Report Number: WUCSE-2005-33

2005-07-22

A Collision Detection Chip on Reconfigurable Hardware

Nuzhet Atay, John W. Lockwood, and Burchan Bayazit

Collision detection algorithms check the intersection between two given surfaces or volumes. They are computationally-intensive and the capabilities of conventional processors limit their performance. Hardware acceleration of these algorithms can greatly benefit the systems that need collision detection to be performed in real-time. A Field Programmable Gate Array (FPGA) is a great platform to achieve such acceleration. An FPGA is a collection of digital gates which can be reprogrammed at run time, i.e., it can be used as a CPU that reconfigures itself for a given task. In this paper, we present an FPGA based collision detection chip. The... Read complete abstract on page 2.

A Collision Detection Chip on Reconfigurable Hardware

Nuzhet Atay, John W. Lockwood, and Burchan Bayazit

Complete Abstract:

Collision detection algorithms check the intersection between two given surfaces or volumes. They are computationally-intensive and the capabilities of conventional processors limit their performance. Hardware acceleration of these algorithms can greatly benefit the systems that need collision detection to be performed in real-time. A Field Programmable Gate Array (FPGA) is a great platform to achieve such acceleration. An FPGA is a collection of digital gates which can be reprogrammed at run time, i.e., it can be used as a CPU that reconfigures itself for a given task. In this paper, we present an FPGA based collision detection chip. The chip can be used as a co-processor for a traditional computer or several of them can be utilized to work in parallel to create a very fast collision detection server for real time environments. In our experiments we have seen speeds-up of 36 with respect to a fast Pentium 4 chip. Further improvements are possible by using more advanced collision detection techniques

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2005-33

A Collision Detection Chip on Reconfigurable Hardware

Authors: Atay, Nuzhet; Lockwood John W; Bayazit Burchan

July 22, 2005

Abstract: Collision detection algorithms check the intersection between two given surfaces or volumes. They are computationally-intensive and the capabilities of conventional processors limit their performance. Hardware acceleration of these algorithms can greatly benefit the systems that need collision detection to be performed in real-time. A Field Programmable Gate Array (FPGA) is a great platform to achieve such acceleration. An FPGA is a collection of digital gates which can be reprogrammed at run time, i.e., it can be used as a CPU that reconfigures itself for a given task.

In this paper, we present an FPGA based collision detection chip. The chip can be used as a co-processor for a traditional computer or several of them can be utilized to work in parallel to create a very fast collision detection server for real time environments. In our experiments we have seen speeds-up of 36 with respect to a fast Pentium 4 chip. Further improvements are possible by using more advanced collision detection techniques.

A Collision Detection Chip on Reconfigurable Hardware*

Nuzhet Atay John W. Lockwood Burchan Bayazit
Washington University in St. Louis
{atay,lockwood,bayazit}@cse.wustl.edu

Abstract

Collision detection algorithms check the intersection between two given surfaces or volumes. They are computationally-intensive and the capabilities of conventional processors limit their performance. Hardware acceleration of these algorithms can greatly benefit the systems that need collision detection to be performed in real-time. A Field Programmable Gate Array (FPGA) is a great platform to achieve such acceleration. An FPGA is a collection of digital gates which can be reprogrammed at run time, i.e., it can be used as a CPU that reconfigures itself for a given task.

In this paper, we present an FPGA based collision detection chip. The chip can be used as a co-processor for a traditional computer or several of them can be utilized to work in parallel to create a very fast collision detection server for real time environments. In our experiments we have seen speeds-up of 36 with respect to a fast Pentium 4 chip. Further improvements are possible by using more advanced collision detection techniques.

1 Introduction

A wide range of applications, such as robotics, animation systems, physical simulations, virtual prototyping (CAD/CAM), biomedicine, require collision detection in order to perform particular tasks. In the simplest definition, the collision detection is the task of deciding whether a geometric contact between two objects exists. The objects can be represented in several different ways, such as polygonal models, using constructive geometry, as parametric or implicit surfaces. In robotics, for example, the physical environment and the robot itself can be represented as a set of polygons. Then a motion planning algorithm can be used to find a collision-free path that allows the robot to navigate through the environment. In animation, physical properties of an object, such as deformation, can require finding colliding surfaces of the object with the environment. Because of their importance, collision detection algorithms are well studied and continuously being improved [12, 16].

With the advance of graphics processing units (GPUs), we have started to see their utilization in collision detection. However, GPUs are primarily designed to increase the speed of graphics computation. The limitations on the resolutions or need to use a limited shading language effects the collision detection algorithms running on them. For example, [2, 14] can handle general polygonal models only when they are represented as unions of convex objects. More general algorithms [5, 6] usually require both CPU and GPU resources. On the other hand, Field Programmable Gate Arrays (FPGAs) can be used to implement a dedicated collision detection chip capable of working without any other resources from CPU. Since an FPGA is a collection of digital gates which can be reprogrammed at run time, it can be reconfigured to suit a given task. FPGAs are continuously improving while suppressing the Moore's law. A recent analysis shows that it is expected that their floating-point performance will overpass the general purpose CPUs by 2009 [22].

In this paper, we present a collision detection chip based on FPGAs. Our design takes advantage of inherent parallelism of collision detection algorithms and checks collisions in parallel. Since we hope to utilize our system in motion planning algorithms, we have defined our collision detection problem as the following: *"given an environment, a dynamic object within this environment, and a set of confi gurations of the dynamic object, decide if the dynamic object at the given confi gurations is in collision*

*Authors' address: Dept. of Computer Science and Engineering, Washington University, St. Louis, MO 63130.

with the environment". We have both the environment and the dynamic object represented as triangular meshes. Our models are general, non-convex objects. Each configuration of the dynamic object is represented with six parameters (three for the position and three for the orientation). For each configuration our chip finds the transformations of the dynamic object's triangles and do several parallel fast triangle intersection tests against the environment. Although we have a dynamic object and static environment, the design can easily be extended to collision detection between a pair of dynamic objects. Similarly instead of just returning a binary value representing the collision state, the system can return the triangles in collision.

To the best of our knowledge, we are the first to implement a dedicated collision detection chip on an FPGA. Since it is highly parallel and use simpler operations, we have used a fast triangle-triangle intersection test to decide the collisions. Please note that our system is highly modular and we can replace our basic collision detection module with other more advanced collision detection modules such as hierarchical algorithms [4] or algorithms that can utilize temporal coherence [11]. Current FPGA technology lets us do up to 25 triangular collision detections in parallel. We have noticed that even with one collision detection circuit, because of the highly parallel nature of the matrix multiplication and other operations, our chip can still do the collision check faster than a general purpose CPU. In fact, with 25 collision detection circuit our design shows speed-ups of around 36 with respect to a Pentium 4, 3 Ghz.

In the next section we will give a brief information about FPGAs. Related work is discussed in Section 3. Section 4 presents our architecture and Section 5 describes individual modules. Experimental results are in Section 6 and Section 7 concludes our paper.

2 Field Programmable Gate Arrays (FPGA)

A field-programmable gate array (FPGA) is an integrated circuit (IC) that can be programmed in the field after it is manufactured. FPGAs are similar in principle to, but have vastly wider potential application than,

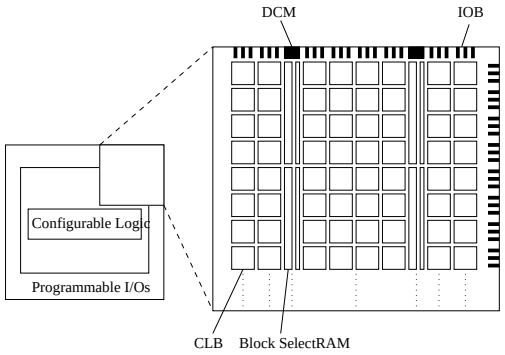


Figure 1: A typical FPGA layout: configurable logic cells (CLBs), block RAMs, digital clocks (DCMs) and input output buffers (IOBs).

programmable read-only memory (PROM) chips. FPGAs can be defined as reconfigurable processors that can be used for testing and implementing designs. Designs loaded on FPGAs are not final. As a result, the configuration of the circuit can be updated whenever needed.

A typical FPGA circuit (see Figure 1) consists of configurable logic cells (CLB) which are the primitive elements of FPGAs. Each CLB contains look-up tables (LUT) where combinatorial logic is stored. LUTs are memory elements which are addressed by inputs of circuit. Instead of rearranging gates according to design, the design is converted into LUTs. Capacity of LUTs is limited with the number of inputs, not with the complexity of design. By this way, delay through a LUT is constant. Large circuit designs can be performed by connecting CLB modules together. This structure is supported by on-chip and external memories. On-chip memories are block rams and distributed ram. These memories have low capacity (up to 1.5 MByte) but have very low access time (1 clock cycle) and large data width (up to 256 bits). Many low capacity (up to 336) memory modules (4 Kbit-18Kbit) can be accessed independently. This is an important advantage over general purpose processors where memory is a serious bottleneck. External memory has much higher capacity (up to terabit) but has higher access time. For synchronous circuits, programmable clocks are provided where each clock has frequency up to 500 Mhz.

To define the behavior of the FPGA it is required to use a Hardware Description Language (HDL) or a schematic

designed using an Electronic design automation tool. Either of these, when compiled, will generate a net list, that can be mapped to the actual FPGA architecture. When done the binary file generated is used to (re)configure the FPGA device. Common HDL's are VHDL and Verilog. A good introduction to FPGAs can be found in [15].

3 Related Work

A good survey of recent developments in collision detection algorithms can be found at [12, 16]. Recently a new class of collision detection algorithm, *image space collision detection* [1, 2, 3, 4, 7, 9, 10, 14, 19, 20, 21, 23, 25] is proposed to take the advantage of graphics hardware (GPUs). Such algorithms project the object geometry onto the image plane and do the collision check in this reduced space while using the depth map information stored in the GPU. They are fast, but they are not always accurate when the objects are far away. They are also limited by the resolution of the viewport. As a solution, hybrid methods are suggested where the parts of the algorithm takes advantage of GPUs and refine the result on CPU [5, 6, 8, 13].

To the best of our knowledge, we are the first researchers implementing a collision detection chip on an FPGA. However, there has been a recent design on Application-Specific Integrated Circuit (ASIC) [26]. Their design is based on dynamically aligned DOP-trees [7, 25]. In addition to advantage of FPGAs over ASIC, such as more flexible design and low cost, we believe our design is more modular, i.e., the collision detection module can be replaced by any other collision detection algorithm and the further performance gain can be achieved. Also, an FPGA can be reprogrammed by the user as oppose to the manufacturer when ASICs are used. This way, the user can program the hardware to use slow but accurate collision detection algorithm or fast but approximate collision detection algorithm. Please also note that, while their results are all based on simulations, our design is also implemented on a real FPGA.

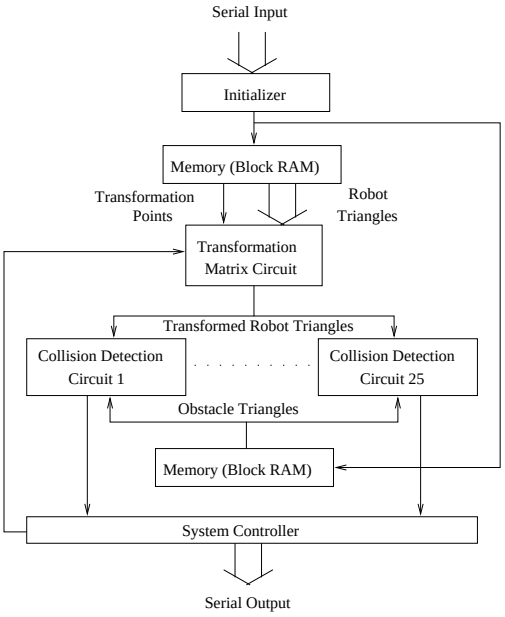


Figure 2: System overview.

4 System Overview

Our collision detection chip communicates with the host through serial port. The chip has a local memory to store object representations as well as the configurations of the dynamic object. A transformation circuit translates the dynamic object to given position and orientation. There are n parallel collision detection circuits each of which get a transformed triangle of dynamic object and check it against the environment's in parallel. The results of collision detection at each configuration is then sent back to the host. Note that this system is very flexible, by replacing the collision detection circuit, we can implement different collision detection algorithms.

5 Collision Detection Chip

The collision detection chip has four major modules (see Figure 3): (i) *I/O* is responsible from communication between the host computer and the chip, (ii) *memory* stores the object models, (iii) *transformation* transforms the dynamic object, and, (iv) *collision detection* checks

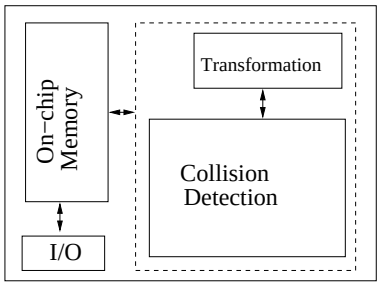


Figure 3: The collision detection chip.

the triangular intersection between the dynamic object and the environment.

5.1 I/O

In the current implementation, the host computer accesses to the collision detection chip using RS-232 serial communications. However since it is modular, it can easily be replaced by a PCI interface. It is used to get the geometrical models for the dynamic object and the environment to the chip. The configurations of the dynamic object is also sent through I/O module. Once all the configurations are processed, the chip returns the binary results (True or False) for each configuration through I/O.

5.2 Memory and Data Structures

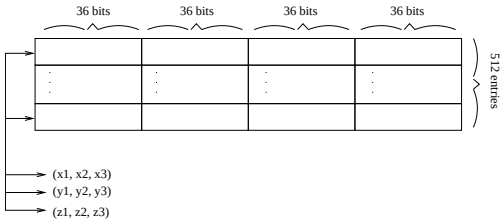


Figure 4: Storage of object triangles in block RAMs. Each Block RAM has two data paths, path A and path B. If the address of path A is k, then path B's address is $n - k$ when n is the maximum address on Block RAM.

The memory module of the collision detection chip is responsible from storing the object models, the configurations of the dynamic object and results of the collision detection for each configuration. The objects are represented as triangular meshes. In order to avoid costly floating point operations, we are using 32-bit fixed point arithmetic. Please note that this does not effect the performance of our chip since we normalize the coordinates before sending them to the chip. Each triangle is represented using 288 bits of data (each vertex is three 32-bit number, i.e., 96 bits, hence each triangle is 96×3).

Instead of having one large memory, FPGAs usually have several small memories which are called Block RAMs. Each Block has data paths to CLBs. The advantage of using such a distributed memory is that several memory block can be accessed in parallel. However, the designer has to be careful to maintain data consistency when the data is distributed among the memory blocks. The amount of the data that can be transfered from memory module to the computational components at each clock cycle depends on the data width. In our implementation, each Block RAM has a data width of 36 bits. There are two data paths from each block, effectively doubling the data width to 72 bits. Hence we can get whole triangle data, including vertex points in 4 clock cycles. Instead of waiting 4 clock cycles, we distributed our triangle data to four Block RAMs resulting in one triangle read per clock cycle. The structure of memory is shown in Figure 4. Block RAMs are cascaded to obtain data width of 144 bits for one data path, and 288 for two paths.

One Block RAM has a capacity of 18K bits, so we can store up to 576 32-bit numbers, allowing 4 Block RAMs to store up to 256 triangles. Current FPGA chips can contain up to 336 Block RAMs which increases the total number of triangles to 86016.

5.3 Transformation and Collision Detection

As we have mentioned before, our system utilizes intersection checks between triangles to decide a collision. Since the dynamic object is moving, its triangles must be transformed to new positions as the dynamic object changes its position and orientation. Our triangle-triangle intersection test is based on the fast triangle-triangle intersection test described in [18]. Next, we will briefly summarize this algorithm and then show how it can be applied in hardware.

Fast Triangle-Triangle Intersection Detection. This algorithm considers three cases: (i) triangles lie in the

half-planes of each other, (ii) the triangles are coplanar, (iii) the triangles are not coplanar. It works in the following way:

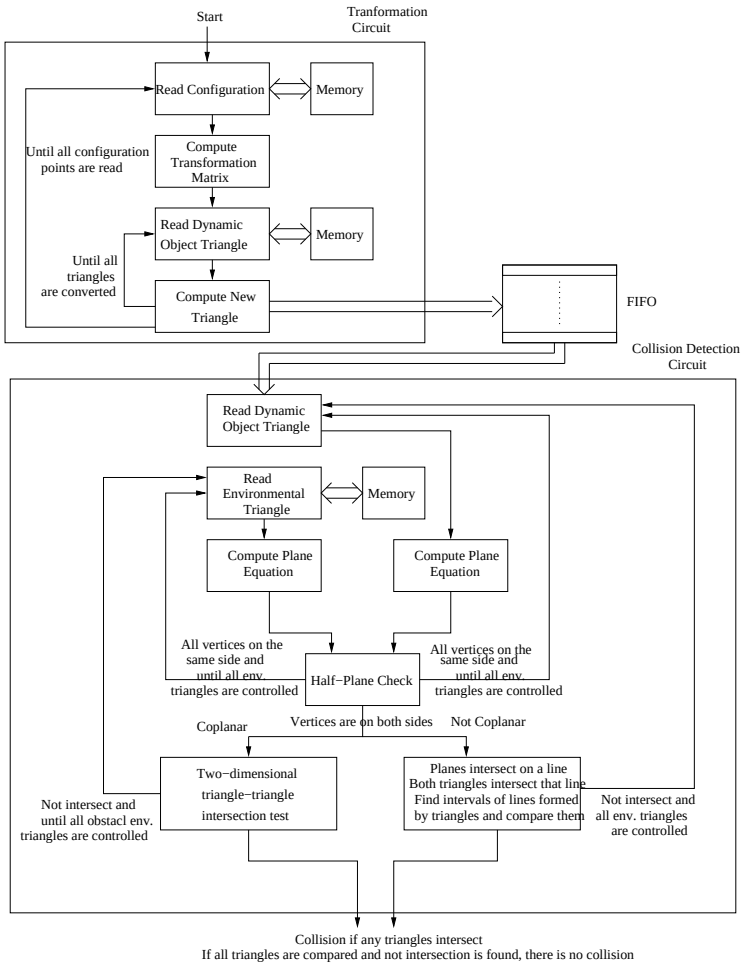
- *Half-plane check:* If all the vertices of one triangle lies on the same half-space of the other triangle, there is no way these triangles intersects so just return *collision free*.
- *Coplanar:* If the triangles are coplanar, project them onto the axis-aligned plane where the areas of the triangles are maximized. Then do a two-dimensional triangle-triangle overlap test.
- *Not Coplanar:* If L is the line at the intersection of two planes containing each triangle, both triangles are guaranteed to intersect with L. Find the intersection intervals for each triangle and check if they overlap (*collision*).

Collision Detection Hardware. Our hardware implementation closely matches the above algorithm. Figure 5 shows the internal structures of the transformation and collision detection circuits. Once a configuration is read, then the elements of transformation matrix is computed in parallel. When the matrix is found, we start transforming the dynamic object's triangles. For this purpose, we get a triangle from the memory (all vertices in parallel) and compute the new position of the triangle (by transforming each vertex in parallel). Once a triangle is transformed, it can be directly sent to the collision detection circuit to check the collision with environment. However, since the overall collision check is a slower process than the triangle transformation (number of triangles compared is much higher than the number of triangles transformed), we use a buffer (FIFO) to store the results of transformations while waiting for the collision detection circuit become ready.

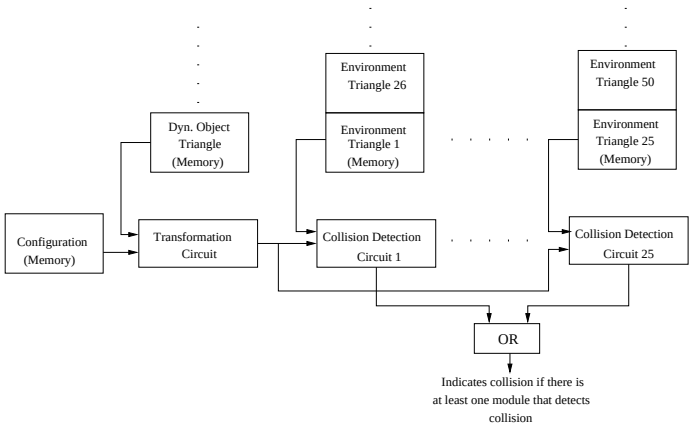
The collision detection circuit gets the next dynamic object triangle (T_{do}) from the buffer, and gets the next environmental triangle (T_e) from the memory. It computes the plane equations for both triangles in parallel. Next, it checks if all the vertices of the T_{do} lies in one side of T_e (all vertices are checked in parallel). If that is the case, then there is no collision, so the circuit moves to the next environmental triangle. Otherwise, the circuit checks if the planes are coplanar. If that is the case, then it performs a triangle-triangle collision test in 2D. Otherwise, finds the line L at the intersection of the planes containing T_{do} and T_e. The final collision detection test is then to find the intersection of L and each triangle (in parallel) and check if they overlap.

Collision detection between the dynamic object and the environment continues until either all triangle pairs are compared or one triangle-triangle intersection test returns collision. In order to increase the parallelism of the system, further collision detection circuits can be added to the chip. The number of the total parallel collision detection circuits is only limited by the number of logic slices on the FPGA chips. When several collision detection circuits are employed in parallel, the transformed triangle T_{do} can be compared with several environmental triangles in parallel. A block diagram of this approach is given in Figure 5(b).

Design Issues. We have used VHDL to develop our chip. Since trigonometric functions and multiplication are not directly supported in VHDL, we have used the Xilinx CoreGen tool to generate lookup tables (LUTs) that contain the results of trigonometric functions. We created a trigonometric module for sine and cosine functions which has symmetric output and uses distributed memory. When distributed memory is used for a LUT, the circuit has a latency of 2 clock cycles compared to 3 with Block RAM. Since speed is the most important issue in our problem, we preferred distributed memory. Input precision of 10 bits was specified, and the output precision was set to 32 bits. Similarly, the multiplications are also generated with CoreGen. It is a parallel signed multiplier with minimum pipelining and has a latency of 2 clock cycles. If maximum pipelining were chosen, the latency would be 6. This circuit also uses LUTs constructed on distributed memory. Inputs are 32 bits wide and output is 64 bits wide which satisfies our 32-bit number representation. For divisions, we have used right shifts since CoreGen divisions are very costly (both space wise and latency wise). Although this reduced the precision, we haven't observed any problem with it so far. Other operations are completed using standard VHDL functions. The algorithm works as a state machine with approximately 50 states.



(a)



(b)

Figure 5: Internal structures of transformation and collision detection circuits: (a) collision detection path, (b) modular representation.

6 Experiments

Our target FPGA chip is Xilinx Virtex-4 XC4VLX200 [24]. This chip allows us to create up to 25 collision detection circuits in parallel and can run our design at the clock rate of 50 MHz. We compared our solution to the sequential execution running on a workstation with Pentium-4 processor at 3 GHz with 1 GB memory. We have both simulation and real hardware results.

In our simulation experiments, we have used ModelSim XE II/Starter 5.8c by Mentor Graphics [17] edition. ModelSim is an HDL (Hardware Description Language) high-end simulator. It takes the VHDL description of the design as well as input values, then compiles the design, and runs it in the defined clock frequency. Outputs and internal signals of circuit can be examined while simulation is running. Simulation runs circuit in real-time so timing is accurate, i.e. same results are obtained when design is loaded to FPGA chip. Our development board has a 40-MHz clock on it, so we have simulated circuit using this frequency.

In order to test our chip, we designed an experiment where a dynamic object is placed at different configurations in an environment. The object is made-up of 12 random triangles. Next we have randomly placed 1600 obstacle triangles in a workspace which has 10 times length, width and height of the bounding box covering the dynamic object. We have checked the collision of the dynamic object at 20 different configurations.

In our experiments, we have executed a functionally similar algorithm in a workstation. In the worst case, each collision check between the moving object and the environment takes 19200 triangular intersection checks. For 20 different configurations of the dynamic object, we have to perform 384,000 triangle comparisons. Collision checking for 20 configurations took 247 milliseconds on the workstation. Next, we run the same experiment using our design in ModelSim. The same set took 170.28 msec when our chip contained only one collision detection circuit. When we have included 12 collision detection circuits, it took 13.44 msec to finish. Finally when we used 25 collision detection circuits, the collision detection time was 6.85 msec. Our results can be seen in Figure 6. These results show that even without any parallel collision detection circuit, our chip performs faster than a Pentium 4. Its performance can be explained by the parallelism in individual circuits. This increases the speed-up up to 36 when collision detection circuits are replicated and run concurrently. Another advantage is the FIFO between transformation circuit and collision detection circuit. This lets transformation circuit time overlap collision detection circuit time. As a result, the time we obtained is only the time for collision detection circuit. Please note that there are two main modules in the system, transformation matrix circuit and collision detection circuit. These circuits include algorithms that can be changed or optimized according to application. In our experiments, we have noticed that transformation circuit takes 710 ns and collision detection circuit takes 440 ns in the worst case with 40 MHz clock frequency. These times do not depend on anything except clock frequency. Moreover, the time of transformation matrix totally overlaps with collision detection time except the time it takes in the beginning of the execution. The FIFO between them lets constant data flow between these modules. As a result, in the worst case (where there are not collisions, and triangles are arranged so that one vertex lies on one half space and the others lies on the other half space) total time of execution is defined as:

$$time = \frac{N_{dyn} * N_{env} * N_{conf} * T_{col}}{\#\ of\ collision\ detection\ modules} + T_{trans}$$

where N_{dyn} is the number of triangles in the moving object, N_{env} is the number of the triangles in the environment, N_{conf} number of configurations, T_{col} is the collision detection time in one circuit and T_{trans} is the transformation time for a triangle.

To verify that our system is working, we have also tested it on Virtex-4 ML401 evaluation board. Unfortunately, this board contains a limited capacity FPGA (Virtex-4 XC4VLX25), we could only test one collision detection circuit in the real FGPA. We were successfully run the same experiment on this board and found similar results to the simulation.

7 Conclusion

We have presented a collision detection chip based on an FPGA. Our chip takes the advantage of inherited parallelism of collision detection algorithms and can

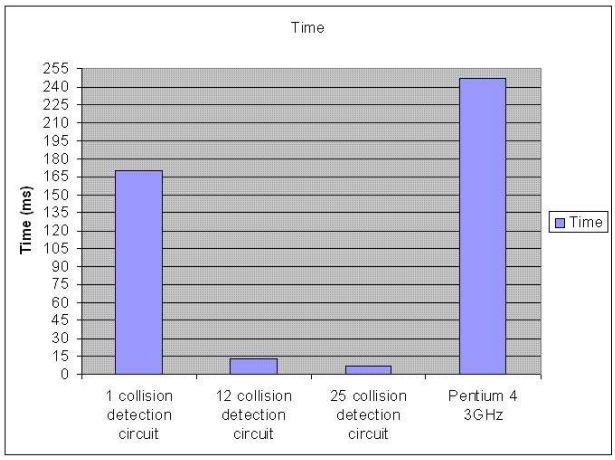


Figure 6: Collision detection chip with 1, 12 and 25 collision detection circuit vs. Pentium 4 for 384K triangle intersection test.

compute the collision detection up to 36 times faster than a Pentium-4, 3Ghz CPU. Our current chip uses fast triangle-triangle intersection test to check collision. Our future work includes implementing more advanced collision detection algorithms on the FPGA and introducing pipelining to our chip. We would also like to compare our chip's performance to other collision detection algorithms running on the CPU. Finally, we would like to migrate our design to custom hardware platforms to get even further speed-ups.

References

[1] G. Baciu and S. K. Wong. Image-based techniques in a hybrid collision detector. In *IEEE Trans. on Visualization and Computer Graphics*, 2002.

[2] G. Baciu, W.S.-K. Wong, and H. Sun. Recode: an image-based collision detection algorithm. *The Journal of Visualization and Computer Animation*, 10(4):181–192, 1999.

[3] J. Eckstein and E. Shomer. Dynamic collision detection in virtual reality applications. In *In WSCG'99*, pages 71–78, 1999.

[4] S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *SIGGRAPH*, pages 171–180, 1996.

[5] N. K. Govindaraju, M. C. Lin, and D. Manocha. Fast and reliable collision culling using graphics hardware. In *Virtual Reality Software and Technology (VRST)*, 2004.

[6] N. K. Govindaraju, S. Redon, M. C. Lin, and D. Manocha. Cullide: Interactive collision detection between complex models in large environments using graphics hardware. In *Graphics Hardware*, 2003.

[7] A. Gress and G. Zachman. Object-space interference detection on programmable graphics hardware. In *In SIAM Conf. on Geometric Design and Computing*, 2003.

[8] B. Heidelberger, M. Teschner, and M. Gross. Real-time volumetric intersections of deforming objects. In *Proc. of Vision, Modeling and Visualization*, 2003.

[9] K. Hoff, A. Zaferakis, M. Lin, and D. Manocha. Fast and simple 2d geometric proximity queries using graphics hardware. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 145–148, 2001.

[10] P. M. Hubbard. Collision detection for interactive graphics applications. In *IEEE Transactions on Visualization and Computer Graphics*, volume 1, pages 218–230, 1995.

[11] M. Hughes, M. C. Lin, D. Manocha, and C. Dimattia. Efficient and accurate interference detection for polynomial deformation and soft object animation. In *Proc. of Computer Animation*, pages 155–166, 1996.

[12] P. Jimenez, F. Thomas, and C. Torras. 3D collision detection: A survey. *Computers & Graphics*, 25(2):269–285, 2000.

[13] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. In *Proc. of ACM Symposium on Computer Animation*, 2002.

[14] D. Knott and D. K. Pai. Cinder: Collision and interference detection in real-time using graphics hardware. In *Proc. of Graphics Interface*, 2003.

[15] H. Krupnova and G. Saucier. FPGA technology snapshot: Current devices and design tools. In *11th IEEE International Workshop on Rapid System Prototyping (RSP 2000)*, pages 200–2005, 2000.

[16] M. C. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, volume 1, pages 602–608, 1998.

[17] Modelsim. http://www.model.com/.

[18] Tomas Moller. A fast triangle-triangle intersection test. *J. Graph. Tools*, 2(2):25–30, 1997.

[19] K. Myszkowski, O. G. Okunev, and T. L. Kunii. Fast collision detection between complex solids using rasterizing graphics hardware. In *The Visual Computer*, volume 11, pages 497–512, 1995.

[20] J. Rossignac, A. Megahed, and B. D. Schneider. Interactive inspection of solids: cross-sections and interferences. In *Proceedings of ACM SIGGRAPH*, 1992.

[21] M. Shinya and M. C. Forgue. Interference detection through rasterization. In *The Journal of Visualization and Computer Animation*, volume 2, pages 131–134, 1991.

[22] Keith Underwood. FPGAs vs. CPUs: trends in peak fbating-point performance. In *FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pages 171–180, New York, NY, USA, 2004. ACM Press.

[23] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. In *Computer Graphics Forum (Proc. of Eurographics'01)*, volume 20, pages 260–267, 2001.

[24] Xilinx. http://www.xilinx.com/.

[25] G. Zachman. Rapid collision detection by dynamically aligned dop-trees. In *Proc. of IEEE Virtual Reality Annual Internation Symposium, VRAS'98*, pages 90–97, 1998.

[26] G. Zachman and G. Knittel. An architecture for hierarchical collision detection. In *The 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2003*, pages 149–156, 2003.