A Parallel Version of the MiNa's Quantum Dot Cellular Automata IDE - QCADesigner

Riccardo Cattaneo, Giuseppe Chindemi

Politecnico di Milano
HPPS

June 22 2010

Why?

- Si based technology is reaching its physical limits due to aggressive miniaturization, progressively increasing packaging densities, clock distribution and dissipation issues
- Alternatives are being studied right now in order to overcome these limits (different technologies)
- QCA cells are one of these
- QCADesigner is the most mature tool for layouting and simulating QCA cells based circuits, yet, in the original implementation it is unreasonably slow to simulate more bigger-than-toy ones
- Exploitation of other approach in order to obtain significant speedups

QCA cells? Why?

Limits of current transistor technology

- **Shrinking size** in practice: no less than $20\mu m$ (nowadays: $45\mu m$)
- **Reduced clock frequency** in practice: in the order of magnitude of $10^9 Hz$
- **Power dissipation problem**

QCA cells technology

- **Improved clock frequency** in practice: in the order of magnitude of $10^{12} Hz$
- **Power dissipation** in practice: in the order of magnitude of $10^{-18} W$
- **Device density** in practice: in the order of magnitude of $10^{10} W$

QCA cells? What?

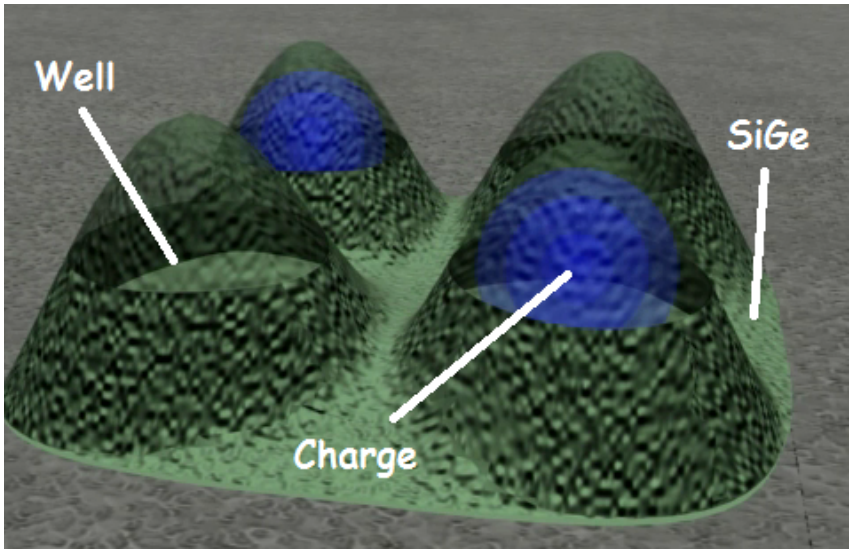


Figure: QCA cells: physical view.

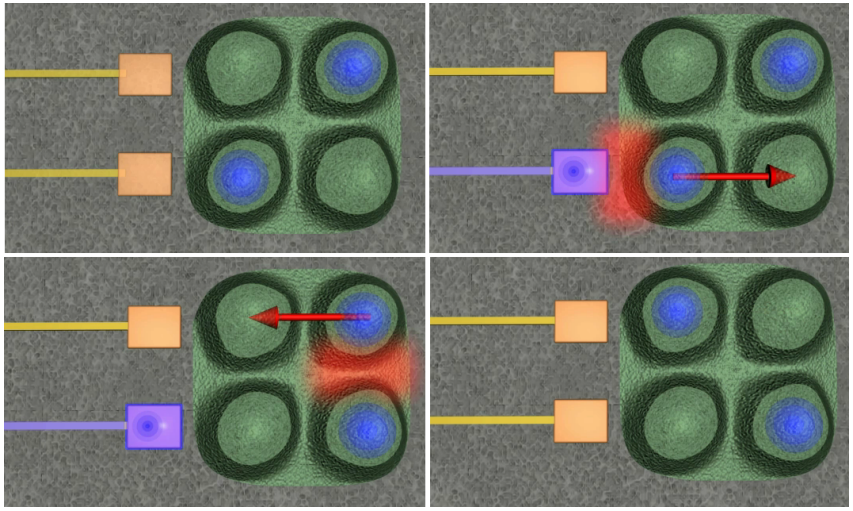


Figure: QCA cells: evolution of local state.

QCA are CA: evolution depends on previous status of cell itself and neighborhood

CUDA Overview

What is CUDA?

- It is a software layer that allow programmers to exploit the capability of Nvidia GPUs as general purpose processors.

Why CUDA for QCAD?

- Because QCA are parrallel by nature.
- Because GPUs are good in arithmeric.
- Because GPUs offer the lowest price per core.

GPU Logical Organization and Programming Model

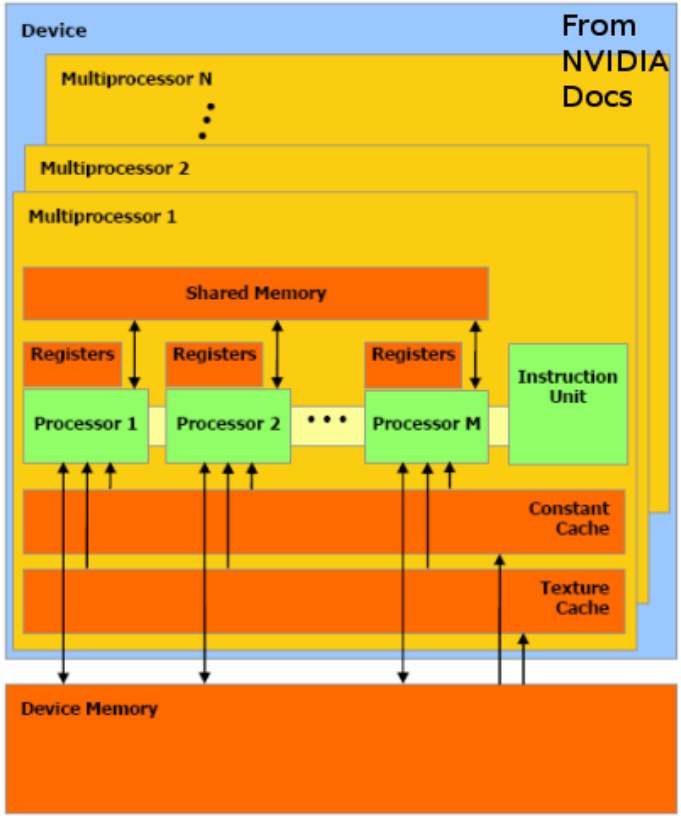


Figure: CUDA GPUs: A MIMD Array of SIMD processors

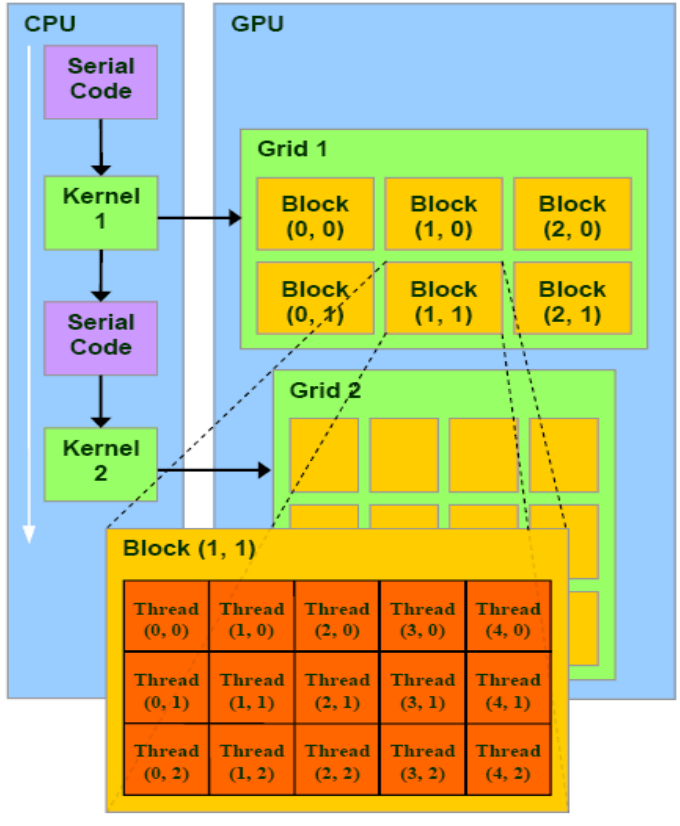


Figure: CUDA GPUs: Heterogeneous Programming

Original Implementation

- Every cell-one thread approach
- Additive error when evolving system
- Time complexity: $O(2^i * n * b)$

Every cell is simulated one after the other even though they could be evolved in parallel

CUDA Implementation

- One cell-one thread approach
- No additive error when evolving system
- Time complexity: $(\frac{2^i * n * b}{T})$ where T is the number of running threads

Every thread is responsible for the evolution of its cell. The larger the number of running threads, the better the performances (upper bound: T=number of cells in the layout)

Tests Description

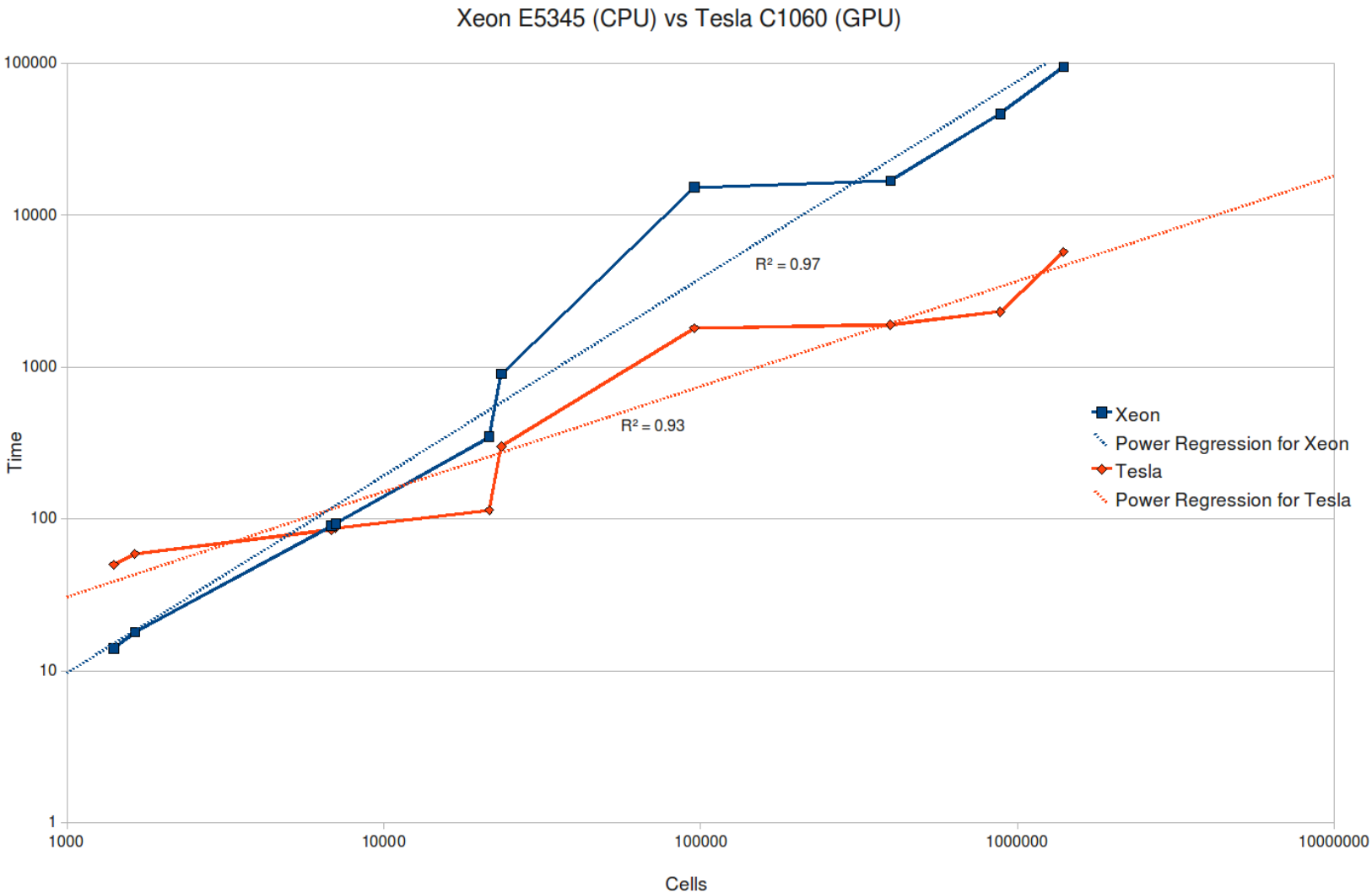
The "Lucifero" Workstation

- CPU Intel Xeon E5345
- GPU Nvidia Testa C1060

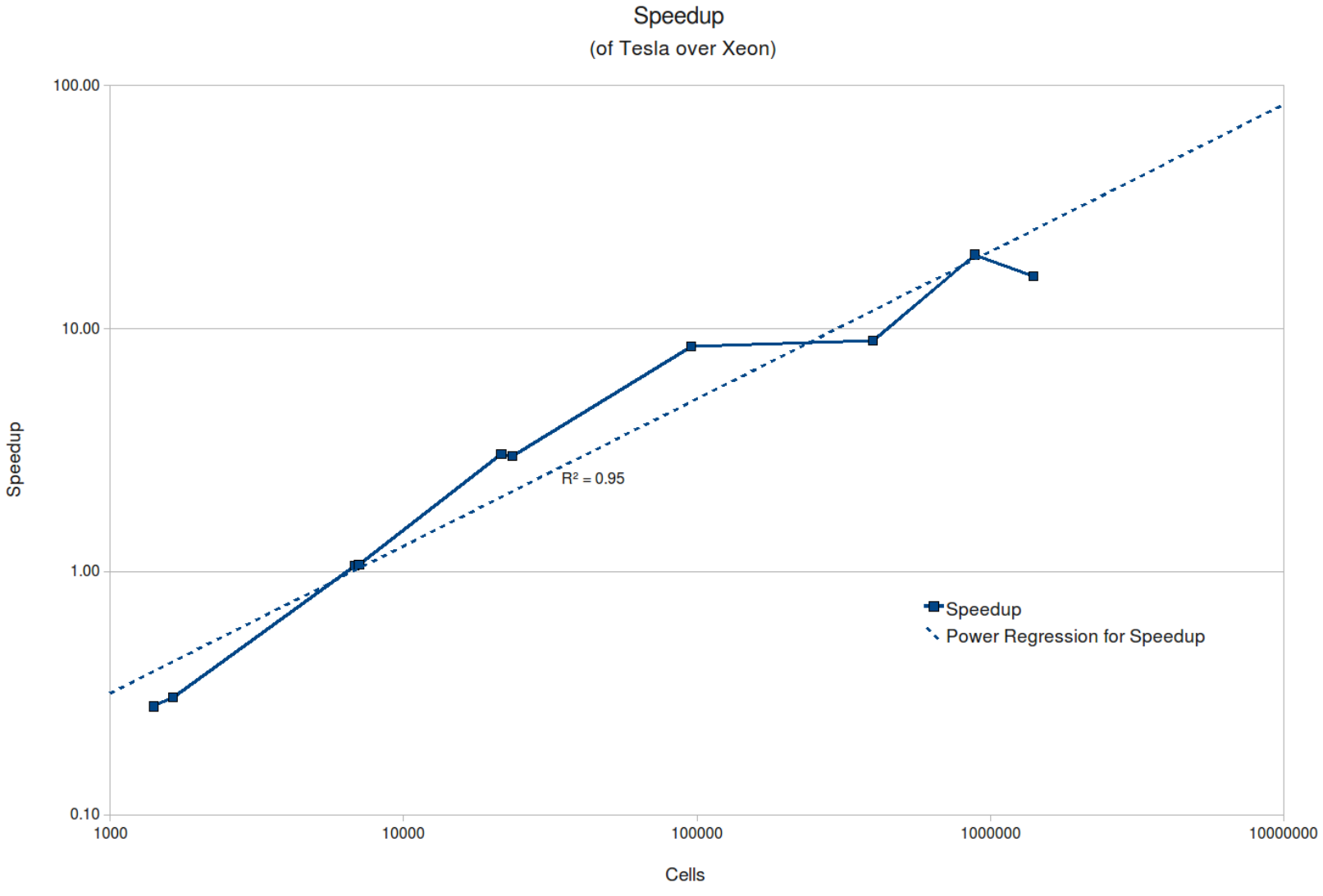
Which Tests?

- Test 1 QCAD vs CUDAQCAD
- Test 2 Memory Transfert Rate
- Test 3 GPU Occupancy

Test 1: QCAD vs CUDAQCAD



Test 1: QCAD vs CUDAQCAD



Test 2: Memory Transfert Rate

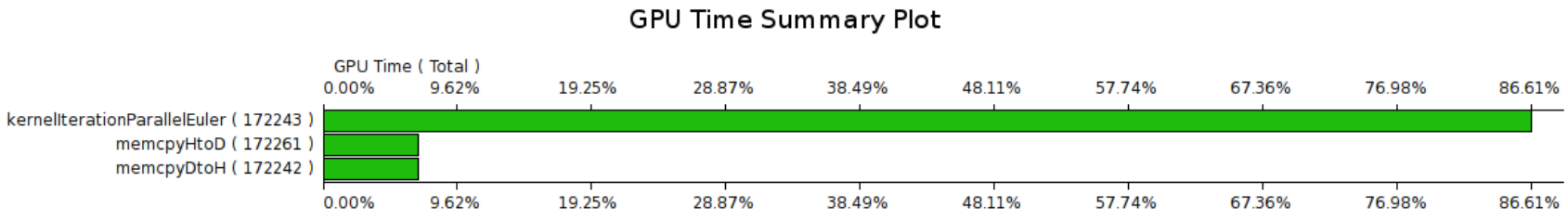


Figure: Memory Tranfer for NAND circuit (1642 cells)

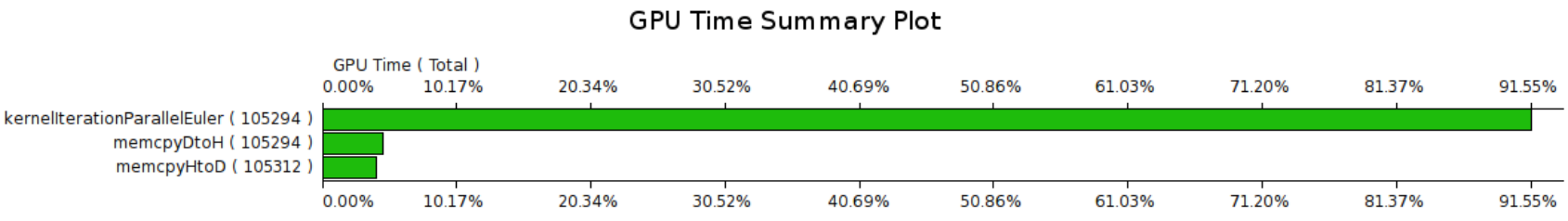
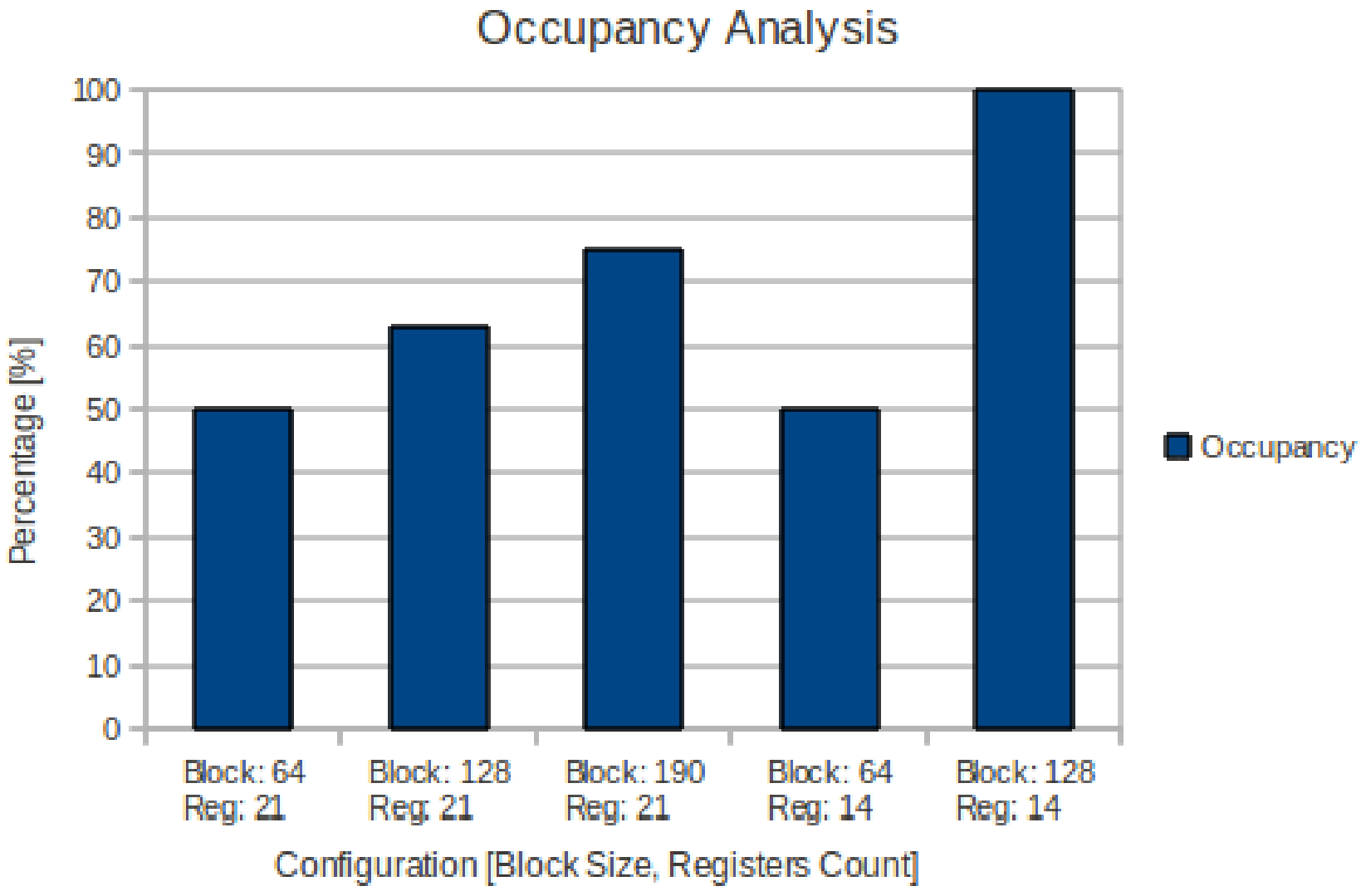


Figure: Memory transfers for MUX42 circuit (21551 cells)

Test 3: GPU Occupancy



Questions?

Questions?