This Publication's Reference:

[1] Andreas Raabe, Andreas Nett, and Andreas Niers. A Refinement Case-Study of a Dynamically Reconfigurable Intersection Test Hardware. In *ReCoSoc'08*, July 2008.

A Refinement Case-Study of a Dynamically Reconfigurable Intersection Test Hardware

Andreas Raabe Andreas Nett Andreas H. Niers

University of Bonn
Technical Computer Science
Römerstr. 164, 53117 Bonn, Germany
{raabe,nett,niers}@cs.uni-bonn.de

Abstract

This paper introduces a dynamically reconfigurable intersection test hardware for virtual 3D objects. A static hardware design is rendered dynamically reconfigurable. The algorithm for the narrow phase of the intersection test is exchanged during the system's run-time, enabling intersection testing of objects constructed of varying primitives. In this article triangles and quadrangles are considered. The according primitive tests undergo a complete refinement from an untimed functional implementation down to a synthesisable RTL description. The impact of reconfiguration on timing behaviour, controlling, and topology of the system is considered in the process.

1. Introduction

Collision detection between graphical objects is a major bottleneck in all areas of physically based simulation. Most approaches firstly place objects at certain positions, check for collisions and then calculate forces and positions that remove the collisions. In conjunction with the hard real-time constraint to complete all collision checks within a simulation cycle, this reactiveness demands collision queries at tremendous rates. [14] report that up to 95% of the overall effort in physically based simulations is spent on collision detection. Since collision detection is such a performance hungry and yet fundamental task it is highly desirable to have hardware acceleration available. The benefit is an increased number of objects that can be processed per simulation cycle and therefore can populate the simulated environment. Additionally the CPU is liberated from processing collision queries.

This is underlined by the fact that a growing problem in high-performance computing is the increasing energy consumption of state-of-the-art processing devices. This ongoing trend favours specialised hardware in performance hungry areas of application, since these are in general far more energy efficient than calculations done on general purpose CPUs.

Still, collision detection algorithms are under heavy research and it is not yet clear what the final solution will be. It is very likely that there will not be a single algorithm that will be used for all possible cases but multiple different approaches being used simultaneously. Especially the choice of primitives used for modelling 3D environments is unlikely to be ultimately decidable, since they come with very different advantages and disadvantages. The most prominent primitives currently in use are triangles, quadrangles, point-clouds and NURBS. Hence, collision detection hardware needs to remain flexible, in such a way that different primitives can be used to model the underlying geometry. Even concurrent use of objects represented using different primitives is imaginable. In this case $\Omega(n^2)$ intersection test algorithms need to be provided, if n is the number of supported primitives. This directly leads to the use of dynamically reconfigurable hardware that allows exchange of the sub-system performing the primitive intersection test to avoid an explosion of the resource consumption.

Considerable work on collision detection in software was done in a wide variety of publications (e.g., [4, 13, 5, 11, 12, 7]).

Within the field, hardware acceleration is a recent development. [15] introduced the first hardware implementation of a collision test, while [17] presented the first FPGA architecture. Our architecture reuses parts of the latter, which will therefore be briefly recapped in Sec. 2.

The first commercially available dedicated hardware for

physically-based simulation in gaming-platforms was introduced to the market by [1]. Little detail was published and hence its inner structure is unknown to the public. In [22] an FPGA architecture is presented that pre-selects objects for collision detection. [2] presents a triangle intersection test hardware based on the common line interval approach, widely known as "Möller test". Since this is a rather sequential approach it is very resource consuming. It additionally relies on an extremely fast interface to the triangle data. Only triangle intersection testing is covered by the approach.

To the best of our knowledge this is the first publication on run-time reconfigurable collision detection hardware architectures. Thus it is the first work to provide both: hardware acceleration and the full flexibility of featuring intersection tests for multiple primitive.

There are multiple approaches towards description and simulation of dynamically reconfigurable HW, too. [19] introduce OSSS+R, an extension to OSSS [10], which itself is an extension to SYSTEMC, enabling object oriented HW development. In our setting some basic parts of the collision detection algorithm were already completely described in standard, component based SYSTEMC. Since OSSS+R models reconfiguration via polymorphism it imposes the use of the object orientation paradigm and thus does not allow reuse of components.

The ADRIATIC project [21] targets transaction level (TL) description and simulation of reconfiguration aspects, focusing on early evaluation of the performance impact of reconfiguration. An automated tool is introduced that analyses static TL models and identifies components that could be made reconfigurable. In [3] a modified SYSTEMC kernel is presented that allows enabling and disabling of processes. It is used for modelling and simulation of a reconfigurable car application. Here process disabling is executed unconditioned whenever it is requested. No additional means for implementing the necessary synchronisation is provided and thus render the usage a time-consuming task.

Both, [21] and [3], leave open how the refinement needs to proceed from this point to yield a HW-implementable description.

Most recently [16] introduced the SYSTEMC extension RECHANNEL. Since it enables both component reuse and description of reconfiguration on all levels of abstraction including synthesisable RTL, it is a native choice for a refinement case study.

In the following Section a brief introduction into the collision detection architecture is given. Sec. 3 presents the refinement of the dynamically reconfigurable primitive intersection test sub-system (primtest). To maintain synthesisability of the intersection test modules with standard SYSTEMC tools, their source codes are not manipulated (i.e., they are treated as closed source components).

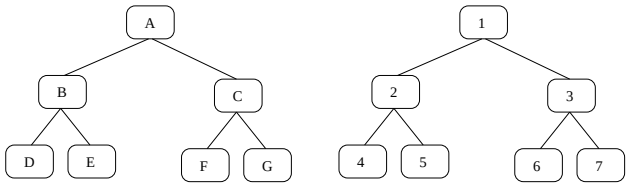

Figure 1

2. The Collision Detection Architecture

A two-staged hierarchical collision detection algorithm for rigid objects is implemented in this paper. To avoid checking each primitive of object O against each primitive of object Q for intersection a bounding volume hierarchy (BVH) for each object is used. In the BVH each leaf represents a primitive. A node represents a bounding volume (BV) which encloses all primitives in its subtree. To achieve an efficient hardware implementation a binary tree is used but n-ary trees could also be considered. Discretely oriented polytopes with 24 orientations (24-DOPs) are used as bounding volumes because they performed well in software [23] and hardware [17, 15]. To check object O and object Q for collision their BVHs are traversed simultaneously. If two BVs intersect, the next level of the BVHs is checked for intersection. This is the algorithm's broad-phase. In the narrow phase the resulting leaves (i.e., the primitives) have to be checked against each other. Fig. 1 shows two BVHs.

Specialized variants of the Separating Axis Test (SAT) [9, 20] are used to check DOP pairs and primitive pairs for intersection. It is a general way to test two convex polytopes for intersection by projecting them on several axes. If the projections do not intersect, the polytopes do not intersect either. According to [8] it suffices to test all axes orthogonal to a face of either polytope (i.e., DOP or primitive) or orthogonal to an edge of each polytope. If all of the resulting intervals overlap, the polytopes intersect.

Fig. 2 shows the architecture of the COLLISIONCHIP-design. An application running on a host PC communicates over a PCI interface with the hardware part. The traversal of the BVHs is controlled by a specilized controller module which determines the data that has to be loaded next. This information is passed to the getData module which fetches the requested data from the DDR-RAM and routes it to the corresponding pipeline. In case of DOPs the data is routed to the BV-test pipeline. In case of primitives it is routed to the primtest pipeline. A major bottleneck

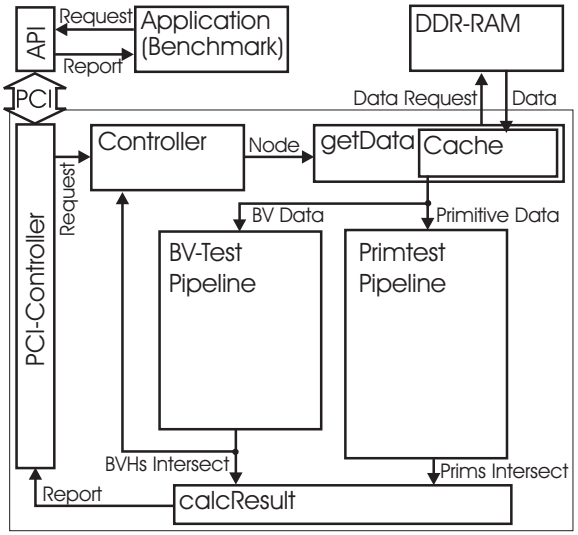

Figure 2

in this scheme is the communication between the DDR-RAM and the two pipelines. Therefore a lockable two-way set-associative cache (LTA) [18] is implemented in the getData module. The calcResult module combines the results of the BV-test pipeline and the primtest pipeline and calculates the overall intersection result for the current object pair. This result is reported back to the host PC via the PCI interface. Details on the hierarchy traversal and the bounding-volume test were previously published in [17]. Thus we will concentrate on the primitive tests in the following.

3. Refinement

In the following we will discuss intersection tests for three primitive combinations: Triangle-triangle, quadrangle-quadrangle, and triangle-quadrangle. As previously discussed, the separating axis test is a general way to test two convex polytopes for intersection. Although triangles and quadrangles are 2D objects embedded in 3D both can be interpreted as (degenerated) convex polytopes. Thus they are treated very similarly and only the triangle-triangle test is discussed in detail.

Let T_A and T_B be triangles with vertices $V^0, V^1, V^2 \in \mathbb{R}^3$, and $W^0, W^1, W^2 \in \mathbb{R}^3$ respectively. In case they are in general position it suffices to test 11 axes. In the following, wrap-around indexing is used for the vertices. Let k, l $\in \{0, 1, 2\}$. Normals n^A, n^B and cross-edge-axes $C_{k,l}$ are then calculated using

$$\begin{aligned} n^A &:= \left(V^1 - V^0\right) \times \left(V^2 - V^0\right) \\ n^B &:= \left(W^1 - W^0\right) \times \left(W^2 - W^0\right) \\ C_{k,l} &:= \left(V^{k+1} - V^k\right) \times \left(W^{l+1} - W^l\right) \end{aligned} \tag{1}$$

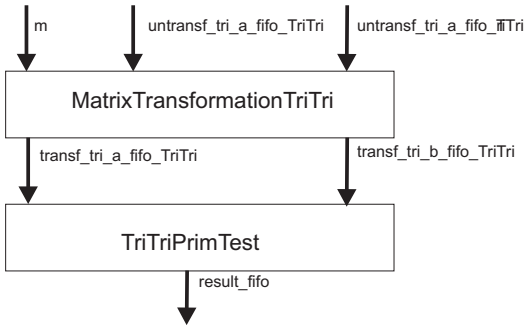

Figure 3

Then the triangles' points are projected onto any test axis L out of this set by

$$\begin{aligned} p_{V^k} &:= V^k * L \\ p_{W^l} &:= W^l * L \end{aligned} \tag{2}$$

The projection intervals on the axis are

$$\begin{aligned} I_A &:= \left[\min\left\{p_{V^0}, p_{V^1}, p_{V^2}\right\}, \max\left\{p_{V^0}, p_{V^1}, p_{V^2}\right\}\right] \\ I_B &:= \left[\min\left\{p_{W^0}, p_{W^1}, p_{W^2}\right\}, \max\left\{p_{W^0}, p_{W^1}, p_{W^2}\right\}\right] \end{aligned} \tag{3}$$

Due to definition 1 every test axis is orthogonal to at least two triangle edges. Thus the projections of the two triangle points defining this axis are identical. This is used to reduce the number of necessary point projections to four. The number of comparisons decreases accordingly.

To avoid coplanarity problems the triangles can be extruded to 3 dimensional quadrihedra of thickness ε by treating the cross products of normals and edge orientations as additional normals [6]. This results in 6 additional axis tests. Moreover, it comes with almost no extra development effort.

For initial testing in conjunction with the bounding-volume test, the primitive tests were implemented in a plainly functional description style. In the given framework, the relative position and orientation of the two primitives under test is given by a matrix m. Thus we need to transform one of them into the relative coordinate system of the other by applying this matrix. To enable fast implementation data is transferred via FIFOs exclusively. Fig. 3 shows the resulting subsystem.

3.1. Rendering the Design Reconfigurable

The RECHANNEL library introduces some new language constructs to enable description of reconfiguration in SYSTEMC. The main constructs used in the following are `rc_reconfigurable`, `rc_portal`, and `rc_control`. The following is done for all primtests.

Preparing the Modules - `rc_reconfigurable` enables declaration of reconfigurable components, which

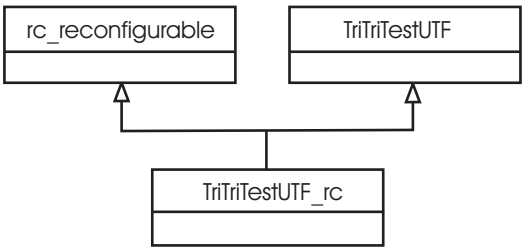

Figure 4

can then be equipped with reconfiguration properties such as reconfiguration timings and handshaking behaviour. To yield modules that can be reconfigured and are functionally equivalent to the static primitive tests, they are derived from the according test module and `rc_reconfigurable`. Fig. 4 shows this derivation for the untimed functional triangle-triangle test.

Adding Portals - Instances of `rc_portal` (portals) enable the binding of ports of multiple reconfigurable components to the interface of a single static channel. They allow re-channeling the data streams from and to these components according to their current reconfiguration state. Components that are exclusively connected to portals can communicate with their environment only if they are currently active (configured). This enables correct simulation behaviour while also providing an abstract model of low-level reconfiguration primitives, such as bus macros. Therefore it is necessary to provide all ports of all primtests with portals and connect those to the according channels. In the first refinement step the primtests are described in an abstract style and communicate via blocking accesses to their ports. In this case additional synchronisation is usually required, to prevent deactivation of a module which has already read data from a port, but has not written the according output yet. But the tests connect to their environment exclusively via FIFOs and thus via FIFO-portals. The latter prevent deactivation of their associated module by default, if the FIFO is not empty. Since within the given framework, primitive tests will only be deactivated between two object tests, the FIFOs will always be empty if a deactivation is requested. Therefore the built-in synchronisation mechanism suffices to guarantee correct behaviour and no additional synchronisation is necessary.

Configuration Control - The RECHANNEL library distinguishes two aspects of simulation controlling. Hardware properties of the reconfigurable hardware (i.e., the FPGA and its host system) are represented by the class `rc_control`. Instances of `rc_control` manage reconfigurable components and allow the designer to activate and deactivate them, while respecting their reconfiguration timings. Therefore, they allow for the description of custom configuration controllers (C^3), which represent the second aspect of the controlling. Here the designer needs

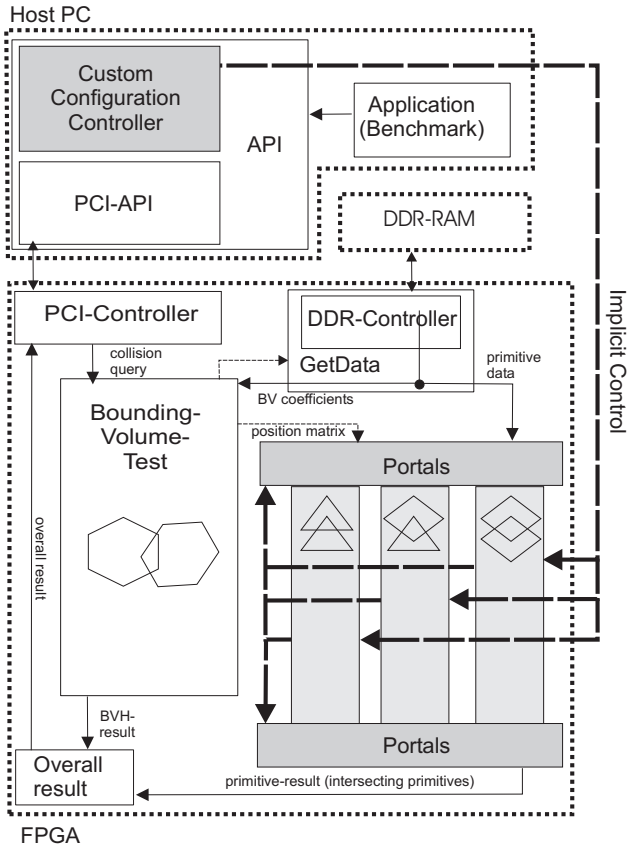

Figure 5

to specify how reconfiguration requests are arbitrated.

In our design the C^3 needs to activate the correct primtest depending on the primitive types of the object pair currently under test. The API is the only instance within the design which needs to be informed of the type of primitives an object is constructed of, disregarding any reconfiguration issues. Therefore the simplest approach is to implement the C^3 as a part of the API. This might not be fast enough to maintain a reasonable system-performance, because the API is supposed to run in software. Therefore the performance impact of this decision could be evaluated when the refinement proceeds. Since the development overhead of a plainly functional C^3 description is very low, this would not significantly increase development time. But, as will be discussed in the following, this is not necessary in the given scenario. The reconfigurable system is depicted in Fig. 5.

3.2. Performance Evaluation

As was discussed in [17] the static collision detection architecture, using triangle objects exclusively, accelerates collision queries by a factor of four, when compared to software. Although the dynamic system will perform iden-

tically if not reconfigured, it is of major importance to exclude that the system performance is impaired by reconfiguration.

For this, the configuration delay needs to be modelled. This is best done by modelling the timing behaviour of the reconfigurable target hardware in use. Therefore, we derive a specialised simulation controller from `rc_control`, implementing the delay of configuring a module depending on its bitfile size.

We assume the usage of a XILINX VIRTEX II FPGA on an ALPHA-DATA ADM-XRC II board connected via PCI to a host PC. The target architecture provides means to (re-)configure the on-board FPGA using API calls, which transmit the according bitstream via PCI and local bus. The transmission delay of the module's bitstream via the buses can be neglected, since the on-board configuration controller is the limiting element. The configuration itself is executed in 8-bit chunks at a maximum frequency of 50 MHz and thus the configuration delay $D_{configuration}$ of a bitstream of size B can be approximated as follows:

$$D_{configuration}\ \text{[ns]} \approx \frac{B\ \text{[Bit]}}{8\ \text{[Bit]}} \cdot \frac{1}{50\ \text{[MHz]}} \qquad (4)$$

This is modelled within the simulation controller derived from `rc_control`. The bitfile sizes of the primtests are results of a previous static VHDL implementation and are added to the modules as additional `sc_attributes`. If the exact bitfile sizes were unavailable, high-level synthesis results could be used as worst-case estimates. Although the bus delay is neglected in approximation 4, it is respected that the target architecture provides a single PCI interface for reconfiguration and data transmission. If a reconfiguration needs to be performed the data's transmission delay and the reconfiguration delay sum up to an overall initialisation delay.

$$D_{\text{overall}} = D_{transmission} + D_{configuration} \qquad (5)$$

In the calculation of the transmission delay the board's local bus is the limiting element. Since it transmits data in 32-bit chunks, the transmission delay $D_{transmission}$ of a data packet of size P-bit of data at the maximum bus frequency of 66MHz can be approximated as follows.

$$D_{transmission}\ \text{[ns]} \approx \frac{P\ \text{[Bit]}}{32\ \text{[Bit]}} \cdot \frac{1}{66\ \text{[MHz]}} \qquad (6)$$

Now we compare the dynamic architecture to the static one which uses the identical transmission delay, no configuration delay, and implements *two* primtests (triangle-triangle and quad-quad). Note that these two primtests do not operate concurrently in this evaluation, since we do not consider mixed-primitive objects. The static implementation is still advantaged, since twice the chip-area would have to be spent for primitive testing.

Design	**Bitfilesize** $[bit]$
Triangle-triangle intersection test	5,504,088
Quad-quad intersection test	5,409,976
Simulation	**Run-time** $[ns]$
Two static primitive tests	274,948,800
Reconfiguration using a single Bus	274,887,700

Table 1

Table 1 presents the architecture run-times, along with the bitfile sizes used for simulation. First two triangle objects were tested for intersection, followed by two quadrangle objects. This requires reconfiguring once per object pair and thus is a worst-case scenario, which overestimates the performance impact of reconfiguration. Still, the run-time is increased when using reconfiguration by less than 0.1%. Further optimisations, e.g., implementing the C^3 in HW, will therefore not result in any significant speed-up of the overall design and are discarded. It can be concluded that reconfiguration is an effective way to enable use of multiple primitive intersection tests without impairing the responsiveness of the system.

3.3. Synthesizable RTL

Proceeding to RT-level the interface of the intersection tests changes significantly, since the FIFO communication needs to be substituted by signals.

To enable implementation of the SAT algorithm on RT-level, it can be sequenced into identical calculations: One for each axis. Thus in the worst case it takes as many clock cycles to process the overall result as axes are considered.

Fig. 6 shows the refined triangle-triangle test. All interfaces are now composed of signals (Note, that the individual names are of no further relevance in the following). The primitive test itself is divided into a controller, which sequences the axes tests and a pipeline module, which does the actual projection.

Fig. 7 shows the division of the (triangle-triangle) pipeline into macro stages in a simplified block-diagram. In the first stage the triangle points used for the generation of the current axis are chosen. Since the test axes are generated from triangle points, there are always two pairs of points whose projections are identical. Thus only four out of the six triangle points need to be projected per axis test. Since they differ according to the axis under test they also need to be multiplexed. This is also done in the first macro stage of the triangle pipeline.

In the pipeline's second macro stage the test axes are actually generated from the points singled out in the previous stage. This consumes six multipliers and nine adders. The third stage implements four vector multipliers used to

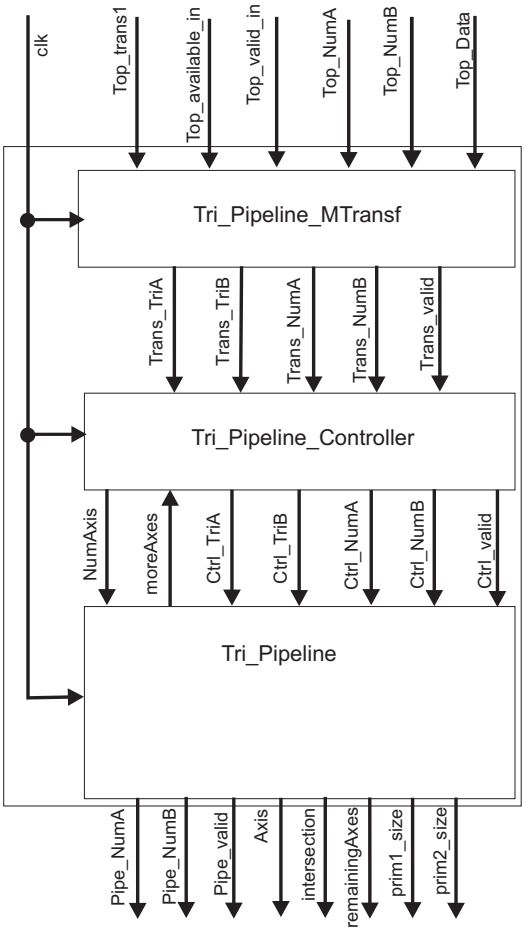

Figure 6

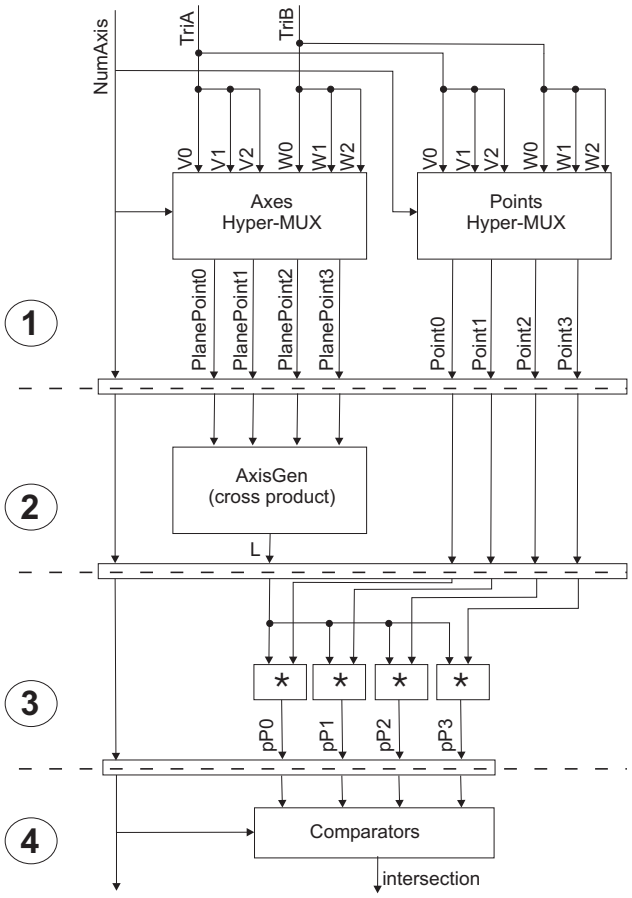

Figure 7

project the four triangle points onto the test axes. The implementation of the axis generator and the vector multiplier modules are straight forward hardware implementations of the according equations. They are further divided into ten substages to maximise pipeline throughput. They consume three multipliers and two adders each.

Stage four compares the results. Due to the projection of only four points, the line intervals need to be generated differently for different axes. This is done with respect to the current test axis, consuming only two adders.

Topology - Since the FIFO communication is substituted by signals it is necessary to exchange the portals in use accordingly. RT-level interfaces require a vast number of signals. A direct consequence is the requirement of as many portals, which not only need to be instantiated, but need to be bound, too. Since multiple reconfigurable modules need to be bound to the dynamic end of theses portals, the number of binding statements multiplies accordingly. Implementing this is tedious, error-prone work and the resulting code is difficult to understand and maintain. Therefore it is advisable to reduce the number of necessary binding statements. For this RECHANNEL provides grouping of ports and grouping of portals. The modules can now provide all of their ports as a single port group. Thus a complete port group of a module can be bound with a single statement to the according portal group. This effectively simplifies binding and renders the resulting description easy to understand and maintain.

External Reset - As previously pointed out, the implementations of the intersection tests are treated as closed source components. A limitation in rendering static closed source components reconfigurable is the necessity for an externally initiated reset, since RECHANNEL does not feature any reset-on-configuration (ROC) for components that do not provide this functionality themselves. If the designer does not consider this, the component will behave unpredictably.

Thus an additional process is implemented after derivation of the primitive tests from `rc_reconfigurable`. Now the RECHANNEL process macros `RC_METHOD` and `RC_THREAD` can be used. They behave identically to their SYSTEMC counterparts, but are reset on configuration by

```
virtual void write_roc() {
 roc_out.write( true );
 wait(); wait(); // wait two clock cycles
 roc_out.write( false );
}
```

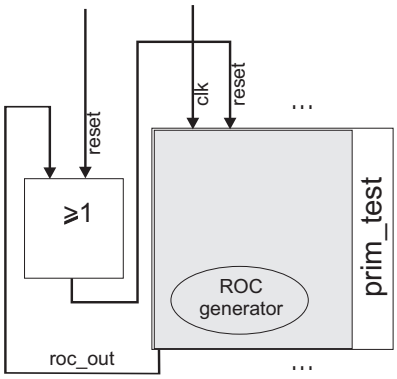

Figure 8

the library.

The additional process generates a reset every time the test is configured. The according signal is externally fed back into the reset port of the primitive test. Listing 3.3 shows the implementation of the very simple ROC generator, while Fig. 8 illustrates how it integrates with the primitive test. As can be seen, only a minimum amount of additional logic and development effort is necessary.

4. Conclusion

This paper introduced a dynamically reconfigurable intersection test hardware for virtual 3D objects. The narrow-phase intersection test is exchanged during run-time, while the broad-phase test remains in operation. This enables intersection testing of objects constructed of varying primitives, such as triangles and quadrangles. The timing analyses of the resulting dynamic system shows that the run-time is increased by the use of reconfiguration by less than 0.1%. Although it provides the flexibility to model geometry with multiple primitives the responsiveness of the system is not impaired. It can be concluded that reconfiguration is a very efficient way to enable use of multiple primitive intersection tests. The primitive tests were refined from an un-timed functional implementation down to a synthesizable RTL description. To provide synthesizability of the intersection tests with standard tools they were written in plain SystemC. Thus they were treated as closed-source components in all aspects concerning ReChannel and reconfiguration. Therefore an external reset mechanism is presented, enabling reset-on-configuration for closed source components. This effectively proves the applicability of the refinement methodology proposed by the ReChannel approach.

5. Future Work

An open problem in collision detection is the fact that rounding errors in the calculation can lead to false reports of non-collision (false negatives). This type of error can render a physically-based simulation using the according algorithm unstable. Although this occurs only very occasionally, it is unacceptable in safety-critical applications. We are currently developing a triangle intersection test which classifies its results as being a definitive result or an educated guess.

Another interesting topic is the inclusion of deformable objects. To enable deformation in hardware accelerated collision detection, it is necessary to provide an architecture which is able to construct BVHs from objects.

Furthermore, we will evaluate applicability of hardware acceleration for intersection testing of further kinds of primitives, like NURBS and point-clouds.

References

[1] Ageia. White paper, May 2005.

[2] N. Atay and B. B. John W. Lockwood. A Collision Detection Chip on Reconfigurable Hardware. In *13th Pacific Conference on Computer Graphics and Application*, 2005.

[3] A. V. Brito, M. Kuhnle, M. Hubner, J. Becker, and E. U. K. Melcher. Modelling and Simulation of Dynamic and Partially Reconfigurable Systems using Systemc. *ISVLSI*, 00:35–40, 2007.

[4] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In P. Hanrahan and J. Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 189–196. ACM SIGGRAPH, Apr. 1995. ISBN 0-89791-736-7.

[5] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6:381–392, 1985.

[6] D. Eberly. Intersection of Convex Objects: The Method of Separating Axes. Oktober 2007. `http://www.geometrictools.com`.

[7] S. Fisher and M. Lin. Fast penetration depth estimation for elastic bodies using deformed distance fields. In *Proc. International Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[8] S. Gottschalk. Separating Axis Theorem. Technical Report TR-96-024. Technical report, 1996.

[9] S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, pages 171–180. ACM SIGGRAPH, Addison Wesley, Aug. 1996.

[10] E. Grimpe and F. Oppenheimer. Aspects of Object Oriented Hardware Modelling With SystemC-Plus. In *System on Chip Design Languages. Extended papers: Best of FDL'01 and HDLCon'01.*, pages 213–223. Kluwer Academic Publ., 2002.

[11] L. J. Guibas, D. Hsu, and L. Zhang. H-walk: Hierarchical distance computation for moving convex bodies. In W. V. Oz and M. Yannakakis, editors, *Proc. ACM Symposium on Computational Geometry*, pages 265–273, 1999.

[12] T. Larsson and T. Akenine-Möller. Collision detection for continuously deforming bodies. In *Eurographics*, pages 325–333, 2001. short presentation.

[13] B. Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Trans. Graph.*, 17(3):177–208, July 1998.

[14] E. Plante, M.-P. Cani, and P. Poulin. A layered wisp model for simulating interactions inside long hair. In N. M.-T. Marie-Paule Cani, Daniel Thalmann, editor, *Computer Animation and Simulation 2001Proceeding*, Computer Science. EUROGRAPHICS, Springer, Sept. 2001. Proceedings of the EG workshop of Animation and Simulation.

[15] A. Raabe, B. Bartyzel, J. K. Anlauf, and G. Zachmann. Hardware Accelerated Collision Detection — An Architecture and Simulation Results. In *Design Automation and Test (DATE)*, pages 130–135, Munich, Germany, Mar.7–11 2005. IEEE Computer Society.

[16] A. Raabe, P. A. Hartmann, and J. K. Anlauf. Rechannel: Describing and Simulating Reconfigurable Hardware in SystemC. *ACM Transactions on Design Automation of Electronic Systems*, 13(1):1–18, 2008.

[17] A. Raabe, S. Hochgürtel, G. Zachmann, and J. K. Anlauf. Space-Efficient FPGA-Accelerated Collision Detection for Virtual Prototyping. In *Design Automation and Test (DATE)*, pages 206–211, Munich, Germany, 2006.

[18] A. Raabe and F. Zavelberg. Defying the Memory Bottleneck in Hardware Accelerated Collision Detection. In *WSCG '2008*, University of West Bohemia, Plzen, Czech Republic, 2008.

[19] A. Schallenberg, F. Oppenheimer, and W. Nebel. Designing for dynamic partially reconfigurable FPGAs with SystemC and OSSS. In *Forum on Specification and Design Languages*, Lille, France, Sept. 2004.

[20] G. J. A. van den Bergen. *Collision Detection in Interactive 3D Computer Animation*. PhD dissertation, Eindhoven University of Technology, 1999.

[21] N. S. Voros and K. Masselos. *System Level Design of Reconfigurable Systems-on-Chip*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[22] M. Woulfe, J. Dingliana, and M. Manzke. Hardware accelerated broad phase collision detection for realtime simulations. In J. Dingliana and F. Ganovelli, editors, *VRIPHYS 2007: 4th Workshop on Virtual Reality Interaction and Physical Simulation*, 9–10 Nov. 2007.

[23] G. Zachmann. Rapid collision detection by dynamically aligned DOP-trees. In *Proc. of IEEE Virtual Reality Annual International Symposium; VRAIS '98*, pages 90–97, Atlanta, Georgia, Mar. 1998.