An FPGA-Based Dynamic Load-Balancing Processor Architecture for Solving N-body Problems

Jonathan Phillips, Matthew Areno, Brandon Eames, and Aravind Dasu
Utah State University
{jdphillips@cc}{matthewareno@cc}{beames@engineering}{dasu@engineering}.usu.edu

Overview

Reconfigurable computing has emerged as a key technology in the field of high performance and embedded computing. Our research focuses on the use of reconfigurable processors for solving computationally intensive scientific applications, with an added twist. We are designing an architecture that can retarget its resources dynamically based on run-time and data-driven variation in task level parallelism, as typically seen in radar-based target tracking, high performance video compression, and N-body simulations.

Our approach advocates the derivation of such runtime-reconfigurable architectures through a set of compile time analyses, integrating resource derivation and allocation with a set of architectural features that permit rapid run-time adaptation of reconfigurable data paths. We present an initial demonstration of our approach targeting a molecular dynamics simulation algorithm as a first experiment. The molecular dynamics algorithm under consideration has two primary computations: a distance calculation (DC) and an evaluation of Lennard-Jones potentials (LJP).

The algorithm simulates interactions of a set of molecules based on a set of forces, one of which is the Lennard-Jones potential. The strength of the potential between two molecules is a strong function of the distance separating the molecules. If two atoms or molecules in a system are separated by a distance less than a specified threshold, the Lennard-Jones potential between them is calculated. For each molecule in the simulation, we integrate all such potentials, and use that result to compute future molecule positions in real space (through Newtonian dynamics). The simulation progresses through multiple time steps, with the system calculating forces and positions at each step.

Architecture

Our platform consists of a Xilinx V4FX60 FPGA, on which discrete units are placed to perform either DC or LJP calculations. Global data is stored in a feeder FIFO, which feeds data to DC units and stores data as it is produced by LJPC units. This feeder FIFO is actually a string of FIFOs, where data is passed across all DC units to facilitate calculations for every two-atom pair. The feeder FIFO makes extensive use of the hardware FIFOs embedded in the FPGA fabric, thus avoiding large amounts of logic slice consumption. Data produced by DC units is sent to an intermediate buffer which feeds the LJPC units. Due to the dynamic, data-dependent nature of the molecular dynamics algorithm, it is not possible to accurately predict the percentage of DC calculations that will lead to LJPC calculations. Consequently, we employ a fixed, equal number of DC compute units and LJP compute units, together with a set of reconfigurable units called FLEX processors. The DC unit is responsible for determining whether to release a pair of molecules to the LJP calculators, based on their proximity. .

Because the ratio of actual DC to LJP computations is not known at design time, a static allocation of DC and LJP computation units can lead to inefficient resource utilization. Either too many atom pairs could qualify for LJP computations and hence flood the intermediate buffer or too few atoms might require the use of LJP units, in which case the LJP units would sit idle. The former case results from a shortage of LJP units and in the latter case, a shortage of DC units. To address the inadequacies of static load balancing, we integrate a flexible computational module (FLEX). The role of the FLEX processor is to function in either DC or LJP mode, depending on the types of calculations that need to be performed. This architecture is illustrated in figure 1. The mode of each FLEX processor is controlled via an on-chip PowerPC processor, which dynamically analyzes the state of the computation and determines, based on runtime information, an appropriate mix of DC and LJP compute units in order to maximize computational throughput.

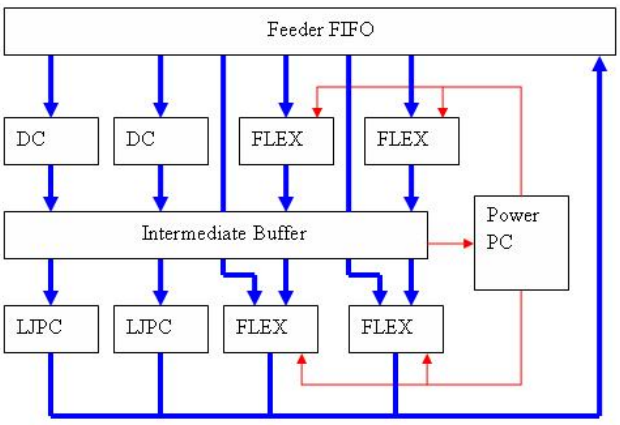

Figure 1: System block diagram.

All computations are IEEE-754 single precision floating point. Preliminary results indicate that we can fit two DC, two LJP, and four FLEX units on the V4FX60 device. To obtain efficient resource utilization in the FLEX block, we

first ran the data flow graphs of one instance of both the DC and the LJPC intermediate representation graphs obtained from a standard gcc compiler though a force-directed scheduler. The union of these schedules was used to obtain a set of resources. The optimality of the resource set is measured by the number of bubbles introduced in the pipelines of the floating-point arithmetic units. These resources are then fed to a list scheduler that maps an optimal number of parallel DC or LJP calculations onto the FLEX processor. We have used a VLIW architecture to implement the flow of instructions through the processor. Instructions are derived directly from the instruction schedule. Programs for DC and LJP computations are stored in separate program memories on the FPGA.

Results

This project will yield a system that efficiently performs dynamic load balancing, based upon the characteristics of a set of input data. It is anticipated that our novel architecture of combining static DC and LJPC units with FLEX processors on a Xilinx V4FX60 FPGA running at 200 MHz will provide substantial speedup over a dual-processor Pentium server machine. Our data flow analysis and application-specific design will also provide better performance than a general-purpose VLIW processor. Synthetically generated data sets of varying sizes and with varying proportions of atom pairs on which LJPC calculations must be performed will be utilized to generalize system performance.

Table 1 shows the types and numbers of arithmetic units that are employed for each type of computation module. The FLEX module is more complex, as it must be able to perform both DC and LJP computations.

Table 1: Arithmetic Unit Distributions.

Module	+	-	*	÷	≤
DC	2	2	2	0	2
LJPC	3	1	3	1	0
FLEX	3	2	3	1	1

Preliminary resource utilization estimates for the different blocks are shown in table 2. The system will consist of two DC units, two LJPC units, and four FLEX units.

Table 2: Estimated Resource Utilization.

Module	Slices	DSP48s	BRAMs
DC	1,642	8	0
LJPC	2,514	12	0
FLEX	3,610	12	20

Under steady-state conditions, each module will produce a valid output on each clock cycle, except the DC unit which produces two per cycle. Given a 200 MHz clock, this implies a throughput of 200 million results per second for a LJPC or FLEX and 400 million results per second for a DC unit. Under circumstances when all atom pairs flow through both DC and LJPC units, maximum throughput of 1 billion results per second are attainable.

Results of load balancing performance across several synthetically generated data sets and results of speed performance versus the dual-processor Pentium will be presented at the workshop.