特輯

고성능 그래픽스 전용 프로세서 개발

崔相吉, 崔秉均, 文相鎬, 魏榮徽
컴퓨터 構造 硏究室 三星綜合技術院

Ⅰ. 서 론

최근 자동화와 정보사회의 대두로 컴퓨터 그래픽의 응용분야가 급속히 확대되고 있는 가운데 3차원 그래픽을 실시간으로 처리하기 위한 고속 그래픽 가속기의 개발이 활발히 진행되고 있다. 현실감이 있는 영상을 만들기 위해서는 고해상도의 display(1280x1024이상)가 필요하며 1백만 개 이상의 화소에 대해 transformation과 색상계산 등을 직접 수행할 경우 엄청난 량의 계산을 해야한다. 이와같은 많은 계산량을 줄이기 위하여 물체의 곡면을 작은 polygon의 mesh로 나타내고 polygon의 각 vertex에 대한 transformation, 색상계산등을 수행한 다음 rasterizer에서 polygon 내부의 각 화소에 해당하는 색상의 근사치를 구하는 방법(Gouraud shading, Phong shading등[1, 2])이 많이 쓰이고 있다.

본 논문에서는 먼저 3차원 그래픽 시스템의 구조에 대하여 살펴본 다음, rasterizer 시제품을 ASIC으로 설계 및 제작한 Raster Engine Junior(REjr)의 기능 및 성능을 분석하고 REjr의 단점 및 기능 보완을 고려한 Raster Engine(RE)의 설계에 관하여 논의하기로 한다. 특히 4장에서는 RE에서 구현될 triangle input 처리 방법, fast Phong shading, texture mapping, alpha blending등의 고급 3차원 그래픽 기능들에 관하여 논의한다.

3차원 컴퓨터 그래픽 시스템의 일반적인 구조는 그림 1과 같은 pipeline 방식으로 구성되며 object를 나타내는 기본요소(primitive)들이 geometry processor와 rasterizer를 거치면서 화소 data로 변환되어 메모리(framebuffer)에 저장된다. Geometry processor는 geometric transformation, light modeling, clipping, perspective projection등의 pipeline으로 구현된다. Rasterizer는 primitive의 각 화소에 대해 shading model에 의거하여 칼라값을 결정하고 Z-buffer를 이용하여 은면 제거를 하는 일을 주로 한다. Geometry processing에서는 소량의 (polygon의 각 vertex에 대한) 복잡한 연산이 수행되는 반면에 rasterizing에서는 다량의 (polygon 내부의 각 pixel에 대한) 비교적 간단한 연산이 반복적으로 수행됨에 따라 고성능 그래픽 시스템에서는 Raster Engine으로 proprietary VLSI chip을 각자가 설계, 제작하여 사용하고 있다.

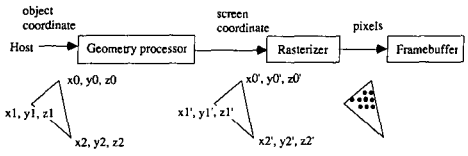


그림 1. Graphics pipeline

Rasterizer는 graphics pipeline에서 세가지 중요한 bottleneck의 하나로서 rasterizer의 ASIC화에 관한 많은 연구가 진행되고있다. Akeley와 Jarmoluk[3]는 edge processor를, Kirk와 Voorhies[4]는 span processor를 rasterizer와 더불어 사용하도록 설계하였고 Dunnett등[5]은 삼각형 단위로 입력을 받아들여 span generation을 하지않고 직접 삼각형 내부에 대한 interpolation을 처리할 수 있도록 설계하였다. Fuchs등[6]은 동일한 pixel processor

chip을 여러 개 사용하여 rasterization 성능 및 memory bandwidth를 높이도록 설계하였다.

REjr는 Gouraud shading을 위한 span interpolation과 vector drawing(styled line 포함), area filling, plane masking과 같은 기본적인 기능을 고속으로 수행할 수 있으며 은면 제거를 위한 z-buffering 기능, animation을 위한 double buffering 기능, 12 bit true color를 위한 dithering 기능등에 필요한 logic을 갖추고 있으며 multi-window 환경을 효율적으로 지원 할 수 있도록 window clipping 기능을 제공한다. 그리고 memory control을 위해 DRAM refresh, screen refresh를 수행하도록 설계되었다. REjr는 40MHz로 동작하며 3 cycle에 1 pixel을 처리하게 설계되어 초당 13만 Gouraud shaded polygon(100pixels/polygon)을 그릴 수 있다. REjr는 LSI Logic사의 1.0μ CMOS gate array로 제작되었다.

Ⅱ. 기능 및 구현 방법

1. Gouraud shading

Gouraud shading [1] 은 real time system에서 흔히 사용되는 기법으로, intensity interpolation shading 혹은 color interpolation shading 이라고도 불리운다. Gouraud shading은 polygon 내부의 색상을 선형적으로 변화시킴으로써 이웃한 면과의 광 불연속을 제거한다. 이 방법에서는 면을 지나는 모든 scan line 상의 광도 값을 면 경계에서의 광도로부터 linear interpolation 하여 구한다. REjr에서는 geometry processor로부터 span data를 받아들여 span 단위로 shading을 수행한다. 즉 span 왼쪽 끝점의 R, G, B 값과 △R, △G, △B와 화소의 갯수 N으로부터 각 화소의 R, G, B 값 각각을 병렬로 interpolation한다. 이와함께 은면 제거를 위한 Z interpolation 및 Z check도 병렬로 처리한다. 위의 연산과 함께 memory read/write operation이 interleaving 형식으로 수행되게 함으로써 REjr는 3 cycle (25nsec/cycle)당 1 pixel씩 처리하여 초당 약 13M pixel, 따라서 초당 약 13만 Gouraud shaded polygon(100pixel/polygon)을 처리할 수 있다.

2. Double buffering

Double buffering은 animation시 연속되는 scene이 완전히 그려진 다음 display 함으로써 scene이 그려지는 과정을 보이지 않게 하고 screen clear로 인한 flickering 현상을 제거하기 위한 방법이다. Double buffer는 framebuffer를 하나 더 추가하거나 image plane을 둘로 나누어서 front buffer, back buffer로 구성하여 front buffer와 back buffer는 각각 image update와 display를 교대로 수행하게 한다. Image plane을 나누어서 사용하는 방법은 메모리 사용량을 줄이는 장점이 있으나 image의 quality가 저하되는 단점이 있다. REjr에서는 이를 보완하기 위하여 dithering 기법을 이용하여 12 bit image plane으로 24 bit true color와 거의 비슷한 image를 생성하도록 하였다.

3. Dithering

Image plane으로 표현할 수 있는 색상의 수가 충분하지 못할 경우 표면에 밝거나 어두운 광도의 띠가 생기는 Mach band effect [2] 가 발생하며 dithering은 이를 제거하는 한 방법이다. Dithering [2] 은 광도의 변화가 이루어지는 pixel 사이에서 발생하는 광도의 띠와 같은 변화 부분을 부드럽게 처리해 주기 위하여 dithering noise(random error)를 각 pixel에 random 하게 추가하므로써 이루어 진다. 완벽한 random error의 generation은 구현이 용이치 않고 구성이 복잡하므로 반복되는 checkboard pattern 형태로 recursive하게 정의된 error pattern이 사용된다. 이를 ordered dithering 이라 하며 기본 매트릭스 $D^{(2)}$로부터 식 (1)에의하여 ordered dither matrix $D^{(n)}$ 이 정의된다. $U^{(n)}$은 각 entry가 1인 n x n matrix이다.

$$D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}, \quad D^{(n)} = \begin{bmatrix} 4D^{(n/2)} + D^{(2)}_{00}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{01}U^{(n/2)} \\ 4D^{(n/2)} + D^{(2)}_{10}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{11}U^{(n/2)} \end{bmatrix} \quad (1)$$

REjr에서는 dither matrix 값이 식 (1)과 같이 regular하게 정의됨을 이용하여 matrix의 각 entry를 저장하지 않고 pixel address를 이용하여 직접 구하도록 설계하였다.(특허 출원중) 위의 방법은 table look up 방법에서 필요한 n^2 개의 register 사용을 그림 2의 좌상단 block과 같은 간단한 logic으로 대처하고 table initializing을 하지 않아도 되는 장점

이 있다. 8 x 8 dither matrix를 사용할 경우 X,Y address 각각의 최하위 3 bit로부터 dither matrix entry들은 식 (2)에의하여 구할수 있으며 그림 2의 좌상단 block과 같은 간단한 logic으로 구현된다.

$$D[0]=Y[2];\ d[1]=X[2]\oplus Y[2];\ D[2]=Y[1];$$
$$D[3]=X[1]\oplus Y[1];\ D[4]=Y[0];\ D[5]=X[0]\oplus Y[0]; \quad (2)$$

8 x 8 ordered dither matrix 를 사용한 dithering logic은 그림 2와 같이 구현될수있다. R, G, B 각각의 10 bit color값의 최상위 4 bit는 true value로, 최하위 6 bit 는 dither value로 입력된다. True value는 가산기를 거쳐 '1'이 증가한 값과 bypass 된 값이 multiplexer에 입력되고, 최하위 6 bit는 자신의 pixel address에 의하여 계산된 dither matrix 값과 비교하여 크면 '1'을, 작거나 같으면 '0'을 발생시켜 multiplexer의 선택 신호로 할당한다. 이때 '1'이 할당되면 true value + 1 값이, '0'이 선택되면 bypass 된 값이 최종 color 값으로 선택되어 dithering이 이루어진다. 4 x 4 dither matrix를 사용하려면 X,Y address 각각의 최하위 2 bit를 가지고 D [0] 에서 D [3] 까지를 구하여 8 x 8 dithering과 유사한 방법으로 구현할 수 있다. 실제로 REjr를 사용하여 같은 object에 대한 24 bit true color와 dithering한 12 bit color의 image를 생성하여 비교한 결과 image quality가 거의 비슷하였다.

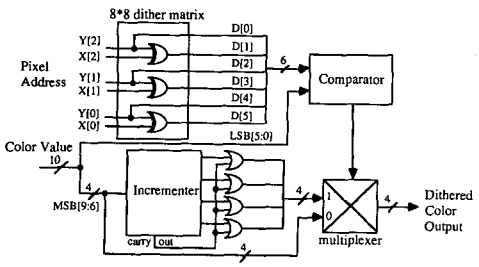


그림 2. ordered dithering unit scheme

4. Z-buffering

3차원 컴퓨터 그래픽스에서는 은선/은면 제거를 위하여 각 화소의 color 값 이외에 Z-depth 값을 Z-buffer에 저장한다. Z-buffering을 수행하기 위하여 REjr는 span 시작점에서의 Z 값과 값을 받아들여 각 화소의 Z값을 R, G, B 값과 동시에 interpola-tion 하여 구한다. 이렇게하여 계산된 Z값과 메모리에서 읽어온 기존의 Z값을 비교하여 pixel image data를 update할 것인지를 결정한다. REjr에서는 이를 위하여 read-modify-write mode를 제공하며 메모리 bandwidth를 높이기 위하여 interleaving 방식으로 수행한다. (Ⅲ.1 절 참조)

5. Window clipping

Window clipping은 크게 두 가지 방법으로 분류되는데 하나는 primitive가 rendering되기 이전에 clipping하여 새로운 primitive들을 형성하는 방법과 rendering 과정에서 화소 단위로 clipping하는 방법이 있다. REjr에서는 후자의 방법을 제공하며 다음과 같이 수행된다. 각 화소에 대하여 window clipping ID buffer에 window들의 priority에 따라 window ID값이 저장되게 하고 REjr에서 생성된 화소의 window ID값과 window clipping ID buffer에서 읽어온 값이 일치할 경우에만 생성된 pixel image data를 update 시키므로써 window의 경계를 벗어나거나 다른 window에 의하여 가려지는 부분을 그리지 않게 한다. [7]

Ⅲ. Architecture

REjr를 테스트하기 위한 전체 시스템의 구성은 그림 3과 같다. i860 을 CPU로 사용한 workstation이 host system으로 geometry processing도 담당하게 된다. Video Controller(VC)와 Video Signal Generator(VSG)는 각각 FPGA로 설계 및 제작하였다. 그리고 RAMDAC으로는 multiwindow 환경을 위하여 pixel 단위로 window type table을 지정할 수 있는 Brooktree사의 Bt463을 사용하였다.

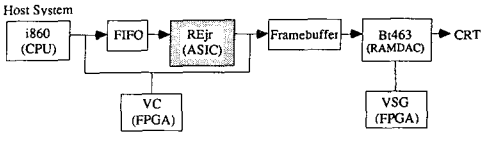


그림 3. REjr test board의 구조

1. REjr 구성

REjr는 그림 4와 같이 12개의 블록으로 구성되어 있다. Geometry processor에서 생성된 instruction은 FIFO를 거쳐 Input Register File(IRF)에 저장된다. IRF는 입력 fetch 시간을 줄이기 위하여 IRF1과 IRF2로 구성되어 있으며 IRF2의 instruction에 의해 REjr가 동작을 하는 동안 IRF1은 다음 입력을 받아들인다.

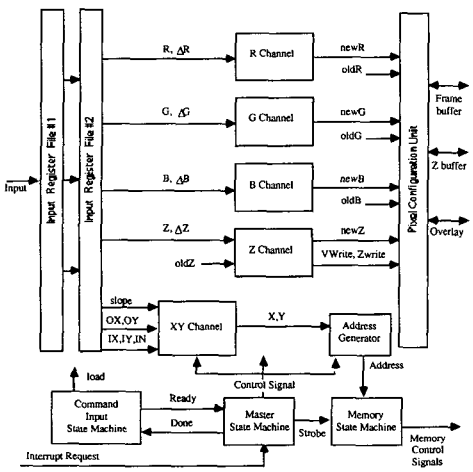


그림 4. Block Diagram of REjr

R, G, B channel은 같은 구조로 만들어져 있으며 초기값 R, G, B에 증가분 △R, △G, △B값을 더함으로써 칼라값을 parallel interpolation한다. Fill instruc-tion이 수행될 때는 R, G, B channel을 거치지 않고 IRF에서 바로 Pixel Configuration Unit(PCU)으로 R, G, B값이 전달된다. Dithering은 R, G, B channel 내에 포함되어 있는 dithering logic에 의해 수행된다.

은면 제거를 위한 Z-depth interpolation, Z check, window clipping을 위한 window ID check는 Z channel에서 수행된다. Window ID buffer와 Z-depth buffer에 저장된 값을 읽어와서 window ID가 일치하고 새로 interpolation된 Z 값이 읽어온 Z 값보다 작은지를 check한다. Z fill instruction을 수행할 때는 Z 값이 Z channel을 bypass하여 PCU로 전달된다. 위의 과정은 framebuffer, Z buffer, window ID buffer로 부터 R, G, B, Z, window ID를 읽어오는 과정, R, G, B, Z interpolation 수행 및 Z, window ID check 과정, 새로운 R, G, B, Z write 과정의 read, modify, write 과정으로 나누어지며 그림 5와 같이 interleaving하여 수행된다.

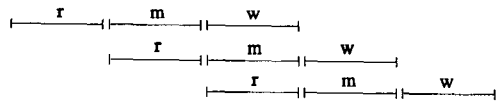


그림 5. REjr Operation Cycle

XY channel은 starting xy address와 화소의 갯수, 기울기를 받아들여 polygon span drawing 또는 vector drawing시에 다음 화소의 xy 좌표를 생성하는 일을 한다. Address Generator는 XY channel로부터 xy 좌표를 받아들여 framebuffer의 Row/Column address를 만든다. Z check와 window ID check에 의해 framebuffer의 update 여부가 결정되고 window type(III.2. 절 참조)에 따라 여러가지 칼라 모드를 가지게 된다. PCU는 이러한 writing logic을 간단하게 하기 위하여 설계된 것으로 Z-depth 값을 읽어오면서 동시에 image 값도 읽어온 다음 window type에 따라 해당하는 bit를 masking함으로써 그림 7과 같은 여러가지 칼라 모드를 지원한다.

REjr에는 세 개의 state machine이 있다. Command Input State Machine은 IRF를 control하며, Master State Machine은 R, G, B, XY channel과 Address Generator를 control하며 DRAM refresh나 screen refresh같은 interrupt를 받아서 처리한다. Memory State Machine은 RAS, CAS, read/write enable등의 framebuffer control 신호를 생성하는 일을 수행한다.

2. Framebuffer Organization

Framebuffer는 그림 6과 같이 각 화소에 대해 image용으로 24bitplane, Z-depth용으로 16bitplane, window clipping ID를 위해 8bitplane, overlay를 위해 4bitplane, window type을 위해 4bitplane을 할당하여 모두 56bitplane으로 구성되어 있다. Image buffer는 direct color mode일 경우, single buffered mode에서는 true color(24bit) image를 저

장하고 flicker free animation을 위한 double buffering mode에서는 front buffer와 back buffer로 각각 12bit씩 나뉘어져서 사용된다. Indexed color mode에서는 front buffer와 back buffer로 각각 8bit를 할당한다. 그림 7은 REjr가 지원하는 칼라 모드에 따른 image buffer의 pixel configuration을 나타낸다.

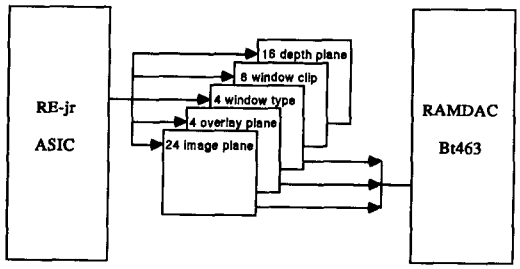
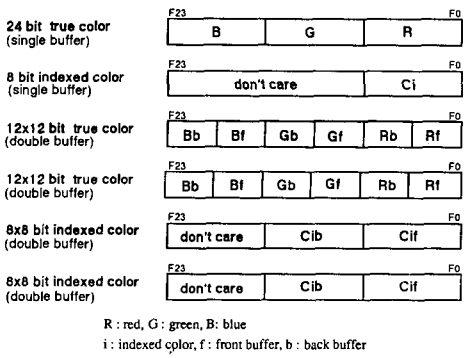


그림 6. Frame Buffer Organization

Z-depth buffer는 은선/은면을 제거하는 Z-buffering algorithm을 수행하기 위해 Z-depth 값을 저장한다. REjr는 각 화소의 image 값을 update할 때 저장된 image의 Z-depth 값을 읽어서 새로 update 하고자 하는 Z-depth 값과 비교하여 새로운 image가 관찰자의 눈에 더 가까이 있을 경우만 image값과 Z-depth값을 update한다.



R:red, G:green, B:blue, i:indexed color, f:front buffer, b:back buffer

그림 7. Pixel Configuration

Window clipping ID buffer는 각 화소가 속해 있는 window의 window ID를 저장한다. REjr에서 생성된 image의 window ID값과 window clipping ID buffer에 저장된 값을 비교하여 일치하면 image data를 update시키고 그렇지않으면 무시해 버린다. Window clipping ID buffer는 8bit로 구성되어 256개의 window까지 지원할 수 있다.

Window type buffer는 각 화소의 칼라 format에 해당하는 window type table의 index를 저장하고 있다. RAMDAC으로 사용되는 Bt463은 각 화소의 window type에 따라 image plane의 내용을 여러가지 칼라 모드로 display할 수 있다. 이러한 기능은 각 window마다 image buffer의 칼라 모드를 다르게 지정하는데 이용할 수 있다. 예를들면, 동시에 한 window는 single buffer 24bit true color mode로쓰고 다른 window는 8bit indexed color mode로 쓸 수 있다.

Overlay buffer는 image buffer에 저장된 image data를 그대로 유지하면서 또다른 image를 겹쳐서 display할 때 이용되는 buffer이다. Overlay buffer는 pop-up menu등의 display시에 사용한다.

Ⅳ. Raster Engine

REjr의 성능 및 기능을 보완하기 위하여 RE에서 고려되는 주요 사항은 geometry processor의 span generation에 필요한 계산량을 줄이기 위한 triangle input 처리 및 memory bandwidth를 높이기 위한 memory interface 부분의 구조개선과 antialiasing, texture mapping, fast Phong shading, alpha blending등의 고급 3차원 그래픽 기능을 추가하는 것이다.

1. Triangle input processing

REjr에서는 처리될 span data(시작점의 R, G, B, Z값과 △R, △G, △B, △Z값)는 geometry processor에서 만들어지는데 geometry processor에서 span data를 만드는데 걸리는 부하가 일반적으로 전체 geometry processing에 걸리는 부하의 절반 정도를 차지한다. 10만 polygon/sec 성능을 기준으로 span generation을 제외한 geometry proces-

sing의 계산량이 약 30MFLOPS인데 반하여 span generation에 필요한 계산량은 약 15MFLOPS가 요구된다. REjr의 성능 테스트 결과, geometry engine (i860)에서 span을 generate 하는 경우 REjr가 약 70%이상 idle 하게 되었다. 삼각형에서는 각 span의 △R, △G, △B, △Z가 같으므로 RE가 triangle 단위로 processing하면 geometry processor의 부하를 줄임과 동시에 RE로 전달되는 data의 량도 줄일 수가 있다. Triangle processing [5]은 삼각형의 한 vertex에서 출발하여 삼각형의 boundary를 check하면서 삼각형의 내부를 search 한다. 이와 동시에 dR(GBZ)/dx, dR(GBZ)/dy 값을 증감하면서 각 화소의 R, G, B, Z 값을 interpolation unit에 의하여 생성한다. Boundary checking 또한 [5]의 방법에 의해 interpolation unit으로 처리된다.

2. Fast Phong shading

Gouraud shading에서는 양 끝점의 색상을 linear interpolation하는 반면에, Phong shading에서는 양 끝점의 surface normal을 interpolation 한 다음 각 화소당 light model을 적용하여 칼라값을 계산한다. 이러한 Phong shading은 복잡한 연산을 동반하여 하드웨어로 처리하기가 어려우며 소프트웨어로 처리할 경우 많은 계산 시간이 요구된다. 예를들면 diffuse reflection의 경우 surface normal vector N의 interpolation과

$$\cos\phi = \frac{L \cdot N}{|L||N|} \tag{3}$$

을 구하기 위하여 각 화소당 7 addition, 6 multiplication, 1 division, 1 square root의 연산이 필요하다.

일반적으로 light model은 근사적으로 계산한 경우에도 image quality가 거의 비슷하게 된다. 이에 따라 Taylor series로 식 (3)을 approximation한 다음 quadratic interpolation을 2nd order forward difference method를 사용하여 각 화소당 2 addition으로 처리하는 방법 [8]이 고안되었다. 위의 방법에서 span 단위로 처리할 경우 setup time이 너무 크고 triangle 단위로 처리할 경우에는 삼각형 내부를 traverse하는 방법에 따라 계산량의 차이가 크다. 예를들면 [5]의 Gouraud shading 방법과 같이 각 화소마다 3방향(좌, 우, 하)의 traversal을 하게되면 $\triangle_x f(x, y)$ 외에도 $\triangle_y f(x, y)$가 각 화소에 대해 필요하게되고 여러 개의 register 및 복잡한 control을 포함하게 된다. RE에서는 triangle 단위로 처리하는 방법을 쓰되 [5]에서의 traversal 방법을 fast Phong shading에 알맞게 수정하여 사용한다. [9]

3. Texture mapping

Object의 표면에 어떤 image를 입히는 것을 texture mapping이라 한다. Texture mapping [10]에는 texture 및 object의 특성과 speed등을 고려한 다양한 방법이 있다. Hardware로 구현할 경우 scan line을 따라서 interpolation 하는 방법이 많이 쓰이고 있다. RE에서는 scan line 방법을 쓰며 quadratic interpolation을 한다. Linear interpolation을 perspective factor가 고려되지 않아 texture가 꺽이거나 굴곡된 형태로 나타 날 수 있다. Interpolation은 fast Phong shading을 위한 quadratic interpolation unit을 공유하여 수행된다.

4. Alpha blending과 Antialiasing

Alpha blending은 image 들을 특정한 비율로 섞어 주는 기능이다. 즉 주어진 color C_1, C_2 및 blending factor α로부터 $C = C_1\alpha + C_2(1-\alpha)$을 구하게 되며 interpolation unit의 일종인 α-channel 에서 수행된다.

Boundary가 기울기를 갖는 image들은 boundary 부분이 resolution의 유한성에 의하여 계단 모양 으로 나타나는데 이러한 현상을 aliasing이라고 한다. 이를 제거하기 위한 방법으로는 resolution을 높이는 효과를 나타내는 super-sampling 방법과 pixel coverage를 이용한 area sampling 방법이 있다. RE에서는 area sampling 방법을 사용하며 α-channel에서 수행 될 수 있다.

V. Implementation

REjr의 제작을 위하여 functional simulation은

Cadence사의 Verilog HDL을 사용하였고 또한 효율적이고 정확한 simulation 위하여 C 언어로 REjr의 아키텍쳐를 모델링하여 원하는 image가 제대로 생성되는지를 검증하였다. 특히 C model을 통하여 칼라값을 계산하는 interpolator의 bit수를 조정함으로써 image의 선명도를 최적화하도록 하였다. ASIC화를 위한 timing simulation과 physical level simulation은 LSI Logic사의 'lsim'과 'lcap'을 사용하였고, state machine과 behavioral model들을 LSI Logic사의 netlist로 변환하기 위하여 LSI Logic사의 logic synthesizer인 'les'를 사용하였다. ASIC 설계후 설계된 ASIC에 대한 검증을 위하여 Verilog HDL을 사용하여 structural model을 작성한후 각 기능별로 image를 생성하여 C model에 의해 생성된 image와 비교해 보았다. REjr은 LSI Logic사의 1.0μ gate array 기술을 이용하여 약 2만 gate로 제작되었고, 40MHz로 동작하며, 초당 13만 개의 polygon을 처리할 수 있다. REjr의 동작 및 성능의 테스트는 video signal generator, video RAM controller, host interface 회로등을 설계하여 제작한 graphic board를 이용하여 수행하였다.

Ⅵ. Conclusion

본 논문에서 기술된 REjr는 polygon Gouraud shading을 위한 span interpolation과 vector drawing, area filling, plane masking과 같은 기본적인 그래픽 기능을 고속으로 수행할 수 있으며, Z-buffering 기능, animation을 위한 double buffering 기능, window clipping 기능을 수행하고, 12 bit true color를 위한 dithering 하드웨어와 line style을 생성할 수 있는 하드웨어를 갖추고 있으며, multi-windowing을 지원할 수 있도록 설계되었다. REjr는 LSI Logic사의 1.0μCMOS gate array로 제작되었다. REjr은 40MHz로 동작하며 parallel interpolation과 fast memory interface 설계로 초당 13만 Gouraud shaded polygon을 그릴 수 있다.

REjr는 첫번째 시제품 제작에서 모든 기능이 정확하게 작동하였다. 이는 REjr의 구조를 C 언어로 모델링한 functional verification 결과와 Verilog HDL을 사용한 functional simulation 결과를 비교하여 function의 정확성을 높이고, Verilog HDL을 사용한 functional simulation 결과와 LSI Logic tool을 사용한 physical simulation 결과를 비교하여 ASIC 설계의 정확도를 높이기 위한 세밀한 검토를 반복적으로 수행한데 있다고 본다.

현재는 REjr의 기본 구조를 바탕으로 antialiasing, texture mapping, fast Phong shading등과 같은 고급 그래픽 기능의 추가와 성능 향상을 위한 triangle input 처리 및 high bandwidth memory interface가 고려된 rasterizer의 연구 및 설계가 진행되고 있다.

參 考 文 獻

[1] H. Gouraud, "Continuous Shading of Curved Surfaces," *IEEE Transactions on Computers*, vol. C-29, no. 6, June 1971, pp. 623-629.

[2] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics : Principles and Practice, Second Edition*, Addison-Wesley, 1990.

[3] K. Akeley and T. Jermoluk, "High Performance Polygon Rendering," *Computer Graphics(SIGGRAPH'88 Proceedings)*, vol. 22, no. 4, Aug. 1988, pp. 239-246.

[4] D. Kirk and D. Voorhies, "The Rendering Architecture of the DN10000 VS," *Computer Graphics(SIGGRAPH'88 Proceedings)*, vol. 24, no. 4, Aug. 1990, pp. 299-307.

[5] G. J. Dunnett, M. White, P. F. Lister and R. L. Grimsdale, "The Image Chip for High Performance 3D Rendering," *Computer Graphics and Applications*, vol. 12, No. 6, nov. 1992, pp. 41-51.

[6] H. Fuchs et al., "Pixel-Planes 5 : A

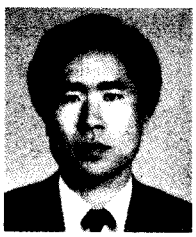
Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memory," Computer Graphics(SIGGRAPH'88 Proceedings), vol. 23, no. 3, Jul. 1989, pp. 79-88.
[7] D. Pinedo, "Window Clipping Methods in Graphics Accelerators," Computer Graphics and Applications, vol. 11, no. 3, May. 1991, pp. 75-84.
[8] G. Bishop, "Fast Phong Shading", Computer Graphics (SIGGRAPH '86 Proceedings), vol. 20, August 1986, pp. 103-106.
[9] Y. C. Wee, "An Implementation of Fast Phong Shading", Technical Report no. C-93-01, Samsung Advanced Institute of Technology.
[10] G. Wolberg, Digital Image Warping, IEEE Computer Society Press, 1990.

筆 者 紹 介

崔 相 吉
1963年 2月 3日生
1985年 부산대학교 전자공학과 공학사 학위 취득
1987年 한국과학기술원 전기 및 전자공학과 공학 석사 학위 취득
1992年 ~ 현재 삼성종합기술원 기반기술 연구소 선임연구원
주관심분야 : Computer Aided Design, Graphics Hardware

崔 乘 均
1959年 4月 14日生
1982年 경북대학교 전자공학과 공학사 학위 취득
1984年 한국과학기술원 전산학과 이학 석사 학위 취득
1984年 2月 ~ 1987年 8月 현대전자산업(주)
1987年 9月 ~ 1988年 10月 LSILogic of KOREA
1988年 11月 ~ 1990年 2月 Valid Logic System, KOREA
1990年 3月 ~ 현재 삼성종합기술원 기반기술 연구소 선임연구원
주관심분야 : Electronics Design Automation, Graphics Hardware

筆 者 紹 介



文 相 鎬
1960年 5月 15日生
1982年 경북대학교 전자공학과 공학사 학위 취득
1984年 경북대학교 전자공학과 공학석사 학위 취득

1983年 10月 ~ 1986年 10月 삼성전자(주)
1986年 11月 ~ 현재 삼성종합기술원 기반기술 연구소 선임연구원

주관심분야 : Graphics Hardware, Computer Architecture

魏 榮 徹
1955年 2月 7日生
1983年 미국 State University of New York at Albany 전산학과 이학사 학위 취득
1985年 미국 State University of New York at Albany 전산학과 이학석사 학위 취득
1992年 미국 State University of New York at Albany 전산학과 이학박사 학위 취득

1989年 ~ 1990年 State University of New York at Albany 연구원
1990年 ~ 현재 삼성종합기술원 기반기술 연구소 선임연구원

주관심분야 : Computational Geometry, Graphics Hardware