

Hardware-accelerated MC Simulation for PDT Treatment Planning using FPGAs and GPUs

William Chun Yip Lo

University of Toronto, Department of Medical Biophysics
610 University Avenue, Toronto, ON, Canada M5G 2M9
wlo@uhnres.utoronto.ca

1. Introduction

Photodynamic therapy (PDT) is an emerging treatment modality in oncology and other fields. Improvements in PDT efficacy, particularly for interstitial applications, require faster computational tools to enable efficient treatment planning beyond relying on diffusion theory and homogeneity [1]. To maximize PDT efficacy, accurate models of light propagation should be employed to account for target volume geometry and its heterogeneity.

Light dosimetry plays a critical role in PDT treatment planning. For computing light dose (fluence) distribution, the Monte Carlo (MC) method is often employed due to its flexibility and accuracy. Unfortunately, MC-based models are very time-consuming, particularly for iterative optimization problems such as PDT treatment planning, highlighting the need to accelerate MC-based light dose computation.

To explore this possibility, this work aims to accelerate an MC simulation for PDT, based on the widely accepted Monte Carlo for Multi-Layered media (MCML) code [2]. We demonstrate the feasibility of achieving this goal using a development platform with multiple field-programmable gate arrays (FPGAs) and another system with multiple graphics processing units (GPUs) comprising a total of 480 cores. Particularly, the GPU-based approach has gained popularity with the release of the Compute Unified Device Architecture (CUDA) [3], a C-like programming interface for NVIDIA GPUs, as it suits general-purpose applications much better than traditional GPU languages. However, the underlying NVIDIA architecture presents several unique challenges that necessitate a new approach to code optimization. Conversely, an FPGA-based approach offers greater flexibility as one has the opportunity to customize the underlying architecture to suit the application. The results for both approaches are presented and the development process will be compared to highlight the key differences between the two approaches.

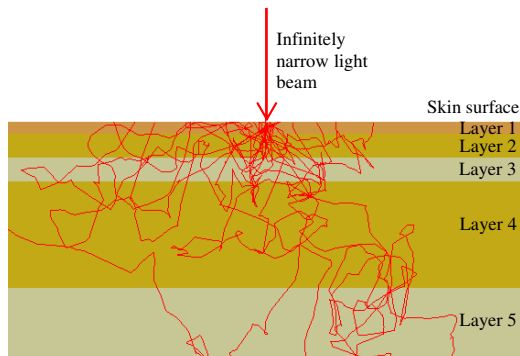


Figure 1. Photon propagation in a 5-layer realistic skin model using an infinitely narrow beam at 633nm

Table 1. Optical properties of five-layer skin tissue (633nm)

Layer	μ_a (cm^{-1})	μ_s (cm^{-1})	g	n	Thickness (cm)
1. epidermis	4.3	107	0.79	1.5	0.01
2. dermis	2.7	187	0.82	1.4	0.02
3. dermis with plexus superficialis	3.3	192	0.82	1.4	0.02
4. dermis	2.7	187	0.82	1.4	0.09
5. dermis with plexus profundus	3.4	194	0.82	1.4	0.06

2. Monte Carlo for Multi-Layered media (MCML)

The MCML code provides an MC model of steady-state light transport in multi-layered media. It assumes infinitely wide layers and models an incident pencil beam perpendicular to the surface. Figure 1 depicts an example of the simulation using as input a realistic, five-layer skin model [4], showing the paths of photon packets from an infinitely narrow beam. In MCML, absorption is stored in a two-dimensional array $A[r][z]$, representing the photon absorption probability density [cm^{-3}] as a function of position, which can be converted into photon fluence [cm^{-2}] to obtain the isofluence lines for treatment planning. Millions of photon packets are required to generate low-noise isofluence maps. In MCML, the simulation of each photon packet involves the same sequence of steps and is independent of other photon packets (except the memory access to $A[r][z]$).

3. Methods

A. FPGA-based approach

A field-programmable gate array (FPGA) chip is a pre-fabricated silicon chip that can be programmed electrically to implement virtually any digital design, including the design of custom hardware for the acceleration of computationally intensive applications such as MCML. Its flexibility is derived from its underlying architecture, consisting of an array of programmable logic blocks (logic elements) interconnected by a programmable routing fabric. Compared to GPUs, FPGAs offer greater flexibility in the design as one has the ability to customize the underlying architecture, instead of being constrained by it. For example, applications with undesirable memory access patterns and significant divergent behavior may achieve very poor performance on GPUs due to the constraints imposed by the NVIDIA architecture.

Using an FPGA-based platform, custom hardware was designed *de novo* to perform the MCML simulation. To maximize parallelism and computational throughput, three hardware acceleration techniques were applied in the design. First, floating point operations were converted to fixed point operations to greatly reduce the resource usage of a computational unit (e.g., adder, multiplier, and divider). Second, look-up tables were created in on-chip memory to pre-compute values for expensive operations (such as trigonometric functions), thereby saving a large number of logic blocks. The third technique applied was pipelining. Similar to an assembly line, the complex MCML simulation was broken down into simpler stages. Since each stage performs its task independently, the net throughput is increased, thereby speeding up the computation. More details about the hardware implementation will appear in the JBO [5].

B. GPU-based approach

To achieve high performance on NVIDIA GPUs, it is important to understand how the CUDA code maps to the NVIDIA-specific GPU architecture. In CUDA, the programmer writes GPU code in the form of *kernels*, which are similar to regular C functions, except that multiple copies are executed in parallel by the GPU threads [3]. To execute a kernel, the programmer specifies a *kernel configuration* - the organization of GPU threads. GPU threads are organized into blocks, which in turn contains multiple threads.

CUDA also requires the programmer to explicitly manage data transfer between the GPU main memory (called *global memory* [3]) and the host (CPU) memory space. NVIDIA GPUs also have a unique memory hierarchy. Therefore, to maximize performance, registers and read-only *constant* memory, which are usually much faster than the *global* memory, were used to store read-only variables such as tissue geometry, photon data structures and temporary variables in the GPU-based MCML. The 2-D absorption array $A[r][z]$ was stored in global memory. The basic parallelization scheme involves processing multiple photon packets in parallel on multiple cores.

The implementation of the GPU-based MCML consisted of four stages: single-precision floating point conversion, creation of a GPU program skeleton, kernel packaging and memory management, and optimizations. First, all double-precision floating point operations were converted into single-precision in MCML to maximize performance. Second, a simple program skeleton was created to ensure that the memory management code, kernel configuration, and host CPU code were implemented correctly. Third, the skeleton was replaced by the actual MCML computations. To construct a unique seed for random number generation in each thread, the block id and thread id were used. Finally, the kernel code was optimized to maximize the level of parallelism while minimizing single-thread execution time. The kernel configuration was fine-tuned. The main kernel code was modified to reduce the usage of the *local* memory, another piece of GPU memory that is >100 times slower than the registers [3]. To reduce local memory usage, a different random number generator (Tausworthe) [6] that does not use arrays was adopted, as the compiler automatically allocates arrays into local memory instead of registers.

4. Results

For validation and performance comparison, a skin model was selected as the input to MCML. The tissue optical parameters are presented in the Table 1. The execution time of *FPGA-MCML* was measured on a platform called the TM-4 with four Altera Stratix I FPGAs and on the modern DE3 board with one Stratix III FPGA. The execution time of *GPU-MCML*, compiled using the CUDA Toolkit version 2.0, was measured on a platform with 2 NVIDIA GeForce GTX280 graphics card with a total of 480 processing cores. The baseline performance for *CPU-MCML*, compiled with the highest optimization level, was measured using the unmodified MCML on a 3-GHz Intel Xeon processor and a modern 3-GHz Xeon 5160 processor. All runtimes included the main simulation and all pre-/post-processing operations. For validation, isofluence maps were generated using 10^8 photon packets. The relative shift in the position of the isofluence lines was compared against the gold standard MCML output.

A. Performance

Table 2 shows that the average execution times for all three implementations of MCML: *FPGA-MCML*, *GPU-MCML*, and *CPU-MCML*. Compared to *CPU-MCML* executed on the modern Xeon 5160 processor, *FPGA-MCML* achieved a 56x speedup (or 14x per Stratix I FPGA) and 28x speedup per Stratix III FPGA, while *GPU-MCML* achieved a 75x speedup (or 37.5x per GTX280 card). Note that the speedup would be 1.5 times greater in all cases using the older Xeon processor for baseline comparison, which is a fairer comparison for the old Stratix I platform.

Table 2. Average Execution Time of *FPGA-MCML*, *GPU-MCML*, and *CPU-MCML* at 10^8 photon packets (4 runs).

Machine	Clock Speed	Execution Time (s)	Speedup (vs. *)	Speedup (vs. **)
1 x Xeon processor (Pentium 4) *	3 GHz	9150	1	0.67
1 x Xeon 5160 processor (<i>CPU-MCML</i>) **	3 GHz	6102	1.5	1
2 x GeForce GTX280 (<i>GPU-MCML</i>)	1.296 GHz	81	113	75
4 x Stratix I FPGAs (<i>FPGA-MCML</i>)	41 MHz ^a	109	84 ^b	56
1 x Stratix III FPGA (<i>FPGA-MCML</i>)	80MHz	220	42	28

^a Note that the Stratix I FPGA has the same process technology as the old Xeon processor (130nm).

^b The high performance achieved despite the gap in clock speed highlights the efficiency of the custom MCML hardware

B. Validation

The isofluence lines generated based on the skin tissue geometry are plotted in Fig. 2. Since *GPU-MCML* produced similar isofluence lines, only the isofluence lines for *FPGA-MCML* are plotted. As shown in Fig. 2, the isofluence lines produced by *FPGA-MCML* and *CPU-MCML* matched very well. A shift in the position of the isofluence lines was only noticeable for fluence levels at 0.00001cm^{-2} (8 orders of magnitude smaller than the fluence near the centre - 1000cm^{-2}). The detected shift was approximately 0.1mm, which is of little significance in PDT treatment planning.

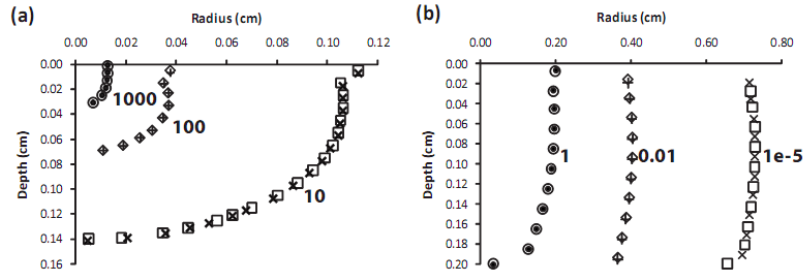


Figure 2. Comparison of the isofluence lines generated by *CPU-MCML* (○, ◇, and □) and *FPGA-MCML* (●, +, and ×) using 100 million photon packets. Fluence levels range from 1000cm^{-2} to 0.00001cm^{-2} .

5. Conclusions

Both the FPGA-based approach and GPU-based approach offer the scalability to achieve the desired performance given N FPGAs or GPUs. The dramatic reduction in treatment planning time achieved by either an FPGA or GPU-based platform may potentially enable real-time treatment planning based on the most recent images of the treatment volume. Currently, pre-treatment models assume constant values for tissue optical properties and ignore the dynamic nature of tissues, which could directly affect treatment outcomes in interstitial PDT [7]. The significant performance gain provided by the hardware approach can enable PDT treatment planning in heterogeneous, spatially complex tissues using more sophisticated MC-based models.

6. References

- [1] T. J. Dougherty, "Photodynamic therapy," *Photochemistry and Photobiology*, vol. 58, pp. 895-900, 1993.
- [2] L. Wang, S. L. Jacques, and L. Zheng, "MCML—Monte Carlo modeling of light transport in multi-layered tissues," *Computer Methods and Programs in Biomedicine*, vol. 47, pp. 131-146, 1995.
- [3] Nvidia, "Compute Unified Device Architecture, Programming Guide, version 2.0," 2007.
- [4] V. V. Tuchin, "Light scattering study of tissues," *Physics-Uspekhi*, vol. 40, pp. 495-515, 1997.
- [5] W. Lo, K. Redmond, J. Luu, P. Chow, J. Rose, and L. Lilge, "Hardware Acceleration of a Monte Carlo simulation for PDT treatment planning," *Biomedical Optics*, vol. 14, Jan/Feb 2009.
- [6] P. L'Ecuyer, "Maximally equidistributed combined Tausworthe generators," *Mathematics of Computation*, vol. 65, pp. 203-213, 1996.
- [7] A. Johansson, J. Axelsson, S. Andersson-Engels, and J. Swartling, "Realtime light dosimetry software tools for interstitial photodynamic therapy of the human prostate," *Medical Physics*, vol. 34, p. 4309, 2007.