{-i} - x_i t_i p_i 2^{-2i}) \\ w_{i+1} &= K_i(w_i + z_i p_i 2^{-i}) \\ x_{i+1} &= K_i^2(x_i + u_i p_i 2^{-i} - y_i t_i 2^{-i} - v_i t_i p_i 2^{-2i}) \\ y_{i+1} &= K_i^2(y_i + v_i p_i 2^{-i} + x_i t_i 2^{-i} + u_i t_i p_i 2^{-2i}) \\ z_{i+1} &= K_i(z_i - w_i p_i 2^{-i}). \end{aligned} \tag{5}$$

Thus the CORDIC algorithm is well suited for reconfigurable hardware because of its shift and add operations, and due to its ability to faciltate a fine-grain precision control by providing precision proportional to the number of iterations computed.

3. Methodology

This section discusses the methodology followed to characterize power savings and other demerits such as the introduced error, to arrive at realistic control criterion that provides maximum power savings yet with a graceful quality degradation.

3.1. Power Estimation

The power estimation methodology uses analytical as well as experimental approaches. Analytical methods are used to arrive at realistic first-order estimates for power savings at a faster rate. For example analytical estimates are used to obtain the power ratio between Phong and Gouraud shading as a function of various primitive dimensions.

3.1.1. Power Ratio Between Gouraud and Phong Shading. In this section, a simple model is introduced to show power savings possible due to adaptive shading. In order to simplify the model and to minimize the computation difference between the two shading algorithms, the following assumptions are made:

(a) All functional units have the same word length.
(b) Only one light source is present.
(c) Specular reflection parameter $s = 1$.
(d) For the specular term, $\vec{N} \cdot \vec{H}$ [27] is used instead of $\vec{R} \cdot \vec{V}$. Since halfway vector $\vec{H}$ between $\vec{L}$ and $\vec{V}$

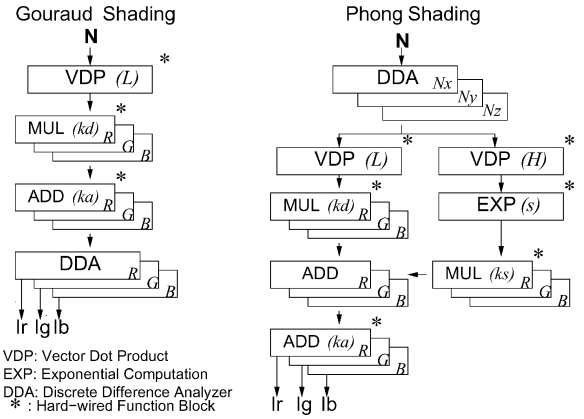

Figure 4. Basic data flow of Gouraud and Phong shaders.

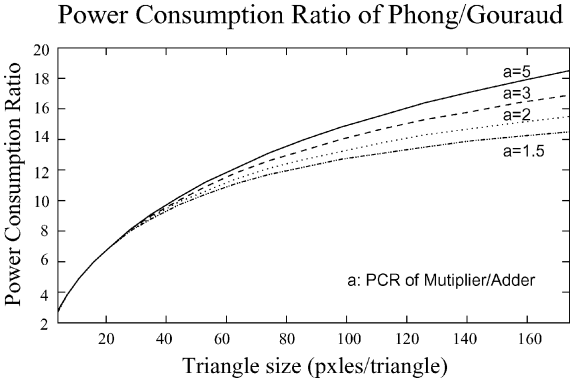

Figure 5. Power consumption ratio of Phong shading and Gouraud shading: one triangle shading.

is same for all pixels of an object, while $\vec{R}$ varies for each pixel.

(e) The functional units adder, multiplier, and divider have normalized power consumptions of 1, 1.5 and 4, respectively [28] (pp. 2–6).

Figure 4 shows the basic data flow block diagram of the Gouraud and Phong shader.

Simplified power model for Gouraud and Phong shading based on the above assumptions can be expressed by Eqs. (6) and (7), respectively.

$$P_{\text{gouraud}} = P_{\text{dda}} + P_{\text{diff}} \times 3 \quad (6)$$

$$P_{\text{phong}} = P_{\text{dda}} \times 3 + (P_{\text{diff}} + P_{\text{spec}}) \times p \quad (7)$$

where P_{dda}, P_{diff} and P_{spec} are the power consumed by the DDA algorithm, diffusion term and specular term, respectively. p is the number of pixels in a triangle.

The first term P_{dda} of Eq. (6) is the power consumed to compute the inner pixels of a triangle given the pixel values of its three vertices. Although P_{dda} of a triangle varies due to the position and shape, the variation could be ignored by assuming that all triangles are equilateral. Therefore, the P_{dda} is:

$$\begin{aligned} P_{\text{dda}} &= (x+1)(P_{\text{div}} + P_{\text{sub}}) + [p - (3x-3)]P_{\text{add}} \\ &= 5(x+1) + (p - 3x + 3) = p + 2x + 8 \end{aligned} \quad (8)$$

where x and p are the number of pixels of a side of a triangle and the number of pixels within a triangle, respectively. The relationship between x and p can be expressed by $0.43x^2 + 0.13x - 0.56 = p$. This x and p relationship equation is derived from the simple equilateral triangle trigonometry. We also assume that $P_{\text{sub}} = P_{\text{add}}$. Unlike the P_{dda} of P_{gouraud}, P_{dda} term of Eq. (7) is the power consumed to interpolate the normal vector $\vec{N}$. Since $\vec{N}$ has three components (x, y, z), P_{dda} is multiplied by 3.

From Eq. (1), it can be seen that the diffusion term needs one vector dot product and one multiplication with the diffusion coefficient, and one more addition is needed to add it to the ambient term for obtaining the overall intensity. Hence $P_{\text{diff}} = 3 \times P_{\text{mul}} + 2 \times P_{\text{add}} + 1 \times P_{\text{mul}} + P_{\text{add}} = 4 \times P_{\text{mul}} + 3 \times P_{\text{add}} = 9$. With the assumptions c. and d., P_{spec} is equal to $P_{\text{diff}} - 1 = 8$. By substituting these numbers into Eqs. (6) and (7), the ratio of P_{phong} to P_{gouraud} can be expressed as in Eq. (9). Figure 5 shows the overall power consumption ratio for different power consumption ratios of the adder and the multiplier.

$$\begin{aligned} \frac{P_{\text{phong}}}{P_{\text{gouraud}}} &= \frac{3(p+2x+8) + (9+8)p}{(p+2x+8) + (9 \times 3)} \\ &= \frac{20p + 6x + 24}{p + 2x + 35} \end{aligned} \quad (9)$$

where $p \geq 3$.

3.1.2. Experimental Power Estimation Methodology. The analytical estimates are approximate, and cannot properly model certain situations like the suitability of the architecture to the implementation mechanism and the impact of input data-patterns on power consumption. An experimental estimation methodology can effectively address these concerns.

Figure 6 shows the experimental methodology adopted for the research. The first step involves the creation of an application using OpenGL programming and production of an animation such as a teapot or

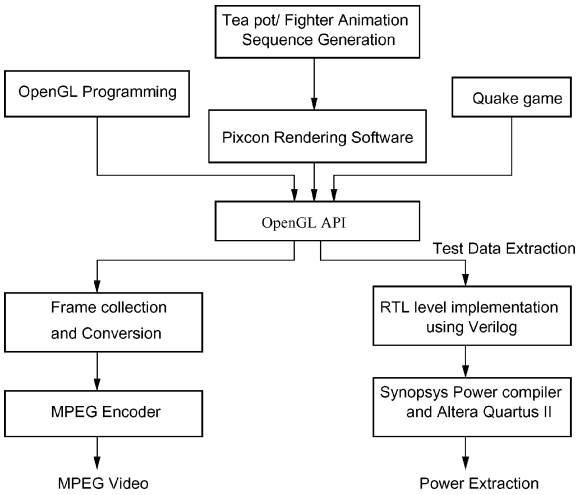

Figure 6. Experimental methodology for power savings estimation using HVP based adaptive computations.

fighter using Pixcon software or playing of a 3D game such as *Quake2* [1]. Open Graphics Library (OpenGL) functions as the interface between the program and computer hardware. The application test data is traced by modifying the OpenGL library to monitor the inputs to both shading and the texture mapping units. while, OpenGL programming can be used to simulate both Gouraud shading and texture mapping, Phong shading is not supported. Hence Pixcon software is used to perform Phong shading. *Quake2* uses texture mapping excessively, and hence provides a more realistic benchmark for texture mapping.

Simultaneously, a hardware model of the functions of interest is implemented at the Register Transfer Logic (RTL) level using Verilog Hardware Description Language(HDL). The power consumption of the hardware model is simulated using Taiwan Semiconductor Manufacturing Company (TSMC) 0.25 μm Application Specific Integrated Circuits (ASIC) library and Synopsys Power Compiler based methodology or a Field Programmable Gate Array (FPGA) based cell library using the Altera QuartusII tool. The test application data obtained above is used to excite the hardware modules and obtain realistic power values. The methodology is also used to derive HVP based control criterion. The characteristics such as the speed and distance of an object from the viewer are specified in the application program. The 3D graphics still image rendered by OpenGL is observed to determine if high or low quality is needed.

3.2. *Image Quality Measurement*

In general, 3D graphics image quality is measured by human observation. For example, the level decision criteria of the LOD algorithm is pre-decided by trial and error even though a method based on the human visual system is introduced. In order to develop criteria for shading selection, the term picture signal to noise ratio(PSNR) used for video image comparison (Eq. (10)) has been utilized. PSNR cannot be used alone to decide the relative quality of a given graphic image, since different images with the same PSNR may not have the same quality. Because of this reason, human observation is used in addition to PSNR to judge the relative difference in image qualities.

$$\mathrm{PSNR} = 10 \log \frac{255^2 m^2}{\sum_{n=1}^{m} (R_n - C_n)^2} \qquad (10)$$

where m, R, and C are the number of pixels in an image frame, the reference image, and the compared image, respectively. Figure 7 shows stills from the two

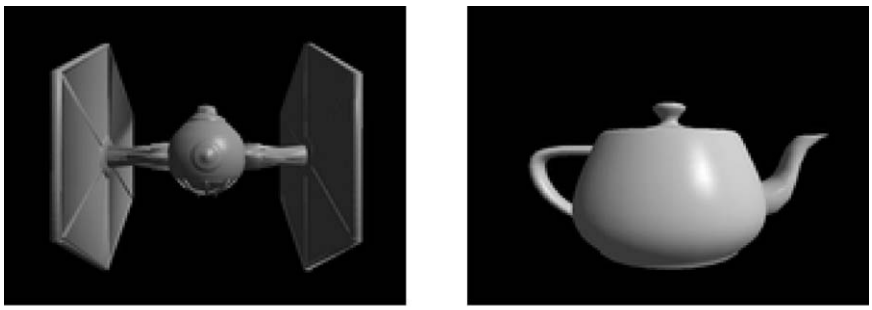

Figure 7. Simulation model: (a) Fighter with 3843 triangles and 2014 vertices (b) Teapot with 2257 triangles and 1180 vertices.

objects used for the simulation. Although just monochrome stills, the shading artifacts can be easily discerned.

4. Algorithmic Reconfiguration in Shading

This section explains the various techniques proposed for adaptive shading. The adaptive shading uses an appropriate shading algorithm from a set of alternatives. The limits of a human visual system like the after image phenomenon, its relative insensitivity to objects at a distance and in motion, are leveraged to select a low quality shading algorithm. The information regarding scene change occurrence, the distance between camera and object and speed of a moving object are required for adaptive shader control. As this information is available only at the application stage of graphics processing, help from this stage is required for successful deployment of these techniques. Note that this information is readily available in 3D graphics while it must be derived for video sequences. The various adaptive shading techniques are.

A. Distributed Computation Over Frames

In a layered image rendering system [29], scene change is one of the cases pushing hard on the system, since each object of all the layers should be rendered in one frame time. Thus a scene change often determines the rendering system clock. The after image phenomenon of the human visual system can be used to ease the system clock by rendering new objects over several frame times. The clearly noticeable artifacts that may result due to progressive rendering can be mitigated by using an accumulation buffer. From Eq. (1), overall intensity requires computation of the diffusion and specular terms over all the light sources. If there are four light sources, the system clock should be fast enough to iterate the diffusion and specular term four times in a frame time. As shown in Fig. 8, by adding an accumulation buffer and executing diffusion and specular term iterations over four frame times, the system clock can be lowered to $1/4$ of the initial value. This potentially allows the lowering of system clock speed.

If the normal vector for each pixel is cached in a local memory, it need not be re-computed for each light source. For a 640×480 image frame size rendering system with a 16 bit normal vector and support for hidden surface removal algorithm, the hardware cost is 1.85 MB memory and 922 KB for normal vector caching and accumulation buffer respectively. The power savings obtained from this approach could be up to $75\% - P_{\text{mem}}$, where P_{mem} is the power consumed for access and maintenance of the memory. The power reduction varies according to the contents of the images. Since the coverage of artifacts resulting from progressive rendering over frames could be proved only by displaying the image sequence, the results are not shown in this paper.

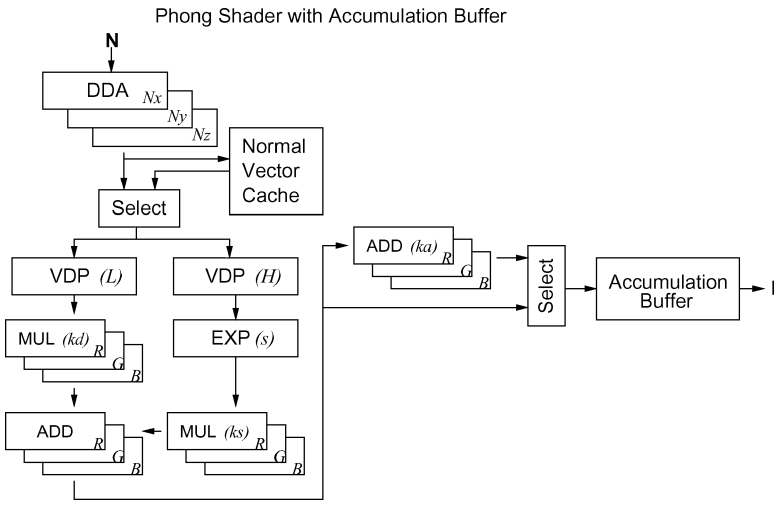

Figure 8. Phong Shader with accumulation buffer.

B. *Adaptive Shading on a Moving Object*

An object can be shaded using either Gouraud or Phong shading algorithms, the dataflow of which is shown in Fig. 4. If the speed and distance of the object from the camera are above a particular threshold, the low complex Gouraud shading can be used without perceivable quality degradation. In order to determine the threshold values, the PSNR value over a range of speeds and distances is noted. The PSNR graphs for the models of the focus, *Teapot* and *Fighter*, are depicted in Figs. 7 and 9.

A decision rule graph which indicates the algorithm to be used at a given speed and distance is then generated based on the chosen PSNR. Though, a chosen PSNR may lead to a contour shaped decision line, it can be approximated by a straight line to simplify the controller. Since the 3D models may differ in a variety of attributes like light reflection, surface complexity, color, etc., a unique decision rule might be required for each 3D model or a set of closely behaved models.

The difference between Phong and Gouraud shaded images can be noticed clearly in the boxed area of Fig. 10, even though motion blurring is applied, since the images are being observed in still mode. However, when the image sequence generated by adaptive shading is played in real time, it is hard to notice the difference with an appropriate decision rule. The decision line in Fig. 10 can be altered to modify the amount of time low quality shading is used, thus controlling the power savings.

Table 1 shows the power savings possible by rendering the *Fighter* and the *Teapot* 3D models in fast motion. The *Teapot* model saves more power than the *Fighter* model since it has a greater number of pixels per triangle. The savings estimates are measured with $s = 1$. At the s values of 20 and 40 which are realistic for the *Fighter* and the *Teapot* models, respectively, the power savings could be higher than that shown in Table 1.

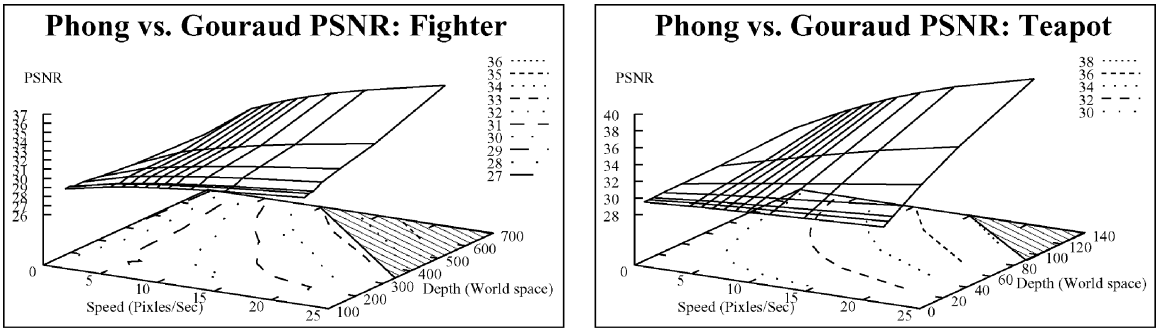

Figure 9. PSNR of adaptive shading on moving objects: (a) Fighter and (b) Teapot.

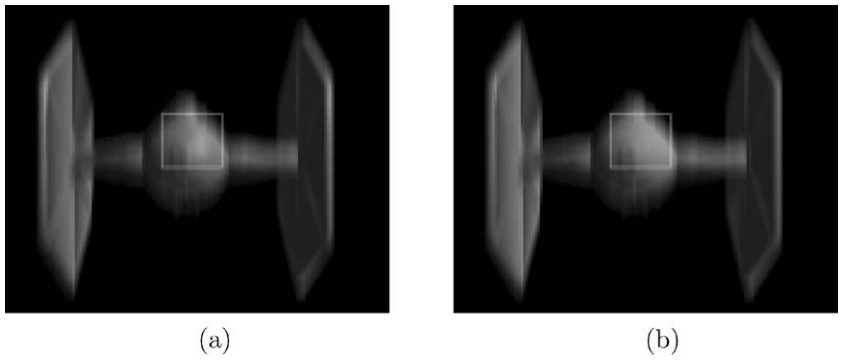

Figure 10. Blurred Fighter image (21 pixels/frame): (a) Phong shading and (b) Gouraud shading.

Table 1. Average power saving for 5 different distances.

Distance	Avg. Pixel/Tri	Fighter	Teapot	Avg. Pixel/Tri
1	23.5	7.35	10.36	51.8
2	18.8	6.56	8.52	32.1
3	15.8	5.99	7.10	21.9
4	11.2	4.94	5.06	11.7
5	6.7	3.66	3.23	5.4
Avg.		5.7	6.85	
Power saving %		82.5%	85.4%	

C. *Adaptive Specular Term Computation*

Evaluation of the specular term of Eq. (1) involves an expensive exponential computation. The exponential computation can be implemented using a variety of methods [30], however, we consider the general implementation of exponential computation as the reference for which the computational cost grows at least logarithmical in s [31].

An alternative approach exists for computing the specular term called Fast Phong shading [32], which is given by Eq. (11). Since this equation requires one multiplication, one subtraction, one addition, and one division, it is not power efficient for all values of the specular reflection parameter s. For small values of s, using just the iterative multiplication consumes less power.

$$Specular\_term = k_s \frac{ns}{(\vec{N}\cdot\vec{H}) - (\vec{N}\cdot\vec{H})ns + ns}. \tag{11}$$

If the reference exponential computation block requires 8 multiplications when $s = 256$, the power consumption is $8 \times 1.5 = 12$, while the power consumption of the fast phong equation is $1 + 1 + 1.5 + 4 = 7.5$, leading to a 37.5% power saving. Figure 11 shows the difference between Phong and Fast_Phong with $s = 64$ which yields 16.7% power saving.

The adaptive specular term computation uses the same criteria and one more input parameter s for decision rule making. Figure 12 shows the PSNR graph used to make the decision line. Unlike the graph from the adaptive shading, the Fast Phong shading can be used for all objects in motion regardless of the speed

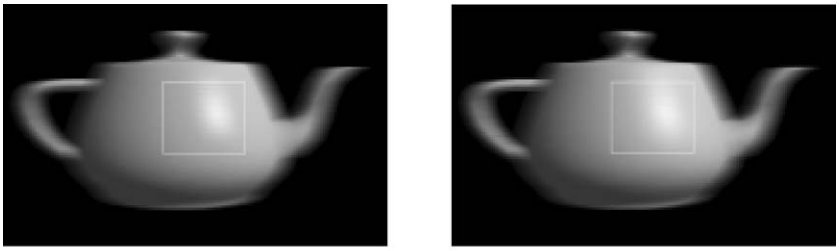

Figure 11. Blurred Teapot image (21 pixels/frame): (a) Phong shading and (b) Fast Phong shading.

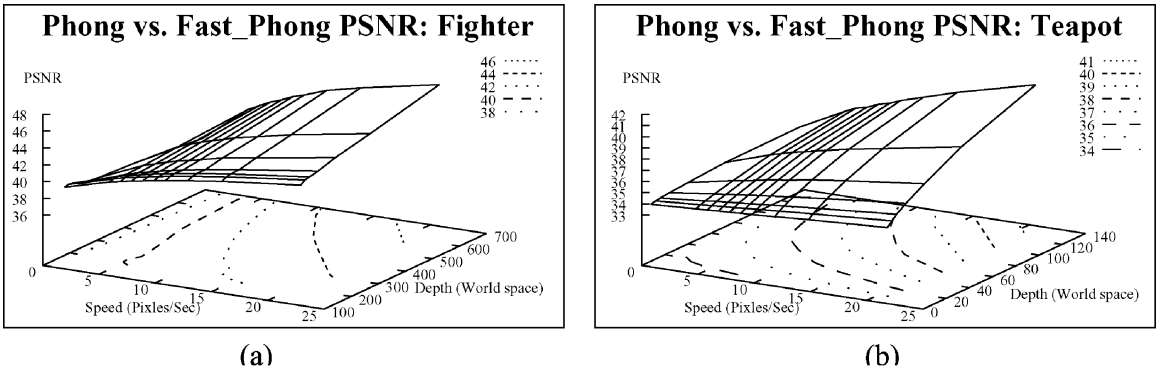

Figure 12. PSNR of adaptive specular term computation: (a) Fighter and (b) Teapot.

and depth, since the PSNRs are much higher than those of the adaptive shading.

5. CORDIC Based Vector Interpolator

In a power-aware 3D graphics system [23], it is preferred to have a flexible hardware block that provides precision control. Three dimensional vector interpolation required by Phong shading [33], is one functional block where multiple precision capability could help in a power-aware system design.

5.1. 3D Vector Interpolator

The vector interpolation involves interpolating the surface normal vectors of primitive vertices to obtain surface normal vectors of pixels lying on the edges and within the primitive. Figure 13 shows the four intermediate vectors obtained by interpolating the vectors $\mathbf{V}_1$ and $\mathbf{V}_2$.

The vector interpolation can be performed using a variety of algorithms such as spherical interpolation, linear interpolation, CORDIC based vector interpolation, etc. The spherical interpolation though sophisticated, involves complex computations such as sine, division and multiplication. The linear interpolation is less involved computationally, yet demands complex post normalization which requires one square-root, three divisions, three multiplications and two additions. The CORDIC based interpolator is more flexible and efficient, and uses CORDIC algorithm in the rotation mode for performing vector interpolation.

5.2. 3D CORDIC Vector Interpolation

The proposed 3D CORDIC interpolator is based on 3D redundant CORDIC algorithm introduced in Section 2. It operates on polar components instead of cartesian components because of its hardware implementation benefits [34]. If polar coordinated data is not supported by the system, the CORDIC architecture itself can perform the conversion, without any additional hardware. The CORDIC vector interpolator consists of two steps. The first step interpolates the polar components of the two given vectors linearly according to the position of the intermediate point. As shown in Fig. 13, θ and ϕ are the differences between polar components of the two vectors. This step is the same as that in the cartesian component interpolation method, except that the computations are now performed on two parameters (θ, ϕ), rather than three (x, y, z). Instead of vector normalization, the CORDIC vector rotation is performed to produce the normalized vector in the second step. This output of the CORDIC vector rotation is ready to be used in the next step of the Phong shader.

Due to the inherent characteristic of linear interpolation, the CORDIC vector interpolator produces an interpolation path which is different from the analytically estimated path. The gap between the two paths tends to become greater for larger γ. However, as shown in Fig. 14, over 90% of γ is less than 30 degrees (0.52 radians) which is small enough for the path difference to be ignored. The quality of interpolation is proportional to the bit precision $(N_i - 1)$ or the number of iterations (N_i), because convergence error becomes more prominent at smaller numbers of iterations. It can be seen from Fig. 15 that 10 iteration CORDIC (9-bit

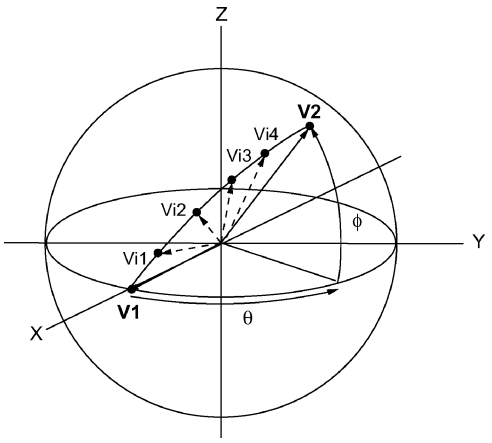

Figure 13. Vector interpolation.

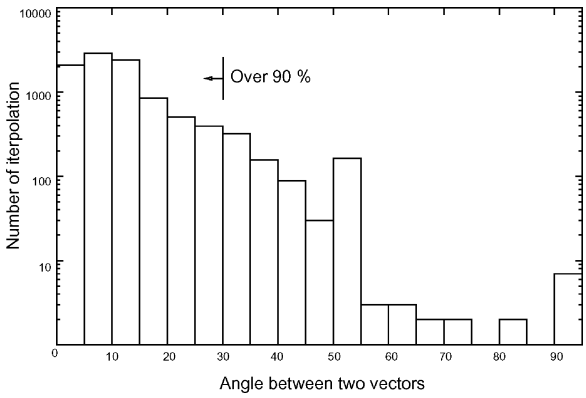

Figure 14. Statistics of angles between two vectors: Teapot.

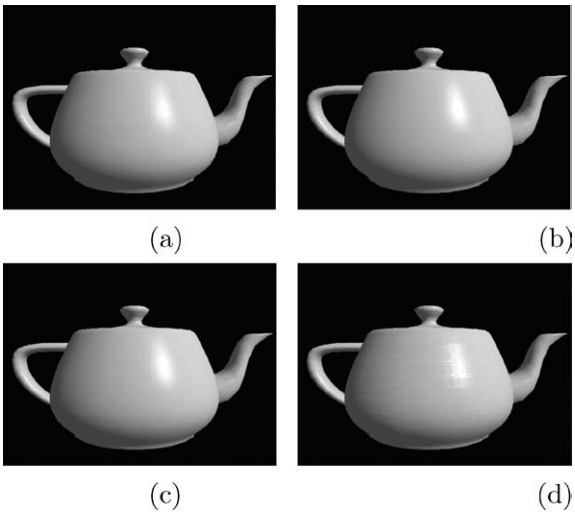

Figure 15. Teapot with different number of CORDIC iterations: (a) spherical interpolation, (b) 10 iteration CORDIC, (c) 8 iteration CORDIC and (d) 6 iteration CORDIC.

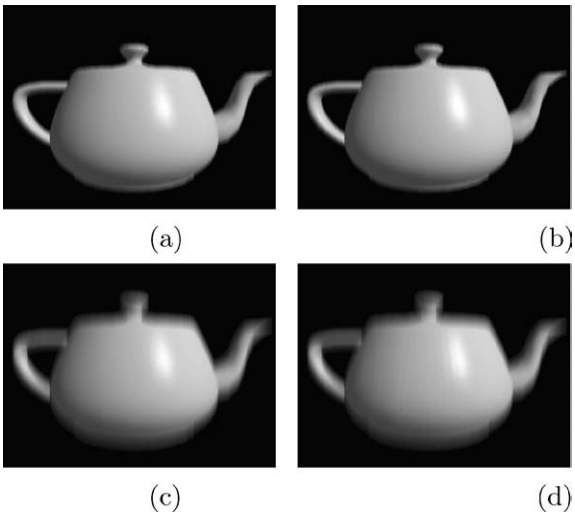

Figure 16. Motion blurred image: (a) 10 pixel motion blurred teapot (spherical interpolation), (b) 10 pixel motion blurred teapot (8 iteration CORDIC), (c) 30 pixel motion blurred teapot (spherical interpolation) and (d) 30 pixel motion blurred teapot (6 iteration CORDIC).

data + 1-bit sign) can produce almost similar interpolation results as that of other sophisticated algorithms, and can meet the normal perceptual quality requirements for the teapot image.

5.3. HVP Based Adaptive Control

The error introduced due to CORDIC based vector interpolation is readily apparent at reduced number of iterations as shown in Fig. 15. However, if the depth or speed of the object increases, the artifacts may not be easily noticeable due to HVP limits. In this work, the impact of increased object speed on the number of iterations required for producing an image with out any perceivable quality degradation is investigated. The image of fast moving objects is simulated by applying motion blurring on the respective objects at steady state. It can be seen from Fig. 16 that, six iterations for a 30-pixel and eight iterations for a 10-pixel motion blurred teapot can be used without any noticeable degradation. The optimum number of iterations required at various speeds, thus obtained, can be stored as a control map for future adaptive control. The map can be created for each object separately or for a group of objects having similar surface complexity. The same control map based approach can be extended to account for other HVP parameters such as the number of frames per second, the size of the image and the viewer distance from the display device.

5.4. CORDIC Architecture for Vector Interpolation

The proposed CORDIC architecture is based on Eq. (5) and uses redundant arithmetic CORDIC instead of cascaded 2D CORDIC functional blocks. It uses more computations but provides easy precision control. The function block for the first step of the CORDIC vector interpolator is the same as that of the cartesian component interpolator. The inputs of the second step (u, v, w, x, y, z) assumes the values $(0, 0, 0.673, \pm 0.3696, 0, 0)$, respectively. The initial value of x is selected according to the target angle θ. The pre-scaled initial values of x and w are chosen to remove the post-scaling computation of the rotated vector.

The 10 iteration mode requires 80 shift operations and 140 additions if an optimized 4-input adder block is employed. The number of iterations can be reduced further by storing a pre-computed dataset in memory of the first few iterations and using it as required. As shown in Table 2, 122.9 Kbits are required to save all possible data values for the first five iterations. This storage requirement can be satisfied by recent FPGA devices such as the Altera EP20K series device.

If a pre-computed data set for 5 iterations is used, the total number of computations for 10 iterations is reduced to 60 additions and 30 shift operations. Similarly, for the 6 iteration CORDIC mode, only 12 additions

Table 2. Computation savings for different memory size usage (A: add, S: shift).

No. iterations to save	No. data (n) to store	Required memory ($n \cdot 6 \cdot 10$ bit)	Reduced computations
2	2^5	1.9 Kb	32A, 20S
3	2^7	7.7 Kb	48A, 30S
4	2^9	30.7 Kb	64A, 40S
5	2^{11}	122.9 Kb	80A, 50S

and 6 shift operations are needed. Thus CORDIC based scheme is very efficient compared to the conventional vector normalization requiring three multiplications, three divisions, one square-root and two additions for each vector. More details about the architecture can be found in [35].

5.5. Results

Table 3 shows the energy consumption and energy saving ratios for different number of iterations. The energy consumption values are computed based on the power results obtained using *Synopsys Power Compiler* with TSMC 0.25μ library. Note that the power savings doesn't include the extra power involved in accessing the pre-computed dataset from the internal memory. It can be seen that the energy savings from dynamic operation can be up to 72% when all objects in a scene are rendered in 6 iteration CORDIC mode.

To compare the energy consumption with the conventional vector normalization, a 10-bit divider is implemented and is found to consume $0.7 \cdot 10^{-6}$ (J). Since vector normalization performs 3 divisions, $2.1 \cdot 10^{-6}$ (J) is required only for the division operations. This demonstrates that energy consumption of the presented CORDIC vector interpolator could be competitive in the 10 iteration mode, and better in lower iteration mode as compared to the conventional hardware.

Table 3. Power consumption ratio for different number of iterations.

No. iterations	Energy consumption (J)	Energy savings
10	$2.24 \cdot 10^{-6}$	–
8	$1.21 \cdot 10^{-6}$	46%
6	$0.5 \cdot 10^{-6}$	78%

6. Adaptive Texture Mapping

The texture mapping step of the 3D Graphics rendering pipeline involves a significant amount of computations and memory accesses for performing interpolation. The demand for these operations is aggravated by the increasing need for high quality real-time texture mapping and associated increase in texture resolutions. Various techniques have been proposed to improve texture mapping performance.

Previous Work

Texture mapping using compressed textures has been proposed by Beers et al. [36], to alleviate the texture transmission and storage requirements. However some artifacts are introduced and the amount of interpolation computations remain unaltered. Kugler [37] proposed space-variant filter in the texturing unit to minimize the artifacts. Hakura [38] and Cox et al. [39] demonstrated the use of a single- and multi-level texture caching for improving bandwidth and latency of the texture memory system. The intra and inter frame texture locality were exploited to achieve these improvements. Igehy et al. [40] used a prefetching cache scheme to overcome the texture memory latency.

A few techniques have been proposed for adaptive texture mapping. Rosman et al. [41] presented a scheme for dynamically lowering the quality of interpolation based on the fractional level of depth value. HVP based adaptive texture mapping has been developed by Dumont et al. [20]. They leveraged visual importance of the textures to enhance the efficiency of the texture memory system. Their primary goal to provide high frame rates is different from our objective.

In this work the fact that human visual system is less sensitive to certain complex contents and moving objects is leveraged to dynamically perform lower quality interpolation. HVP based control criterion is proposed for directing the interpolation algorithm selection process.

6.1. HVP Based Algorithm Selection

For adaptive control, object velocity and texture spatial frequency are used as the input parameters. According to the value of these two parameters, the adaptive controller decides which interpolation will be used for the incoming data. Figure 17(a) shows the contrast

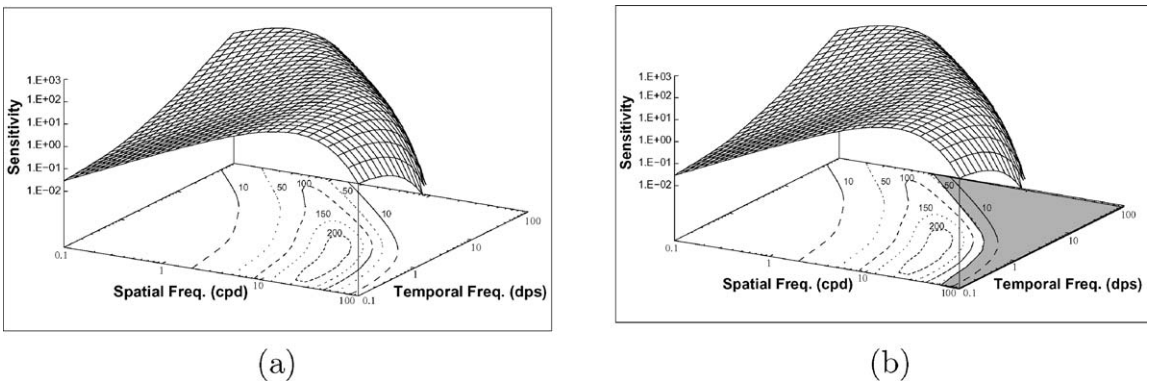

Figure 17. (a) Spatio-temporal contrast sensitivity function (CSF) plot and (b) Shaded area: sensitivity less than 50.

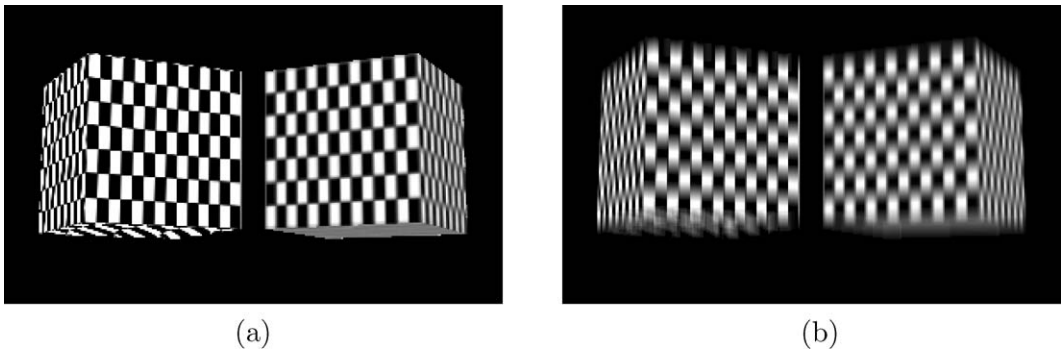

Figure 18. (a) Bilinear interpolated cube (left) and Trilinear interpolated cube (right) and (b) Motion blurred by 10 pixels.

sensitivity function graph [13] which is the base of decision rule and an example is depicted in Fig. 17(b).

In Fig. 17(b), if the velocity and the texture spatial frequency of an object fall in shaded area, mode controller sets the texture mapping system to bilinear interpolation. Decision making in the mode controller can be implemented with a memory map that has the decision rule graph. Figure 18(a) shows the quality difference between bilinear and trilinear interpolations. Although there are some noticeable aliasing artifacts in Fig. 18(a), it is hard to distinguish the difference in Fig. 18(b). Thus bilinear interpolation requiring less computations can be used instead of trilinear interpolation.

In general, trilinear interpolation function block consists of two bilinear interpolation blocks operating in parallel at the supply voltage and clock values of $V_{\text{dd,full}}$ and Clk_{full} to obtain a given throughput of k pixels/second. At the same throughput, one of the two bilinear interpolation function blocks will be idle in the bilinear mode. A technique called Dynamic Voltage Scaling (DVS) can be used to obtain quadratic power efficiency by operating the two bilinear units concurrently in the bilinear mode at reduced parameters $V_{\text{dd,half}}$ and Clk_{half} required to provide an individual throughput of $k/2$.

6.2. Results

Experimental results are obtained using the RTL level implementation and power analysis methodology explained in Section 3. Since TSMC 0.25 μm library supports only 2.5 V V_{dd}, the T_p vs. V_{dd} relationship is used to derive the $V_{\text{dd,half}}$. $V_{\text{dd,half}}$ is estimated to be 1.6 V leading to a power reduction ratio of $(1.6/2.5)^2$, 0.41. Thus, the architecture uses two modes; 2.5 V V_{dd} at 100 MHz clock for trilinear mode and 1.6 V V_{dd} at 50 MHz clock for bilinear mode. It is assumed that these voltages and clocks are supplied by the system level source simultaneously and a couple of multiplexers are used to select appropriate voltage and clock values without any significant delay.

Table 4. Power saving from adaptive interpolation: Quake2 10 frames.

Converted ratio (%)	Number of Bi-linear computation	Number of Tri-linear computation	Power consumption (mW)	Power savings (%)
0	3619749	3808894	454.68	0
0, DVS only	3619749	3808894	371.19	18.36
5	3810193	3618450	358.59	21.13
15	4191083	3237560	333.37	26.68
25	4571972	2856671	308.16	32.22
100	7428643	0	119.06	73.81

Table 4 shows power consumption of the adaptive texture mapping system for various interpolation conversion ratios. Interpolation conversion means that bilinear interpolation is performed for a certain pixel which is supposed to be interpolated in the trilinear mode. Since we do not have the source code of Quake2, the various conversion ratios are manually selected. As shown in Table 4, bilinear operation with DVS and without the HVP model leads to a 18.4% power saving and up to 73.8% savings are possible by setting the adaptive texture mapping system completely in bilinear mode using DVS (no trilinear interpolation).

7. Parameterized Texture Mapping

Section 6 discussed algorithm level techniques for low power texture mapping. More power savings can be achieved from these techniques by exploiting the texture mapping data and interpolation characteristics. These characteristics can be monitored using the two parameters, texel weight and its intensity. Significant amount of Multiply-Accumulate (MAC) operations form the core of interpolation as given in Eqs. (2) and (3). The number of MAC computations can be reduced by leveraging each of these parameters by:

(1) *Elimination of MAC operations based on a weight threshold (weight based technique):* The weights W_i, where $i \in \{0, 1, 2, 3\}$ as used in Eq. (2) are calculated using the formulae:

$$\begin{aligned} W_0 &= (1.0 - xf) \times (1.0 - yf) \\ W_1 &= \quad xf \quad \times (1.0 - yf) \\ W_2 &= (1.0 - xf) \times \quad yf \\ W_3 &= \quad xf \quad \times \quad yf. \end{aligned} \tag{12}$$

Here xf = FRAC(x) and yf = FRAC(y) respectively. Hence, as seen above, when (x, y) is closest to a particular position $i (i \in \{0, 1, 2, 3\})$, W_i is the largest and W_{3-i} is the smallest. If xf and yf assume random values between 0.0 and 1.0, then, M_i, the probability that $W_i \leq k$, $k \in \{0.0..1.0\}$, is equal for all i, and increases with k. It has been observed emperically that, at least $15 - 20\%$ of the MAC operations involve weights below a threshold k of 0.05, and if it is assumed that the texel intensities are comparable, these MACs can be eliminated while introducing a very minor error, thus saving power.

(2) *MAC to add transformation based on spatial correlation in texel intensities (intensity based technique):* The intensity values used in calculating $I_{(x,y)}$ in Eq. (2) are those of the neighboring texel values. If the textures exhibit significant amount of spatial correlation, the neighboring texels will be equal with a high probability. If any two texels, T_i and T_j are equal, a MAC operation can be transformed to an addition with out any degradation in precision using Eq. (13).

$$T_i \times W_i + T_j \times W_j = T_i \times (W_i + W_j). \tag{13}$$

There can exist at most three pairs of equal intensities among the four intensities. When the number of such equal pairs is 1, 2 and 3, the number of MAC operations converted to additions is 1, 2 and 4, respectively. Extra logic needed for identifying the equality of texel intensities $T_i, i \in \{0 \ldots 3\}$, is found to be less than that of a multiplier. Thus this technique leads to reduced computations and thus power when the number of equal pairs is ≥ 1. The number of bits of the intensity value, N, used for equality determination can be parameterized to facilitate variable degrees of precision and power savings.

7.1. Experimental Results

In order to obtain reasonably accurate estimates for power savings, experiments are performed on a trace obtained by running the first ten frames of the *Quake2*. Figure 19 depicts the percentage of less expensive MAC operations for a given weight threshold k. Figure 20 demonstrates the absolute error statistics for a set of weight threshold(k) values. Each ordinate value in

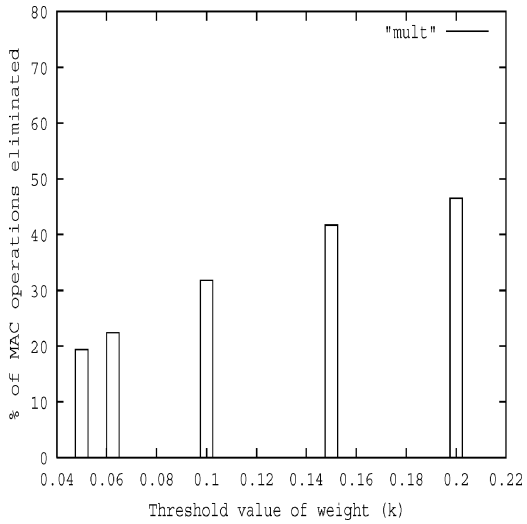

Figure 19. MAC operations saved in weight based technique.

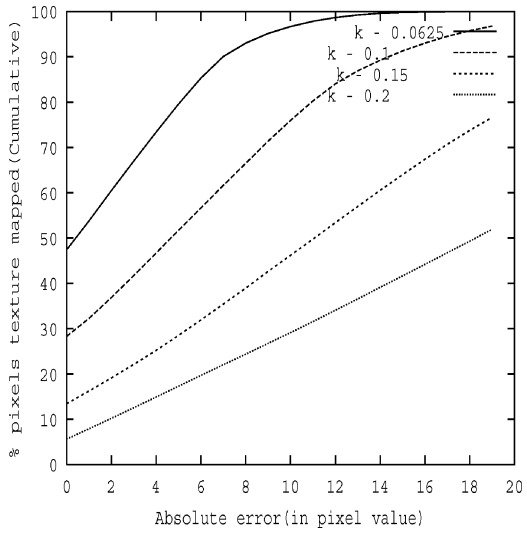

Figure 20. Absolute error statistics in weight based technique.

Fig. 20 represents the percentage of the texture mapped pixels with absolute error less than the corresponding abscissa value (out of a maximum pixel value of 255). It can be seen that up to 47% of the MACs can be eilminated at $k = 0.2$. However, to keep the absolute error within acceptable bounds, a value for k between 0.05 and 0.1 is preferable, resulting in a meager 19 to 31% reduction of MAC operations.

Figure 21 presents the percentage of MAC operations transformed into additions using the intensity based approach for a given number of MSB (most significant bit) positions N, ($N \in \{4 \ldots 8\}$) used for

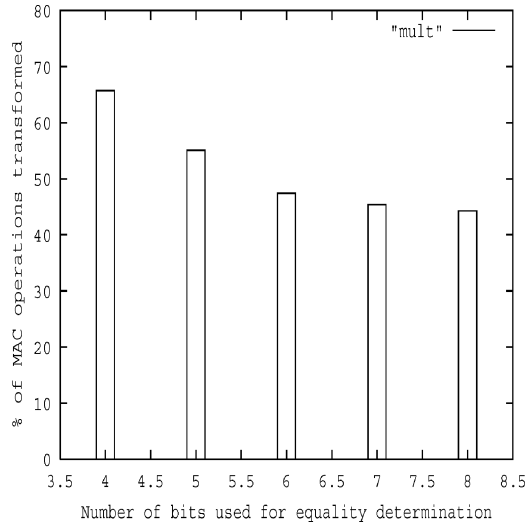

Figure 21. MAC operations transformed in intensity based technique.

equality detection. About 44% of the MACs can be transformed to additions without any error. The very low absolute error makes even the 4-bit ($N = 4$) mode of operation feasible. Hence, up to 67% of the MACs can be transformed to additions, resulting in significant power savings. It can be seen from Figs. 19–22 that the intensity based approach is very superior compared to the weight based approach, both from the point of view of power savings and precision.

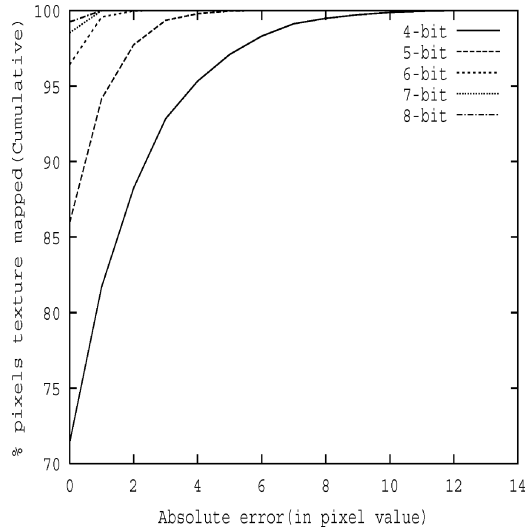

Figure 22. Absolute error statistics in intensity based technique.

8. Conclusions and Future Work

8.1. Conclusions

In this work, a novel power-aware graphics rendering system performing the 3D graphics shading and texture mapping is presented. The proposed system is based on the concept of AGR, according to which the amount of quality increases continuously with an increase in expended power. The image information along with the HVP characteristics are used to select the minimum perceptual quality required, thus leading to power savings. A methodology to estimate the power reduction and error involved in using approximate techniques is also presented.

Algorithm level power savings are demonstrated for both shading and texture mapping units. A CORDIC based 3D vector interpolator is presented to support dynamic control of computing precision ranging from 5–9 bits. A technique to exploit the spatial correlation of the texture maps used in texture mapping is also presented for enhanced savings.

8.2. Future Work

The HVP concepts are to be extended to facilitate power-awareness in other parts of the 3D Graphics system. The power savings possible using other reconfigurable techniques such as hardwired multiplier, distributed computations, etc., should be evaluated. Instead of optimizing graphics processing on the general purpose reconfigurable systems, the scope for developing graphic domain-specific reconfigurable units should be investigated.

Acknowledgments

We would like to acknowledge NSF CCR-9988238 for partially supporting the work. Sincere thanks are also due to ID Software Corp. for making the Quake2 game available, David Bucciarelli for providing the textured teapot animation, Brian Paul et al., for the Mesa 3D Graphics library, and Artisan Corp., for supplying the TSMC library. The authors would like to thank Aiyappan Natarajan, Arun Ramanathan and other anonymous reviewiers for their help with reviewing the document.

References

1. ID software, http://www.idsoftware.com/games/quake/quake/.
2. "Digital domain," http://www.d2.com/.
3. "VRML consortium," http://www.vrml.org/.
4. "Buzz 3D-PC," http://www.vrmarketing.com/buzz3d/b3d.Index.htm/.
5. A. Chandrakasan and R. Brodersen, "Low Power Digital CMOS Design," *Journal of Solid-State Circuits*, vol. 27, 1992, pp. 473–484.
6. R. Graybill and R. Melhem, *Power Aware Computing*, Kluwer Academic/Plenum Publishers, May 2002.
7. J. Rabacy, "Reconfigurable Processing: The Solution to Low Power Programmable DSP," in *International Conference on Acoustics, Speech, and Signal Processing*, 1997.
8. W. Burleson et al., "Dynamically Parameterized Algorithms and Architectures to Exploit Signal Variations for Improved Performance and Reduced Power," in *International Conference on Acoustics, Speech, and Signal Processing*, 2001.
9. T. Moller and E. Haines, *Real-Time Rendering*. AK Peters, 1999.
10. S. Nawab, A. Oppenheim, A. Chandrakasan, J. Winograd, and J. Ludwig, "Approximate Signal Processing," *VLSI Signal Processing, vol*. 15, nos. 1–2, 1997, pp. 177–200.
11. A. Sinha, A. Wang, and A. Chandrakasan, "Algorithmic Transforms for Efficient Energy Scalable Computation," in *International Symposium on Low Power Electronics and Design (ISLPED)*, Aug. 2000.
12. M. Reddy, "Perceptually Optimized 3D Graphics," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, Sept./Oct. 2001, pp. 68–75.
13. D. Kelly, "Motion and Vision II: Stabilized Spatio-Temporal Threshold Surface," *Journal of the Optical Society of America*, vol. 79, no. 10, Oct. 1979, pp. 1340–1349.
14. C.J. van den Branden Lambrecht, "A Working Spatio-Temporal Model of Human Visual System for Image Restoration and Quality Assesment Applications," in *International Conference on Acoustics Speech and Signal Processing*, May 1996.
15. R. Pajarola and J. Rossignac, "Compressed Progressive Meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, 2000, pp. 79–93.
16. J. Cohen, M. Olano, and D. Manocha, "Appearance-Preserving Simplification," in *SIGGRAPH'98*, July 1998, pp. 115–122.
17. A. Gueziec, G. Taubin, F. Lazarus, and W. Horn, "Simplical Maps for Progressive Transmission of Polygonal Surfaces," in *VRML '98*, Feb. 1998.
18. H. Hoppe, "Progressive Meshes," in *SIGGRAPH '96*, July 1998, pp. 99–108.
19. H. Yee, S. Pattanaik, and D. Greenberg, "Spatiotemporal Sensitivity and Visual Attention for Efficient Rendering of Dynamic Environments," *ACM Transactions on Graphics*, vol. 20, no. 1, Jan. 2001, pp. 39–65.
20. R. Dumont, F. Pellacini, and J. Ferwerda, "A Perceptually-Based Texture Caching Algorithm for Hardware-Based Rendering," in *Eurographics Workshop on Graphics Rendering*, June 2001, pp. 249–256.
21. T. McReynolds, T. Blythe, B. Grantham, and S. Nelson, "Programming With OpenGL: Advanced Techniques," in *SIGGRAPH '98*, July 1998.
22. J. Snyder and J. Lengyel, "Visibility sorting and Compositing Without Splitting for Image Layer Decomposition," in *SIGGRAPH '96*, July 1996, pp. 219–230.

23. J. Euh and W. Burleson, "Exploiting Content-Variation and Perception in Power-Aware 3D Graphics Rendering," in *Power Aware Computer Systems*, LNCS: Springer, 2000, pp. 51–64.
24. J. Euh, J. Chittamuru, and W. Burleson, "A Low Power Content-Adaptive Texture Mapping Architecture for Real-Time 3D Graphics," in *Power Aware Computer Systems*, LNCS: Springer, 2002.
25. O. Mencer, L. Semeria, M. Morf, and J. Delosme, "Application of Reconfigurable CORDIC Architectures," *The Journal of VLSI Signal Processing, Special Issue on Reconfigurable Computing*, March 2000.
26. C. Clarke and G. Nudd, "A Redundant Arithmetic CORDIC System with a Unit Scale Factor," in *IMA Conference on Mathematics in Signal Processing III*, 1994, pp. 63–73.
27. J. Blinn, "Simulation of Wrinkled Surfaces," in *SIGGRAPH '78*, 1978, vol. 12, pp. 286–292.
28. A. Nannarelli, "Low Power Division and Square-Root," Ph.D, Dissertation, U.C. Irvine, 1999.
29. J. Torborg and J. Kajiya, "Talisman: Commodity Real-Time 3D Graphics for the PC," in *SIGGRAPH*, 1996, pp. 353–364.
30. H. Shin, J. Lee, and L. Kim, "A Minimized hardware Architecture of Fast Phong Shader using Taylor Series Approximation in 3D Graphics," in *IEEE International Conference on Computer Design*, Oct. 1998, pp. 286–291.
31. T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw-Hill, 1990.
32. C. Schlick, *Graphics Gems IV*, AP Professional, 1994.
33. B. Phong, "Illumination for Computer Generated Pictures," *Communications of the ACM (CACM)*, June 1975, pp. 311–317,
34. T. Ikedo and J. Ma, "The Truga001: A Scalable Rendering Processor," *IEEE Computer Graphics and Applications*, vol. 18, no. 2, April 1998, pp. 59–79.
35. J. Euh, J. Chittamuru, and W. Burleson, "CORDIC Based Vector Interpolator for 3D Graphics," in *IEEE Workshop on Signal Processing Systems*, Oct. 2002, pp. 240–245.
36. A. Beers, M. Agrawala, and N. Chaddha, "Rendering From Compressed Textures," *SIGGRAPH '96*, 1996, pp. 373–378.
37. A. Kugler, "Designing a High-Performance Texturing Circuit," in *Proceedings of the Electronic Imaging Symposium on Multimedia Hardware Architectures*, Feb. 1997.
38. Z. Hakura and A. Gupta, "The Design and Analysis of a Cache Architecture for Texture Mapping," in *24th International Symposium on Computer Architecture*, 1997, pp. 108–120.
39. M. Cox, N. Bhandari, and M. Shantz, "Multi-Level Texture Caching for 3D Graphics Hardware," in *25th International Symposium on Computer Architecture*, June 1998, pp. 86–97.
40. H. Igehy, M. Eldridge, and K. Proudfoot, "Prefetching in a Texture Cache Architecture," in *Eurographics/SIGGRAPH Workshop on Graphics Hardware*, 1998, pp. 133–142.
41. A. Rosman and M. Pimpalkhare, "Adaptive Tri-Linear Interpolation for Use When Switching to a New Level-of-Detail Map," United States Patent: US 6,184,894 B1, Feb. 2001.

Jeongseon Euh obtained his Ph.D. from University of Massachusetts Amherst in 2002. Prior to that, he completed his B.S. degree in Electrical Engineering from Yonsei University, Seoul, Korea and M.S. Degree in Electrical Engineering from Fairleigh Dickinson University, NJ, USA. He was a member of the research staff at the Hyundai Electronics Industries, Co, Korea, from 1989 to 1993, where he had worked in the image data codec developing team for developing a High Definition Television. His main interests lie in the development of adaptive systems for 3D graphics and Multimedia, by exploiting the Human Visual Perception Criterion. Since 2002, he is working in the Digital Media Group of Samsung Electronics Co., Korea.

Jeevan Chittamuru obtained his MSECE from University of Massachusetts Amherst in 2003 and B.Tech from Regional Engineering College, Warangal, India, in 2000. He was associated with the VLSI Signal processing Laboratory at Amherst during 2001–2003, where he had worked on the development of algorithms and architectures for adaptive low-power 3D Graphics. His main research interests include chip-wide power reduction of computing systems, graphics domain-specific reconfigurable processing and design methodologies for adaptive systems. He is a student member of IEEE.
jchittam@ecs.umass.edu

Wayne Burleson is an Associate Professor of Electrical and Computer Engineering at the University of Massachusetts Amherst where he has been since 1990. He has a BSEE and MSEE from MIT and

a PhD in ECE from the University of Colorado. He has worked as a custom chip designer for VLSI Technology and Fairchild and as a consultant for Digital, Compaq, Intel, Datafusion and Tensorcomm. His research specialties include circuits for low-power, long interconnects, clocking and mixed signals with funding from NSF and Intel. He also conducts research in reconfigurable computing for content-adaptive signal processing, smart cards and multimedia instructional technologies with funding from NSF. He has published over 100 technical papers in these areas. He is even more proud of his former students who work as chip designers and CAD developers at Compaq, Intel, AMD, Agilent, Xilinx, Synopsys, IBM, Analog Devices, Cadence and as professors at several leading universities.

He was co-chair and co-program chair of the Signal Processing Systems Workshop. He has been an Associate Editor of IEEE Transaction on VLSI and IEEE Transactions on Circuits and Systems II. He won the Best Paper Award at the 1999 Frontiers in Education Conference.