Hardware acceleration of a Monte Carlo simulation for photodynamic treatment planning

William Chun Yip Lo
University of Toronto
Department of Medical Biophysics
Rm. 8-324,
610 University Avenue
Toronto, Ontario M5G 2M9 Canada

Keith Redmond
Jason Luu
Paul Chow
Jonathan Rose
University of Toronto
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
10 King's College Road
Toronto, Ontario M5S 3G4

Lothar Lilge
Princess Margaret Hospital
Ontario Cancer Institute
University of Toronto
Department of Medical Biophysics
Rm. 7-416,
610 University Avenue
Toronto, Ontario M5G 2M9 Canada

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 MC simulations would enable the use of MC-based models for accomplishing such tasks. To explore this possibility, a digital hardware implementation of a MC 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 MC simulation on average 80 times faster and was 45 times more energy efficient than the MCML software executed on a 3-GHz Intel Xeon processor. The resulting isofluence lines closely matched those produced by MCML in software, diverging by only less than 0.1 mm for fluence levels as low as 0.00001 cm^{-2} in a skin model.
[DOI: 10.1117/1.3080134]

Keywords: Monte Carlo simulation; Monte Carlo multilayered media; photodynamic therapy treatment planning; hardware acceleration; pipelining; field-programmable gate array.

Paper 08332R received Sep. 12, 2008; revised manuscript received Dec. 2, 2008; accepted for publication Dec. 17, 2008; published online Feb. 27, 2009.

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 is the administration of a photosensitizer, followed by the irradiation of the target volume with light of a specific wavelength to activate the photosensitizer locally.[1–4] Advances in PDT have enabled 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.[5–9] Compared to conventional treatments such as surgery, radiotherapy, and chemotherapy, PDT is a minimally invasive procedure that achieves tumor destruction without systemic toxicity. This is especially beneficial for head and neck cancers, since the surgical resection of even small tumors can lead to functional impairment or disfiguration.[10] 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 (in J cm^{-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 3-D geometries of the tissue with varying optical properties and light sources with predetermined emission patterns, its high reproducibility, and its accuracy.[11] Similarly, Monte Carlo (MC) simulations are used widely as the gold standard in radiotherapy treatment planning and there is a clear trend toward adopting the MC method for clinical radiotherapy dose calculations in commercial treatment planning systems.[12,13] Unfortunately, such simulations are also known to be very time consuming and

Address all correspondence to: Lothar Lilge, Department of Medical Biophysics, Ontario Cancer Institute, Princess Margaret Hospital, Rm. 7-416, 610 University Avenue, Toronto, ON, Canada M5G 2M9. Tel: 416–946–4501 x5743; Fax: 416–946–6529; E-mail: llilge@uhnres.utoronto.ca

1083-3668/2009/14(1)/014019/11/$25.00 © 2009 SPIE

different variance reduction schemes or efficiency-enhancing methods are traditionally introduced to reduce the computation time.[14] However, the computation time for MC remains high and this limits its use to solving only the forward solutions for a given source geometry, emission profile, and optical interaction coefficients of the tissue. PDT treatment planning requires solving the inverse solution to achieve a given desired dose distribution. Accelerating MC simulations would enable the use of MC-based models for solving such inverse problems.

Attempts to accelerate MC 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.[15,16] One potential problem with the software parallelization approach is the need for dedicated access to a computer cluster to achieve the desired performance. Overall, this approach is not easily accessible as the costs of high-end computing infrastructure are substantial, thus hindering the deployment of complex MC-based models. This paper explores the use of custom-built hardware to accelerate the MC simulation for computing light dose in PDT. The key advantages of this approach include the greater scalability, portability, as well as lower power consumption due to the use of dedicated hardware. In addition, a purpose-built system could be significantly cheaper than a large-scale computer cluster.

Using the widely accepted MC for multilayered media[17] (MCML) code as the gold standard, this paper demonstrates the feasibility of the hardware-based approach for accelerating MC 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 3-GHz 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 *de novo* on programmable logic devices called field-programmable gate arrays[18] (FPGAs).

The remainder of this paper discusses the FPGA-based hardware design, called here FPGA-based MCML or FBM for short. Beginning with a brief overview of MCML, the unique aspects of the design are explained to highlight how various hardware acceleration schemes were utilized to achieve the reduced computation time. The validation results and performance analysis are presented, followed by the possible implications of the significant reduction in computation time for MC-based models within the context of PDT treatment planning.

2 MCML

2.1 *Overview*

The MCML approach and code[17] provides an MC model of steady state light transport in multilayered media. It assumes infinitely wide layers and models an incident pencil beam that is 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[19]). Three physical quantities are scored in MCML, namely absorption, reflectance, and transmittance. Absorption in the tissue is stored in a 2-D array $A[r][z]$, which represents the photon absorption probability density as a function of radius r and depth z (measured in cm^{-3}, and normalized to the total number of photon packets). It can be further converted into photon fluence (measured in cm^{-2}). To obtain the isofluence lines of various light sources such as a Gaussian beam the CONV program is used to parse the simulation output file and generate a new file with the locations of the contour lines.

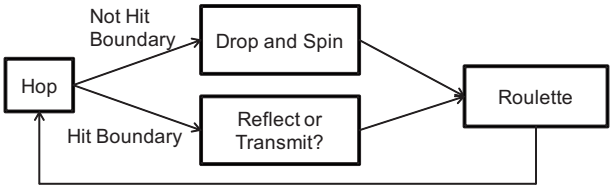


Fig. 1 Key steps in the MCML program.

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 still required for generating a low-noise fluence 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 (Fig. 1). The hop step moves the photon packets from the current position to the next interaction site by computing the step size through 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.[20] When a photon packet exits the tissue through the top or bottom layer, it is terminated. 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 due to energy conservation requirements.

2.2 *Modifications*

Considering the envisioned application in PDT treatment planning and the hardware design requirements, two key modifications were made to the MCML program. 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 2-D absorption array $A[r][z]$ is 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 nontrivial conversion was necessary because floating-point hardware is very 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 general hardware design on FPGA-based platforms, primarily for readers interested in exploring hardware acceleration for their applications. The details of the FBM hardware design are given in Sec. 4.

3.1 *FPGAs*

An FPGA chip is a prefabricated 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 nonprogrammable 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. The subtlety of the NVIDIA GPU architecture, for example, can make it difficult to achieve high performance for certain applications, such as those with significant divergent behaviors leading to warp serialization and undesirable memory access patterns. Therefore, the FPGA-based approach was selected to create hardware architecture that is tailored to the data flow and computation for the specific application.

The design presented in this paper was implemented on a multi-FPGA platform called the Transmogrifier-4 (TM-4),[21] developed at the University of Toronto. This platform contains four FPGAs from the Altera Stratix I device family (Altera Corporation, San Jose, California) 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: cycle-accurate design and structural design. Cycle-accurate design requires a hardware design that 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 profile the code to determine and reduce performance bottlenecks). Structural design requires that a hardware design specifying 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 or considered by the programmer.

To simplify the design flow in hardware development, Computer-aided design (CAD) tools are used, which are analogous to the compiler used by the software programmer. CAD tools typically accept a hardware description language as input, which is a textual description of the circuit structure. The tools perform many sophisticated optimizations to determine the precise 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 subproblems, 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. Verilog[22,23] was selected as the hardware description language, and Altera Quartus II 7.2 software was the CAD tool to synthesize the Verilog design into hardware structures as well as to configure the FPGA. A C-based hardware modeling language called SystemC[24] was used to develop the cycle-accurate intermediate stage between software and hardware design.

3.3 *Hardware Acceleration Techniques*

An FPGA can implement any digital circuit including those with significant amounts of computation and the custom circuit can run faster than software on a processor for two reasons. First, an FPGA can implement many computational units in parallel and second, it allows exact 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 number of logic elements 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 precompute values for expensive operations (such as trigonometric functions), thereby saving a large number of logic elements. The third key technique is pipelining, 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 Fig. 2, where the calculation $Y=aX^2+b$ is broken down in a pipelined fashion into three stages. Therefore, a continuous stream of a 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 fac-

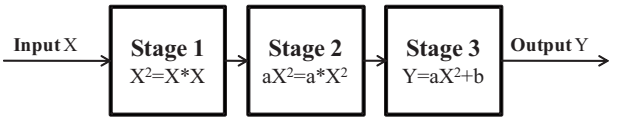


Fig. 2 Example of a three-stage pipeline for computing $Y=aX^2+b$.

tor of three. While pipelining leads to a significant performance gain, the complexity involved in designing and verifying the individual stages increases appreciably in sophisticated designs such as MCML.

4 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 MC simulation. The software on the host computer performs the preprocessing steps and postprocessing 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 Fig. 3.

The TM-4 platform[21] 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 MC 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 Fig. 4. The table below the architectural diagram outlines the on-chip memory usage, key

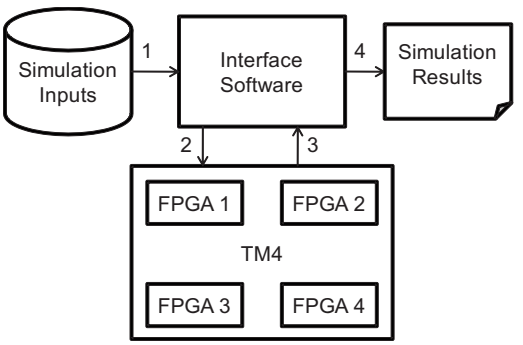
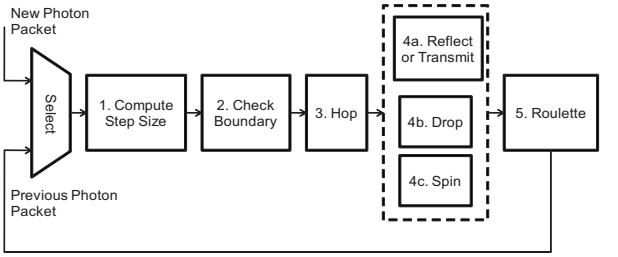


Fig. 3 Overall program flow: step 1, parsing of the simulation input file; step 2, transfer of initialization information to the TM-4; step 3, transfer of simulation results from the TM-4; and step 4, creation of the simulation output file.



Module	On-chip memory usage	Key Computational blocks (Resource Intensive)	Latency (Number of clock cycles)
1. Compute Step Size	Log lookup table	3 multipliers	1
2. Check Boundary	---	3 multipliers, 1 divider	60
3. Hop	---	3 multipliers	1
4. a) Reflect or Transmit	---	---	37
b) Drop	Absorption array	3 multipliers, 1 square root	37
c) Spin	Trig lookup tables	---	37
Shared Resources	Fresnel and other trig function lookup tables	15 multipliers, 1 divider, 1 square root	N/A
5. Roulette	---	---	1

Fig. 4 Pipelined architecture of FBM.

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 eighteen multipliers, two square root blocks, and one 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 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 (Fig. 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, as already discussed. An example of using this technique is module 2, which lies in the critical path of the circuit. Sixty 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,[16] is described here.

$$\mu_x' = \frac{\sin\theta(\mu_x\mu_z\cos\psi - \mu_y\sin\psi)}{\sqrt{1-\mu_z^2}} + \mu_s\cos\theta. \quad (1)$$

A direct implementation of this computation would be 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

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

Layer	u_a (cm^{-1})	u_s (cm^{-1})	g	n	Thickness (cm)
1 (epidermis)	4.3(32)	107(165)	0.79(0.72)	1.5	0.01
2 (dermis)	2.7(23)	187(227)	0.82(0.72)	1.4	0.02
3 (dermis with plexus superficialis)	3.3(40)	192(246)	0.82(0.72)	1.4	0.02
4 (dermis)	2.7(23)	187(227)	0.82(0.72)	1.4	0.09
5 (dermis plexus profundus)	3.4(46)	194(253)	0.82(0.72)	1.4	0.06

* Tissue optical properties according to Tuchin.[25]

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 Fig. 4. This is possible because the result from only two of the three 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 *Trade-Offs*

Due to the resource constraints on the prototyping platform used, several important trade-offs were made on the final hardware design. First, the size of the on-chip memory (7.4 Mbits 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 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 highly sensitive within the range of 0 to 1—the expected range of values provided by the uniformly distributed random number ξ, as shown in Eq. (2):

$$s = -\ln(\xi)/\mu_t. \tag{2}$$

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

5 Validation Procedures

5.1 *Validation Model*

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.[25] The optical parameters of the skin for two wavelengths were used, namely 633 nm and 337 nm. 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.

5.2 *FPGA System-Level Validation*

System validation consisted of three phases. The first phase involved verifying the FBM simulation output against the gold standard MCML executed on an Intel Xeon processor. Since MC simulations are non-deterministic, it is important to separate the error introduced by the hardware implementation from the statistical uncertainty inherent in an MC 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. The output is a 2-D array that scores the absorbed photon probability density (in cm^{-3}) as a function of radius and depth. To quantify the difference between these arrays, the relative error $E[i_r][i_z]$ between the corresponding elements is computed using the following formula:

$$E[i_r][i_z] = \frac{|A_s[i_r][i_z] - A_h[i_r][i_z]|}{A_s[i_r][i_z]}, \tag{3}$$

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 2-D color map was generated, showing the relative error in percent as a function of position. 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.

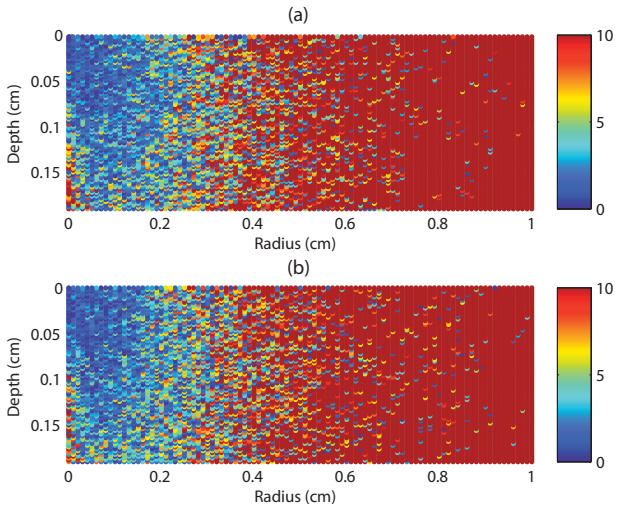


Fig. 5 Distribution of relative error as a function of radius and depth using 100 thousand photon packets (633 nm): (a) FBM (100 thousand) versus MCML (100 million) and (b) MCML (100 thousand) versus MCML (100 million). The bar represents percent error from 0 to 10% (values above 10% are represented by the same color as the maximum value in the color scale).

To summarize the effect of varying the number of photon packets, the mean relative error [Eq. (4)] was computed by averaging the relative error in all elements in the absorption array with values above a randomly selected threshold ($0.00001\ \mathrm{cm}^{-3}$). The setting of a threshold is necessary since 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.

$$E_{\mathrm{ave}} = \frac{\sum_{i_z=1}^{n_z}\sum_{i_r=1}^{n_r} E[i_r][i_z]}{n_r n_z}, \tag{4}$$

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

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.

6 Results

6.1 *Validation*

Figures 5 and 6 show the distribution of the relative error for 10^5 and 10^8 photon packets, respectively, using Tuchin's skin

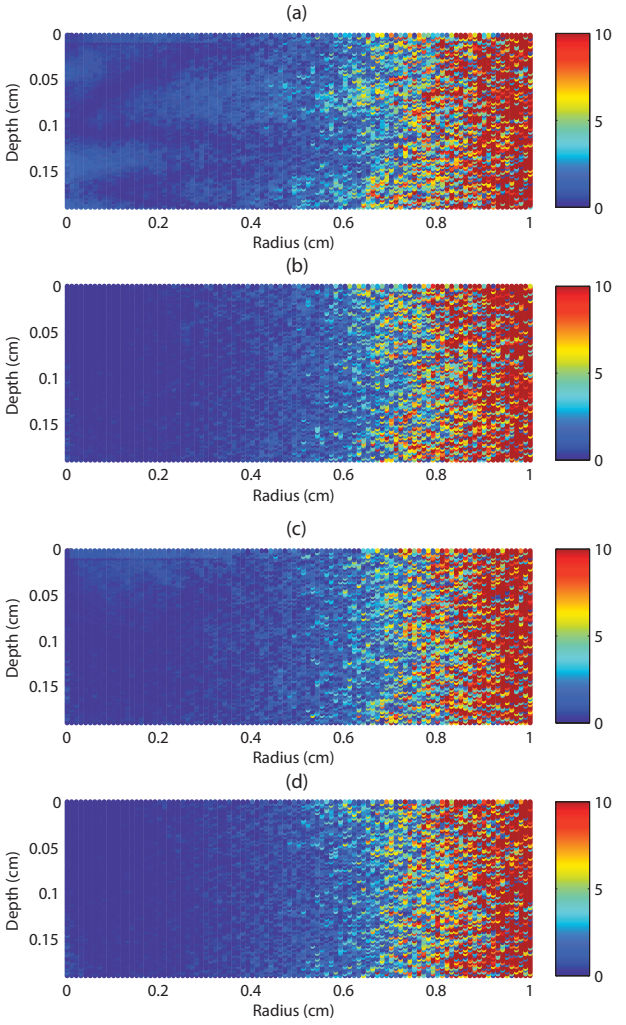


Fig. 6 Distribution of relative error as a function of radius and depth using 100 million photon packets (633 nm): (a) FBM versus MCML (run 2), (b) MCML (run 1) versus MCML (run 2), (c) FBM versus MCML with Tausworthe generator (run 2), and (d) MCML (run 1) versus MCML (run 2) both with Tausworthe generator. Color bar represents percent error from 0 to 10%.

model at $\lambda=633$ nm. In both cases, the accuracy of FBM was comparable to that of MCML, as demonstrated by the similarity between the two error distributions [Figs. 5(a) and 5(b)]. 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% error (Fig. 6). This is expected as the variance in MC simulation decreases by $1/\sqrt{n}$, where n equals the number of photon packets. Figure 6(a) also shows some slight differences of about 1 to 2% (manifesting as an S-shaped region with lower error) in the region with a radius of 0.5 cm (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 the 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 se-

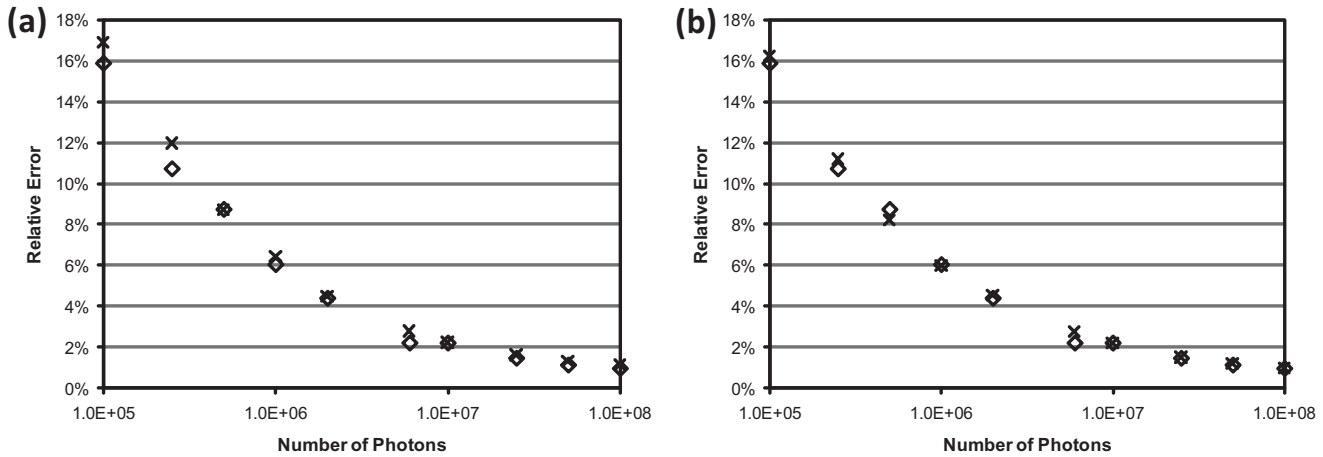


Fig. 7 Relative error as a function of the number of photon packets simulated (633 nm). The horizontal axis is in logarithmic scale: ◇, relative error between two independent MCML runs. (a) ×, relative error comparing the results produced by FBM and MCML and (b) ×, relative error of the results produced 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.

quence generated by two different random number generators [Figs. 6(c) and 6(d)].

To analyze the effect of photon packet number on the accuracy of the simulation, the mean relative error was computed [Eq. (4)]. Figure 7(a) 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 7(b) 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 to 0.5%.

In the second phase of the validation, the mean relative error as a function of the albedo was plotted [Fig. 8(a)]. The results show that for albedo values above 0.7, the increase in error was 0.5 to 1%, while for albedo values less than 0.7, the added error was up to 2%. This increase was mainly 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 non-zero elements out of 65536 (256 by 256) elements, while at an albedo of 0.5, only 351 non-zero elements were left. The small voxel size used (dr=0.01 cm and dz=0.002 cm) also contributed to this difference.

To investigate the impact of 1–2% additional error within the context of PDT treatment planning, the isofluence lines for the impulse response based on the simulation input parameters from Table 1 were plotted (Fig. 9). The isofluence lines produced by FBM and MCML matched very well. A shift in the position of the isofluence lines was only noticeable for fluence levels at 0.00001 cm^{-2} (eight orders of magnitude smaller than the fluence near the center—1000 cm^{-2}). The detected shift was only around 0.1 mm, which is of little significance in PDT treatment planning. Note that the 1 to 2% error introduced is well within the uncertainties due to the variations in tissue optical properties, as shown by Rendon et al.[27]

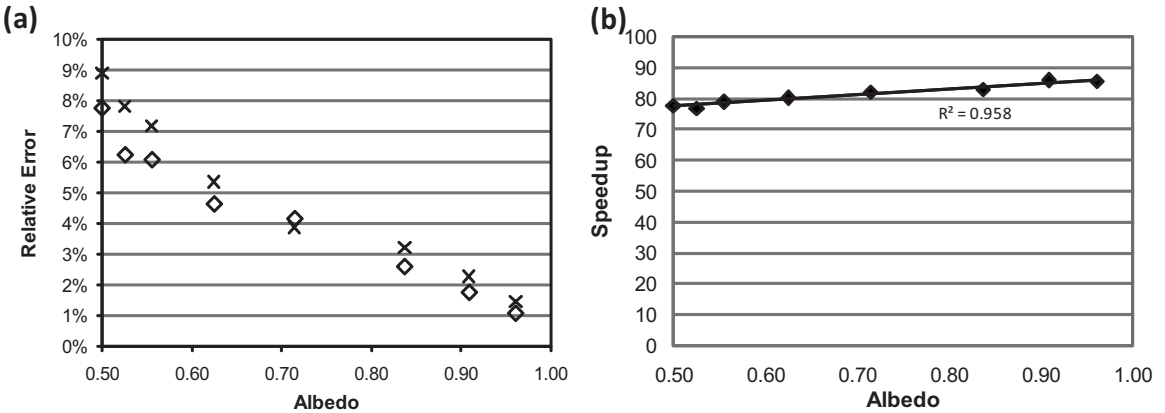


Fig. 8 (a) Relative error with varying albedo (10^8 photon packets); ◇, relative error between two independent MCML runs at 10^8 photon packets; ×, relative error between the results produced by FBM and MCML. (b) Speedup at different albedo values (10^8 photon packets). Each point represents the mean obtained from four simulation runs.

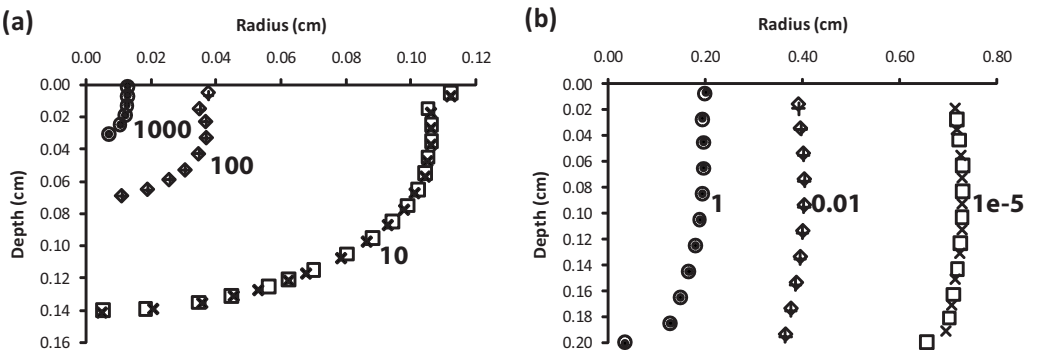


Fig. 9 Comparison of the isofluence lines for the impulse response generated by FBM and MCML using 100 million photon packets (633 nm): open symbols ○, ◇, and □ results from MCML; and ●, +, and ×, results from FBM. (a) Isofluence lines for fluence levels at 1000, 100, and 10 cm^{-2}, as indicated on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

6.2 *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 MC 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).[28]

As shown in Table 3(a), the runtime of the MC simulation using 100 million photon packets was reduced from over 2.5 h in software to approximately 2 min in hardware for the 633 nm case. The overall speedup was 78 times including a data transfer time of 8 s. Using the tissue optical properties at 337 nm [Table 3(b)], the overall speedup was 66 times, mainly due to the much shorter execution time and hence the relative 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 (peripheral component interconnect) bus] supports a bandwidth of 266 MBytes/s for writes to the FPGA and 154 MBytes/s for reads from the FPGA to the host.[21] Currently, it takes 8 s to transfer 610 kBytes of data. Hence, the use of commercial prototyping platforms with fully functional communication channels should yield a net 84 times speedup for the 633 nm case and 80 times speedup for the 337 nm case without any modifications to the design. Figure 8(b) 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 times compared to that of Xeon processor.

Table 4 shows the resource utilization (number of logic elements, DSP blocks, and on-chip memory usage) and clock speed of FBM on a Stratix I FPGA device and a modern Stratix III FPGA device. On Stratix I, only one instance of the current design can be accommodated. On Stratix III, two instances of the same design can be replicated for an additional twofold speedup. Although this design only occupies about 16.3% of the available logic elements on Stratix III, on-chip memory size restrictions limit the number of replicas to two. Also, the FBM design can run at 1.5 times the clock speed on Stratix III. As the figures presented in Table 4 are based on the full compilation report generated by Altera Quartus 7.2 using an existing Stratix III device, it is possible to increase the speedup by an additional factor of 3 to a projected speedup of 240 times given a platform with four Stratix III FPGA chips and a high-speed communication interface.

Table 5 shows the power consumption and energy efficiency of MCML on a network cluster with 84 cores versus the TM-4 with the same performance. 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. The power consumed by memory, Ethernet, and other off-chip components was ignored. The worst-case power for the TM-4 was based on the maximum power consumed by all four Strattix I chips on the TM-4. While a power simulation of the FBM design using the Quartus tool showed a much lower power consumption, the worst case power (60 W) was chosen to remain conservative in the comparison. 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 Intel Xeon processor based on worst-case power consumption.

Table 2 Specifications of test platform.

University Health Network Linux Cluster.	
Processor	Intel Xeon 3-GHz CPU[a] (130 nm)
Memory	2 GBytes RAM
Cache	512 kBytes
Operating system	Red Hat Linux 3.2.2-5
Compiler	gcc 3.2.2

[a]Processor architecture can affect execution time.

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

Device	Clock Speed	Simulation Time (s)	Data Transfer Time (s)	Total Runtime (s)	Overall Speedup	Speedup excluding data transfer
Intel Xeon	3.06 GHz	9150	0	9150±1	1	1
TM-4	40 MHz	109	8	117±1	78±1	84±1

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

Device	Clock Speed	Simulation Time (s)	Data Transfer Time (s)	Total Runtime (s)	Overall Speedup	Speedup excluding data transfer
Intel Xeon	3.06 GHz	3100	0	3100±1	1	1
TM-4	40 MHz	39	8	47±1	66±1	80±1

7 Conclusion

Using the MCML program as the gold standard, custom pipelined hardware designed on a multi-FPGA platform known as the TM-4 achieved an 80 times speedup compared to a 3-GHz 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 0.1 mm shift in the hardware-generated isofluence lines from those produced by MCML for fluence levels as low as 0.00001 cm^{-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.

8 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 will have 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 3-D modeling of tumors, which for realistic cases would require at least $1024 \times 1024 \times 1024$ voxels (a minimum of 4 GBytes assuming 4 bytes per voxel). Finally, the significantly larger memory space offered by external memory will enable further optimization of the number of entries in the look-up tables to improve the accuracy of the simulation. Determining the precise trade-offs 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.[30] Tailoring the FPGA-based

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.

FPGA Device	Number of Logic Elements	Number of DSP Blocks	On-chip Memory	Clock Speed
Stratix I EP1S80F1508C6 (130 nm)	64,000 out of 79,000 LUTs[a]	160 out of 176	4.8 Mbits out of 7.4 Mbits	41 MHz
Stratix III EP3SL340H1152C3 (65 nm)	44,000 out of 270,000 ALMs[a]	104 out of 1152	4.8 Mbits out of 16.7 Mbits[b]	62 MHz

[a]The types of logic elements provided by the Stratix I and Stratix III devices are different.
[b]Although it appears that 3 instances of the design can fit it the Stratix III device, in reality only 2 instances can be accommodated due to memory block size restrictions.

Table 5 Comparison of worst-case chip power consumption and energy efficiency between the TM-4 and the Intel 3.06GHz Xeon processor.

	Worst-case Power (Watts)	Delay for 100M photons (seconds)	Power-Delay Product (Joules)	Normalized Power-delay Product
Processor (single-core)	32.5	9150	29700	45
Cluster (84 cores)	2730	109	29700	45
TM-4	60	109	6540	1

hardware to the system of matrices specific to the application will be a key step in the design process.

The possible implications of this study are twofold. First, the pipelined design could form the basis on which more complex MC simulations or other light transport models can be built. The flexible pipelined architecture enables 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.[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

L. Lilge acknowledges the financial support received through the Canadian Institute for Health Research (CIHR) Grant No. 68951. Two of the authors (J. Luu and W. Lo) also acknowledge the financial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada, in the form of a CGS-M and PGS-M graduate scholarship respectively. K. Redmond acknowledges the financial support of the Ontario Graduate Scholarship. J. Rose acknowledges the financial support received through the NSERC Discovery Grant No. 171074.

References

1. B. W. Henderson and T. J. Dougherty, "How does photodynamic therapy work?" *Photochem. Photobiol.* **55**(1), 145–157 (1992).
2. T. J. Dougherty, "Photodynamic therapy," *Photochem. Photobiol.* **58**(6), 895–900 (1993).
3. S. B. Brown, E. A. Brown, and I. Walker, "The present and future role of photodynamic therapy in cancer treatment," *Lancet Oncol.* **5**(8), 497–508 (2004).
4. T. J. Dougherty, "An update on photodynamic therapy applications," *J. Clin. Laser Med. Surg.* **20**(1), 3–7 (2002).
5. M. D. Altschuler, T. C. Zhu, J. Li, and S. M. Hahn, "Optimized interstitial PDT prostate treatment planning with the Cimmino feasibility algorithm," *Med. Phys.* **32**, 3524–3536 (2005).
6. A. Johansson, J. Axelsson, S. Andersson-Engels, and J. Swartling, "Realtime light dosimetry software tools for interstitial photodynamic therapy of the human prostate," *Med. Phys.* **34**, 4309 (2007).
7. R. A. W. John Trachtenberg, S. R. H. Davidson, M. A. Haider, A. Bogaards, M. R. Gertner, A. Evans, A. Scherz, J. Savard, J. L. Chin, B. C. Wilson, and M. Elhilali, "Vascular-targeted photodynamic therapy (padoporfin, WST09) for recurrent prostate cancer after failure of external beam radiotherapy: a study of escalating light doses," *BJU Int.* **102**(5), 556–562 (2008).
8. L. K. Lee, C. Whitehurst, Q. Chen, M. L. Pantelides, F. W. Hetzel, and J. V. Moore, "Interstitial photodynamic therapy in the canine prostate," *Br. J. Urol.* **80**(6), 898–902 (1997).
9. M. Biel, "Advances in photodynamic therapy for the treatment of head and neck cancers," *Lasers Surg. Med.* **38**, 349–355 (2006).
10. I. B. Tan, P. D. Md, H. Oppelaar, and M. C. Ruevekamp, "The importance of in situ light dosimetry for photodynamic therapy of oral cavity tumors," *Head Neck* **21**, 434–441 (1999).
11. B. C. Wilson and G. Adam, "A Monte Carlo model for the absorption and flux distributions of light in tissue," *Med. Phys. (Lancaster)* **10**(6), 824–830 (1983).
12. C. M. Ma, E. Mok, A. Kapur, T. Pawlicki, D. Findley, S. Brain, K. Forster, and A. L. Boyer, "Clinical implementation of a Monte Carlo treatment planning system," *Med. Phys.* **26**, 2133–2143 (1999).
13. E. Heath, J. Seuntjens, and D. Sheikh-Bagheri, "Dosimetric evaluation of the clinical implementation of the first commercial IMRT Monte Carlo treatment planning system at 6 MV," *Med. Phys.* **31**, 2771 (2004).
14. I. J. Chetty, B. Curran, J. E. Cygler, J. J. DeMarco, G. Ezzell, B. A. Faddegon, I. Kawrakow, P. J. Keall, H. Liu, and C. M. C. Ma, "Report of the AAPM Task Group No. 105: issues associated with clinical implementation of Monte Carlo-based photon and electron external beam treatment planning," *Med. Phys.* **34**, 4818 (2007).
15. A. J. Page, S. Coyle, T. M. Keane, T. J. Naughton, C. Markham, and T. Ward, "Distributed Monte Carlo simulation of light transportation in tissue," in *Proc. 20th Int. Parallel and Distributed Processing Symp.*, IEEE, p. 4 (2006).
16. A. Colasanti, G. Guida, A. Kisslinger, R. Liuzzi, M. Quarto, P. Riccio, G. Roberti, and F. Villani, "Multiple processor version of a Monte Carlo code for photon transport in turbid media," *Comput. Phys. Commun.* **132**(1–2), 84–93 (2000).
17. L. Wang, S. L. Jacques, and L. Zheng, "MCML—Monte Carlo modeling of light transport in multi-layered tissues," *Comput. Methods Programs Biomed.* **47**(2), 131–146 (1995).
18. J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of field-programmable gate arrays," *Proc. IEEE* **81**(7), 1013–1029 (1993).
19. L. Wang, S. L. Jacques, and L. Zheng, "CONV—onvolution for responses to a finite diameter photon beam incident on multi-layered tissues," *Comput. Methods Programs Biomed.* **54**(3), 141–150 (1997).
20. L. G. Henyey and J. L. Greenstein, "Diffuse radiation in the galaxy," *Ann. Astrophys.* **3**, 117–137 (1940).
21. J. Fender, J. Rose, and D. Galloway, "The transmogrifier-4: an FPGA-based hardware development system with multi-gigabyte memory capacity and high host and memory bandwidth," in *Proc. IEEE Int. Conf. on Field-Programmable Technology*, pp. 301–302 (2005).

22. S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, McGraw-Hill, New York (2002).
23. D. E. Thomas and P. R. Moorby, *The Verilog Hardware Description Language*, Kluwer, Dordrecht, Netherlands (2002).
24. T. Grötker, *System Design with SystemC*, Kluwer, Dordrecht, Netherlands (2002).
25. V. V. Tuchin, "Light scattering study of tissues," *Phys. Usp.* **40**(5), 495–515 (1997).
26. P. L'Ecuyer, "Maximally equidistributed combined Tausworthe generators," *Mathematics of Computation*, **65**, 203–213 (1996).
27. A. Rendon, R. Weersink, and L. Lilge, "Towards conformal light delivery using tailored cylindrical diffusers: attainable light dose distributions," *Phys. Med. Biol.*, **51**, 5967–5975 (2006).
28. R. M. Stallman and F. Mass, *Free Software, Using GCC: The GNU Compiler Collection Reference Manual*, Free Sostware Foundation, GNU Press (2003).
29. M. Schweiger, S. R. Arridge, M. Hiraoka, and D. T. Delpy, "The finite element method for the propagation of light in scattering media: boundary and source conditions," *Med. Phys.*, **22**, 1779–1792 (1995).
30. A. R. Lopes and G. A. Constantinides, "A high throughput FPGA-based floating point conjugate gradient implementation," *Lect. Notes Comput. Sci.*, **4943**, 75–86 (2008).
31. A. Johansson, N. Bendsoe, K. Svanberg, S. Svanberg, and S. Andersson-Engels, "Influence of treatment-induced changes in tissue absorption on treatment volume during interstitial photodynamic therapy," *Med. Laser Appl.* **21**(4), 261–270 (2006).