Specialized Hardware for Deformable Object Modeling

Benjamin Bishop Thomas P. Kelliher
Department of Computer Science
The University of Georgia
Athens, GA
{bishop@cs.uga.edu, kelliher@bluebird.goucher.edu}

Abstract— **Deformable object modeling is a technology that will enable a broad range of new applications. Unfortunately, current techniques are far from being able to offer interactive performance for realistic scenes. In this paper, we examine the idea of using low-cost, highly specialized hardware in order to accelerate deformable object modeling. Details are presented regarding two generations of such experimental systems.**

I. Introduction

Recently, there has been a great deal of interest in physical-based modeling from the graphics research community. In the context of graphics, physical modeling is used to simulate the behavior of objects for the purpose of animation. Physical modeling applies to a broad range of applications with highly variable requirements.

Much of the previous work in physical modeling has come from groups interested in generating animation sequences for use in the film industry. However, there are many untapped, emerging application areas such as: warfare simulation, interactive entertainment, and virtual reality for example. It is important to note that the requirements for these applications differ greatly. In physical modeling for film, interactivity is not required and a large number of machines could potentially be used. On the other hand, consumer-grade electronic entertainment requires interactivity and very low hardware cost.

In physical modeling of solid objects, a number of simulation techniques exist. In rigid body simulation, objects are not allowed to deform. This lack of deformation allows a great deal of precomputation, which makes interactive simulation possible for certain scenes. However, the lack of deformation can be an important limitation. Deformable object techniques allow for the simulation of a much wider variety of materials, but have very high computational requirements.

In this work, we explore the idea of using low cost, highly specialized hardware in order to achieve interactive deformable object modeling performance for complex scenes. Figure 1 illustrates the complexity that is present in real scenes.

Thomas P. Kelliher is with Goucher College, Baltimore, MD

Fig. 1. An example of a complex scene

II. Previous and Related Work

A wide variety of previous work is relevant to this project. A number of algorithms, general purpose CPU extensions and specialized hardware systems relate to this work.

A. Algorithms

In addition to solid objects, a number of research groups are working toward interactive fluid dynamics simulations. Similar to deformable solid object modeling, the computational requirements for this type of simulation are high for reasonably complex flows. Stam [1] gives a particularly interesting example of work in this area.

In the area of solid object modeling, currently rigid body techniques [4], [2], [3] are very popular. This is because they perform well on even general purpose hardware. Since the objects do not deform, the only significant computational challenge is collision detection. However, precomputation can be used in order to handle this efficiently. For example, one possibility would be to use a hierarchy of object-oriented bounding boxes. Lin et al. [8] give a good survey of collision detection.

There has also been a great deal of previous work in algorithms for deformable object modeling [5], [6], [7]. While more computationally expensive, this simulation method is more versatile than the rigid body approach. Care must be taken when using these techniques to guarantee stability, allow for stiffness, and achieve high performance. For example, one approach would be to represent the object as a system of masses connected by springs. Hookes law would govern the behavior of the system. Using implicit integration (for stability), a sparse system of equations is generated. This system can be linearized and an iterative method can then be applied in order to update the system for the next simulation step.

B. General Purpose Hardware

One important development in the area of physical modeling on general purpose hardware is the introduction of special streaming floating-point instructions [9], [10]. These extensions add a number of SIMD floating point instructions, allowing for greater floating point performance. However, this approach results in a rather incremental performance improvement and compiler support can be problematic.

A number of new 3D graphics accelerator cards are now offering some on-board programmability. In the future, it may be possible to perform physical modeling directly on the graphics card, which would avoid bus bottlenecks and could potentially offer higher floating point performance. However, this scheme too is unlikely to offer orders of magnitude improvement in floating point performance.

C. Specialized Hardware

In application domains other than physical modeling, there have been some examples of specialized hardware systems that greatly exceed general purpose system performance for a small number of applications.

The GRAPE [11] (GRAvity PipE) project is a family of experimental systems for gravity computation developed at the University of Tokyo. The N-body gravity computations are used in order to simulate astronomical phenomena, such as galaxy formation. The current GRAPE-6 system is reported to achieve a peak performance of 100 Tflops.

One additional example is that of 3D graphics hardware. 3D graphics cards achieve very high performance at a very specific task (graphics pipeline). The commercial success of these cards illustrates the point that some types of specialized hardware are viable if they offer significant performance advantages over general-purpose machines.

III. Our Work

The basic idea of this project is to construct a specialized hardware system (similar to the GRAPE) in order to accelerate physical modeling. Ideally, this acceleration would allow for the simulation of complex scenes at interactive speeds.

Functional Unit	Number
Adder	11
Multiplication	9
Square Root	1
Reciprocation	1
Memory	1
Misc	5

TABLE I
Functional Unit Details

In order to evaluate these ideas, we constructed a proof-of-concept system, which implements a mass-spring deformable object simulation. This system uses a forward Euler solver. The system was implemented on a high-density Altera EPF10K250 FPGA (250k gates) using AHDL. This FPGA was placed on a custom printed circuit board (PCB), which was then connected to a host machine (for graphic output) via a parallel cable. Figure 2 shows the final populated PCB.

The organization of the specialized processor is pipeline-oriented. The idea is to take the Euler solver function and basically unroll it into a pipeline on the FPGA. Figure 3 shows a diagram of the solver pipeline. Table I shows the number and type of functional units used in the processor. Unfortunately, it is impossible to give speed/area figures for the functional units since this would vary depending on placement within the FPGA.

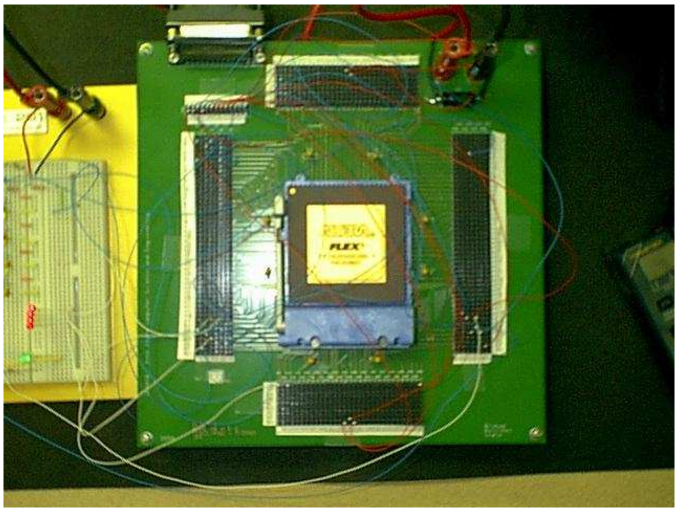

Fig. 2. The final printed circuit board

The biggest problem during the project was utilization on the FPGA. This was a result of the well-known problems in implementing floating point arithmetic in programmable logic [12]. In a nutshell, poor utilization results from the high interconnect requirements for shifting in floating point arithmetic.

In order to compress the design so that it would fit on the FPGA, a number of simplifications were necessary. These simplifications include a removal of pipeline registers, two-dimensional simulation, reduced floating point precision

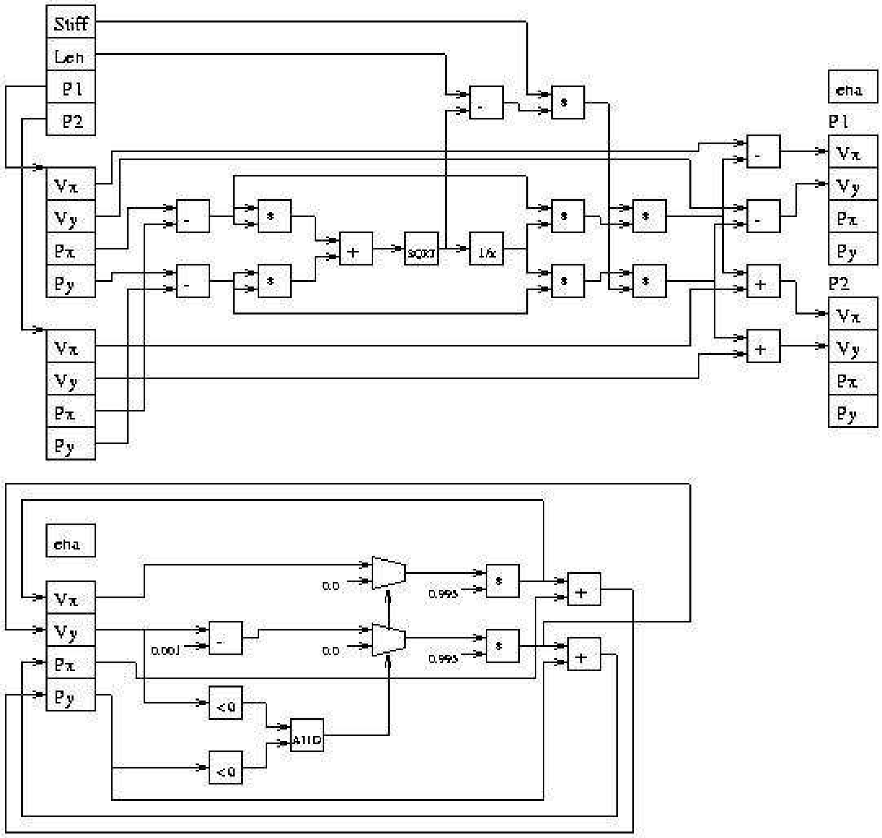

Fig. 3. A diagram of the processor architecture

(the mantissa was reduced to get a 16 bit floating point word), and simple collision detection. The final FPGA utilization was 79%.

After the simplifications were made, the project was completed and successfully tested. Figure 4 shows an animation sequence that was generated on the hardware system. We can examine the performance of this system in comparison to existing general purpose machines. For the 2D simulation above, a Pentium II/300mhz machine can achieve .32M iterations per second. If we assume pipelining and 300mhz operation, we calculate that the specialized system could achieve 30M iterations per second. This represents a 92X speedup over the general purpose machine. This would, in fact, be higher for 3D simulation since the pipeline could exploit greater parallelism.

IV. Next Generation System

A number of lessons were learned in the construction of the first generation proof-of-concept system. We are designing the next generation system in order to make improvements based on this information. The most important distinctions are that the new system must implement an implicit solver (stability in forward Euler is very poor) and will be constructed as a custom integrated circuit (IC).

A. Description of Architecture

The custom integrated circuits will be fabricated using TSMC through MOSIS. Several of these chips will populate a custom printed circuit board. This board will then be connected to a Linux host system through the PCI bus. The bus interface will be implemented as an FPGA using a PCI core. The PCI interface will increase the project complexity, but it is necessary in order to supply simulation results to the host quickly enough to achieve real-time display updates.

A.1 Processor Element Cell Design

The goal of new architecture is to achieve the highest performance at low cost for iterative floating point intensive computation. Since the codes typically perform a large number of relatively simple iterations, it is possible for the new system to distribute computation to a large number of independent processing elements. An efficient schedule of operations for each iteration can be determined a priori and stored locally in each processing element.

The processing element will consist of a floating point functional unit, operand memory, control memory, and a simple control unit. Most processing elements will contain a multiply accumulate (MAC) functional unit. Although there will also be a divide/reciprocation unit as well, simu-

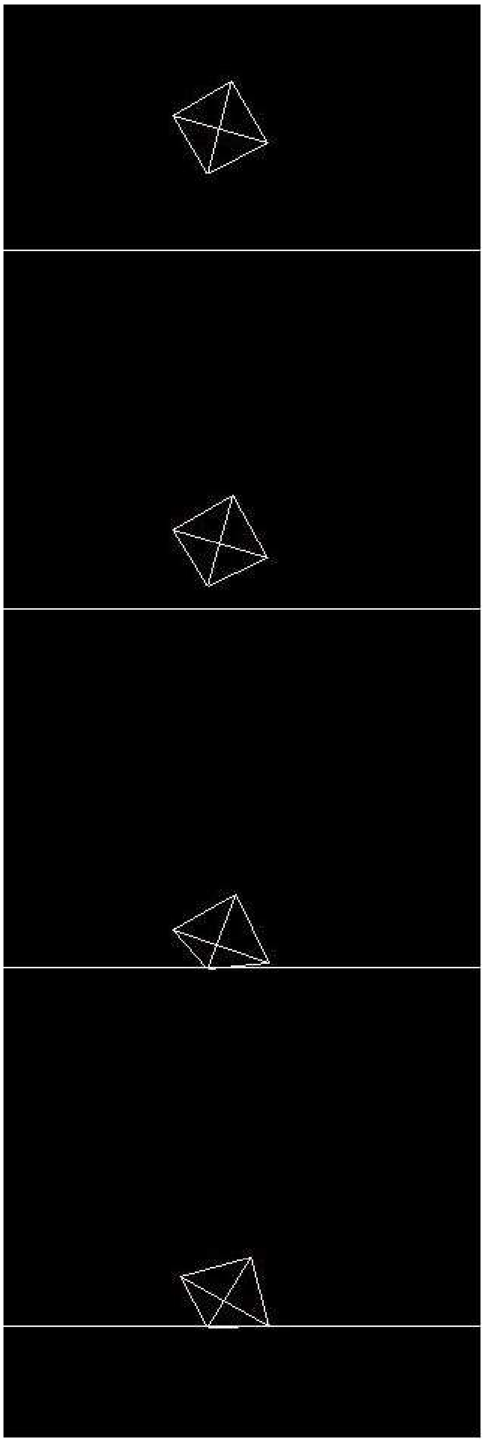

Fig. 4. Animation sequence generated in hardware

lations show that divide is rare. Figure 5 shows a diagram of the single chip organization.

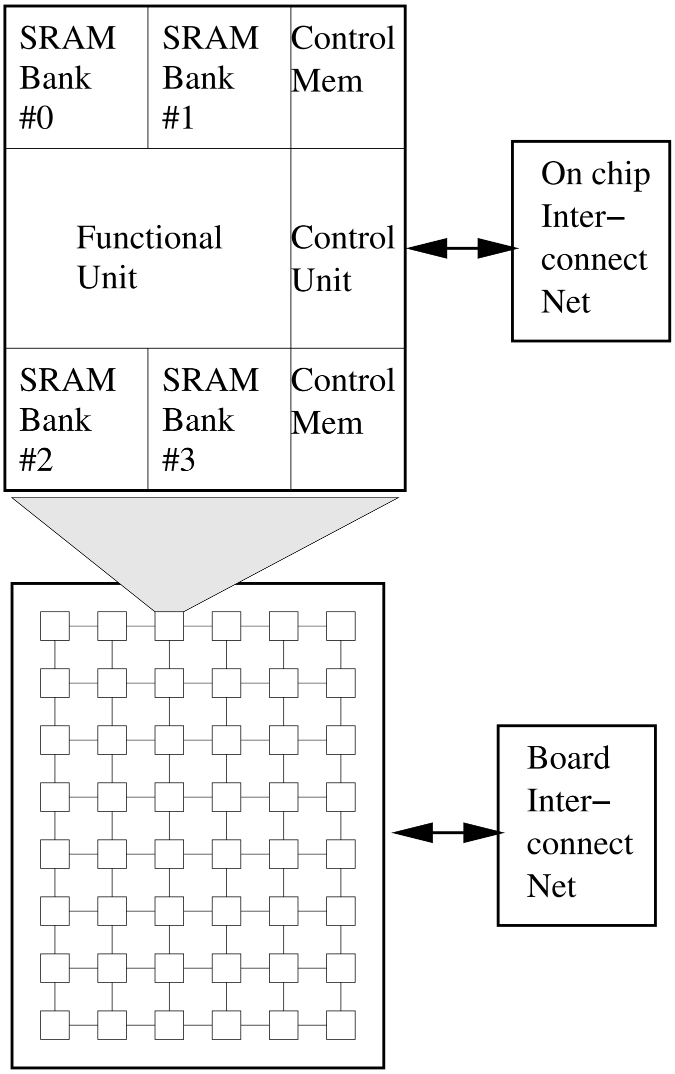

Fig. 5. Diagram of custom integrated circuit

In each processing element, there will be local SRAM for operand and control storage. Distributing a large amount of high speed, high bandwidth SRAM such as this will offer very high memory performance for the target application. In order to achieve optimal performance, it is estimated that the operand storage should contain four banks of 128 32-bit words. The four banks will allow for operations in the form A * B + C $\Rightarrow$ D. The control memory will be separate, and relatively small in comparison.

Since most of the processing element area is consumed by memory, memory cell area is of critical importance. DRAM offers considerable advantages in this regard. However, SRAM has be selected initially for robustness. Specifically, we are concerned about noise affecting sensing on the memory bitlines.

A.2 Communication

Since computation will be distributed over a number of processing elements, efficient communication is very important. A simulator is being used to evaluate different interconnection schemes at both the board and chip level.

At the board level, interconnect will likely be more limited than within the IC. A large number of 32 bit buses will cause packaging difficulties, increase chip to chip delay due to routing congestion, and will increase heat dissipated in the chip I/O pads.

Within the IC, the interconnect method may be more flexible. The primary concern is to allow for efficient communication without significantly increasing area overhead. Delay is not expected to be a problem since the operand SRAM bitlines will have a higher capacitive load. The main interconnect options being evaluated are a nearest neighbor style vs. a hierarchical approach.

Using a nearest neighbor interconnect strategy, processing elements could be efficiently connected to their cardinal neighbors (north, south, east, west). This would result in a simple and regular layout. In a nearest neighbor interconnect strategy, a number of options exist for treating boundaries (torus, wrap, etc.), however, in this design wiring complexity could be reduced by disallowing on-chip boundary communication.

A hierarchical or tree-based approach would offer the advantage of faster communication between distant neighbors. This interconnection method is commonly used in modern FPGAs. Hierarchical communication tends to be necessary when the number of processing elements are large (as in fine grain FPGAs). Given the fact that the processing element organization is relatively coarse, the additional overhead for this scheme may be difficult to justify.

In each method, a processing element must be allowed to quickly broadcast its data to all other processing elements. This rules out 1-dimensional interconnection methods. However, broadcast could be performed over multiple cycles by rippling through a nearest neighbor network, for example.

Clock distribution will also be an important consideration. The primary concern, with respect to this project, is to minimize the probability of a system crippling design error. Since PLL design is a fairly complex and error-prone process, it may be safer to combine two (or more) out-of-phase board level clock signals in the IC.

A.3 Power/Heat Considerations

This project attempts to pack as much computation into as small a space as possible. This computation density should lead to a higher level of switching activity than what is seen in general purpose processors. Therefore, power consumption and heat generation may become problematic. Indeed, experience with the MGAP-II [13] has shown that this sort of specialized processor can generate excessive heat. In the case of the MGAP-II, this heat generation constrained the clock rate.

It may be possible to apply a number of techniques in order to reduce the effect of high switching activity. These techniques include heat-tolerant packaging, low-swing interconnect, and possibly even scheduling operations in order to reduce switching activity.

V. Conclusions

We have presented details on two physical modeling experimental systems. The first system, which was implemented on an FPGA, proves the concept that specialized hardware can be used to dramatically improve the performance of physical modeling codes. We estimate a 92X speedup over a general purpose machine. The second generation system incorporates lessons learned from the first generation. This system is more flexible and will allow for stable implicit solvers.

References

[1] J. Stam, "Stable Fluids", *Proceedings of the Conference on Computer Graphics* pp. 121-128, 1999.

[2] D. Baraff, "Fast Contact Force Computation for Nonpenetrating Rigid Bodies", *Computer Graphics Proceedings*, July 1994.

[3] B. Mirtich, "Timewarp rigid body simulation ", *Proceedings of the conference on Computer Graphics*, pp. 193-200, 2000.

[4] W. Armstrong, M. Green, "The dynamics of articulated rigid bodies for purposes of animation", *The visual computer*, Springer-Verlag, 1985.

[5] A. Witkin, W. Welch, "Fast Animation and Control of Nonrigid Structures", *Computer Graphics*, Vol. 24, No. 4, August 1990.

[6] D. Baraff, A. Witkin, "Large Steps in Cloth Simulation", *Computer Graphics Proceedings*, July 1998.

[7] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, "Elastically Deformable Models", *Computer Graphics*, Vol. 21, No. 4, July 1987.

[8] M. Lin, S. Gottschalk, "Collision Detection between Geometric Models: A Survey", *Proc. IMA Conference on Mathematics and Surfaces*, 1998.

[9] "Inside 3DNow! Technology", `http://www.amd.com/products/cpg/k623d/inside3d.html`

[10] "Discover the New PentiumIII Processor" `http://developer.intel.com/design/PentiumIII/prodbref/`

[11] J. Makino, T. Fukushige, M. Koga, "A 1.349 Tflops Simulation of Black Holes in a Galactic Center on GRAPE-6", *Proceedings of High Performance Networking and Computing Conference (SC '00)* , 2000.

[12] W. Ligon, S. McMillan, G. Monn, K. Schoonover, F. Stivers, K. Underwood, "A re-evaluation of the practicality of floating-point operations on FPGAs", *IEEE Symposium on FPGAs for Custom Computing Machines* pp. 206-215, 1998.

[13] E. Gayles, T. Kelliher, R. Owens, M. Irwin, "The design of the MGAP-2: a micro-grained massively parallel array", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(6), Dec. 2000.