IMPLEMENTATION OF FINITE DIFFERENCE SCHEMES FOR THE WAVE EQUATION ON FPGA

E. Motuk, R. Woods, and S. Bilbao

Sonic Arts Research Centre, Queen's University of Belfast, Belfast, Northern Ireland
{e.motuk, r.woods, s.bilbao@qub.ac.uk}

ABSTRACT

The computational requirements of finite difference schemes for the solution of the wave equation for physical modelling can be huge. Field programmable gate arrays (FPGAs) provide an ideal platform for performing highly parallel DSP computations but the challenge is to be able to quickly and efficiently implement complex systems on FPGA platforms. The paper presents a system level design approach based on dataflow model of computation using a particular finite difference scheme for the solution of 2+1-D wave equation. The results suggest that 84000 nodes could be accommodated on a single Virtex II FPGA.

1. INTRODUCTION

Physical modelling based sound synthesis and acoustical simulation deals with the solution of partial differential equations representing physical phenomena of sound production and propagation. The wave equation is an N+1 dimensional hyperbolic partial differential equation having N space and 1 time dimensions. It describes displacement on a membrane in 2+1-D and sound propagation in spaces in 3+1-D form. Finite difference (FD) schemes transform the partial differential equation into a difference equation by discretizing time and space on an N+1 dimensional grid. For accurate and stable approximations, these schemes employ high sampling rates resulting in high computation. Parallel processors have been used to speed up the computation [1], but they are expensive. Field programmable gate arrays (FPGAs) provide enormous potential as it is possible to derive the architecture to best match the computational requirements. However, realisation is largely a hardware design process and can be very tedious and time-consuming and it is clear that a good high-level design approach is needed.

In this paper, an approach based on the data flow graph (DFG) model of computation is proposed. The particular focus is to realise the explicit FD scheme for the 2+1-D wave equation. Some custom FPGA hardware for FD including FDTD algorithms for solving Maxwell's equations in 2-D or 3-D cases has been reported [2, 3].

The paper is organised as follows. Background on FD schemes is given in section II and the system level design methodology is presented in Section III. Section IV presents the DFG specification and partitioning of the FD algorithm using GEDAE. Section VI outlines the FPGA implementation. Section VII gives an analysis of the FD mesh implementation on the Xilinx Virtex II FPGA.

2. FD SCHEMES FOR 2+1-D WAVE EQUATION

Numerous two-step explicit FD schemes exist, which use discretization of time and space on a structured rectangular grid for the solution of 2+1-D wave equation given below, with certain initial and boundary conditions.

$$\frac{\partial^2 u}{\partial t^2} = v^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \quad (1)$$

When the variable $u(x,y,t)$ is approximated by a grid function $u_{i,j}^n$, where the grid points are defined as $x=i\Delta x$, $y=j\Delta y$, and $t=n\Delta t$ with $\Delta x = \Delta y$, and two-step central differences are substituted for the second derivatives, we obtain the explicit FD form, which is also called centred in time and space (CTCS) scheme [4].

$$u_{i,j}^{n+1} = \alpha^2\left[u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n\right] + 2u_{i,j}^n(1-2\alpha^2) - u_{i,j}^{n-1} \quad (2)$$

where $\alpha = v\Delta t/\Delta x$. From the von Neumann analysis, the Courant-Friedrichs-Lewy condition is derived, which places a restriction, $\alpha \le 1/\sqrt{2}$ for the stability of the FD scheme [5]. For the particular case where $\alpha = 1/\sqrt{2}$, the equation 2 reduces to the simple scheme,

$$u_{i,j}^{n+1} = \frac{1}{2}\left[u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n\right] - u_{i,j}^{n-1} \quad (3)$$

In this case, the FD scheme becomes identical to the 2-D rectangular waveguide mesh representation of the wave motion, which is composed of a network of bi-directional delay elements and 4-port scattering junctions [6]. Fig. 1 shows the graphical representation of the FD scheme in eqn. 3.

From the von Neumann analysis, it can be shown that for the FD scheme corresponding to a rectangular mesh, the speed of propagation of the numerical solution

depends on the frequency and the direction - called the dispersion error [4]. This can be compensated for by increasing the density of the grid points. From the waveguide mesh point of view, interpolated and frequency warped rectangular meshes provide reduction of dispersion error at the expense of increased number of operations per grid point [7]. Triangular waveguide meshes, which are based on a topology of junctions connected to their 6 neighbours, provide better directional dispersion characteristics than rectangular meshes, but this is not the subject of this paper.

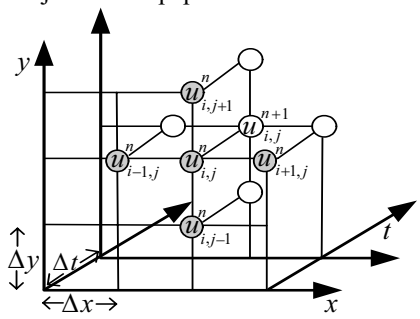

Fig. 1. FD scheme graphical representation

3. IMPLEMENTATION OF FD SCHEMES

The computational requirements of a FD scheme depend on *o*, the number of operations per grid point, the size of the grid and the update rate, *f*. For a two dimensional medium of size (*LXM*), the number of operations per second, is $f \times o \times LM/\Delta x^2$, where Δx is the grid spacing. From the sampling theorem, the FD update rate determines the frequency bandwidth of the wave to be propagated, and as a result, the stability condition. When $\alpha = 1/\sqrt{2}$, the update rate is $f = 1/\Delta t = v\sqrt{2}/\Delta x$, and therefore the grid has to be denser for higher bandwidths. For the FD scheme in equation (3), 5 operations namely 1 multiplication, 3 additions and 1 subtraction are needed to update the grid point. In order to solve the wave equation by the FD scheme (eqn. 3) for a two-dimensional representation of a room with the audio sampling rate of 44.1 kHz and a grid spacing, Δx, of 0.0109m ($343\sqrt{2}/44100$ where v=343m/s), the total number of operations per second for a room (4mx5m) for a real-time application is 36.5×10^9. In addition, the different boundary conditions and oversampling the mesh to reduce dispersion error result in increased computation.

The explicit FD schemes naturally lend themselves to parallel implementations as the same operations are being applied to different data in the problem domain, and there is no limiting temporal dependencies, and FPGAs would be appear to be an ideal platform. A system level design flow based on dataflow computational model would appear to offer an ideal design flow [9] as outlined in Fig. 2.

4. HIGH LEVEL REPRESENTATION BY DATA FLOW NETWORKS

For data parallel algorithms, dataflow graph (DFg) representations allow high-level specification that is independent of the underlying hardware. It utilizes data dependencies to fully exploit parallelism in an algorithm. The communication mechanism in data flow networks is asynchronous message passing [10], which is suited for algorithms having locality of communication like FDs. The model also allows the use of a visual syntax based on block diagrams for specifying the algorithms, which simplifies the design process.

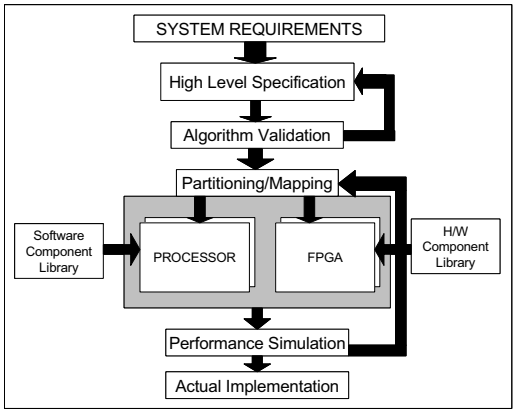

Fig. 2. System level design flow

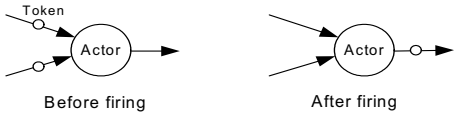

Fig. 3. Simple data flow example

In DFG, the algorithm is specified by a directed dataflow graph where the nodes (actors) represent computations and the arcs represent totally ordered sequences (streams) of events (tokens) [11]. The nodes are hierarchical structures that may represent other directed graphs and can also be implemented as either high level language or behavioural HDL code. The tokens are data structures that can range from scalars to matrices. Whenever a specific set of input arc of a node has data then the node is able to fire. Firing of a node is the computation of the function associated with that node and involves consuming tokens from its inputs and producing tokens on its outputs (Fig. 3).

GEDAE is a block based graphical development platform based on DFG for rapidly implementing DSP algorithms onto multiple processors [12]. In GEDAE, each node can be associated with a point in the FD grid (see Fig. 1) allowing the update equation associated with each grid to be computed. The arcs connect the nodes

representing the neighbouring grids to each other in order to exchange values needed for the update as data streams. Fig. 4 shows the GEDAE DFG that represents a 3X3 square mesh for the implementation of the FD scheme in equation 3. Fig. 4 also shows the inner DFG of a grid point. According to equation (3), a grid point requires the previous iteration values of 4 neighbouring points and its own two previous iteration value to calculate its current iteration value. In DFG representation, this is implemented by the use of two delay boxes where the input to the first is the calculated value and the input to the second is the previous iteration value. This value is also put on the arcs that connect the node to its neighbours. Excitation of the mesh is done by changing the stored previous iteration value of the node that is to be excited. When represented as a DFG, this requires the use of a control stream and a merge box that selects between the excitation value and the previous iteration value of the node to be transferred to the neighbouring nodes and the second delay box, thus realising conditional data flow.

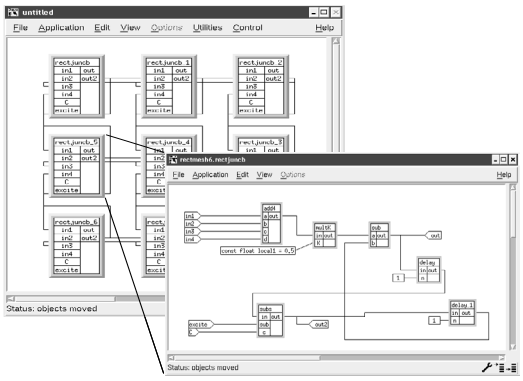

Fig. 4. GEDAE DFG of a 3X3 mesh

Domain decomposition method is the parallelism strategy which involves partitioning the domain into sub-domains [13] as shown in Fig. 5 for a 2-D rectangular grid. Sub-domains are mapped onto the processing elements (PEs) in mesh connectivity. According to the FD scheme formulation, updating the value of a grid point requires the values of the neighbouring grid points, therefore values on the sub-domain boundaries have to be transferred between the neighbouring sub-domains in each iteration period. Therefore, this communication locality is exploited by the block partitioning method.

5. FPGA IMPLEMENTATION

Each PE realizes the operations related to a grid point in the mesh according to the data flow shown in Fig. 4. The communication between the PEs is handled automatically by GEDAE provided that there are hardware structures supporting the token based point-to-point communication structure. Therefore, each PE is equipped with send and receive signals for the transfer of data values as tokens. The communication between the PEs is point-to-point and buffers are implemented as registers to hold the token values. Fig. 6 shows the PE in block form and the hardware structure inside. The PE has 4 inputs and an output to be connected to its neighbours. The control input accepts the control tokens from the host to know whether the node is excited by the value at the excite input.

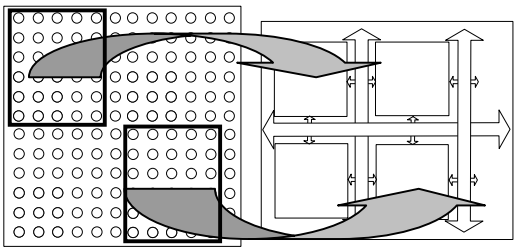

Fig. 5. Example of domain decomposition

In the inner structure of the PE, the interface unit is responsible for communication between PEs and has buffers. The memory is implemented as registers to store the values of the previous and two previous iterations. The update operation is pipelined. The control unit generates the signals for the timing and flow of data between the units. The PE is coded in VHDL and synthesized for Xilinx VirtexII FPGAs using the Synplify synthesis tool.

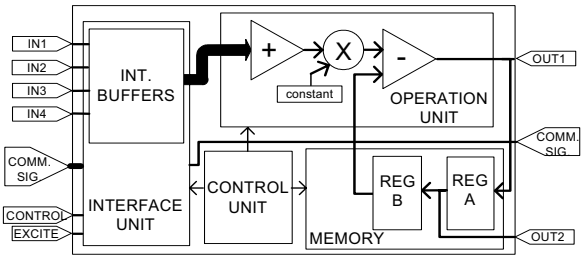

Fig. 6. Details of the PE

Depending on the partitioning, mapping of the nodes to the PEs can either be on a one-to-one or many-to-one basis. The first of the two factors that determine the mapping is the level of parallelism required for real-time execution of the algorithm taking into consideration the communication overhead. The second factor is the FPGA size which determines the number of PEs. When more than one node in the DFG is mapped onto a PE, the communication buffers and registers that store the previous iteration values can be scaled, and they are implemented in the FPGA in either block or distributed form. Distributed memory uses up FPGA slices and is suitable for small memory structures, whereas block memory can be used for larger memory blocks. Therefore, when a large number of nodes are to be mapped on to a single PE, the storage memory should be implemented as

block RAM. The communication buffers in the interface unit can be implemented as distributed memory.

6. PERFORMANCE ANALYSIS

Table-1 shows the synthesis results for a single PE and gives clock cycles for computation and communication. This means that 400PEs can fit onto the largest Xilinx Virtex (XC2V8000 device) and would take 11 clock cycles to complete one iteration of the FD calculation. This gives a maximum iteration frequency of the FD scheme of 16.6 MHz which indicates that to produce 1s of sound sampled at 44.1 kHz, the computation will take 0.0026 seconds which is much faster than the real-time.

Table 1. Xilinx Virtex-II FPGA results for a single PE

Slices	Max. Freq. (MHz)	No. of clock cycles	
		(Computation)	(Communication)
111	182.7	5	6

As the PE can execute much faster than the sampling rate of the algorithm, mapping many nodes to a PE is a feasible option to implement larger meshes. This changes the size of PE's communication buffer, internal memory, and controller unit. Table II gives details when a block partition of size 15x15 is mapped onto one PE.

Table 2. 15x15 node mapping to a single PE

Storage memory locations	Interface memory locations	No. of clock cycles	
		(Computation)	(Communication)
450	120	2025	1140

The storage memory locations are double the total number of nodes and interface memory locations is double the nodes on the partition boundary. As the operation unit is pipelined, ideally it should take 1 clock cycle per node to calculate the next iteration value. However, 4 memory reads to supply the pipeline and 2 memory writes at the end to update the stored values increase the number of clock cycles per node to 9. The number of clock cycles for communication is determined by the number of boundary nodes, and to increase the throughput the communication and the computation can be interleaved. Without interleaving, it takes 3165 clock cycles to complete one iteration period, and when the PE is run at 180 MHz, the iteration rate is 56.87 kHz. In this case it will take 0.775 seconds to produce 1s of sound sampled at 44.1 kHz.

The limiting factor that determines the number of PEs will be the total amount of memory on the device, rather than the number of logic slices. The total amount of memory on a XC2V8000 device is 3 Mbits arranged as 168 18Kbit blocks. When each 18Kbit memory block is dedicated to a PE, 168 PEs can be accommodated which means 500 nodes to be mapped onto a single 16-bit PE. This gives a mesh size of 168x500 or 84000 nodes. However, in this case, the current throughput rate will not be able to satisfy the real-time constraint. Better PE design should increase the running frequency, and interleave communication and computation, thus increasing the throughput.

7. CONCLUSIONS

In this paper, a system level DFG-based design methodology for implementing FD schemes involving the use of data flow networks has been presented. The work suggests that a mesh with 84,000 nodes can be implemented on a single FPGA. The advantage of the approach means that this system will be easily extended to a heterogeneous platform comprising processor and FPGAs which has not been possible before.

8. REFERENCES

[1] A. Villareal, J. A. Scales, "Distributed Three-dimensional Finite-difference Modeling of Wave Propagation in Acoustic Media", Computers in Physics, Vol. 11, No. 4, pp. 388-389, 1997

[2] W. Chen, P. Kosmas, M. Leeser, and C. Rappaport, "An FPGA Implementation of the Two-dimensional Finite-difference Time-Domain (FDTD) Algorithm", in IEEE Proc. FPGA, Monterey, USA, pp. 213-222, 2004.

[3] J. P. Durbano, et al. "Hardware Implementation of a Three-dimensional Finite-difference Time-domain Algorithm", IEEE Ant. and Wireless Propagation Letters, Vol. 2, No.1, pp. 54-57, 2003.

[4] S. Bilbao, J. O. Smith, III, "Finite Difference Schemes and Digital Waveguide Networks for the Wave Equation: Stability, Passivity, and Numerical Dispersion", IEEE Trans. on SAP, Vol. 11, No. 3, pp. 255-266, 2003

[5] J. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Pacific Grove, CA: Wadsworth and Brooks/Cole, 1989

[6] S. A. Van Duyne, J. O. Smith, III, "Physical Modeling with the 2-D Digital Waveguide Mesh", in Proc. Int. Computer Music Conf., pp.40-47, Tokyo, Japan, 1993

[7] L. Savioja, V. Valimaki, "Reducing the Dispersion Error in the Digital Waveguide Mesh Using Interpolation and Frequency-Warping Techniques", IEEE Trans. on SAP., Vol. 8, No. 2, pp. 184-194, 2000.

[9] K. Keutzer, et al., "System Level Design: Orthogonalization of Concerns and Platform-based Design", IEEE Trans. on CAD of Circuits and Systems, Vol. 19, No. 12, pp. 1523-1543, 2000.

[10] W. Johnston, J. P. Hanna, and R. Millar, "Advances in Dataflow Programming Languages", ACM Computing Surveys, Vol. 36, No. 1, pp. 1-34, 2004.

[11] E. A. Lee, T. Parks, "Dataflow Process Networks", Proc. of IEEE, Vol. 83, No. 5, pp. 773-801, 1995.

[12] Gedae Technical Brochure, www.gedae.com

[13] I. Foster, *Designing and Building Parallel Programs*, Addison-Wesley, 1995.