

# Four-way VLIW Geometry Processor for 3D Graphics Applications

● Hajime Kubosawa   ● Naoshi Higaki   ● Hiromasa Takahashi

*(Manuscript received December 1, 1999)*

**A four-way Very Long Instruction Word (VLIW) geometry processor that can be applied to PC-based CAD systems has been developed. PC-based CAD systems require a very high graphics performance and a very high cost effectiveness. To achieve a high performance, Single Instruction Multiple Data Stream (SIMD) instructions specialized for geometry operations were implemented and a unique architecture called the "software bypass mechanism" was adopted. To reduce system cost, PCI and AGP interface logics were implemented within the processor, so the processor can be applied to PC-based systems without using bus interfacing LSIs. The processor can issue up to four instructions at a time based on a four-way VLIW architecture. A performance of 2.5 GFLOPS and 6.5 Mp/s (mega polygons per second) was achieved at an operating frequency of 312 MHz. The processor was fabricated with a 0.21  $\mu\text{m}$  CMOS process technology on a 9.18 mm  $\times$  9.11 mm die.<sup>1),2)</sup>**

## 1. Introduction

Recently, there has been an increase in the need for high-performance 3D graphics systems in graphics applications such as engineering CAD systems and video games. Because these applications require a lot of floating point calculations, the floating point calculation capability of systems must be greatly increased. There are two approaches to improving floating point performance. One of them is to increase the parallelization of floating point calculations, and the other is to accelerate the operating frequency of the system.

Increased parallelization can be realized by constructing a system using multiple RISC processors.<sup>3)</sup> However, this approach needs a lot of resources and is too expensive to adopt in a PC-based system. Another way to increase parallelization is to add graphics-centric instructions to a conventional RISC instruction set. These special instructions can effectively improve performance if the instructions are prepared properly.

However, the previous work in this area has not resulted in geometry processing with sufficient performance.<sup>4)</sup> This paper describes a 3D geometry processor that achieves excellent performance through graphics-centric instructions that include SIMD instructions and a VLIW architecture.

Accelerating the operating frequency is a direct and simple approach to improving performance. We achieved a high operating frequency in our new processor by using a new clock distribution methodology and the latest process technology.

Another important requirement in graphics systems is to reduce the development time without compromising performance. Because 3D graphics performance is expected to increase by a factor of almost 8 every 18 months,<sup>5)</sup> the hardware should be as simple as possible to make development easy and also should have a high calculation capability to meet the performance demand. To fulfill the requirement of a short development

time, we chose the VLIW architecture for the processor because it is logically and physically easy to design. Another reason for adopting the VLIW architecture is that its instruction issue logic is less complicated than in other architectures.

Achieving a high cost-effectiveness is also important, especially in PC-based graphics systems. To provide this feature, the new processor has both a core unit which executes the geometry processing and a bus unit which can be connected directly via a standard bus interface. Consequently, the processor can be applied to conventional PC-based graphics systems without requiring additional LSIs, and therefore high-performance graphics systems can be created at minimum cost.

Chapter 2 of this paper describes the basic architecture of the geometry processor, and Chapter 3 describes the implementation of the processor in detail. Then, Chapter 4 describes the clock design methodology of the processor, Chapter 5 presents some evaluation results, and Chapter 6 concludes the paper.

## 2. Basic architecture

### 2.1 Graphics system

Figure 1 shows the block diagram of a graphics system that employs the 3D geometry processor. The bus unit receives polygon data from the host CPU via an AGP interface and outputs the geometry processing results to a rendering system over a 66 MHz PCI interface. The core unit receives data from the bus unit, performs

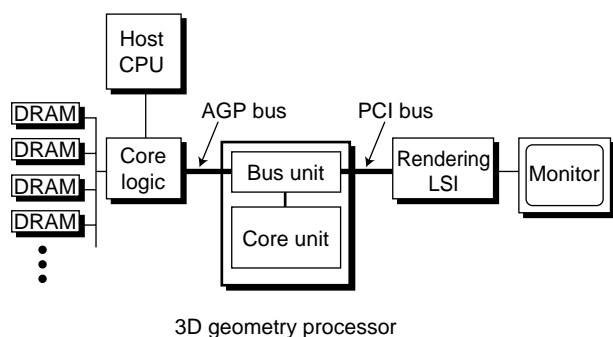


Figure 1  
Graphics system.

geometry processing, and outputs the processed data back to the bus unit.

### 2.2 VLIW geometry processor

The 3D geometry processor has a four-way VLIW architecture. One of the reasons we chose a VLIW architecture is that it can provide a highly parallel execution capability with a less complicated instruction dispatch logic than other architectures. The VLIW architecture has this advantage over other architectures because it does not require scheduling logic to dispatch instructions to execution units in parallel. Often, the instruction dispatch logic is the most critical circuit-delay path, and in a non-VLIW processor such as a superscalar processor, it is the most difficult part to design logically. Minimizing the circuit delay and completing the logic design usually takes a lot of time, so the VLIW architecture also has the advantage of reducing design time without compromising performance. Furthermore, because geometry processing has a lot of inherent data parallelism, it can be performed very efficiently using a VLIW architecture.

The processor has the four-way VLIW architecture reported in Ref. 6). A VLIW instruction is 120 bits long and consists of four 30-bit instruction elements. The position of an instruction element in a VLIW instruction is called a "slot." Operations executed by an instruction element are defined according to the slot where the element is located. There are four slots: Slots A to D. Slot A and Slot C execute floating point operations. Slot B and Slot D execute integer operations, including memory address calculations. To provide the parallel floating point execution capability required in geometry processing, floating point instructions located in Slot A and Slot C can be executed simultaneously. The implementation of the floating point execution unit is discussed in the next section.

### 3. Implementation

#### 3.1 Floating point unit

Because the floating point data format is used in geometry processing and a very high performance is required by the target applications, the processor should have strong floating point parallel execution capabilities based on the VLIW architecture and SIMD instructions. **Figure 2** shows the block diagram of the processor's core unit. The core unit has four floating point pipelines (Fpipes 0 to 3), an integer pipeline, a 64 KB code RAM, two 8 KB data RAMs, two floating point registers, and an integer register. Each Fpipe consists of a multiplier and an adder and can execute multiply-accumulate instructions, which is very effective for improving geometry processing performance. The Fpipes are tightly coupled in pairs that form two sub-units. The Fpipes in each pair share a floating point register file and data RAM. Instruction elements in Slot A can manipulate FR 0 and Data RAM 0 and can activate Fpipe 0 and Fpipe 1. Instruction elements in Slot C can manipulate FR 1 and Data RAM 1 and can acti-

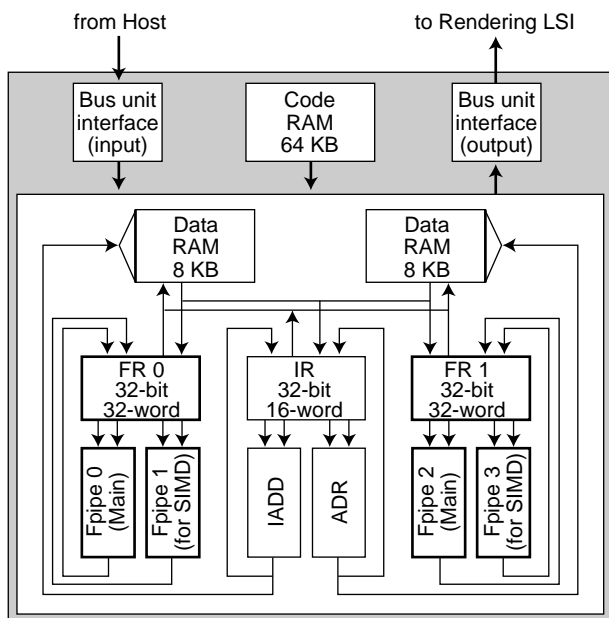


Figure 2  
Core unit of 3D geometry processor.

vate Fpipe 2 and Fpipe 3. By adopting this clustered and symmetrical structure, the numbers of register file ports and cross-bar switches on the data bus line can be reduced to roughly one half of the numbers in conventional non-clustered structures, which greatly reduces the delays in the circuits and wires of register files and switches. Consequently, the register file access time and critical path delay can be reduced.

The block diagram of an Fpipe is shown in **Figure 3**. The output of the multiplier is connected to the adder as well as to the floating point register for writing back. When a floating point multiply-accumulate instruction (called an fmac instruction) is executed with this block, two operands are fed into the multiplier and the multiplication result is sent to the adder and then added to the data stored in a register for accumulation. Multiplication and addition each take two stages, consequently, an fmac takes four stages. The Fpipes support the standard IEEE 32-bit single precision floating point data format.

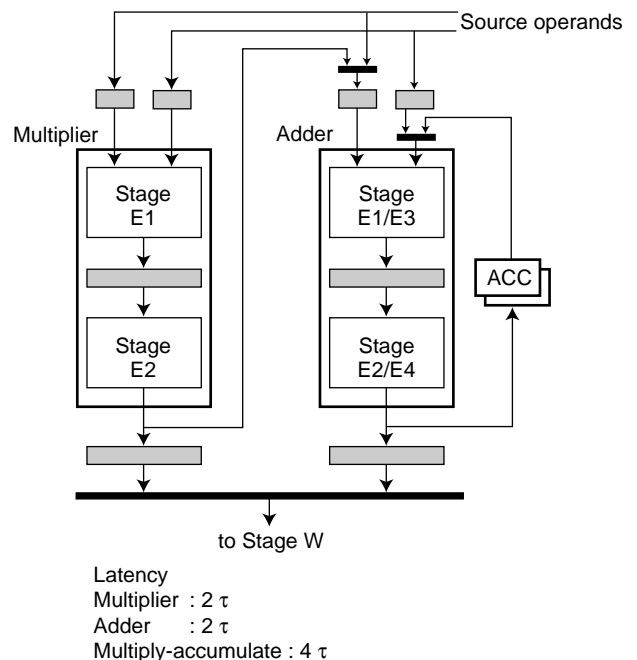


Figure 3  
Floating point pipeline.

**Figure 4** shows the block diagram of an Fpipe multiplier. Stage E1 consists of a Wallace tree and a Booth decoder. To adjust the circuit delay for a target cycle time, only the lower 25 bits of the sum and carry outputs from the Wallace tree are added in the Stage E1 mantissa adder. The input check block generates input operand types that are used for setting constants and exception flags. In Stage E2, the upper 27 bits of the sum and carry outputs and the carry from the lower 25 bits are added. The results are then normalized by a right shifter and an exponent adder.

**Figure 5** shows the block diagram of an Fpipe adder. In Stage E1, there are two right shifters, two mantissa adders, a data packer, a zero filler, an input checker, and various other blocks. The two mantissa adders generate addition and subtraction results, and the 2's complemeter generates the complement for data format conversion instructions between an integer and a floating point. In Stage E2, the correct result is selected according to the instruction being executed and various flags generated in Stage E1.

Newton's method was adopted for division operations in the processor. The processor has four ROMs which store approximate reciprocal values for iterative calculation of Newton's method. Each Fpipe has its own ROM and can execute an inde-

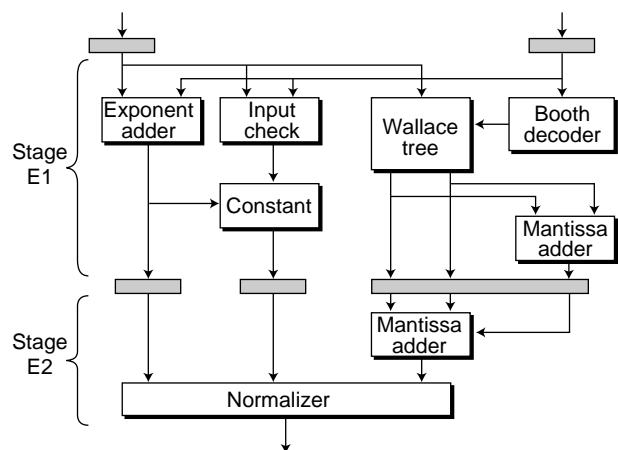


Figure 4  
Multiplier of a floating point pipeline.

pendent divide operation from other Fpipes simultaneously. The latency of the divide operation is 14 clock cycles. For square root operations, the processor has ROMs which store approximate reciprocal values of the operations. The approximate values are precise enough to apply them to geometry operations directly. The latency of the ROM look-up instructions is two cycles for division and square root operations.

### 3.2 SIMD instructions

Because a multiply-accumulate instruction involves two operations and because two SIMD instructions can be executed simultaneously, a total of eight floating point operations can be executed simultaneously. To activate four sets of floating point pipelines simultaneously, SIMD type instructions need to be implemented. One pair of Fpipes, for example, Fpipe 0 and Fpipe 1, is activated by an SIMD type instruction. The multipliers and adders in the Fpipes are activated by an fmac instruction. Therefore, if two SIMD type fmac instructions are executed, the processor could perform eight floating point operations simultaneously.

**Figure 6** shows a diagram of  $4 \times 4$  array multiplication using an SIMD type fmac instruction called "qfmac." This type of multiplication is quite common in 3D graphics processing, for example, coordinate transformations. To execute the array multiplication, 16 multiplications and

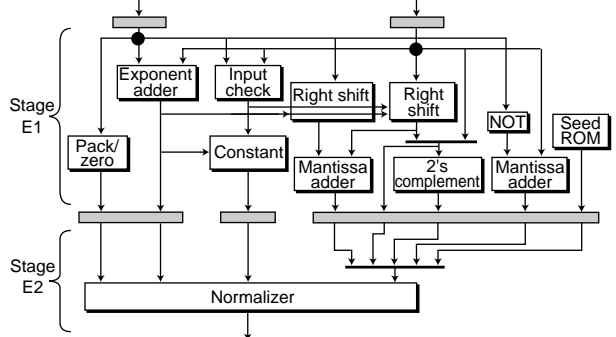


Figure 5  
Adder of a floating point pipeline.

12 additions are necessary. However, as shown in Figure 6, it can be performed by using only eight qfmac instructions. In this example, the x and z values are calculated in Fpipe 0 and Fpipe 1. The calculation is as follows:

$$x' = a0*x + a1*y + a2*z + a3*w$$

$$z' = c0*x + c1*y + c2*z + c3*w$$

The first qfmac operation calculates the values of a0 times x' and c0 times x' and stores the results in the ACC simultaneously by using Fpipe 0 and Fpipe 1, respectively. The second qfmac operation calculates the values of a1 times y' and c1 times y' and adds the values stored in each ACC. The third and fourth operations are identical to the second. Consequently, the x and z values can be calculated by using four qfmac operations in Fpipe 0 and Fpipe 1. In Fpipe 2 and Fpipe 3, the values of y and w are calculated in the same way. Therefore, the four-way VLIW architecture and the SIMD type multiply-accumulate instructions make it possible to perform an array multiplication in only 10 clock cycles.

### 3.3 Software bypass

The software bypass is a unique feature of

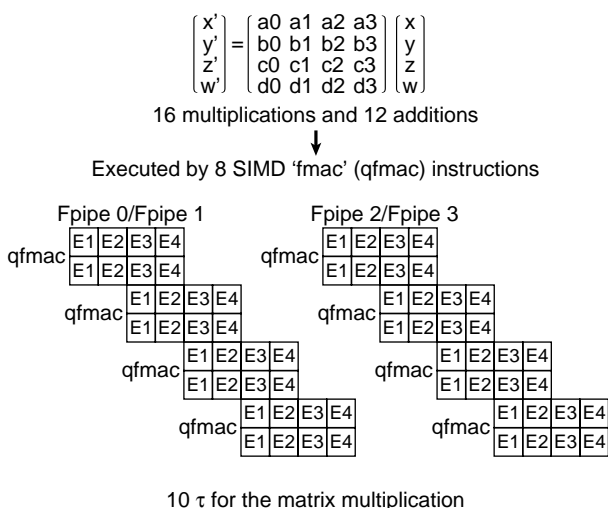


Figure 6  
4 × 4 array multiplication.

this chip. This mechanism enables the compiler to control hardware bypass lines, which, in conventional microprocessors are controlled by the hardware at runtime. **Figure 7** shows the bypass logic and a simple example of the assembler notation for its operation. The difference between the processor we propose and other processors is that, in our processor, the compiler selects which data is fed into the buffer register. In most other processors, this is done by the hardware. As shown in Figure 7, each multiplexer has five inputs, but it is not necessary to implement hardware logic to control the inputs. This mechanism made it possible to eliminate 20 000 gates from the bypass control logic compared with a previous design we made.<sup>7)</sup> As a result, logical verification and critical path timing optimization of the bypass control logic were not required, which reduced the development time.

### 3.4 Clipping

Clipping is one of the basic operations in 3D geometry processing. The operation determines whether objects to be displayed are inside a display screen. To do this, 21 conditions have to be checked for each independent triangle polygon formed by the vertices of the displayed object. To

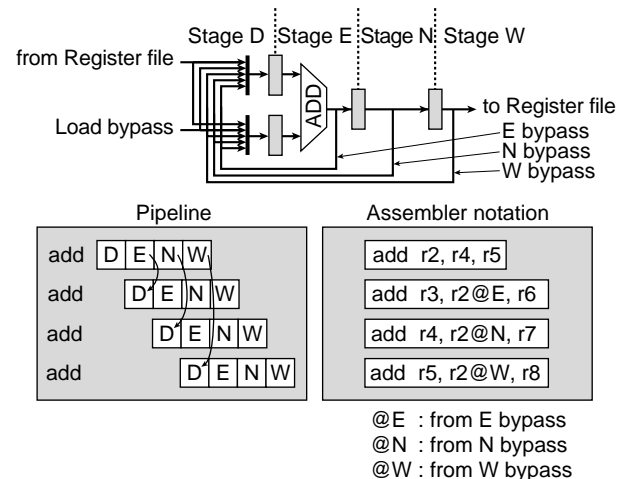


Figure 7  
Software bypass logic and example assembler notation for its operation.

accelerate the clipping operation, the processor sets the 7 bits of a condition code register (CCR) and the 14 bits of two condition code backup registers (CCBRs), which hold the two immediately preceding CCR values (see **Figure 8**). The branch

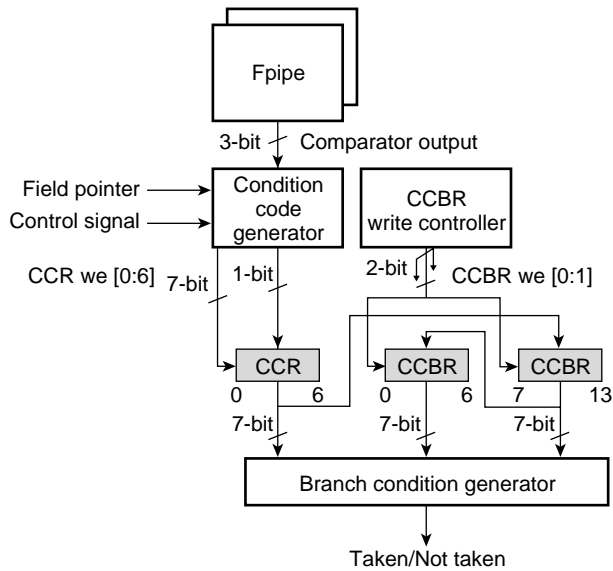


Figure 8  
Branch condition generator logic.

condition generator decides whether the object to be displayed is inside the display screen by checking whether the contents of the CCR and CCBRs are all zero.

These features are helpful for the clipping operation because the operation is usually applied to strips of triangles and we have to check seven conditions for each vertex of the triangles to decide whether they are inside the display area. In the conventional approach to the clipping operation, combinations of subtraction, move, and logical OR instructions are used. As a result, 20 steps would be required to check seven conditions for one vertex. With the features we propose, it is possible to check whether a vertex is inside the screen in only seven steps. Moreover, by using CCBRs, the load/store operations needed for the conventional approach can be eliminated. This is possible because three sets of comparison results corresponding to the three vertices of a triangle are already held in the CCR and CCBRs. Consequently, by using these features, it is possible to determine whether a triangle is inside the screen

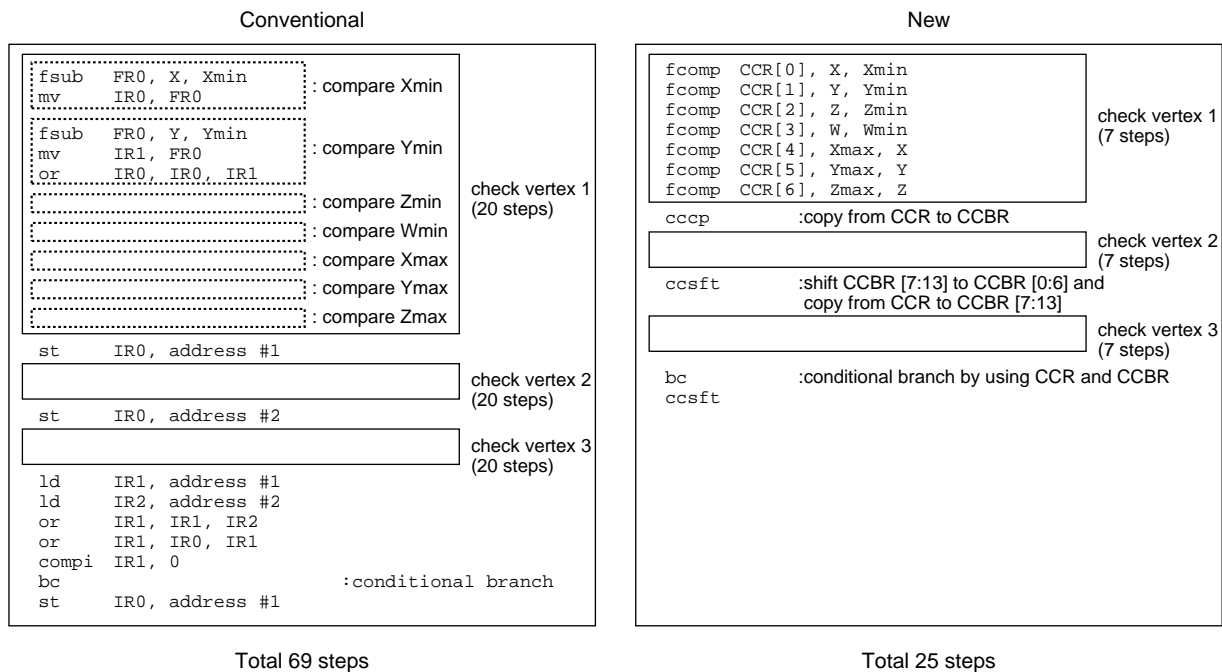


Figure 9  
Comparison of code for clipping in conventional and new methods.

in only 25 steps. Without these features, it would require 69 steps. **Figure 9** compares the code required for clipping in the conventional method and the new method.

#### 4. Clock design methodology

The clock design methodology is a key factor in designing high-performance processors in a short time. For the bus unit, the ability to set the clock signal early or late is a highly requested feature because the AC specification of the AGP is very tight.<sup>8)</sup> **Figure 10** shows the block diagram of the bus clock tree we adopted. To fulfill the requirements of the hold time, a through latch is used between input I/O and FFs so that the latch can hold valid data until the first half of the clock period. Therefore, it is necessary to input an early clock to the latches. On the global clock distribution, the clock trunks are automatically jogged to make them equidistant. On the local clock distribution, the clock buffers are automatically inserted by a CAD tool. A single-level buffer is inserted for the latch clock, and multi-level buffers are inserted for the FF clock. In this way, the clock skew between the latch clock and FF clock

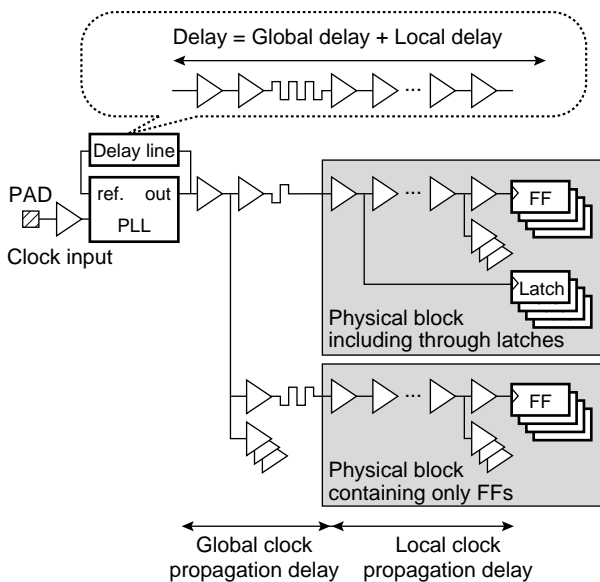


Figure 10  
Bus clock tree.

is automatically established.

The only part of the bus clock tree requiring manual adjustment is the propagation delay of the delay line connected to the reference signal input of the PLL. This delay line delays the reference signal by the average of the clock propagation delay from the PLL output to the latches and the propagation delay from the PLL output to the FFs. Because the delayed clock is in phase with the External AGP clock, the timing relationship shown in **Figure 11** is established. This clock design methodology enabled us to quickly design this high-performance processor because only the PLL block had to be optimized manually.

#### 5. Evaluation

To evaluate the 3D geometry performance in terms of polygons per second, we hand-coded a program for use as a benchmark. The input for this program was an x, y, z coordinates value and its normal vector with respect to the vertex it belonged to. For this input, the benchmark program performs transformation of the vertex coordinates by parallel projection, normal coordinate transformation, and lighting, which is a typical geometry-related execution sequence for a polygon to be displayed on the screen. We ran this program on a software simulator<sup>6)</sup> and found that it took only 48 cycles to complete the program. This result corresponds to a performance of 6.5 M polygons

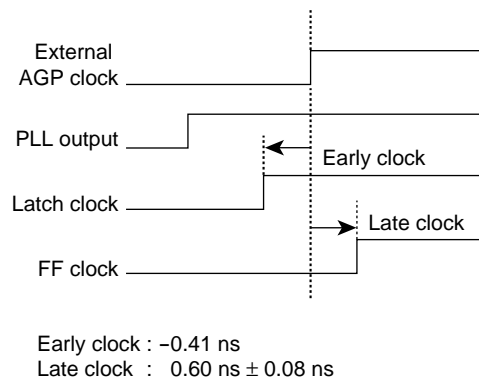


Figure 11  
Timing of bus clock.

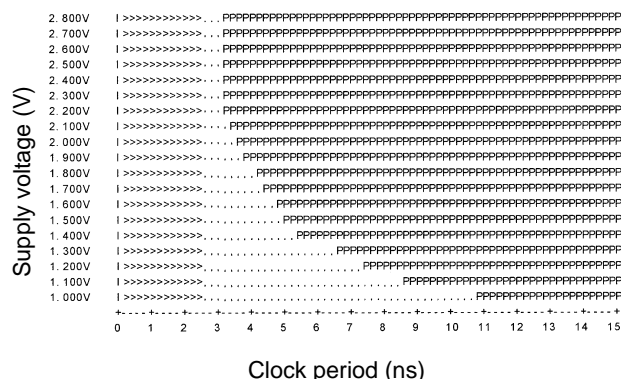


Figure 12 Shmoo plot.

Table 1 Process technology.

Technology	0.21 $\mu$ m, 3 metal layers
Gate length	0.21 $\mu$ m
Gate oxide	5 nm
1st metal pitch	0.88 $\mu$ m
2nd metal pitch	0.88 $\mu$ m
3rd metal pitch	0.88 $\mu$ m
Frequency	312 MHz (core clock)
	66 MHz (bus clock)
Power supply	2.5 V
Power dissipation	7.5 W
Package	BGA 352-pin
Die size	9.18 mm $\times$ 9.11 mm

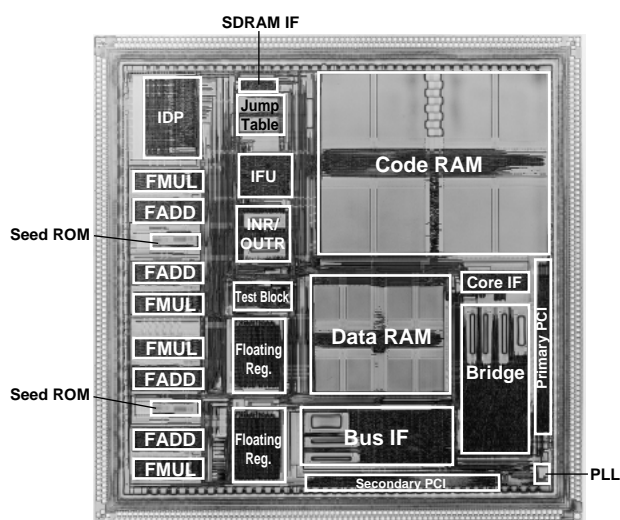


Figure 13 Photograph of 3D geometry processor.

per second when this chip is running at a 312 MHz clock frequency. **Figure 12** shows a Shmoo plot obtained by applying the test vector for activating the critical path estimated by static timing analysis of this chip. The plot predicts 312 MHz (3.2 ns cycle time) operation for the chip at 2.5 V. **Table 1** lists the technology-related features and die characteristics. **Figure 13** shows a photograph of this chip.

### 6. Conclusion

A geometry processor for 3D graphics systems has been developed. The processor has a four-way VLIW architecture with a software bypass mechanism and uses SIMD instructions. A performance of 2.5 GFLOPS and 6.5 M polygons per second has been achieved at 312 MHz. The processor has special condition registers and a branch condition generator which successfully reduces the number of instruction steps for clipping operations. An automatic clock delay tuning methodology is used to achieve a high clock frequency in a short design time. The processor has been implemented with a 0.21  $\mu$ m, 2.5 V, three-layer metal CMOS technology on a 9.18 mm  $\times$  9.11 mm die.

### References

- 1) N. Higaki et al.: A 2.5 GFLOPS 6.5 Million Polygons per Second 4-Way VLIW Geometry Processor with SIMD Instructions and a Software Bypass Mechanism. *ISSCC Digest of Technical Papers*, Feb. 1999, pp.260-261.
- 2) H. Kubosawa et al.: A 2.5 GFLOPS 6.5 Million Polygons per Second 4-Way VLIW Geometry Processor with SIMD Instructions and a Software Bypass Mechanism. *IEEE J. Solid-State Circuits* (to be published).
- 3) A. Krech: Blitzen: Lightning Speed 3D Geometry Accelerator Hot chips, Aug. 17, 1998.
- 4) N. Glaskowsky: Advanced 3D Chips Show Promise. *Microprocessor Report*, 11, 8, June 23, 1997.

- 5) N. Glaskowsky: 3D Graphics and Multimedia. Chips and Choices. Microprocessor forum 97, Seminar Notes, Oct. 13, 1997.
- 6) Y. Kimura et al.: A VLIW Geometry Processor with Software Bypass mechanism. *IEICE transactions on Electronics*, **E81-C**, 5, May, 1997.
- 7) H. Kubosawa et al.: A 1.2W 2.16GOPS/720MFLOPS Embedded Superscalar Microprocessor for Multimedia Applications. ISSCC Digest of Technical Papers, Feb. 1998, pp.290-291.
- 8) Intel Corporation: Accelerated Graphics Port Interface Specification.



**Hajime Kubosawa** received the B.S. and M.S. degrees in Electronics Engineering from Keio University, Yokohama, Japan in 1982 and 1984, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1984 and has been engaged in research and development of processors for floating point, multimedia, and graphics applications. He is a member of the Institute of Electronics, Information and

Communication Engineers (IEICE) of Japan.

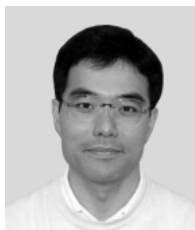
E-mail: kubosawa@flab.fujitsu.co.jp



**Hiromasa Takahashi** received the B.S. and M.S. degrees in Electronics Engineering from Chiba University, Chiba, Japan in 1978 and 1980, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1980 and has been engaged in research and development of various kinds of processors including multimedia processors and embedded processors. He is a member of the Institute of Electronics, Information and

Communication Engineers (IEICE) of Japan.

E-mail: takahasi@flab.fujitsu.co.jp



**Naoshi Higaki** received the B.S. degree in Physics from Hokkaido University, Sapporo, Japan in 1984 and the M.S. degree in Physical Engineering from Tsukuba University, Tsukuba, Japan in 1986. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1986 and has been engaged in research and development of high-speed circuit technologies and computer-aided-design environments. He is a member of the

Institute of Electronics, Information and Communication Engineers (IEICE) of Japan.

E-mail: higaki@flab.fujitsu.co.jp