Hardware acceleration of a Monte Carlo simulation for PDT treatment planning

William Lo, Keith Redmond, Jason Luu, Paul Chow, Jonathan Rose, Lothar Lilge

William Lo, Department of Medical Biophysics, Ontario Cancer Institute, Princess Margaret Hospital, Rm. 8-324, 610 University Avenue, Toronto, ON, Canada M5G 2M9
Phone: 416-946-4501 x5767, Email: wlo@uhnres.utoronto.ca

Keith Redmond, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON, Canada M5S 3G4
Phone: 416-978-1659, Email: keith.redmond@utoronto.ca

Jason Luu, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON, Canada M5S 3G4
Phone: 416-978-1652, Email: jluu@eecg.toronto.edu

Paul Chow, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON, Canada M5S 3G4
Phone: 416-978-2402, Fax: 416-946-8734, Email: pc@eecg.toronto.edu

Jonathan Rose, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON, Canada M5S 3G4
Phone: 416-978-6992, Fax: 416-978-1145, Email: jayar@eecg.toronto.edu

Lothar Lilge, Department of Medical Biophysics, Ontario Cancer Institute, Princess Margaret Hospital, Rm. 7-416, 610 University Avenue, Toronto, ON, Canada M5G 2M9
Phone: 416-946-4501 x5743, Fax: 416-946-6529, Email: llilge@uhnres.utoronto.ca
**Keywords:**

Monte Carlo simulation; Monte Carlo Multi-Layered Media (MCML); Photodynamic Therapy (PDT) treatment planning; hardware acceleration; pipelining; Field-Programmable Gate Array (FPGA)
Abstract

Monte Carlo (MC) simulations are being used extensively in the field of medical biophysics, particularly for modeling light propagation in tissues. The high computation time for MC limits its use to solving only the forward solutions for a given source geometry, emission profile, and optical interaction coefficients of the tissue. However, applications such as Photodynamic Therapy treatment planning or image reconstruction in diffuse optical tomography require solving the inverse problem given a desired dose distribution or absorber distribution, respectively. A faster means for performing Monte Carlo simulations would enable the use of MC-based models for accomplishing such tasks. To explore this possibility, a digital hardware design of a Monte Carlo simulation based on the Monte Carlo for Multi-Layered media (MCML) software was implemented on a development platform with multiple Field-Programmable Gate Arrays (FPGAs). The hardware performed the Monte Carlo simulation on average 80 times faster and was 45 times more energy efficient than the MCML software executed on a 3GHz Intel Xeon processor. The resulting isofluence lines closely matched those produced by MCML in software, diverging by only less than 0.1mm for fluence levels as low as 0.00001cm\(^{-2}\) in a skin model.
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. The fundamental mechanism of PDT involves the localization of a photosensitizer, followed by the irradiation of the target volume with light of a specific wavelength to activate the photosensitizer.\textsuperscript{1-4} Advances in PDT have allowed this therapy to be applied to more complicated treatment volumes, particularly in interstitial applications such as those in the prostate and the head and neck region.\textsuperscript{5-9} In order to maximize the efficacy while reducing complication rates, it is important to employ accurate models of light propagation in turbid media that can take into account complex tumor geometry and the heterogeneity in the tissue’s light interaction coefficient and responsivity to PDT, for clinically robust treatment planning.

Among other factors, light dosimetry plays a critical role in PDT treatment planning. Selective tumor necrosis is largely dependent on reaching a sufficiently high light dose or fluence [J cm\textsuperscript{-2}] within the tumor while not exceeding the threshold level of necrosis in the surrounding normal tissues. Therefore, a successful PDT treatment relies on the accurate computation of the fluence throughout the clinical target volume, which comprises the tumor and other surrounding tissues or organs at risk. Among other techniques for computing the fluence distribution, the Monte Carlo method is often employed due to its flexibility in modeling 3D geometries, its ability to handle varying optical properties and light sources with different emission patterns, as well as its
Similarly, Monte Carlo simulations are widely used as the gold standard in radiotherapy treatment planning and there is a clear trend towards adopting the Monte Carlo method for clinical radiotherapy dose calculations in commercial treatment planning systems.\textsuperscript{11, 12} Unfortunately, such simulations are also known to be very time consuming and different variance reduction schemes or efficiency-enhancing methods are traditionally introduced to reduce the computation time.\textsuperscript{13} Currently, the computation time for MC remains high, preventing its use in iterative forward solutions of photon transport that optimize the source geometry and emission profile for a range of optical interaction coefficients of the tissue to approximate a desired light dose distribution. Accelerating Monte Carlo simulations would enable the use of MC-based models for solving these iterative problems, which can be applied to Photodynamic Therapy (PDT) treatment planning.

Attempts to accelerate Monte Carlo simulations for modeling light propagation in tissues have been limited to software parallelization schemes. For example, one such scheme involved dividing the simulation into many independent groups, each of which was executed on a different computer or processor in parallel.\textsuperscript{14, 15} One potential problem with the software parallelization approach is the need to have dedicated access to a computer cluster in order to achieve the desired performance. This approach is not easily accessible as the costs of a large networked cluster of servers are substantial, thus hindering the deployment of complex MC-based models in iterative optimization problems. The present work explores the use of custom-built hardware to accelerate the Monte Carlo simulation for computing light dose in PDT. Compared to a networked cluster of servers,
custom-built hardware can be easily integrated into an imaging suite for dedicated treatment planning. Also, a purpose-built system with custom hardware delivers the same performance at a much lower cost, both in terms of the cost of parts and the power consumption.

Using the widely accepted Monte Carlo for Multi-Layered media (MCML)\textsuperscript{16} code as the gold standard, this work demonstrates the feasibility of the hardware-based approach for accelerating Monte Carlo simulations applied to the computation of fluence distributions. The final MCML-based hardware design, implemented on a programmable hardware prototyping platform, reduces the computation time of MCML simulation by 80 times compared to a 3GHz Intel Xeon processor. Unlike software-based techniques, this custom hardware design does not use general-purpose processors to execute computationally intensive operations. Instead, the hardware design was created \textit{de novo} on programmable logic devices called Field-Programmable Gate Arrays (FPGAs).\textsuperscript{17}

The remainder of this paper will discuss the FPGA-based hardware design, named here FPGA-based MCML (FBM). This paper first begins with a brief overview of MCML in order to discuss the modifications made to MCML. Next, the FPGA-based hardware design methodology and the key hardware acceleration techniques applied will be presented. The unique aspects of the design will be explained to highlight how various hardware acceleration schemes were applied to achieve the reduced computation time. Finally, the validation results and performance will be analyzed, followed by the possible implications of the significant reduction in computation time for MC-based models.
2 Monte Carlo for Multi-Layered Media

2.1 Overview

The MCML approach and code\textsuperscript{16} provides a Monte Carlo 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. Extended sources and their beam profiles are modeled separately by convolving the photon distribution obtained for a pencil beam (for example, using the CONV program).\textsuperscript{18} Three physical quantities are scored in MCML, namely absorption, reflectance, and transmittance. Absorption in the tissue is stored in a two-dimensional array $A[r][z]$, which represents the photon absorption probability density $[\text{cm}^{-3}]$ normalized to the total number of photon packets launched. It can be converted into photon fluence $[\text{cm}^{-2}]$ by dividing by the local absorption coefficient to obtain the isofluence lines.

To reduce computation time, two variance reduction schemes are employed: scoring in cylindrical coordinates and the use of photon packets. Nonetheless, millions of photon packets are required for generating a low-noise fluence rate distribution map. Each photon packet undergoes three key steps that are repeated continuously in the simulation: hop, drop, and spin, following the naming convention in the MCML program (Figure 1). The hop step moves the photon packets to its next interaction site by a step size obtained from sampling a probability distribution based on the photon’s free path. The drop step adjusts the photon packet’s weight to simulate absorption, based on the absorption coefficient at the site of interaction. Finally, the spin step computes the scattering angle using the Henyey-Greenstein function.\textsuperscript{19} When a photon packet exits the tissue through
the top or bottom layer, it is terminated and its weight is scored as reflectance and transmittance, respectively. If the photon weight has reached a threshold value, a survival roulette is performed to determine if the tracking of the photon packet should end. If the photon survives, its weight is increased according to energy conservation requirements.

2.2 Modifications

Two key modifications were made to the MCML program to meet hardware design requirements and to tailor the solution for the envisioned application: PDT treatment planning. First, since fluence is the quantity of concern in PDT treatment planning, only the absorbed photon probability density as a function of position within the 2D absorption array $A[r,z]$ was recorded. The reflectance and transmittance were ignored to reduce the memory resource requirements in hardware. The second major modification involved the conversion of all floating-point operations into fixed-point operations. This non-trivial conversion was necessary because floating-point hardware is inefficient on FPGAs. One subtle detail of this conversion is the need for look-up tables, commonly used to avoid computationally intense operations such as trigonometric and logarithmic functions.

3 FPGA-based Hardware Acceleration

This section provides background information on hardware design on FPGA-based platforms, primarily for readers interested in exploring hardware acceleration for their applications. The description of the FBM hardware design follows in Section 4.
3.1 Field-Programmable Gate Arrays

A Field-Programmable Gate Array (FPGA) chip is a pre-fabricated silicon chip that can be programmed electrically to implement virtually any digital design. Its flexibility is derived from its underlying architecture, consisting of an array of programmable logic blocks interconnected by a programmable routing fabric. Additionally, modern FPGAs contain two specific structures that are used extensively in this work: on-chip memory blocks that can be used to store reasonable quantities of data (a maximum of about 7 Mbits on the devices used) and hard (dedicated non-programmable circuitry) multipliers. An FPGA chip enables the design of dedicated custom hardware, providing increased performance for computationally intensive applications, without the high power consumption and maintenance costs of networked clusters. Compared to Graphics Processing Units (GPUs), FPGAs offer greater flexibility in the design as one has the ability to customize the underlying architecture, instead of being constrained by it. Although GPUs represent yet another low-cost alternative, the FPGA-based approach was selected to allow maximum control over the computation, which is the key factor contributing to the significant reduction in execution time and increase in energy efficiency of the FPGA-based MCML simulation. Also, due to the random nature of a Monte Carlo simulation, coalescing global memory access on GPUs for Monte Carlo based applications will likely become a challenge. Slow global memory access and small shared memory size are often cited as limiting factors on current GPUs. Conversely, the flexibility of an FPGA allows the memory hierarchy as well as access patterns to be tailored to the application, thus maximizing the performance and energy efficiency of Monte Carlo simulations.
The design presented in this paper was implemented on a multi-FPGA platform called the Transmogrifier-4 (TM-4)\textsuperscript{20}, developed at the University of Toronto. This platform contains four FPGAs from the Altera Stratix I device family (Altera Corporation, San Jose, CA) and is designed to communicate easily with a computer.

3.2 Hardware Design Method

Hardware design requires the explicit handling of two concepts that are normally abstracted from software design: \textit{cycle-accurate} design and \textit{structural} design. Cycle-accurate design requires that the hardware must specify precisely what happens in each hardware clock cycle. A typical software designer will not be concerned with the number of clock cycles consumed in a processor for a section of code (although they do \textit{profile} the code to determine and reduce performance bottlenecks). Structural design requires that a hardware design specify exactly what resources to use and how they are connected. For software design, the underlying architecture and the internal execution units of a processor are not specified by the program and are typically not considered by the programmer.

To simplify the design flow in FPGA-based hardware development, specific Computer-Aided Design (CAD) tools are used, which are analogous to the compiler used by the software programmer. These CAD tools typically accept a Hardware Description Language, a textual description of the circuit structure that must consider both cycle-accurate and structural design, as input. The tools perform many sophisticated
optimizations to determine the logic implementation, location and connectivity routing to create a working high-speed digital hardware implementation.

To implement a large hardware design, the problem must be broken down into smaller sub-problems, each of which is solved by the creation of a module that is simulated in a cycle-accurate manner to ensure data consistency. Due to the vast amount of information gathered, a full system simulation cycle-by-cycle for large designs such as FBM will be too time-consuming.

Therefore, an intermediate stage involving the use of a simpler C-like language that models the cycle-accurate hardware design is employed to simulate the full system more quickly. This stage also allows for the testing and debugging of the additional complexity of cycle-accurate timing before considering structural design necessary in the final hardware design.

The design of an MCML system on the TM-4 followed these hardware design methods including the intermediate cycle-accurate timing stage. A C-based hardware modeling language called SystemC\textsuperscript{21} was used to develop the cycle-accurate intermediate stage between software and hardware design. Verilog\textsuperscript{22, 23} was selected as the hardware description language, and the Altera Quartus II 7.2 software as the CAD tool to synthesize the Verilog design into hardware structures as well as to configure the FPGA.
3.3 Hardware Acceleration Techniques

An FPGA can implement any digital circuit including those with significant amounts of computation and runs faster than software on a processor for two reasons. First, an FPGA can implement many computational units in parallel and second, it allows the organization of the data flow to keep all computational units busy.

A key factor limiting the amount of parallelism and hence the speed of an FPGA-based solution is the amount of logic available on the device. Therefore, minimizing the number of logic elements required for binary logic computation maximizes the performance per FPGA.

To achieve the goal of maximizing parallelism and computational throughput, three hardware acceleration techniques are commonly applied. First, to greatly reduce the size of a computational unit, the conversion from floating point to fixed point data representation is used, although careful design and modeling are essential to ensure that the proper precision level is maintained. Second, look-up tables can be created in on-chip memory to pre-compute values for expensive operations (such as trigonometric functions), thereby saving a large number of logic elements. The third key technique is pipelining\(^2^4\), which optimizes the computational throughput. The pipelining approach, similar to an assembly line, breaks down a complex problem into simpler stages, each of which is responsible for performing a simple task. Since each stage performs its task independently, the net throughput is increased, thereby speeding up the computation. An example of a pipeline is shown in Figure 2, where the calculation \(Y = aX^2 + b\) is broken
down in a pipelined fashion into 3 stages. Therefore, a continuous stream of new input data can be fed into this pipeline. Assuming each stage here takes the same amount of time, pipelining increases the throughput by a factor of three. While pipelining leads to significant performance gain, the complexity involved in designing and verifying the individual stages increases appreciably in sophisticated designs such as MCML.

4 FPGA-based MCML (FBM)

4.1 Hardware Design

The hardware-accelerated MCML design contains both hardware and software components. The hardware component resides on the TM-4 system and performs the core Monte Carlo simulation. The software on the host computer performs the pre-processing steps and post-processing steps. The former includes the parsing of the simulation input file and the initialization of the hardware system based on the simulation input file. The latter includes the transfer of the simulation results from the TM-4 back to the host computer and the creation of the simulation output file containing the absorption array. The absorption array is then used to generate the fluence distribution. The key steps illustrating the overall program flow from the user’s perspective are shown in Figure 3.

The TM-4 platform itself contains four Stratix I FPGA devices and each FPGA device houses one instance of FBM. The four instances together share the execution of the Monte Carlo simulation. FBM in turn consists of two major hardware modules: a controller and a photon simulator core. The controller implements miscellaneous tasks that are not part of the computationally intensive simulation of each photon. It reads
initialization information from the host computer and writes the simulation results back when the simulation is completed. This controller keeps track of the status of the simulation and communicates with the host computer.

The photon simulator is the core of the design and it dictates the overall performance of the system. The architecture of the simulator core is shown in Figure 4. The table below the architectural diagram outlines the on-chip memory usage, key computational blocks, and the latency (number of clock cycles) required by each module. The on-chip memory is mainly dedicated to storing look-up tables and the absorption array $A[r,z]$. As for the key computational blocks, the most resource intensive blocks are listed, such as multipliers, dividers, and square root blocks. The fact that only 18 multipliers, 2 square root blocks, and 1 divider are required by the compute-intense blocks in modules 4a, 4b, and 4c indicates the extensive optimizations applied to the current hardware design. Finally, the latency of each module represents the number of stages in each module of the pipeline. A single pass through the entire pipeline is equivalent to a single iteration in the key loop of the MCML program (Figure 1). The pipeline has 100 stages, meaning 100 photon packets at different stages in the simulation are handled concurrently once the pipeline is filled. Increasing the number of stages serves to decrease the complexity of each stage, thereby improving the clock speed. An example of using this technique is module 2, which lies in the critical path of the circuit. 60 stages were used to increase the clock speed of this part of the circuit and hence the overall clock speed of the pipeline at the expense of increased complexity.
To illustrate the complexity of the photon simulator core, the implementation of the Spin module, which computes the scattering angle and updates the new direction of the photon packet, is described here (Eq. 1).  

\[
\mu = \sin \theta (\mu_x \mu_z \cos \psi - \mu_y \sin \psi) / \sqrt{1 - \mu_z^2 + \mu_x \cos \theta} \quad (1)
\]

A direct implementation of Eq. 1 is very inefficient, resulting in low clock speed and high resource usage for each of the three direction cosines. The Stratix FPGAs on the TM-4 only contain dedicated hard multipliers, and do not contain dedicated hardware to perform division, square root or trigonometric functions. Hence, look-up tables stored in the on-chip memory are used to approximate the trigonometric functions. The division and square root functions are implemented directly in the FPGA programmable fabric since the high precision required here makes a look-up table based solution impractical. As these computations are relatively slow, they are split into many pipeline stages to increase the clock speed. The same pipelining technique is used to improve the performance of multipliers. Wherever possible, multipliers and dividers are shared to reduce resource usage, at the cost of increased complexity of the design.

Another unique aspect of the hardware design is the multiplexing (sharing) of computational units among modules 4a, 4b, and 4c, shown in Figure 4. This is possible because the result from only one of the two modules is used at any given time. The tight coupling of all connected modules is required to minimize resource usage and maximize speed. It is imperative that modules 4a, 4b, and 4c finish all their operations within exactly 37 clock cycles to ensure data consistency. The final stage (Roulette) determines whether a photon packet is still active, in which case it continues iterating at the
beginning of the pipeline. Otherwise, a new photon packet is selected to immediately enter the pipeline.

4.2 Tradeoffs

Due to the resource constraints on the prototyping platform used, several important tradeoffs were made on the final hardware design. First, the size of the on-chip memory (7.4 Mb for the Stratix I chip on the TM-4) limited the precision of each look-up table. Therefore, the number of entries for each look-up table was empirically determined based on the sensitivity of the function in the expected range of values. For example, the logarithmic function used in the computation of the step size $s$ is most sensitive within the range of 0 to 1 - the expected range of values provided by the uniformly distributed random number $\xi$, as shown in Eq. 2.

$$s = -\ln(\xi) / \mu_i$$  \hspace{2cm} (2)

To further maximize the on-chip memory space available to the look-up tables, the absorption array was limited to 256 by 256 elements in the radial and z direction, respectively. Also, the number of layers supported by the hardware was set to a maximum of five, also due to memory constraints. It is important to note that even though the number of layers is fixed at a maximum of five layers, their properties and the dimensions of the voxels ($dr$ and $dz$) can be modified easily through the standard input simulation file format used in the MCML program.
5 Results

For the purpose of validation and performance comparison, a skin model was selected as the simulation input to the MCML program. The tissue optical parameters presented in Table 1 are based on the light scattering study of tissues by Tuchin. The optical parameters of the skin for two wavelengths were used, namely 633nm and 337nm. To test the accuracy and performance of the hardware system with different tissue optical parameters, the absorption coefficient and scattering coefficient were varied systematically in a separate experiment, as described in the next section.

5.1 FPGA System-Level Validation Procedures

System validation consisted of three phases. The first phase involved verifying the FBM simulation outputs against the gold standard MCML executed on an Intel Xeon processor. Since Monte Carlo simulations are non-deterministic, it is important to separate the error introduced by the hardware implementation from the statistical uncertainty inherent in a Monte Carlo simulation. In other words, a fair comparison between MCML and FBM can only be obtained by considering the variance in the output of the MCML simulation, which is a 2D absorption array that scores the photon probability density [cm$^{-3}$] as a function of radius and depth. To quantify differences between these arrays, the relative error $E[i_i,][i_z]$ between corresponding elements was computed as follows:

$$E[i_i,][i_z] = \frac{|A_s[i_i,][i_z] - A_h[i_i,][i_z]|}{A_s[i_i,][i_z]}$$

(3)

where $A_s$ is the gold standard absorption array produced by MCML after launching 100 million photon packets and $A_h$ contains the corresponding elements in the absorption
array produced by FBM. To visualize the distribution of the relative error, a 2D color map showing the relative error in percent as a function of position was generated. For comparison, a reference color map depicts the relative error in the output from MCML compared to the gold standard absorption array to account for the statistical uncertainty between simulation runs. Photon packet numbers ranging from $10^5$ to $10^8$ were simulated.

To summarize the effect of varying the number of photon packets, the mean relative error was computed using Eq. 4, which averages the relative error in all elements in the absorption array with values above a randomly selected threshold ($0.00001\text{cm}^{-3}$). The setting of a threshold is necessary since the relative error is undefined when $A_s[i_r][i_z]$ (gold standard MCML output) reaches zero. This analysis enables the quantification of the impact of look-up tables and fixed-point conversion in the hardware implementation.

$$E_{ave} = \frac{\sum_{i_r=1}^{n_r} \sum_{i_z=1}^{n_z} E[i_r][i_z]}{n_r n_z}$$

(4)

where $E_{ave}$ is defined as the mean relative error and $E[i_r][i_z]$ is the relative error for each element (as defined in Eq. 3) with $n_r=256$, $n_z=256$.

To further characterize the behavior of the hardware system with varying tissue optical parameters, the performance and relative error based on $10^8$ photons were analyzed as a function of the target albedo. In a single-layer geometry, the target albedo, defined as $\mu_s/(\mu_a+\mu_s)$, was systematically varied from 0.50 to 0.96 in order to investigate the effects of tissue optical property on both the speedup and error.
The third phase for system-level validation of the FPGA-based hardware design involved analyzing the effect of the error within the context of PDT treatment planning. Isofluence maps were generated from the FBM output based on \(10^8\) photon packets. The relative shift in the position of the isofluence lines was analyzed by comparing against the gold standard MCML output.

5.2 Validation

Figures 5 and 6 show the distribution of \(E[i,]/[i_z]\) for \(10^5\) and \(10^8\) photon packets respectively, using Tuchin’s skin model at \(\lambda=633\text{nm}\). In both cases, the accuracy of FBM was comparable to that of MCML, as demonstrated by the similarity between the two error distributions (Figures 5a, 5b). The statistical uncertainty decreased for the simulation that used 100 million photon packets, as indicated by the expansion of regions within the \(r, z\) plane showing less than 5% relative error (Figure 6). This is expected as the variance in Monte Carlo simulations decreases by \(1/\sqrt{n}\) where \(n\) equals the number of photon packets. Figure 6a also shows some slight differences of about 1-2% (manifesting as an S-shaped region with lower error) in the region within a radius of 0.5cm (the high fluence region). Further analysis revealed that this S-shaped pattern can be eliminated by replacing the random number generator in the original MCML with the version implemented in hardware (Tausworthe generator\(^{26}\)). The disappearance of the S-shaped pattern with the use of the same random number generator (Tausworthe generator) shows that the minor deviation observed was due to the statistical differences in the random number sequence generated by two different random number generators (Fig. 6c, d).
To analyze the effect of photon packet number on the simulation accuracy, the mean relative error was computed according to Equation 4 (Figure 7). Figure 7a shows that the mean relative error of FBM closely tracked the mean relative error of MCML, both decreasing as the number of photon packets increased. Figure 7b shows the impact of converting from double-precision floating point operations to fixed point operations combined with the impact of the use of look-up tables on the relative error. As shown by the plot, the conversion introduced an increase in relative error of 0.2-0.5%.

In the second phase of the validation, the mean relative error as a function of the albedo was plotted (Figure 8a). The results show that for albedo values above 0.7, the increase in error was 0.5-1%, while for albedo values less than 0.7, the added error was up to 2%. This increase was caused by the significant reduction in the number of non-zero absorption array elements. For example, at an albedo of 0.90, there were 11407/65536 non-zero elements, but only 351/65536 non-zero elements at an albedo of 0.5. This high proportion of zero elements is due to the small voxel size used (\(dr=0.01\text{cm}\) and \(dz=0.002\text{cm}\)).

To investigate the impact of the 1-2% additional error within the context of PDT treatment planning, the isofluence lines for the impulse response based on simulation input parameters from Table 1 were plotted (Figure 9). The isofluence lines produced by FBM and MCML matched each other very well. The shift in the position of the isofluence lines was only noticeable for fluence levels at 0.00001\(\text{cm}^{-2}\) (8 orders of
magnitude smaller than the fluence near the centre – $10^0 cm^{-2}$). The detected shift was only around 0.1mm, which is of little significance in PDT treatment planning. Note that the 1-2\% error introduced is well within the uncertainties due to the variations in tissue optical properties, as shown by Rendon et al.\textsuperscript{27}

5.3 Performance

The execution speed of FBM was compared to the original MCML executed on a single Intel Xeon 3 GHz processor. For a complete end-to-end application runtime comparison, the runtime includes file I/O, system initialization, the Monte Carlo simulation, and all pre-processing/post-processing operations to generate the final simulation output file. Table 2 shows the specifications of the test platform used to execute MCML. The software version of MCML was compiled using full compiler optimizations (gcc -O3 optimization flag\textsuperscript{28}).

As shown in Table 3a, the runtime of the Monte Carlo simulation using 100 million photon packets was reduced from over 2.5 hours in software to approximately 2 minutes in hardware for the 633nm case. The overall speedup was 78 times including a data transfer time of 8 seconds. Using the tissue optical properties at 337nm (Table 1), the overall speedup was 66 times, mainly due to the much shorter execution time and hence the importance of the data transfer time. However, the data transfer rate was far from expected due to a known issue in the communication channel on the TM-4 prototyping system. Normally, the communication channel (host-to-FPGA PCI bus) supports a bandwidth of 266MB/s for writes to the FPGA and 154MB/s for reads from the FPGA to
Currently, it takes 8 seconds to transfer 610kB of data. Hence, the use of commercial prototyping platforms with fully functional communication channels should yield a net 84 times speedup without any modifications to the design for the 633nm case and 80 times speedup for the 337nm case. Figure 8b shows that as the albedo increased, the speedup increased from 77 to 87 times, since the MCML software executes more expensive computations for calculating the scattering angle in the Spin function at higher albedo. The average speedup was 80 times with the current TM-4 platform running at a clock speed of 1/75 compared to the Xeon processor.

Table 4 shows the resource utilization (number of logic elements, DSP blocks, and the capacity of the on-chip memory) and clock speed of FBM on a Stratix I FPGA device and the new Stratix III FPGA device. On Stratix I, only one instance of the current design can be accommodated. On the state-of-the-art Stratix III, two instances of the same design can be replicated for an additional twofold speedup. Although this design only occupies 16.3% of the available logic elements on the Stratix III, on-chip memory size restrictions limit the number of replicas to 2. Also, the FBM design can run at 1.5 times the clock speed on Stratix III. As an existing Stratix III device is specified in the full compilation and all the figures generated in Table 4 are based on the report generated by Altera Quartus 7.2, it is highly probable to increase the speedup by an additional factor of 3 to a projected speedup of 240 times.

Table 5 shows the power consumption comparison of MCML on a network cluster with 84 cores versus the TM-4. The worst-case power for the processor was obtained from the
specifications published by Intel. Perfect parallelism was assumed for a network cluster with no communication overhead. Power consumed by memory, communication through Ethernet, and other off-chip component was ignored. The worst-case power for the TM-4 was based on the maximum power consumed by all four Stratix chips on the TM-4. While a power simulation of the FBM design using the Quartus tool showed a power consumption of 14.4 Watts including I/O power, the worst case, 60 W, was chosen as a conservative approach. The power-delay product is a metric used to compare energy efficiency of different implementations. It provides a convenient way to normalize power and delay trade-offs. The results show that FBM is 45 times more energy efficient than the processor using worst-case power.

6 Conclusion

Using the MCML program as the gold standard, custom pipelined hardware was designed on a multi-FPGA platform known as the TM-4 to achieve an 80 times speedup compared to a 3GHz Intel Xeon processor. The development time was approximately 1 person-year and future modifications can be readily implemented due to the use of a modularized pipelined architecture. Isofluence distribution maps generated by FBM and MCML were compared at 100 million photon packets, showing only a 0.1mm shift in the hardware-generated isofluence lines from those produced by MCML for fluence levels as low as 0.00001cm$^{-2}$. This shift is negligible within the context of PDT treatment planning considering the typically much larger margin of safety for surgical resection or treatment planning in radiation therapy.
7 Implications and Future Work

The limitations of the current prototype design, such as the number of layers, could be
relaxed on newer FPGA platforms which offer more on-chip memory and other
resources. Migrating the current design to modern Stratix III FPGA chips will result in a
projected 240 times speedup, requiring minor modifications to the communication
interface. In future studies, the use of external memory has several implications. First,
more replicas of the design can be accommodated since the on-chip memory space is a
limiting factor, directly translating to an increase in the attainable speedup. Second, using
external memory enables the three-dimensional modeling of tumors, which for realistic
cases requires at least 1024 x 1024 x 1024 voxels (a minimum of 4GB assuming 4 bytes
per voxel). Finally, the significantly larger memory space offered by external memory
will allow further optimization of the number of entries in the look-up tables to improve
the accuracy of the simulation. Determining the precise tradeoffs between accuracy and
resource usage as well as the migration to newer platforms will be the subject of future
work.

For investigators interested in accelerating other light propagation models such as FEM-
based models that solve the radiative transfer equation numerically using the diffusion
approximation$^{29}$, an FPGA-based approach may serve as an alternative. Here, the unique
technical challenges will primarily include mapping the matrix operations onto hardware
and implementing an iterative solver based on techniques such as the conjugate gradient
method\textsuperscript{30}. Tailoring the FPGA-based hardware to the system of matrices specific to the application will be a key step in the design process.

The possible implications of the current study are two-fold. First, the pipelined design could form the basis upon which more complex Monte Carlo simulations or other light transport models can be built. The flexible pipelined architecture allows the addition of extra stages such as those required by external memory accesses without significantly impacting the performance. Secondly, the dramatic reduction in treatment planning time achieved by an FPGA platform may potentially enable real-time treatment planning based on the most recent images of the treatment volume, taking into account the changing tissue optical properties as the treatment progresses. 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.\textsuperscript{31} The significant performance gain provided by the hardware approach can potentially enable PDT treatment planning in heterogeneous, spatially complex tissues using more sophisticated MC-based models.

**Acknowledgments**

The authors acknowledge the financial support from CIHR (Grant No. 68951) and NSERC (Discovery Grant No. 171074) as well as the funding from the NSERC and OGS scholarships.
References


Table 1 Optical properties of the five-layer skin tissue - 633nm (337nm)

<table>
<thead>
<tr>
<th>Layer</th>
<th>(a) (cm(^{-1}))</th>
<th>(s) (cm(^{-1}))</th>
<th>(g)</th>
<th>(n)</th>
<th>Thickness (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (epidermis)</td>
<td>4.3(32)</td>
<td>107(165)</td>
<td>0.79(0.72)</td>
<td>1.5</td>
<td>0.01</td>
</tr>
<tr>
<td>2 (dermis)</td>
<td>2.7(23)</td>
<td>187(227)</td>
<td>0.82(0.72)</td>
<td>1.4</td>
<td>0.02</td>
</tr>
<tr>
<td>3 (dermis with plexus superficialis)</td>
<td>3.3(40)</td>
<td>192(246)</td>
<td>0.82(0.72)</td>
<td>1.4</td>
<td>0.02</td>
</tr>
<tr>
<td>4 (dermis)</td>
<td>2.7(23)</td>
<td>187(227)</td>
<td>0.82(0.72)</td>
<td>1.4</td>
<td>0.09</td>
</tr>
<tr>
<td>5 (dermis plexus profundus)</td>
<td>3.4(46)</td>
<td>194(253)</td>
<td>0.82(0.72)</td>
<td>1.4</td>
<td>0.06</td>
</tr>
</tbody>
</table>

* Tissue optical properties according to Tuchin\textsuperscript{25}

Table 2: Specifications of test platform

<table>
<thead>
<tr>
<th>University Health Network Linux cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Cache</td>
</tr>
<tr>
<td>Operating System</td>
</tr>
<tr>
<td>Compiler</td>
</tr>
</tbody>
</table>

Table 3a: Runtime of MCML and FBM for 100 million photon packets averaged over 4 independent simulation runs (Input parameters from Table 1 – 633nm)

<table>
<thead>
<tr>
<th>Device</th>
<th>Clock Speed</th>
<th>Simulation Time (sec)</th>
<th>Data Transfer Time (sec)</th>
<th>Total Runtime (sec)</th>
<th>Overall Speedup</th>
<th>Speedup excluding data transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon</td>
<td>3.06 GHz</td>
<td>9150</td>
<td>0</td>
<td>9150 ± 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TM-4</td>
<td>40 MHz</td>
<td>109</td>
<td>8</td>
<td>117 ± 1</td>
<td>78 ± 1</td>
<td>84 ± 1</td>
</tr>
</tbody>
</table>

Table 3b: Runtime of MCML and FBM for 100 million photon packets averaged over 4 independent simulation runs (Input parameters from Table 1 – 337nm)

<table>
<thead>
<tr>
<th>Device</th>
<th>Clock Speed</th>
<th>Simulation Time (sec)</th>
<th>Data Transfer Time (sec)</th>
<th>Total Runtime (sec)</th>
<th>Overall Speedup</th>
<th>Speedup excluding data transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon</td>
<td>3.06 GHz</td>
<td>3100</td>
<td>0</td>
<td>3100 ± 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TM-4</td>
<td>40 MHz</td>
<td>39</td>
<td>8</td>
<td>47 ± 1</td>
<td>66 ± 1</td>
<td>80 ± 1</td>
</tr>
</tbody>
</table>
Table 4 Comparison of two FPGA devices and the resource utilization of the current design (one instance of the design on a single chip) on both devices

<table>
<thead>
<tr>
<th>FPGA Device</th>
<th>Number of Logic Elements</th>
<th>Number of DSP blocks</th>
<th>On-chip Memory (Mbits)</th>
<th>Clock Speed (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix I EP1S80F1508C6 (130 nm)</td>
<td>64,000 out of 79,000 LUTs‡</td>
<td>160 out of 176</td>
<td>4.8 Mbits out of 7.4 Mbits</td>
<td>41 MHz</td>
</tr>
<tr>
<td>Stratix III EP3SL340H1152C3 (65 nm)</td>
<td>44,000 out of 270,000 ALMs‡</td>
<td>104 out of 1152</td>
<td>4.8 Mbits out of 16.7 Mbits †</td>
<td>62 MHz</td>
</tr>
</tbody>
</table>

† Although it appears that 3 instances of the design can fit in the Stratix III device, in reality only 2 instances can be accommodated due to memory block size restrictions.

‡ The types of logic elements provided by the Stratix I and Stratix III devices are different.

Table 5 Worst-case chip power and power efficiency comparison between the TM-4 and the Intel 3.06GHz Xeon processor.

<table>
<thead>
<tr>
<th></th>
<th>Worst-case Power (Watts)</th>
<th>Delay for 100M photons (seconds)</th>
<th>Power-Delay Product (Joules)</th>
<th>Normalized Power-delay Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor (single-core)</td>
<td>32.5</td>
<td>9150</td>
<td>297000</td>
<td>45</td>
</tr>
<tr>
<td>Cluster (84 cores)</td>
<td>2730</td>
<td>109</td>
<td>297000</td>
<td>45</td>
</tr>
<tr>
<td>TM-4</td>
<td>60</td>
<td>109</td>
<td>6540</td>
<td>1</td>
</tr>
</tbody>
</table>
Fig. 1 Key steps in the MCML program

Fig. 2 An example of a three-stage pipeline for computing $Y=aX^2+b$

Fig. 3 Overall program flow. Step 1 – Parsing of the simulation input file. Step 2 – Transfer of initialization information to the TM-4. Step 3 – Transfer of simulation results from the TM-4. Step 4 – Creation of the simulation output file.
### Fig. 4 Pipelined architecture of FBM

<table>
<thead>
<tr>
<th>Module</th>
<th>On-chip memory usage</th>
<th>Key Computational blocks (Resource Intensive)</th>
<th>Latency (Number of clock cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Compute Step Size</td>
<td>Log lookup table</td>
<td>3 multipliers</td>
<td>1</td>
</tr>
<tr>
<td>2. Check Boundary</td>
<td>...</td>
<td>3 multipliers, 1 divider</td>
<td>60</td>
</tr>
<tr>
<td>3. Hop</td>
<td>...</td>
<td>3 multipliers</td>
<td>1</td>
</tr>
<tr>
<td>4. a) Reflect or Transmit</td>
<td>...</td>
<td>...</td>
<td>37</td>
</tr>
<tr>
<td>4. b) Drop</td>
<td>Absorption array</td>
<td>3 multipliers, 1 square root</td>
<td>37</td>
</tr>
<tr>
<td>4. c) Spin</td>
<td>Trig lookup tables</td>
<td>...</td>
<td>37</td>
</tr>
<tr>
<td>Shared Resources</td>
<td>Fresnel and other</td>
<td>15 multipliers, 1 divider, 1 square root</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>trig function lookup</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Roulette</td>
<td>...</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>
Fig. 5 Distribution of relative error as a function of radius and depth using 100 thousand photon packets (633nm) (a) FBM (100 thousand) versus MCML (100 million) (b) MCML (100 thousand) versus MCML (100 million). Color bar represents percent error from 0% to 10% (Values above 10% are represented by the same color as the maximum value in the color scale).
Fig. 6 Distribution of relative error as a function of radius and depth using 100 million photon packets (633nm) (a) FBM versus MCML (run 2) (b) MCML (run 1) versus MCML (run 2). (c) FBM versus MCML with Tausworthe generator (run 2) (d) MCML with Tausworthe generator (run 1) versus MCML with Tausworthe generator (run 2). Color bar represents percent error from 0% to 10%.

Fig. 7 Relative error as a function of the number of photon packets simulated (633nm). The horizontal axis is in logarithmic scale. The ‘◊’ symbol represents the relative error between two independent MCML runs. (a) The ‘×’ symbol represents the relative error comparing the results produced by FBM and MCML. (b) The ‘×’ symbol represents the relative error of the results produced by the C program modeling look-up tables and fixed-point operations compared to MCML which uses double-precision floating point operations. Each point represents the mean obtained from four simulation runs.
Fig. 8 (a) Relative error with varying albedo ($10^8$ photon packets). The ‘◊’ symbol represents the relative error between two independent MCML runs at $10^8$ photon packets. The ‘×’ symbol represents the relative error between the results produced by FBM and MCML. (b) Speedup at different albedo values ($10^8$ photon packets). Each point represents the mean obtained from four simulation runs.

Fig. 9 Comparison of the isofluence lines for the impulse response generated by FBM and MCML using 100 million photon packets (633nm). Open symbols ○, ◊, and □ represent results from MCML, while the symbols ●, +, and × represent results from FBM. (a) Isofluence lines for fluence levels at 1000 cm$^{-2}$, 100 cm$^{-2}$, and 10 cm$^{-2}$ respectively as indicated on the figure. (b) Isofluence lines for fluence levels at 1 cm$^{-2}$, 0.01 cm$^{-2}$, and 0.00001 cm$^{-2}$. 