Politecnico di Milano - University of Illinois Chicago
HPPS Project

UIC

QCADESIGNER POWERED BY CUDA

Instructor: Prof.ssa Sciuto Donatella
Tutor: Santambrogio Marco Domenico

Authors:
Gibilisco Giovanni Paolo, Matr. 755066
Marconi Francesco, Matr. 755439
Miglierina Marco, Matr. 754848

2009-2010

Contents

List of Figures

Abstract

2 righe sul lavoro, -¿VENDITI BENE

Chapter 1

State of the Art

- QCA * QCADesigner * Two Engines: BISTABLE and COHERENCE, describe them shortly (see MINA site) -¿ BISTABLE IS JUST A FAST APPROXIMATION to test circuits - CUDA

Chapter 2

Rationale

- Why QCA? * novel emerging paradigm * 2 Thz, low energy consumption and miniaturization * quantum computing - QCADesigner simulator slow on big circuits: * Every sample, each cell's polarization is computed based on the values of his neighbors, sequentially. * Bottleneck from profiling (table with times) * Simulation core: pseudo code * Identical operations repeated for each cell, big circuits -¿ thousands of cells * We chose to speedup this part of the code with CUDA because: SIMD architecture (SIMT): single instruction repeated on different data hundreds of core -¿ many threads running simultaneously, each thread responsible of computing a cell's polarization scalable, adding new cores implies a greater number of cells computed simultaenously, higher speedup - Objective * Speed up simulation for big circuits * batch simulator * Given a file .qca -¿ produce output: binary, continous values, plot on png, log with info of simulation * if same .qca -¿ same results CPU and CUDA

Chapter 3

implementation

3.1 First approach

The original source code of QCADesigner was downloaded from Mina website (ref). We attempted to make it compile as it was but we did not manage to solve several compilation errors. So we started to focus on the identification of the core algorithm supporting the tool in order to obtain a working batch simulator executable on CPU. This operation took us some weeks of work. Meanwhile we were able to deeply analyze the code. We made some hypothesis on the location of possible bottlenecks, we identified the data structures used to represent circuits and started to consider possible transformations that had to be done in order to obtain fast accessible and light weight data structures allocable on the GPU global memory.

3.2 CPU algorithm and profiling

Bistable engine is thought as a fast and approximated simulation, sufficient to verify the logic functionality of a design. Every cell is represented as a simple two-state system. The entire simulation is divided into samples, that are units of time (not yet experimentally defined), and for each sample the state of each cell is calculated with respect to the other cells within a preset effective radius. This operation is iterated until all cells have converged within a predetermined tolerance. Once the entire system converges the output is recorded and the computation goes on with the next sample, after having updated the input cells with new input values. The number of samples required to have a good approximation is known (ref) to be $2000 * 2^N$, where N is the number of inputs. The maximum number of iterations allowed for the convergence within a sample is 100. Thus, during a simulation each cell's

value is computed sequentially $It * 2000 * 2^N$ times, where It is the mean number of iteration needed to reach convergence. The more are the cells, the longer will take each sample to reach convergence.

Once we finished the batch application we started to profile the execution times simulating some circuits. The table (ref) shows

Chapter 4

Results

Chapter 5

Conclusions

Bibliography