ImaginE — The IMAGe engINE

Hans de Vries
Arcobel Graphics BV.
Hambakenwetering 1,
5231 DD 's Hertogenbosch
the Netherlands
Tel: +31 73 444144
Fax: +31 73 444150
Email Business contacts: std@arcobel.nl
Email Author: hdv@arcobel.nl

ImaginE: Bringing high end DTP and 3D graphics to the millions.

The tremendous computing power needed for the interactive processing and generation of visual information is more and more becoming a major technology driver in the consumer market. Illustrative are the benchmarks used by MacWorld magazine recently at the introduction of the new generation of Power PC MACs (a major commercial event) used to compare the processing power of the Power PC with the Intel Pentium and 486: Almost all of them where DTP image processing benchmarks. The majority of the word processor users (estimated at over 100 million people world wide) will evolve to DTP software in the coming years, following the line of ever improving (color) printers and scanners at steadily eroding prices. The use of 3D graphics is quietly growing among professional users like Architects and Engineers. 3D graphics will explode when accelerators will reach price levels low enough for the game industry and will be one of the main ingredients for future multimedia platforms.

Our studies of how the modern general purpose processors handles typical image processing and graphics operations reveal a remarkable lack of efficiency. The actual effective use of transistors lays somewhere between 1% and 4%. This efficiency gap is the main reason for a continuing stream of special purpose integrated circuits. Almost daily new ASICS are developed for filtering, color conversion, compression, raster, screening, alpha blending, affine transformation, 3D rendering etc. These special purpose devices often outperform their general purpose counterparts by a factor of 25 or more, in general with a lot less transistors. These devices are able to reach near 100% efficiency levels.

Arcobel Graphics has devoted the past 3 years on closing this efficiency gap. Based on many years of experience in building high end graphics and image processing accelerators and the knowledge of high end ASIC design it has produced a processor which can execute general purpose C code at a comparable speed but also can replace all of the mentioned special purpose ASICS with a performance which is equal or higher in many cases. It's first generation design (Figure 1), named The IMAGINE (the IMAGe engINE) — available in 50, 67 and 75 Mhz versions — now beats the Power PC MACs with a factor of 50 for almost all DTP image processing functions including all the ones used in MacWorld.

The set of design principles used to reach these speeds (merely the result of efficiency) is referred to as HISC for Hierarchical Instruction Set Computer, much in the same way as RISC was a set of design principles to improve on the processor generation of ten years ago. Hierarchical Instruction Sets allow the programmer to delve deeper and deeper in the available processing hardware in order to improve the efficiency of the available transistors.

The Research and Design phase of the project was however not Processor technology driven but Algorithm driven. The starting point where the building blocks of a typical RISC processor, minimal special purpose hardware was allowed. This gives you the basic ingredients like an ALU, a register file, a barrel shifter, a multiplier and Bus interface units. The internal configuration and the overall composition of these units should adhere to two apparently distinctive worlds: The General Purpose processor, optimised to execute C programs at a comparable speed but it also should be able to mimic a wide

range of special purpose hardware. The configuration of an elementary RISC processor is well defined, but what about all the graphic and image processing algorithms.... Here Arcobel Graphics could build on many years of experience in building special purpose processors based on massively parallel bit-slices and dataflow signal processors and implementing algorithms on these machines.

The Press and PrePress industry has historically been somewhat mysterious, unlike the automotive, petrol or computer industry. It's major players are virtually not known by the general public which has hardly an idea of all the processing steps which are involved in producing the magazines and newspapers which they consume in such big numbers. The first printing machines in the western world where designed some 550 years ago, all in a 250 mile radius from the birthplace of the Arcobel graphics IMAGINE processor by Laurents Janszoon Coster at Haarlem (The Netherlands) and Johann Gutenberg at Mainz (Germany).

If you extent this radius to 500 miles today then the Press and PrePress companies within that region add up to a worldwide number one industrial force in this market segment. This industry has produced many excellent high end graphics computers and special purpose hardware purely for it's own usage. Sometimes with $1,000,000+ price tickets, used by specialists, and known only to industries insiders.

It is hardly surprising that the IMAGINE found it's birthplace in the centre of this area with it's historical industrial reputation kept over half a millennium. The invention of the press more then 500 years ago brought information printed in text and illustrations to millions of individuals. A processor like the IMAGINE turns consumers into producers as did the typewriter and the word processor for text and linework. The IMAGINE enables high resolution photographic (color) illustrations (often containing 32 Mbyte or more per page) to be handled interactively. Editing operations like rotation, scaling, filtering, brush based blending, geometric transformation are all handled in real time as well as printing preparation functions like half toning or error diffusion based rasterisation and RGB to CMYK conversion.

3D perspective texture mapping is another of the IMAGINE 's talents. Texture mapping brings the reality to Virtual Reality applications and games. New Intel/Microsoft standards provide the interface layers between the IMAGINE and 3D standards like Open GL and Hoops and Chicago based 3D texture mapped games, allowing the IMAGINE to be used as a Plug in and Run performance booster.

The successor(s) of the IMAGINE will focus on board level integration and cost engineering to reach price levels which are within everybody's budget. It can do so because of it's inherent efficiency which provides the necessary processing power with a minimal amount of transistors (650,000), considerably lessthan current market leaders like the Pentium and the PowerPC. Key to this efficiency is the set of HISC principles which are explained in more detail below. The small transistor count also enables ultra high performance parallel versions by quadrupling the functional units running at an increased clock speed of 133 MHz to 150 MHz at the first half of 1996. Such a processor provides an astonishing 100 billion 8 bit operations (100,000,000,000) per second. It can run C programs 4 times faster as the current version by using super scalar techniques and runs graphics and image processing functions 10 times faster by up scaling the SIMD word size and vector length.

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 (600×33). 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. Special purpose hardware is typically 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 (Figure 2). 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 for the functional units than those found in a number of some RISC processors which used a comparable process. 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-vector multiplications (specifically 4×4 matrices) or quadruple 4×1 products. 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 (Figure 4) has internal data and co-efficient pipelines to supply the operands for matrix × 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 wordlength 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.

Heterogeneous 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 heterogeneous 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 on 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 is of critical importance.

For example: The C commands P=A≪B and P=A*gg*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 (Figure 4). 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 modes 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 0.11111111×0.nnnnnnnnn, which should always be equal to 0.nnnnnnnn, will in fact be equal to 255/256×0.nnnnnnnnn (i.e. 0.99609370×0.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 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 ef-

ficiency. It specifies a set of design principles such as wordlength partitioning, heterogeneous 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 purpose which will ensure a competitive edge for many years to come.

0

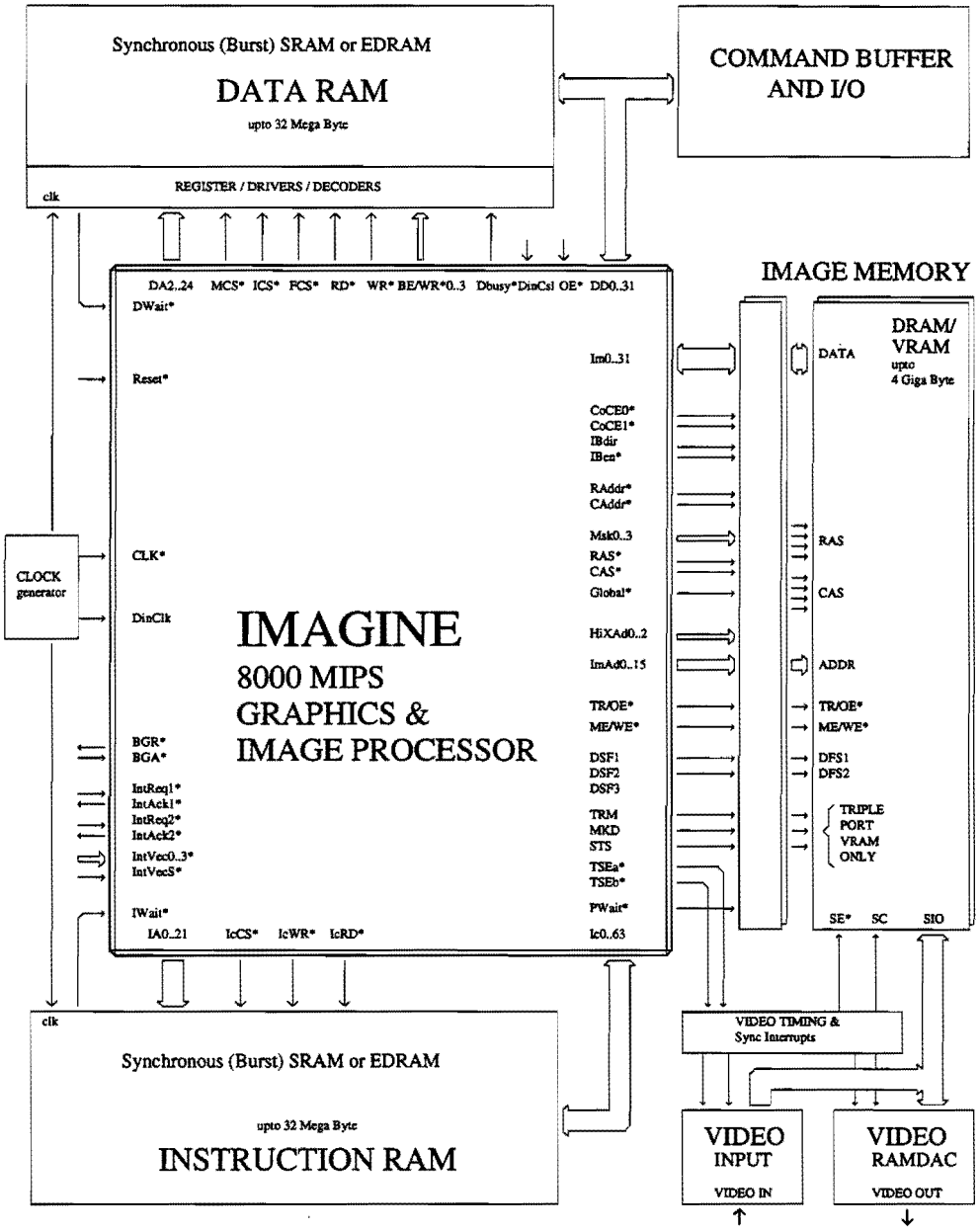


Figure 1: System Concept: IMAGINE plus external memories.

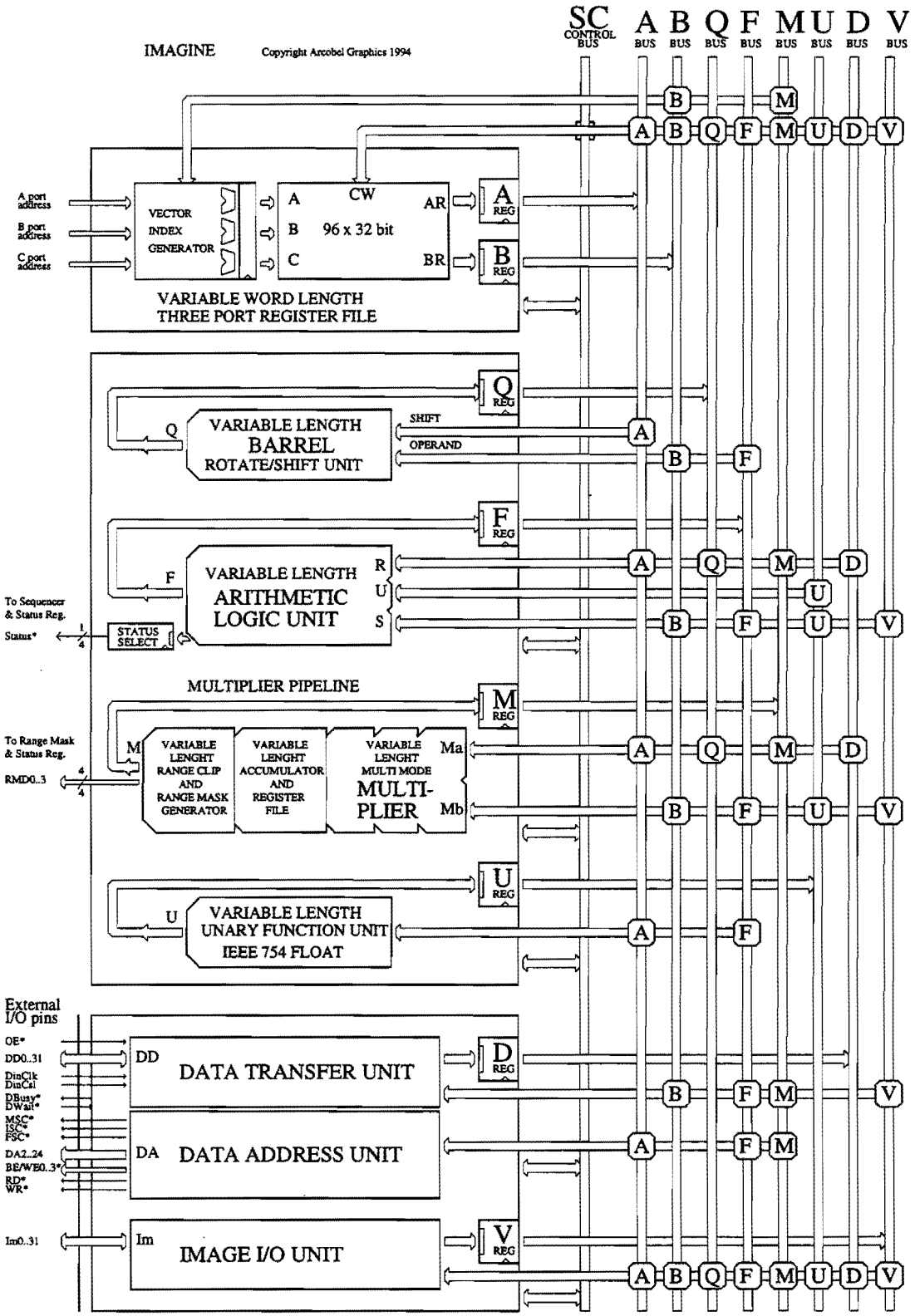


Figure 2: Data Processing Units of the IMAGINE .

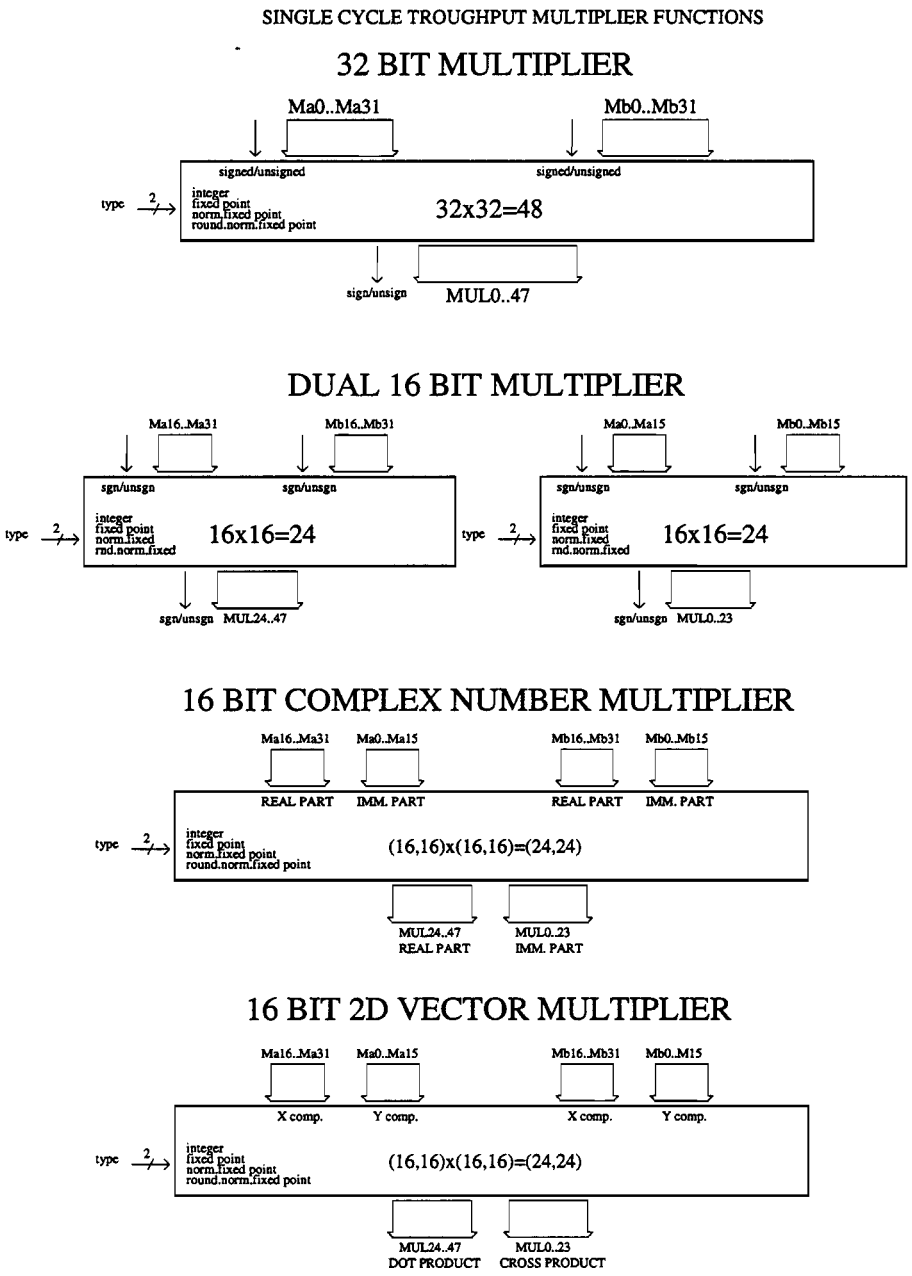
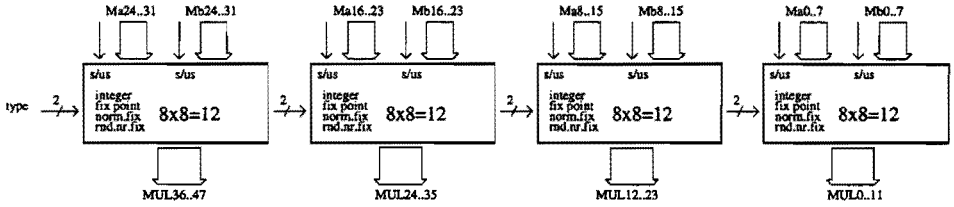


Figure 3: Multiplier modes (32 bit, double 16 bit and quadruple 16 bit).

SINGLE CYCLE TROUGHPUT MULTIPLIER OPERATIONS

QUAD 8 BIT MULTIPLIER



MATRIX x VECTOR MULTIPLIER / QUAD INPRODUCT MULTIPLIER

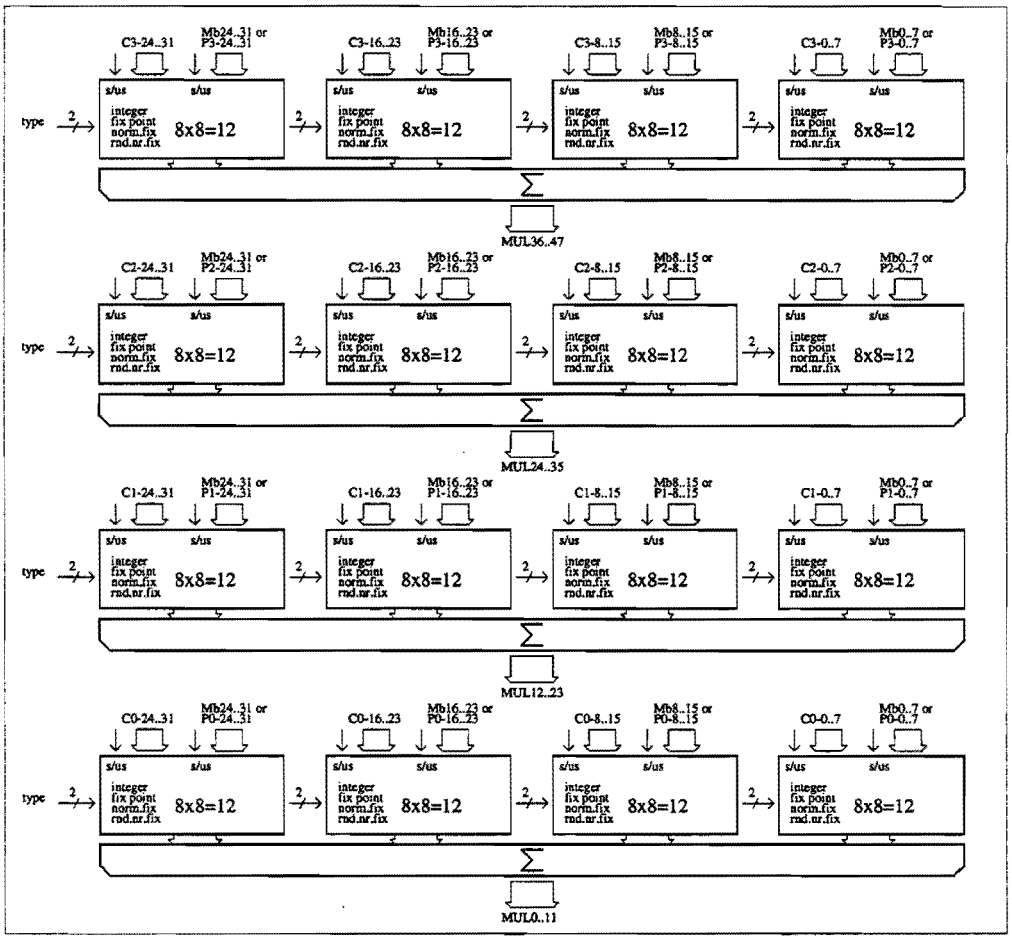


Figure 4: Multiplier modes (quadruple 8 bit and 16 fold 8 bit).

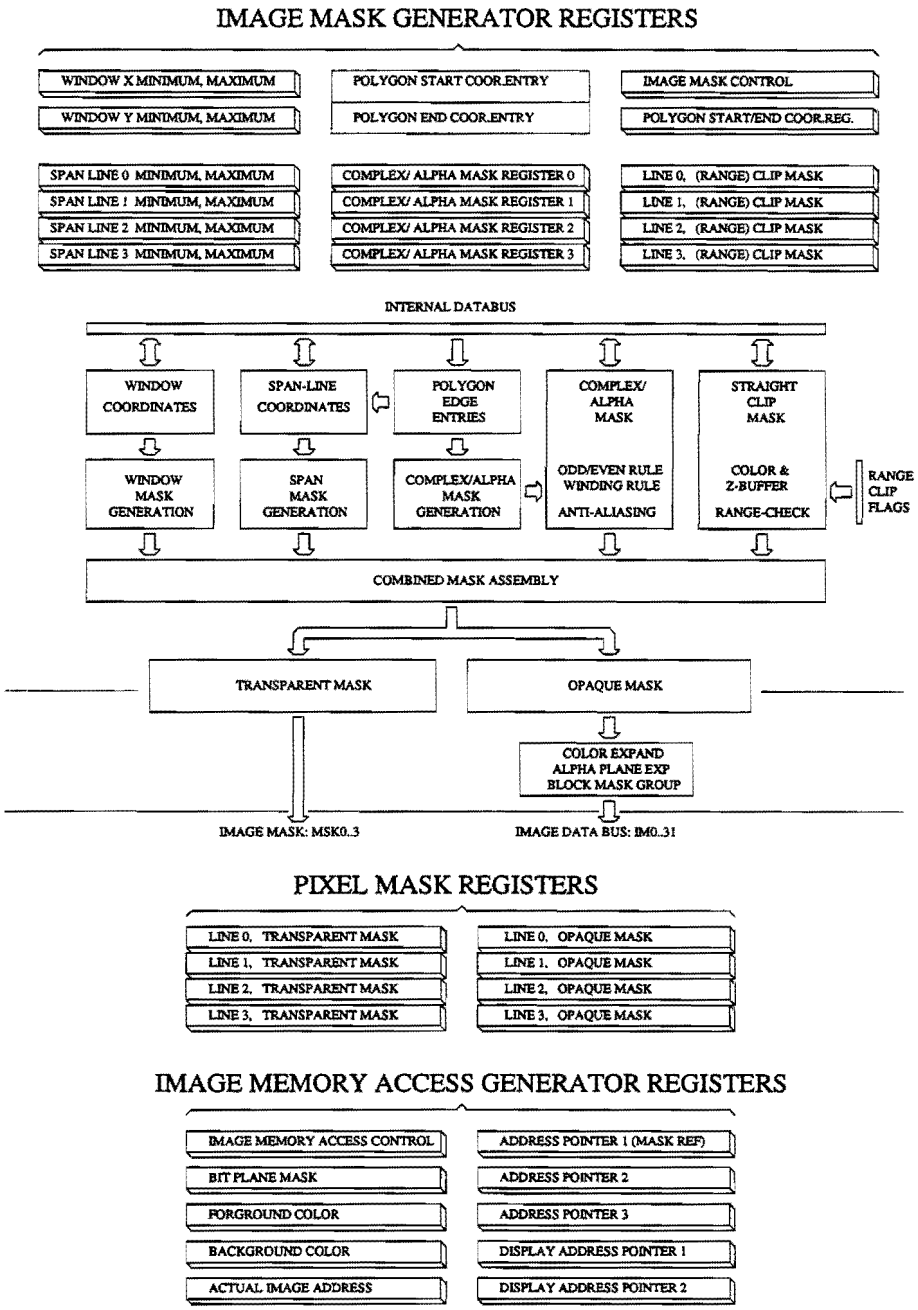


Figure 5: Register model of the Parallel Mask generator.