IMAGINE

the IMAGe engINE

KEY FEATURES

ULTRA HIGH SPEED GRAPHICS & IMAGE PROCESSOR

50 and 67 MHz single chip processor
C and C++ compilers.
Optimised Assembly Function library supporting Windows & X11.5/ X11.6
Performance at 50 MHz:
up to 250 MIPS 32 bit instructions
up to 6000 MIPS 8 bit instructions
upto 100 MFLOPS (block floating point)
350-450.000 X stones @ 8 bit color
>180.000 X stones @ 32 bit color
200 Mbyte/s video memory bandwidth
200 Mbyte/s data memory bandwidth
64 bit optimised instruction word
10 Internal functional units
8 Internal 32 bit data buses
2 External 32 bit data buses
6W at 5V, on chip PLL
299 pin Ceramic Pin Grid Array

Optimised interface to VRAM/DRAM:
Dual & Quad interleaved fast page mode for high speed vector accesses:
(50MHz: 70ns DRAM, 67MHz: 60ns)
Supports all existing VRAM functions
Specialised multimedia interrupt handler: handles up to 4 different video formats simultaneously
Hardware polygon mask generation: convex and complex, based on the odd/even & winding rules
Hardware shape mask generation
Hardware window clipping
Hardware for 3D graphics Z buffer
Hardware for alpha blending
Hardware for anti-aliasing
Hardware color conversion / dithering:
8:8:8:8 <-> 5:6:5, 5:5:5:1 and 4:4:4:4

The IMAGINE offers a five to twenty fold speed improvement over today's and tomorrow's most powerful processors like the Pentium, MIPS R4400, HP PA-RISC and DEC Alpha for graphics and image processing operations (see the benchmark comparison with a 275 MHz DEC Alpha).

The IMAGINE is designed from start to finish to provide the absolute top performance over a very wide spectrum of graphics and image processing. Extensive support for many functions from elementary BitBlt and window system graphics to 3D graphics, (Color) Postscript, Desk Top Publishing, Image Processing and machine vision is included.

The IMAGINE is designed to allow further integration steps in order to reach the mass consumer markets in the near future (internal caches, synchronous DRAM bus interfaces, video timing).

Several methods are used to reach this level of performance with today's technology:

√ 1 HISC: Hierarchical Instruction Set Computer. The programmers model is hierarchical: The IMAGINE can be programmed as a RISC or CISC processor at the simplest level. This level assures compatibility with Graphic and Image processing libraries written in C code. At the most complex level it can be programmed to operate as a parallel pipeline of functions performing complex operations on parallel streams of data.

√ 2 This hierarchical model is based on an extremely efficient architecture: each functional unit can operate on 32 bit words, two 16 bit words or four 8 bit words in parallel. Eg: the multiplier can do single cycle 32 bit multipication, two 16 bit or four 8 bit multiplications 4x4 8 bit matrix times vector multiplications and quadruple inproducts, 16 bit single cycle 2D dot and cross products, 16 bit single cycle complex multiplications, etcetera.

√ 3 Each functional unit can perform single cycle operations, and all at the same time.

√ 4 Parallel conditional operations allow the implementation of a very wide range of algorithms: up to sixteen decisions can be made each and every cycle.

√ 5 Special hardware generates the outlines of convex and complex polygons and arbitrary scan line shapes, and utilises the Z-buffer comparison results, window borders etc. This feature greatly adds to the high speed while simplifying the programmer's job.

√ 6 A highly optimised processor-memory interface: the processor can use low cost DRAM and VRAM memory while maintaining the very high sustained performance. This is an important feature concerning the overall system manufacturing costs.

T

IMAGINE APPLICATIONS

The IMAGINE processor provides the processing and memory handling capabilities for a compact 5000 MIPS graphics and/or image processing board. Only instruction and data RAM, image VRAM and/or DRAM, a DAC and/or ADC plus some glue logic are needed for a basic system. Important for multimedia systems is the multi video timing option which allows for example the display of live PAL, SECAM or NTSC pictures on a non interlaced high resolution screen.

Examples of Single Processor applications

- **'World record slashing' X-accelerators/terminals**
  The IMAGINE can turn the slow, true color X workstations into a comfortable operating environment with split second reaction times. It will be considerably faster than the current world record for monochrome and pseudo color! The IMAGINE is designed to optimally process release 11.5 and 11.6 applications and is therefore the ideal graphics processor for X, PEX and XIE workstations and terminals.

- **True Color PostScript printers**
  A single true color page in high resolution occupies image memory ranging from 16 Mbyte to 64 Mbyte; sometimes even more. An IMAGINE based color PostScript printer has the capability to process these high resolution, full color images in realtime (a 100 MHz FDDI network interface is advised to keep up with the processor speed).

- **Real Time, Photo Realistic 3D graphics boards**
  For high end IBM-PCs, MacIntoshes and Unix Work Stations.
  An IMAGINE added to e.g. your current Intel i860 design propels the speed to 100,000 triangles/s (true color, 50 pixel, Gouraud Shaded Z -buffered and window clipped).
  It does not only make your design 5 to 6 times faster, it also can bring photo realistic quality to the rendered model by using advanced high quality rendering functions. The IMAGINE will do this faster than the i860 manages to draw plain vanilla Gouraud shaded triangles. Have a look at the benchmark figures for high quality anti-aliased Phong shaded triangles (Ambient, Diffuse and Specular lighting) and the truly perspective, full color texture mapped triangles (high quality anti-aliased, transparent and mixed with Gouraud shading).

- **Press and pre-press Desk Top Publishing boards**
  The power of the million dollar costing Image Processor from the eighties now becomes available for each and every user. The arriving 64 Mbit DRAM generation will provide the capacity to handle full color pages with integrated photographic images. One IMAGINE performs all the jobs done by boards crowded with special purpose hardware and arrays of multiprocessors needed in the eighties. This market is expected to be the biggest single market for computers and graphics and will reach the majority of the middle and even small companies.

- **Machine vision boards**
  Look at your machine vision system: nine out of ten it contains loads of special purpose hardware from different vendors for all kinds of different image processing functions. Now compare the speed of the IMAGINE software with the speed of the special purpose hardware and see how it equals the performance in many cases and, in a number of cases, even surpasses the special purpose hardware in speed. Consider a product in which the IMAGINE alone handles the image processing functions including intelligent Video I/O for all those clients who:
  - are happy with the functionality of your current product but want it to be much smaller and cheaper.
  - have their special image processing functions which turn out to be not supported by your current product.

  The IMAGINE processes data in a way which is easy to understand for all of your customers. It does operations on arbitrary areas in VRAM after data is written there autonomously from the Video source. The restrictions and complications which arise in systems linked closely with the dot clock speed and the frame rate do not exist.

Other examples of Single Processor applications:

- Real Time Multimedia boards
- Medical imaging equipment
- Digital Copiers
- Digital Signal Processing Applications which need large amounts of cheap and fast accessible memory

Multi Processor Options

The IMAGINE software contains functions to allow very simple multi processor systems with an almost linear performance increase. This opens the possibility to use for example four IMAGINE graphics boards to get an application almost four times faster. All the programmer has to do is to modify a few parameters. No additional time has to be invested in special multi processor hardware and software.

All functions in the IMAGINE library are, and will be, equipped with an 'early window test' function in order to minimise the loss of time if a given object has no pixel within the visible window.

Four single processor boards may have their analog video outputs linked together while each processes only a quarter of the screen (the screen should be divided in horizontal bands). Each processor may perform the complete drawing task without a division of the objects to draw. Each processor will be effectively close to four times faster because non visible objects are processed with a speed which is 10 to 30 times higher. The processor which is ready first may start with the band which took the longest time to draw in the previous cycle. This simple load balance mechanism assures a good throughput in unequally divided scenes.

Examples of Multi Processor applications:

- Top end 3D graphics workstations
- HDTV real time image manipulation and special effects systems
- Interactive multimedia authoring systems
- Large multi monitor graphic editing stations
- Ultra high performance Flight Simulators and Virtual Reality Simulators with hundreds of thousands polygons per second (anti-aliased and perspective texture mapped)
- Autonomous land vehicle research
- Combined digital signal and image processing for phased array radar and sonar processing

THE GRAPHICS BENCHMARKS

for a single chip 50 MHz IMAGINE with a 32 bit bus to image memory
(all benchmarks include reading and writing to VRAM/DRAM frame buffer and window clipping!)

Elementary BitBlt Graphics:

	8 bit graphics		32 bit rgb,cmyk	
Rectangle Fill (DRAM)	160	Megapix/s	40	Megapix/s.
Rectangle Fill (VRAM)	400	Megapix/s.	100	Megapix/s.
Rectangle Fill (TRAM) (block size 400^2)	12500	Megapix/s.	3100	Megapix/s.
Rectangle Fill through bitmask image (DRAM)	140	Megapix/s.	35	Megapix/s.
Rectangle Fill through bitmask image (VRAM)	340	Megapix/s.	85	Megapix/s.
Rectangle Copy	75	Megapix/s.	19	Megapix/s.
Rectangle Copy through bitmask image.	64	Megapix/s.	16	Megapix/s.
Rectangle Copy zoom factor 2	125	Megapix/s.	31	Megapix/s.
Rectangle Copy zoom factor 4	150	Megapix/s.	37	Megapix/s.
Logic Operation on two rectangles (C = A op B)	50	Megapix/s.	12.5	Megapix/s.
Add/Subtract two rect with scaling (C = a*A +/- b*B)	50	Megapix/s.	12.5	Megapix/s.
Add/Subtract & saturate two rectangles with scaling	50	Megapix/s.	12.5	Megapix/s.
Minimum/Maximum of two rectangles	50	Megapix/s.	12.5	Megapix/s.
Polygon/Shape Fill (DRAM)	160	Megapix/s.	40	Megapix/s.
Polygon/Shape Fill (VRAM)	400	Megapix/s.	100	Megapix/s.
Polygon/Shape Copy	72	Megapix/s.	18	Megapix/s.
Line draw, any direction	16	Megapix/s.	5	Megapix/s.
Line draw, any direction (VRAM)	20	Megapix/s.	5	Megapix/s.
Line draw, any direction & depth cuing	16	Megapix/s.	5	Megapix/s.
Line draw, any direction & pattern	16	Megapix/s.	5	Megapix/s.

Elementary Window Operations
Comparison between various Graphics processors

Processor:	Memory Bus	Draw Line	Fill Rectangle	Copy Rectangle	Zoom Rectangle
Hitachi GDP	32 bit	0.10 ms	37 ms	112.0 ms	600 ms
S3 86C924	32 bit	0.18 ms	19 ms	45.0 ms	--- ms
TI 34020 40MHz	32 bit	0.22 ms	5.6 ms	44.0 ms	227 ms
WD 90C31	32 bit	1.2 ms	17 ms	62.0 ms	--- ms
Sun 64845 SGX	64 bit	0.12 ms	6.6 ms	25.0 ms	--- ms
IMAGINE 50 MHz	32 bit	0.033 ms	1.25 ms	9.5 ms	3.7 ms
IMAGINE 66 MHz	32 bit	0.025 ms	0.94 ms	7.2 ms	2.8 ms

Notes:	All benchmarks operate on 8 bit colors, the IMAGINE uses VRAM image memory.
Line drawing:	500 pixel arbitrary direction
Rectangle fill:	750,000 pixel rectangle
Rectangle copy:	750,000 pixel rectangle
Rectangle zoom:	7000 to 150,000 pixel rectangle, arbitrary scaling

X Window 11.5 3D graphics operations

(BOARD LEVEL SIMULATION RESULTS)
(display list stored in image memory, IEEE 754 floating point coordinates)

Display list processing:	(------ Single Processor--------).	
Vector drawing (polylines)	**8 bit pixel**	**32 bit rgb,cmyk**
(10 pixel vectors)		
2D vectors (precalculated parameters)	1,200,000/sec	400,000/sec.
2D transformed vectors	600,000/sec.	300,000/sec.
3D transformed vectors	500,000/sec.	250,000/sec.
Gouraud shaded polygons (triangular mesh)	**8 bit pixel**	**32 bit rgb,cmyk**
(50 pixel triangles)		
Gouraud shaded polygons (precalculated parameters)	500,000/sec.	330,000/sec.
3D transformed Gouraud shaded polygons	260,000/sec	220,000/sec.
Z buffered Gouraud shaded polygons	120,000/sec.	100,000/sec.
3D transformed Z buffered Gouraud shaded polygons	110,000/sec.	90,000/sec.

(Actual completion speed for complex scenes can be up to 50% faster because hidden parts are not drawn at all.)

Raw pixel drawing speeds for various 3D shading algorithms

The sustained speeds are reached during the rendering of very large triangles. The speeds of the Z buffered functions lay in a range of a minimum and a maximum speed: (min..max). The minimum speed is reached if all pixels are visible. The maximum speed is reached if all pixels are invisible.

Without Z buffer	Peak rates	Sustained speed
8 bit color Gouraud shading	200 Megapix/s.	160 Megapix/s.
32 bit color Gouraud shading	50 Megapix/s.	40 Megapix/s.

With 16 bit Z buffer and all pixels visible	Peak rates:	Sustained speed
8 bit color Gouraud shading	40 Megapix/s.	32 Megapix/s.
32 bit color Gouraud shading	25 Megapix/s.	20 Megapix/s.
Perspective texture mapping/Gouraud shading (32 bit)	12 Megapix/s.	10 Megapix/s.
Anti-aliased/transparent Gouraud shading (32 bit)	12 Megapix/s.	10 Megapix/s.
Texture mapped/anti-aliased Transp. Gour.Shading (32 bit)	9 Megapix/s.	7 Megapix/s.

With 16 bit Z buffer and all pixels invisible	Peak rates:	Sustained speed
8 bit color Gouraud shading	100 Megapix/s.	80 Megapix/s.
32 bit color Gouraud shading	100 Megapix/s.	80 Megapix/s.
Perspective texture mapping/Gouraud shading (32 bit)	100 Megapix/s.	80 Megapix/s.
Anti-aliased/transparent Gouraud shading (32 bit)	100 Megapix/s.	80 Megapix/s.
Texture mapped/anti-aliased Transp. Gour.Shading (32 bit)	100 Megapix/s.	80 Megapix/s.

Transformed 50 pixel Z buffered, true colored Gouraud shaded polygons
Comparison between various graphics processors

Processor:	Memory Bus width	Polygons/ second
Intel 860	64 bit	17,000
Intel 860 + four Toshiba HSP's	192 bit	35,000
IMAGINE 50 MHz	32 bit	90,000
IMAGINE 66 MHz	32 bit	120,000

THE GRAPHICS BENCHMARKS

for a single chip 50 MHz IMAGINE with a 32 bit bus to image memory
(all benchmarks include reading and writing to VRAM/DRAM frame buffer and window clipping!)

HIGH QUALITY 3D GRAPHICS

Preliminary estimated figures

Phong Shaded Triangles:

Size	Triangles/s.	Pixels/s.	
30	46,500	1.395 Megapix/s.	- 32 bit result: RGBx, CMYK
100	23,200	2.320 Megapix/s.	- True Phong: Ambient, Diffuse & Specular
300	10,000	3.000 Megapix/s.	- Four vector interpolations/pix.(48 bit)
1000	3,450	3.450 Megapix/s.	- Two vector inproducts/pixel (16/24 bit)
3000	1,220	3.660 Megapix/s.	- One table look-up per pixel

Perspective Texture Mapped and Gouraud Shaded Triangles:

Size	Triangles/sec.	Pixels/sec.	
30	61,000	1.830 Megapix/s.	- 32 bit result: RGBx, CMYK
100	38,000	3.800 Megapix/s.	- Correct Perspective Texture Mapping of RGBx
300	20,000	6.000 Megapix/s.	- or CMYK images (sampling method)
1000	8,300	8.300 Megapix/s.	- The texture can be mixed with Gouraud
3000	3,250	9.750 Megapix/s.	- shading to mimic reflective objects

Anti-Aliased & Transparent Gouraud Shaded Triangles:

Size	Triangles/sec.	Pixels/sec.	
30	63,000	1.890 Megapix/s.	- 32 bit result: RGBx, CMYK
100	40,000	4.000 Megapix/s.	- High Quality Anti Aliasing:
300	20,700	6.210 Megapix/s.	- Minimum of 9 levels in all directions, which is 1000
8,550		8.550 Megapix/s.	- the equivalent of 8x8 sub pixel rendering
3000	3,300	9.900 Megapix/s.	- Triangles can be transparent
			- (mixed with background, alpha = 0..255)

Anti-Aliased & Transparent Phong Shaded Triangles:

Size	Triangles/sec.	Pixels/sec.	
30	41,000	1.230 Megapix/s.	- 32 bit result: RGBx or CMYK
100	20,600	2.060 Megapix/s.	- High quality anti-aliasing
300	8,800	2.640 Megapix/s.	- True Phong: Ambient, Diffuse & Specular
1000	3,100	3.100 Megapix/s.	- Four vector interpolations/pix. (48 bit)
3000	1,100	3.300 Megapix/s.	- Two vector inproducts/pixel (16/24 bit)
			- One table look-up/pixel

Anti-Aliased & Transparent, Perspective Texture Mapped & Gouraud Shaded Triangles:

Size	Triangles/sec.	Pixels/sec.	
30	49,500	1.485 Megapix/s.	- 32 bit result: RGBx or CMYK
100	30,300	3.030 Megapix/s.	- High quality anti-aliasing
300	15,000	4.500 Megapix/s.	- Correct perspective texture mapping of RGBx
1000	6,000	6.000 Megapix/s.	- or CMYK images (sampling method)
3000	2,250	6.750 Megapix/s.	- The texture can be mixed with Gouraud
			- shading to mimic reflective objects

All these functions are provided with 'early window tests' to decide if the triangle is likely to have pixels within a given window. This allows multi processor systems with an almost linear performance increase in the range from 2 to 8 processor. The work load is divided on screen region base (screen bands).
Input is a display list of a triangular mesh with transformed IEEE 754 floating point coordinates and normal vectors and in some cases vertex colors and/or texture coordinates.

X-STONE (11.4) BENCHMARK PRIMITIVES

for a single chip 50 MHz IMAGINE with a 32 bit bus to image memory
(all benchmarks include reading and writing to VRAM/DRAM framebuffer and window clipping!)

	8 bit graphics		32 bit rgb,cmyk	
Lines (400)	47000 /sec.		17000	/sec.
Lines (100)	170000	/sec.	62000	/sec.
Dashed lines (400)	17000	/sec.	17000	/sec.
Dashed lines (100)	62000	/sec.	62000	/sec.
Wide lines (400)	31000	/sec.	8000	/sec.
Wide lines (100)	87000	/sec.	31000	/sec.
Rectangles (400^2)	16000	/sec.	5000	/sec.
Rectangles (100^2)	56000	/sec.	18000	/sec.
Filled rectangles (400^2) DRAM	1000	/sec.	250	/sec.
Filled rectangles (400^2) VRAM	2500	/sec.	625	/sec.
Filled rectangles (400^2) TRAM	30000	/sec.	10000	/sec.
Filled rectangles (100^2) DRAM	16000	/sec.	5000	/sec.
Filled rectangles (100^2) VRAM	40000	/sec.	12500	/sec.
Filled rectangles (100^2) TRAM	120000	/sec.	40000	/sec.
Tiled rectangles (400^2)	1000	/sec.	250	/sec.
Tiled rectangles (100^2)	16000	/sec.	4000	/sec.
Stippled rectangles (400^2) DRAM	1000	/sec.	250	/sec.
Stippled rectangles (400^2) VRAM	2500	/sec.	625	/sec.
Stippled rectangles (100^2) DRAM	16000	/sec.	4000	/sec.
Stippled rectangles (100^2) VRAM	40000	/sec.	10000	/sec.
Filled polygons, 5 edge (100) DRAM	8000	/sec.	2500	/sec.
Filled polygons, 5 edge (100) VRAM	20000	/sec.	6000	/sec.
Screen copy (400^2)	500	/sec.	125	/sec.
Screen copy (100^2)	8000	/sec.	2000	/sec.
Window scroll (640x400) DRAM	312	/sec.	78	/sec.
Window scroll (640x400) TRAM	(25000)*	/sec.	6250	/sec.
Bit map copy (400^2) DRAM	780	/sec.	220	/sec.
Bit map copy (400^2) VRAM	12500	/sec.	600	/sec.
Bit map copy (100^2) DRAM	12500	/sec	3600	/sec.
Bit map copy (100^2) VRAM	20000	/sec.	8000	/sec.
Invert rectangle (400^2) DRAM	500	/sec.	125	/sec.
Invert rectangle (400^2) TRAM	12500	/sec.	3100	/sec.
Invert rectangle (100^2) DRAM	75000	/sec.	2000	/sec.
Invert rectangle (100^2) TRAM	44000	/sec.	12500	/sec.

()* : Useful only with multiple of 4 line window scrolls.

X11 graphics example with maximal X11.4 functionality: *a portrait with size 23x39 repeated over a 5 side polygon, XORed with the background and then written through the mask of a bouquet of roses, with writing disabled in bitplanes 3 and 5, and everything clipped on the window borders:*

	8 bit graphics	32 bit rgb,cmyk
single chip 50 MHz I[M]AGINE:	42 Megapixels/sec.	12.5 Megapixels/sec.

THE GRAPHICS BENCHMARKS, continued

for a single chip 50 MHz IMAGINE with a 32 bit bus to image memory
(all benchmarks include reading and writing to VRAM/DRAM frame buffer and window clipping!)

Image manipulation operations:

Image merging (RGB, CMYK):	10	Megapixels/sec.
Smooth merging of two images with the use of an alpha buffer and transparency.		

Interpolated affine transformations:
Affine transformation: image rotation, scaling, skew and translation with the use of sub pixel accuracy.

Bi-linear interpolated affine transformation:	25	Megapixs/sec. (8 bit)
Bi-linear interpolated X and Y scaling only:	37	Megapixs/sec. (8 bit)
Bi-cubic interpolated affine transformation:	10	Megapixs/sec. (8 bit)

Affine transformation on RGB or CMYK images
(Increasing quality, image stored in Data Ram)

Sampled transformation	37	Megapixels/sec.
Bi-sampled transformation (sub-sample filter)	18	Megapixels/sec.
Quad-Sampled Transformation (sub-sample filter)	9	Megapixels/sec.
Bi-linear interpolated transformation	5	Megapixels/sec.
Bi-cubic interpolated transformation	1.2	Megapixels/sec.

Image manipulation using data memory:

Nearest neighbour sampling and quasi random displacement sampling, (RGB and CMYK):

Affine transformation:	40	Megapixels/sec.
Third order image warp:	12.5	Megapixels/sec.
16 term parametrised function for both X and Y:		
Free warp with pre-calculated coordinates:	20	Megapixels/sec.
Maps images in an arbitrary way		

Curve drawing functions: (to DRAM/VRAM)

Circle, ellipse: (RGB, CMYK)	6	Megapixels/sec.
Cubic curves: (RGB,CMYK)	6	Megapixels/sec.

Color space conversions

conversion RGB to YUV	20	Megapixels/sec.
conversion YUV to RGB	20	Megapixels/sec.
conversion RGB to HIS	12.5	Megapixels/sec. *)
conversion HIS to RGB	10	Megapixels/sec. *)
conversion CMYK to RGB (tri-linear interp.)	2.2	Megapixels/sec.
conversion RGB to CMYK (tri-linear interp.)	2.2	Megapixels/sec.

(* with 128 kbyte table in data memory)

Postscript functions

Halftoning: Rendering a color A4 sheet of paper (CMYK) with 600 dots per inch for about 80% of its surface: the input is a color photographical image of high resolution in the printing colors. The output are four rotated rasters of variable sized dots in cyan, magenta, yellow and black ink: **ca. 0.800 seconds.**

THE IMAGE PROCESSING BENCHMARKS

for a single chip 50 MHz IMAGINE with a 32 bit bus to image memory

(all benchmarks include reading and writing to VRAM/DRAM frame buffer and window clipping!)

Image Processing Benchmarks

8 bit image processing:		
3x3 convolution:	37	Megapixels/s.
4x4 convolution:	25	Megapixels/s.
8x8 convolution/correlation:	8.7	Megapixels/s.
Grey scale histogram table generation:	160	Megapixels/s.
Grey scale table look-up (small table <= 32):	160	Megapixels/s.
Grey scale table look-up (memory based small table):	40	Megapixels/s.
Grey scale table look-up (memory based large table):	80	Megapixels/s.
Sobel edge detection (r only, abs value)	15	Megapixels/s.
Sobel edge detection (r and phi with table):	10.5	Megapixels/s.
Robinson edge detection (r and phi):	6.2	Megapixels/s.
Median filter (5x5 window):	1.85	Megapixels/s.
Minimum rank filter (3x3) window:	15	Megapixels/s.
Maximum rank filter (3x3) window:	15	Megapixels/s.
Discrete cosine transformation (8x8):	33	Megapixels/s.
Discrete cosine transformation (16x16):	17	Megapixels/s.
16 bit high precision image:		
3x3 convolution	8	Megapixels/s.
16 bit grey scale histogram table generation:	80	Megapixels/s.
16 bit grey scale memory based table look up:	40	Megapixels/s.
16 bit median filter (5x5 window):	1.0	Megapixels/s.
16 bit min/max filter (3x3) window:	7.5	Megapixels/s.
Fast Fourier Transform (512x512 16 bit)	<100	msec.
Color Image processing (RGBa, CMYK, etc.)		
3x3 convolution	9	Megapixels/s.
3D or 4D color space range test	20	Megapixels/s.
Binary Image processing		
Fill binary image	1250	Megapixels/s.
Copy binary image (arbitrary translation)	625	Megapixels/s.
Logical operation on two images	425	Megapixels/s.
Binary erosion (8 or 4 connected)	125	Megapixels/s.
Binary dilation (8 or 4 connected)	125	Megapixels/s.
Binary contour (8 or 4 connected)	125	Megapixels/s.
Binary salt & pepper filter (8 or 4 conn.)	125	Megapixels/s.
Binary skeleton	25	Megapixels/s.
Binary skeleton plus end & single point filter	25	Megapixels/s.
Feature extraction (hit_miss)	125	Megapixels/s.
Voxel Processing		
Extraction of an arbitrary 2D plane from a 256x256x256 Voxel volume 8 bit	40	Megapixels/s.

Various DSP tasks:

1024 point complex FFT, 16 bit fixed point	400	microseconds.
256 tap complex	200	KHz.
3x3 tap FIR (8 bit)	37	MHz.
3x3 tap FIR (16 bit)	8	MHz.
10x10 matrix multiply (32 bit)	2,2	microseconds.
Complex integer multiply (16 bit)	20	nanoseconds.

8 Bit Finite Impulse Response filters:	**nr. taps**	**sample speed**
Image memory VRAM/DRAM -> output or data memory		
	4	128 MHz.
	8	64 MHz.
	16	32 MHz.
	32	16 MHz.
Image Memory VRAM/DRAM -> image memory VRAM/DRAM		
	4	64 MHz.
	8	32 MHz.
	16	16 MHz.

DIRECT CONFRONTATION:

27.5 MHz DEC Alpha against 75 MHz IMAGINE

Desk Top Publishing Benchmarks:

A 'page' is a standard sheet with 600 dpi and contains circa 50 million pixels
A 'display' has a resolution of 1280x1024 with 32 bit colors and contains circa 1.25 million pixels

		27.5 MHz Alpha	75 MHz IMAGINE
RGBa Image Mixing:	pixels:	2 Megapixels/s.	15 Megapixels/s.
(with alpha blending)	display:	1.5 Hertz refresh	12 Hertz refresh
	pages:	0.04 pages/sec.	0.3 pages/sec.
RGBa Image Filtering:	pixels:	0.7 Megapixels/s.	12 Megapixels/s.
(3x3 convolution with overflow	display:	0.55 Hertz refresh	10 Hertz refresh
and underflow saturation)	pages:	0.015 pages/sec.	0.25 pages/sec.
RGBa Image Rotation / Scaling:	pixels:	0.8 Megapixels/s.	7.5 Megapixels/s.
(bi-linear interpolation)	display:	0.65 Hertz refresh	6 Hertz refresh
	pages:	0.015 pages/sec.	0.12 pages/sec.

3D Graphics Benchmarks:

(Triangular mesh of 50 pixel, 3D transformed, lighted, Z-buffered, Gouraud shaded polygons with 24 bit color)

	27.5 MHz DEC Alpha	75 MHz IMAGINE
Gouraud Shaded Polygons:	ca. 70.000 polygons /sec.	ca. 135.000 polygons /sec.

Medical System Image Processing Benchmarks:

A 'display' has a resolution of 2048 x 1536 with 8 bit grey values and contains 3.2 million pixels.

		27.5 MHz Alpha	75 MHz IMAGINE
Image Scaling and Rotation:	pixels:	4 Megapixels/s.	37 Megapixels/ sec.
(8 bit bi-linear interpolation)	display:	1.25 Hertz refresh	11.5 Hertz refresh
Exact Image Scaling/Rotation:	pixels:	1.5 Megapixels/s.	15 Megapixels/ sec.
(8 bit bi-qubic interpolation)	display:	0.45 Hertz refresh	4.5 Hertz refresh
Grey Scale Histogramming:	pixels:	15 Megapixels/s.	80 Megapixels/ sec.
(16 bit / 32 levels)	display:	4.7 Hertz refresh	25 Hertz refresh
Grey Scale Equalisation:	pixels: ca	15 Megapixels/s.	40 Megapixels/ sec.
(16 bit table look-up)	display:ca	4.7 Hertz refresh	12 Hertz refresh

Chapter 2

THE ARCHITECTURE

The basic HISC principles which have guided the design of the IMAGINE are introduced in this chapter. The hierarchical instruction computing model brings a quantum leap in performance improvement with its unequalled level of efficiency. The CISC, RISC, HISC... story describes the basic techniques and background of the HISC architecture.

The functional units of the IMAGINE are introduced one by one in a tour through the chip's architecture.

The Hierarchical Instruction Set Computer (HISC) Principle

CISC, RISC, HISC ....

10 times the Efficiency = 10 times the Performance

The HISC principle has been developed by Arcobel Graphics B.V. to tackle the issue of efficiency and thus of performance of application specific processors. For a wide range of graphics and image processing functions an increase of efficiency in excess of 1000% can be achieved compared with the fastest available RISC and CISC processors.

HISC recognises the fact that performance and efficiency are inextricably linked and that a lack of performance is essentially a lack of efficiency. It offers a set of principles which dramatically improve the efficiency and thus the performance of the processor.

The implementation of HISC principles uses advanced and novel arithmetic hardware design techniques to combine a "faster than RISC" processor with a very wide range of ultra high speed graphics and image processing functionality. The compatibility of HISC with super-pipelining and super-scalar design techniques will ensure leading edge performance levels for many years.

In retrospect it has become apparent that, in reality, the efficiency of general purpose processors has decreased by a factor of 10 in the last 15 years. To illustrate this point consider the dominant family of Complex Instruction Set Computers (CISC) processors over the last 20 years, the Intel 80XXX family.

In 1974, when Intel introduced the 8080 processor, some 5000 transistors were integrated into the device. By 1993, and the launch of the Pentium processor, this figure had rocketed to over
3 million. That is 600 times more than its predecessor. However, not only did the gate count increase dramatically, but so also did the clock frequency, which multiplied by a factor of around 33.

If one ignores the internal usage of the transistors it would be reasonable to expect (though perhaps somewhat naively) a performance improvement of around 20000 (600x33). In reality however, the actual performance improvement (as bench-marked) over the 19 year period, is only in the order of several hundred, not twenty thousand times. Why? Because the main obstacle in fully exploiting increasing hardware densities, shrinking geometries and increasing gate counts,lies in making the most efficient use of these available hardware resources. Then what about the Reduced Instruction Set Computer (RISC)?

A (still growing) set of design techniques is embodied by the RISC concept. One of the original RISC goals of achieving single cycle operations was a big step forward towards more efficient hardware use - the Arithmetic Logic Unit (ALU) could be activated every cycle instead of once every three to six cycles.
A logical development of this technique is that of super-pipelining, for which the same logic can be used two or more times by incorporating intermediate pipeline registers. The first part of the logic can start a new operation whilst the rest is still finishing the previous operation(s).

The RISC concept is therefore based on using as few instructions as possible. The idea behind this is that it will enable the fastest hardware and thus the fastest processors. However, many of the most useful instructions are deliberately omitted because this would make the hardware too complex and therefore too slow. This principle has been shown to be erroneous during the initial design stages of the IMAGINE (the device which will become the tangible implementation of the HISC principle). Hardware efficiency presents almost no problems for special purpose hardware since it is designed to perform a single or a few closely related tasks. Good examples of this type of hardware are image processing and compression/ decompression chips which can reach speeds of billions of operations per second (BOPS) easily.

If, however, a more general set of operations has to be performed, devices have to be added for each and every operation; the efficiency dilemma strikes back in another way. Dedicated special purpose hardware is only truly effective in situations which require **limited functionality**. Typical special purpose hardware is 25 to 100 times faster than general purpose processors with as many or less transistors, depending on the type of operation being performed. This means that a general purpose processor executes graphics and image processing functions with a relative efficiency of only 1% to 4%. In other words, the transistors in the device are only used 1% to 4% of the time or, when they are used, 96% to 99% of the time they are used "in the wrong way". Although it would be unfair to take this statement too literally, it does highlight the fact that there is

considerable scope for the development of innovative hardware design techniques, which can produce spectacular performance gains.

Hierarchical levels

The HISC approach starts at the level of the functional units which are embodied in every RISC and CISC processor. These represent the most basic programming level and at this level compatibility with standard processor design, languages and operating systems can be found. A complete set of basic units is provided at this level and will certainly include an arithmetic logic unit, a barrel shifter and a multiplier/accumulator.
However, although residing at the lowest programming level, these functional units are formed from sub-units, these sub-units from other sub-units, and so on, down to transistor level. At these sub-unit levels techniques can be applied to make most efficient use of the hardware, with a minimum overhead in terms of additional hardware (i.e. transistors).

As mentioned above, The design rule associated with the RISC concept of omitting a large number of instructions has been found to be erroneous during initial design of the IMAGINE.
33% Faster cycle times have been achieved then some of the mainstream RISC processors, like the SUN 2 SPARC and the non-superpipelined MIPS R3000 (after cancellation process parameters of these somewhat older designs). It has become apparent that the techniques developed have enabled the production of faster functional units, in spite of their much richer instruction set.

In order to better understand how this improvement has been achieved, an overview of some of the used techniques is presented below, together with some details on how they can be implemented in a general purpose imaging and graphics processor.

Wordlength partitioning

A good example of low efficiency usage is when operations are performed on short wordlength operands (8 or 16 bit) by 32 bit functional units. A 32 bit processor is not faster when handling 8 bit operations, even though only a proportion of the hardware is utilised. This inability of general purpose processors to deal efficiently with short wordlenghts is one of the key reasons for the performance gap between special purpose and general purpose hardware. The hardware incorporated in a typical 32 bit **ALU** or **barrel shifter** could, if the transistor elements would have been re-arranged and extra control logic would have been added, perform four 8 bit operations or two 16 bit operations per cycle. This efficiency increase would be of a linear nature.

However, a 32 bit **multiplier** requires approximately 16 times as many transistors than an 8 bit multiplier. Consequently, performing four 8 bit multiplications in parallel would only utilize some 25% of the available gates. Using the internal Wallace tree and intelligent control logic, the 32 bit multiplier could perform sixteen 8 bit multiplications and twelve 8 bit additions in a single cycle. These operations can represent matrix x vector multiplications (specifically 4x4 matrices) or quadruple 4x1 inproducts. Functions of this type are particularly useful in both graphics and image processing.

A conventional 32 bit multiplier thus contains almost all the logic required to perform twenty-eight 8 bit operations instead of only one. In effect, we may conclude that something like 96% of the hardware is left unused if a 32 bit multiplier is used for 8 bit multiplications.

In the IMAGINE a 32 bit word can represent a single 32 bit word, two 16 bit words or four 8 bit words. All the functional sub-units can perform SIMD type operations on these parallel data types. The multiplier has internal data and co-efficient pipelines to supply the operands for matrix x vector operations. The ALU can generate four 8 bit based status flags or two 16 bit based status flags. The internal 32 bit register file can be accessed for independent 8 bit and 16 bit words. Conditional accessing and write enabling are possible on an 8 bit and 16 bit basis. The efficiency gain possible by wordlenght partitioning is exploited to the full by the IMAGINE in a way which is optimised for graphics and image processing.

Heterogeneous partitioning

A conventional device has several sections each with its own functionality, for example the ALU, the barrel shifter, the multiplier/accumulator etc. Only one of these sections is used per operation, while the other ones stay idle. Many functions, however, can be mapped on a model in which these sections are separated into distinct and independent functional units. Each functional unit has its own output bus. The inputs to each functional unit are provided by multiplexers which are capable of selecting the input from other functional units. The result from each unit is stored into a register which drives the output bus belonging to that specific unit.

Concatenation of functional units which enables multiple instruction per cycle is especially effective for vector type operations.

The IMAGINE has eight internal buses and eight internal functional units. The functionality and interconnectivity provided are the result of analyzing a very broad range of graphics and image processing functions. Each unit is represented by its own, relatively small, field in the 64 bit instruction word which encodes the basic instruction for that specific unit.

This means that all the units can operate in parallel which, in effect, makes the instruction word a "moderate sized" Very Long Instruction Word (VLIW). This level can be seen as the second programming level, with the first and simplest, being the RISC level. Newer optimising compilers which have sufficient data dependency analysis capabilities, can exploit these to generate faster and more efficient code.

Heterogenous Vector/Stream operations

Processing vectors or streams of data mean that an instruction is repeated a number of times. Typically this will range from 8 to 32 times in continuous bursts, up to several million times in repeated bursts. In this situation there is no need for the instruction to be supplied on each and every cycle.

The IMAGINE will be equipped with more than 600 bits devoted to extended instructions which are stored in control registers located within the various functional units. The basic 64 bit instruction word can select extended functions which use information stored in these control registers. The **actual** instruction word length for these extended operations is thus much longer.

This level can be viewed as the third and most complex programming level. It turns the ineffective functional unit found in standard RISC and CISC processors into an ultra high speed heterogenous multi-vector processor that can perform intelligent conditional operations on parallel streams of data.

Parallel Conditional Processing (General and Application Specific)

It is clear that the most practical ways of obtaining optimum efficiency from arithmetic hardware leads to SIMD and vector type operations. In graphics and image processing terms these can be translated to blocks of pixels which are processed with identical instructions. The pixel is no longer treated as an individual (i.e. point operation) but as an element in a group, upon which certain operations are performed. In many cases however, it is necessary to handle individual pixels without loosing the inherent parallelism provided by this approach.

It is essential to be able to perform if-then-else type operations in a parallel way. For SIMD and vector processing type operations, the program control flow is identical for all pixels. This means that typical conditional control flow, with conditional program jumps and calls, cannot be used.

However, HISC can use parallel conditional data flow instead of serial conditional control flow and considerably enhance the flexibility of the functional units. Many more algorithms can thus be implemented in high speed parallel versions. A general type of parallel conditional processing is implemented within the address generator of the three port register file. Up to sixteen parallel conditional data flow operations can be performed and twelve register addresses can be calculated with conditional offsets and increments. Four conditional write enables are generated each cycle, depending of parallel status information.

Application level parallel conditional processing is used to support a number of algorithms which are typical for many graphics operations. Special hardware is included to generate two-dimensional masks which determine if pixels are inside or outside lines, polygons or other arbitrary shapes.

Functional Completeness

When dealing with low-level efficiency gains, small details become extremely important in sustaining high efficiency levels under many different circumstances. If the basic efficiency level is high, then functional completeness becomes of critical importance.

For example:

The C commands P=A<<B and P=A>>B use the barrel shifter available in almost all of the newer RISC processors. Doing so the command can be executed in a single cycle. In C the operand B can be both positive and negative - when it is negative "shift left" becomes "shift right" and vice versa.

However, popular processors (SPARC, MIPS...) have "copied" the shift left and shift right operations from earlier CISC processors, where B is always positive. Consequently the C compiler has no option but to insert extra code to check the sign of B, perform a conditional branch and then carry out one of the two shift instructions.

Despite the larger number of transistors used to integrate a barrel shifter, the omission of a few extra gates to check the sign of B unfortunately causes the efficiency for this type of operation to drop to around 25%.

Although these extra instructions have relatively little impact on CISC processors (which needed up to 32+ cycles merely for the shifting operation) they cripple the much more efficient RISC processor.

To make matters worse both the SPARC and the MIPS processors only look at the five least significant bits of the B operand in order to determine the number of positions to shift (the 8086 microcode keeps on shifting for thousends of cycles if B is large). This implies, however, that a shift over 35 positions has the same end result as a shift over only 3 positions. This also conflicts with the definition of the C shift functions and the compiler, yet again, has to add extra code to check if operand B is out of range. This obviously compounds the problem and as a result, the efficiency level now drops below 10%. This means that the processor with a barrel shifter is only 2 to 3 times faster than a processor without one.

It is obviously very difficult to predict exactly how hardware will be used in practice and to provide capabilities to address all possible problems. However, by consistently applying the general principle of functional completeness, much can be done to improve efficiency at this level. Thus in the IMAGINE, the barrel shifter will be capable of shifting by a range-tested 2's complement operand.

Completeness is essential in multiplicative operations and so the multiplier in the IMAGINE can orthogonally perform signed, unsigned and mixed mode multiplications for all word sizes and modes. Furthermore words can be interpreted as integers, fixed point and normalised fixed point numbers. All these cases appear frequently in graphics and image processing functions.
(The number of basic multiplications is 786!)

In order to achieve functional completeness, it is sometimes necessary to sacrifice pure mathematical integrity in order to produce a product which will operate satisfactorily over a wide range of functions. For example, a typical mathematical inconsistency can be found in many international graphics and image processing standards, where normalised numbers lie in the range of N = 0.0 to 1.0 (including N = 1.0) and where the numbers are represented by unsigned fixed point numbers in the range of 0 to 255. In this case there are 256 discrete values but the maximum value which may be represented is effectively 255/256 (i.e. less than 1).

Therefore multiplying a value N by the nearest approximation to 1 (255/256) will result in an erroneous value.

Taking the example further, a pixel's transparency value can be represented by an 8 bit unsigned number in the range 0 to 255. Thus 0.0 is (correctly) represented by 0, but 1 will be represented by 255 instead of by 256. this means that 0.11111111 times 0.nnnnnnnn, which should always be equal to 0.nnnnnnnn, will in fact be equal to 255/256 times 0.nnnnnnnn (i.e. 0.99609370 times 0.nnnnnnnn).

Repeated operations in which such differences are neglected will show visible errors. A good example is the fading of the background of a picture scene constructed with high quality alpha plane merging.

Since we cannot change standards to be mathematically consistent it is often necessary to add some "non-mathematical" compensation. The IMAGINE multiplier employs user selectable rounding logic to deal with this kind of effect.

Conclusions

The HISC principle recognises that the lack of performance of CISC and RISC processors compared to special purpose hardware, is essentially a lack of efficiency. It specifies a set of design principles such as wordlength partitioning, heterogenous partitioning and stream processing which can potentially increase performance by a factor of 15 to 35 times for a number of functions. In order to broaden the range of functions which can be implemented, HISC also makes use of the principles of parallel conditional processing and functional completeness.

The IMAGINE is the first processor based extensively on HISC principles and will result in multi-functional arithmetic hardware units which are capable of supporting many different functions, without incurring the performance degradation associated with RISC. In fact design testing shows that IMAGINE provides faster functional units than the leading RISC processors, while retaining the same process technology.

The HISC concept is compatible with super pipelined and super scalar design techniques which it can fully exploit for its own purposes and which will ensure a competitive edge for many years to come.