International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 5, May2014)

NOC Based FPGA Acceleration for Monte Carlo Simulations with Applications to SPECT Imaging

Rohan kumar A
M. Tech, Dept. of ECE, TOCE, Bangalore

Mrs. E.R Suhasini
Asst. Prof, Dept. of ECE, TOCE, Bangalore

***Abstract*- As the number of transistors that are integrated onto a silicon die continues to increase, the compute power is becoming a commodity. This has enabled a whole host of new applications that rely on high-throughput computations. Recently, the need for faster and cost-effective applications in form-factor constrained environments has driven an interest in on-chip acceleration of algorithms based on Monte Carlo simulations. This paper presents a compute architecture for accelerating Monte Carlo simulations based on the Network-on-Chip paradigm for on-chip communication. We demonstrate through the complete implementation of a Monte Carlo-based image reconstruction algorithm for Single-Photon Emission Computed Tomography (SPECT) image that this complex problem can be accelerated by two orders of magnitude on even a modestly sized FPGA. The architecture and the methodology that we present in this paper is modular and hence it is scalable to problem instances of different sizes, with application to other domains that rely on Monte Carlo simulations.**

***Keywords*- Network-on-chip (NoC), field-programmable gate-array (FPGA), Monte Carlo (MC) simulation, SPECT Image.**

I. INTRODUCTION

SINGLE-PHOTON Emission Computed Tomography (SPECT) is a medical imaging modality used clinically in a number of diagnostic applications, including detection of cardiac pathology, various forms of cancer, and certain degenerative brain diseases. Consequently, timely and accurate reconstruction of SPECT images is of critical importance. Although computationally efficient analytical solutions to this extremely complex problem do exist, they are highly susceptible to noise and since image quality has direct implications to patient care, statistical reconstruction methods are typically favored despite their relatively long runtimes. The last decade has seen the development of numerous variance reduction techniques (VRTs) which accelerate these statistical reconstruction methods by optimizing the algorithms of the Monte Carlo (MC) simulations at their core. These have been quite successful in bringing image reconstruction time into a reasonable range for relatively small images. However, because of the underlying structure of MC simulations, we propose that by investigating solutions in parallel computing, images of higher. resolution can be reconstructed without paying a very costly time premium. Naturally, such a parallel solution could also exploit these VRTs That MC simulations are good candidates for parallelization is quite intuitive, as fundamentally they are comprised of huge numbers of independent experiments.

This application falls into a class of Monte Carlo simulations in which all the experiments share data from a common data set which is sufficiently large to prohibit complete reproduction for each processing node. Furthermore, in such experiments, the data access patterns are not known a priori. Other examples from this class include weather, environmental, and risk evaluation simulations. This paper clearly details an insightful and well-executed implementation and the results are very promising In this paper a modular approach that can be reused in other application domains to ease the design effort.

II. IMAGE RECONSTRUCTION ALGORITHM

This section presents the concepts used for image reconstruction in nuclear medical imaging and highlights the algorithmic patterns that have motivated this architectural decisions.

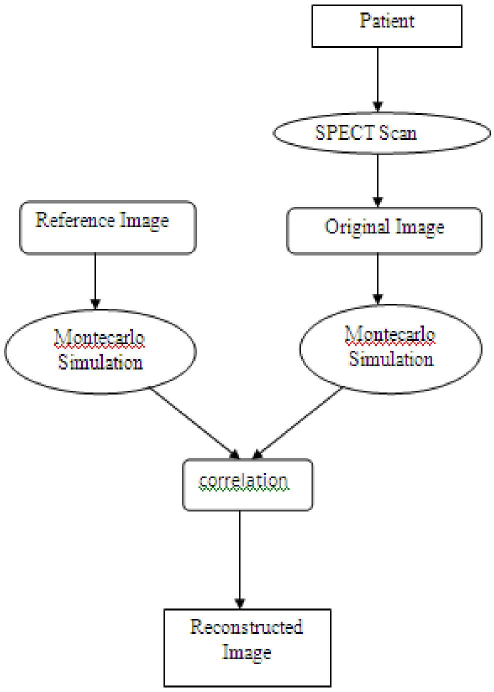


Fig 1 Simulation of Imaging

The physical basis for SPECT imaging is the detection of gamma rays emitted by decaying isotopes, which have been injected into a subject. Prior to detection, these gamma rays may undergo attenuation and a series of scatterings on their course of exit from the patient. This makes determination of the variable of interest namely the distribution of the radioactive source within the patient nontrivial.

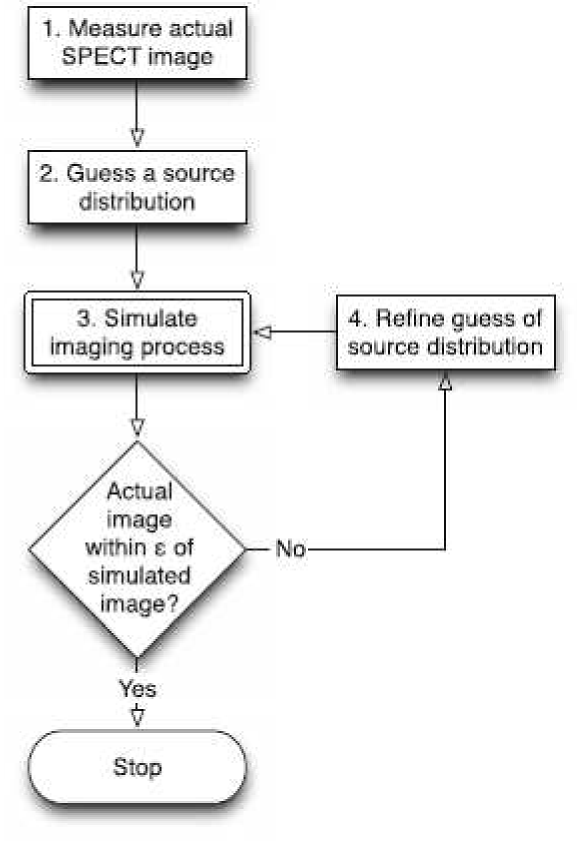

1. Measure actual SPECT image
2. Guess a source distribution
3. Simulate imaging process
4. Refine guess of source distribution
Actual image within ε of simulated image?
No
Yes
Stop

Fig 2 Iterative Reconstruction

The focus of this work is the accelerated simulation of this imaging process (see Fig. 1) using Monte Carlo methods, since this simulation is in the inner loop of a group of iterative reconstruction algorithms depicted in Fig. 2. It should be noted that the actual iterative refinement for image reconstruction is not addressed explicitly by this work. Fig. 1 depicts the simulation of the imaging process. The patient density scan is obtained with a SPECT scanner of the source distribution to simulate the imaging process. These simulated images are compared to the measured images and used to iteratively refine the approximation of the source distribution as shown in Fig. 2.

III. NOC BASED PPARALLEL ARCHITECTURE

In this section, we detail our new architecture for accelerating Monte Carlo simulations. First, we describe the architecture and then we provide implementation details for the on-chip network and processing units illustrated in Fig. 3. The most significant innovation presented in this work is the investigation into a general, scalable architecture for Monte Carlo simulations in which all the experiments share a data set which is too large to replicate for each processing unit—in this case, the density map. The natural problem which arises in this situation is how to arbitrate access between the hundreds of PUs and the single copy of the data set so that each experiment can continue without being data starved. Although it is possible to position the data set centrally and arbitrate access to it, this comes at the price of higher data latency. This can be hidden to some extent by keeping many concurrent experiments at each PU but in such a data-centric simulation, the hardware and time cost of such frequent context switching could become prohibitive. Consequently, we investigated a NoC-based architecture that enabled distributing the data set among the processing units and implementing on-chip communication resources sufficient for each experiment to relocate itself to the PU containing the data it needs. This section describes the design process that was followed, while giving concrete details of the final implementation. In many cases, experimental profiling is used to justify a design decision. It represents the network as a graph and performs cycle-accurate simulation of all network transfers with user-defined functions generating and processing the network traffic. Though there are a number of simulators available for exploration of the NoC design space, we chose to design our own because it allowed us to easily incorporate our specific application into the network simulation. This was important, first because the network traffic is difficult to model since it is heavily instance dependant and hence a much more accurate analysis of the network performance was possible by simulating the network in the context of the application than would have been using a generic traffic model. Furthermore, because the simulator integrated with our application, it was useful not only as a design exploration tool but also as an effective debugging tool. The size of this design

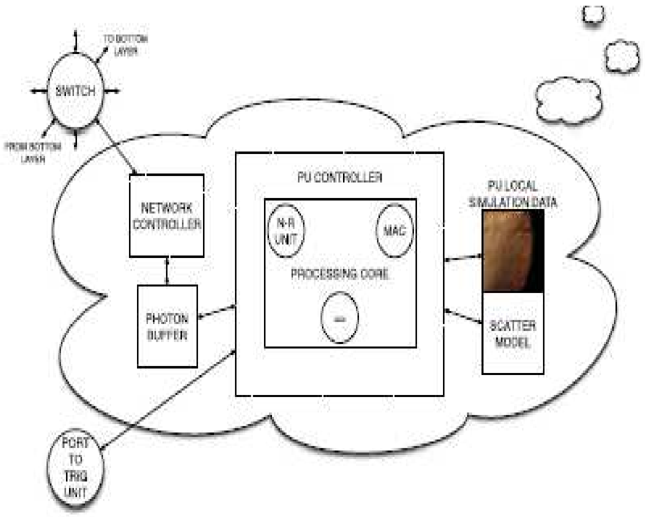


Figure 3. Top Layer of the Network

meant that it was infeasible to perform complete functional hardware simulation for any more than a few milliseconds. However, our network simulator was designed to interface seamlessly with the hardware simulation through the Direct Programming Interface offered by Verilog. Consequently, after the hardware design of the network was functionally verified, while debugging the interaction between processing units the network transfers could be offloaded to our network simulator to be executed at a higher level and hence much more quickly.

A.**s**witching and routing policies

The size of the network again had significant influence on the decision for the routing and switching strategies that were selected. The following were the criteria, in order of importance:

1. No data may be lost, i.e., no packets should be dropped,
2. The network should employ a deadlock free routing strategy,
3. The switching strategy should be simple to allow switches to be built with minimal resources, and
4. The average latency per transfer should be as small as possible.

B.**S**witch Structure

The packet switches are a registered set of input ports with a number of arbitrated paths to the output ports. Ports are 10 bits wide—8 data bits, 1 bit to indicate if a flit is valid or junk and 1 bit for flow control. In addition, there is 1 line from each input port to its feeding output port to indicate packet arrival (discussed at the end of this section). The purpose of the arbitration logic is to assign one, and only one, output port for as many input ports as possible on each clock cycle. In order to minimize the hardware cost of the arbitration unit and output ports, while still implementing the desired routing strategy, limited path switches were constructed. Fig. 6 hows the possible paths a packet can take through a switch.

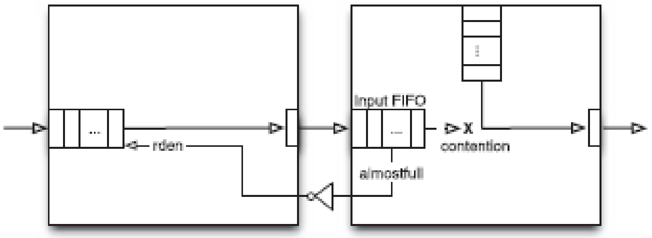


Fig. 4. Wormhole switching.

Packets from the PU can enter the network in any direction and naturally a packet arriving on any input port from the network can be directed out to the PU. As indicated above, the Z direction is routed first and as a result, only packets directly from the PU can exit the Z output port of a switch. Once the photon has reached its correct Z address, it must enter the XY plane in the correct direction. A packet may only make one right turn after entering the XY plane. For example, if a packet must move in the positive X direction and the negative Y direction, it must select the negative Y direction first. This facilitates deadlock-free routing with an extremely simple arbitration unit. Once an input port has been directed to an output port, it locks that port until the entire packet is transmitted indicated by the arrival of the footer.

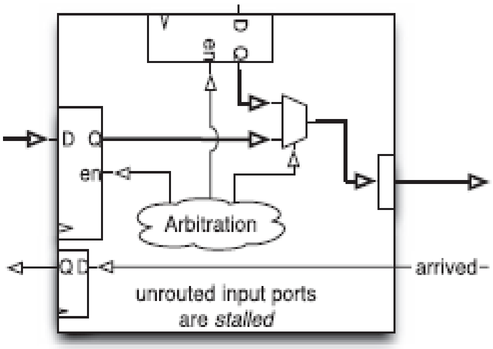


Fig. 5. Modified wormhole switching.

To demonstrate how this process occurs, part of a switch is shown in Fig. 5, with the full implementation being given in Fig. 6. If an output port is unlocked and two or more input ports compete for simultaneous access to it, the assignment is made with the following priority:

1. Straight-through traffic (S),
2. Other in-plane traffic (T),
3. Out-of-plane traffic (Z),
4. Processing unit (P).

Input ports which are not assigned are stalled until the desired output port opens. The mechanism for this stalling is very similar to the strategy employed by wormhole routing in which each input port of a switch has two or more flits of storage. If there is a contention for an output port, the lower priority packet buffers into the storage on the input port and when that buffer has only one space remaining, a stall signal is propagated back to the switch that is feeding that input port as in Fig. 4.

Obviously, with a deeper buffer, the stalled packet will occupy less space in the actual network. This is an extremely effective low-resource switching method; however, the buffer must be at least two levels deep to give the stall signal a clock cycle to propagate back to the previous switch. Unfortunately, there were insufficient resources on chip to provide two buffers for every input port, so a slightly different approach was taken (see Fig. 5). Only one level of input buffering is allocated and if a header arrives in that buffer that cannot be routed, the input port is disabled. This causes dropping of payload flits to occur. To resolve this, the packets are transmitted cyclically from the source PU. Once a packet has locked a path to the destination node, it propagates an arrival signal back to the source, which then transmits the remainder of the packet with a marker in the footer to allow the receiver to align the packet. Naturally, this cyclic transmission results in extra network transfers. Although this can waste some cycles in the source PU's network controller, network throughput is not negatively impacted since these redundant flits are in a portion of the network that is locked to any other traffic and in fact, the blocking of the network controller actually turns out to be a very effective and simple way to limit network congestion.

C. Processing Unit Structure

Based on our choice for the implementation platform, more than 40 percent of the device's logic resources were consumed by the on-chip network. This motivated a PU design that is cost effective without impacting the accuracy of the results.

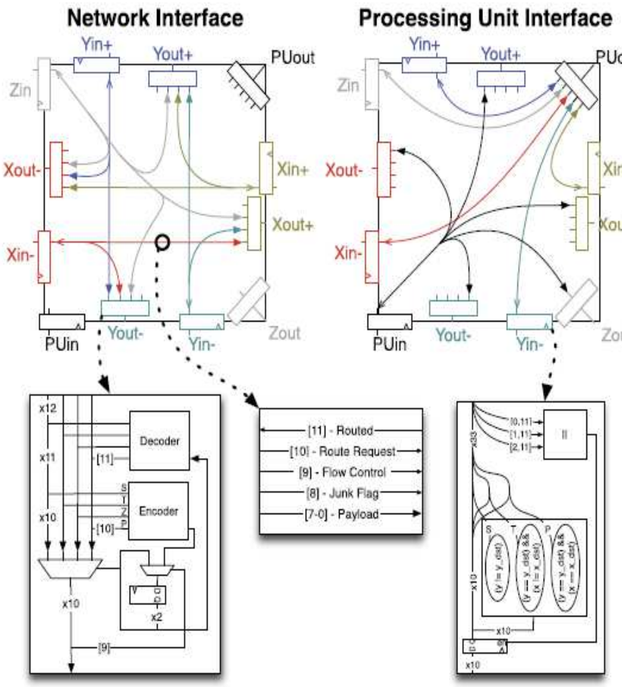


Fig. 6. Routing configuration in a switch.

The ratio of trigonometric calculations to those detailed above is relatively low, yet a trig unit is relatively expensive in terms of hardware resources. Consequently, the decision was made to share a single trigonometry unit, among an entire column of PUs (1 trig processor per 8 PUs). The trig unit is based on the well-documented CORDIC algorithm and the implementation was based heavily on the work. Arbitration is performed on a first-come-first-served basis and each CORDIC unit has one level of input buffering. Our profiling reveals that only 0.0027 percent of CORDIC commands arrive at a full CORDIC unit and need to be backed up into the network and hence we conclude that these units are not bottlenecks to computation. Each processing engine is also equipped with an linear-feedback shift register (LFSR) for pseudorandom number generation, where each unit is selectively seeded for diversity. We readily acknowledge that other methods for random number generation with better statistical properties do exist, however, our choice was motivated by the very tight resource constraints in this application and with the understanding that the focus of this work is on the communication infrastructure

required significant experimental profiling. It is critical in this application to preserve the accuracy of the reconstructed image, yet this must be balanced with the tight resource constraints. The resources required to implement a custom floating-point data path at each PU would drastically limit the number of PUs and hence the possible parallelism. Therefore, the application was mapped into fixed-point arithmetic but the evaluation of the impact of finite precision on the quality of the reconstructed image is complicated by the random nature of the simulation. To overcome this, a twofold approach was taken to establish the appropriate data widths: in the first phase, the experiments were evaluated individually to understand the relative impact on accuracy of each variable. In the second phase, the experiments were evaluated in the context of the simulation.

The signal-to noise ratio (SNR) of the fixed-point images taking the floating-point images as reference is insufficient to draw strong conclusions about the accuracy because of the simulation randomness and because the images are so heavily dependent on the simulation data set. Consequently, the convergence of the images with increasing simulation size was used to evaluate the simulation accuracy .

IV. RESULTS

This paper has the detailed implementation of hardware for the simulation of SPECT imaging based on the Monte Carlo simulation.

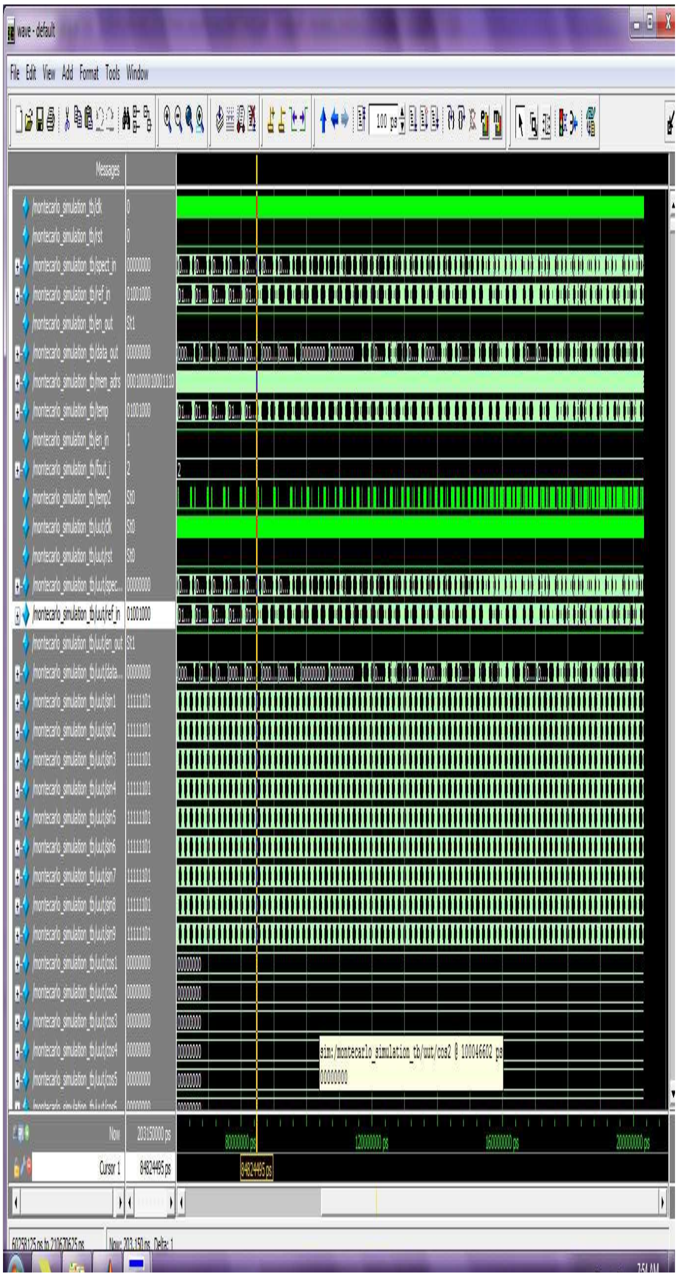
V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a NOC based parallel architecture, and its FPGA implementation, for accelerating Monte Carlo simulations that share common data sets. The case study on SPECT imaging has shown that significant speedups can be achieved over single core implementations, without compromising the image reconstruction accuracy. The proposed architecture, because it is based on an NoC approach, is modular and hence it can easily be expanded to support FPGAs of larger capacity.

VI. RESULT DISPLAY

The Proposed system consists modules: Arbiter, cordic processor, SPECT Image and Montecarlo simulation .

These modules are designed by VERILOG and synthesized by using Xilinx ISE 12.2 version tool. The functionality of the design is verified by simulating the code by using the Modelsim tool and the waveforms are obtained.


wave - default
File Edit View Add Format Tools Window
100 ps
sim:/montecarlo_simulation_tb/uut/cos2 @ 100046602 ps
00000000
80000000 ps
120000000 ps
160000000 ps
200000000 ps
84824495 ps

Messages	
/montecarlo_simulation_tb/clk	0
/montecarlo_simulation_tb/rst	0
/montecarlo_simulation_tb/spect_in	00000000
/montecarlo_simulation_tb/ref_in	01001000
/montecarlo_simulation_tb/en_out	St1
/montecarlo_simulation_tb/data_out	00000000
/montecarlo_simulation_tb/mem_adrs	0001000010001110
/montecarlo_simulation_tb/temp	01001000
/montecarlo_simulation_tb/en_in	1
/montecarlo_simulation_tb/fout_i	2
/montecarlo_simulation_tb/temp2	St0
/montecarlo_simulation_tb/uut/clk	St0
/montecarlo_simulation_tb/uut/rst	St0
/montecarlo_simulation_tb/uut/spec...	00000000
/montecarlo_simulation_tb/uut/ref_in	01001000
/montecarlo_simulation_tb/uut/en_out	St1
/montecarlo_simulation_tb/uut/data...	00000000
/montecarlo_simulation_tb/uut/sin1	11111101
/montecarlo_simulation_tb/uut/sin2	11111101
/montecarlo_simulation_tb/uut/sin3	11111101
/montecarlo_simulation_tb/uut/sin4	11111101
/montecarlo_simulation_tb/uut/sin5	11111101
/montecarlo_simulation_tb/uut/sin6	11111101
/montecarlo_simulation_tb/uut/sin7	11111101
/montecarlo_simulation_tb/uut/sin8	11111101
/montecarlo_simulation_tb/uut/sin9	11111101
/montecarlo_simulation_tb/uut/cos1	00000000
/montecarlo_simulation_tb/uut/cos2	00000000
/montecarlo_simulation_tb/uut/cos3	00000000
/montecarlo_simulation_tb/uut/cos4	00000000
/montecarlo_simulation_tb/uut/cos5	00000000
/montecarlo_simulation_tb/uut/cos6	00000000
Now	203150000 ps
Cursor 1	84824495 ps

Figure 7. simulation waveform montecarlo

.

ACKNOWLEDGEMENT

I would like to thank my guide E.R suhasini of The Oxford College of Engineering Bangalore for supporting the study.

REFERENCES

[1]. F. Angarita, A. Perez-Pascual, T. Sansaloni, and J. Vails, “Efficient FPGA Implementation of Cordic Algorithm for Circular and Linear Coordinates,” Proc. Int’l Conf. Field Programmable Logic and Applications, pp. 535-538, Aug. 2005.

[2]. S. Aslan, E. Oruklu, and J. Saniie, “Realization of Area Efficient QR Factorization Using Unified Division, Square Root, and Inverse Square Root Hardware,” Proc. IEEE Int’l Conf. Electro/ Information Technology (EIT ’09), pp. 245-250, June 2009.

[3]. A. Badal and A. Badano, “Monte Carlo Simulation of X-Ray Imaging Using a Graphics Processing Unit,” Proc. IEEE Nuclear Science Symp. Conf. Record (NSS/MIC), pp. 4081-4084, Oct. 2009.

[4]. F.J. Beekman, H.W.A.M. de Jong, and S. van Geloven, “Efficient Fully 3D Iterative SPECT Reconstruction with Monte Carlo-Based Scatter Compensation,” IEEE Trans. Medical Imaging, vol. 21, no. 8, pp. 867-877, Aug. 2002

[5]. C. Chang, J. Wawrzynek, and R.W. Brodersen, “BEE2: A High-End Reconfigurable Computing System,” IEEE Design Test of Computers, vol. 22, no. 2, pp. 114-125, Mar. 2005.

[6]. G. de Michelis and L. Benini, Networks on Chips: Technology and Tools. Morgan Kaufmann, 2006

[7]. T.C. de Wit, J. Xiao, and F.J. Beekman, “Monte Carlo-Based Statistical SPECT Reconstruction: Influence of Number of Photon Tracks,” IEEE Trans. Nuclear Science, vol. 52, no. 5, pp. 1365-1369, Oct. 2005.

[8]. Y.K. Dewaraja, M. Ljungberg, A. Majumdar, A. Bose, and K.F. Koral, “A Parallel Monte Carlo Code for Planar and SPECT Imaging: Implementation, Verification and Applications in 131-I SPECT,” Proc. IEEE Nuclear Science Symp. Conf. Record, vol. 3, pp. 20/30-20/34, 2000.

[9]. V. Fanti, R. Marzeddu, C. Pili, P. Randaccio, S. Siddhanta, J. Spiga, and A. Szostak, “Dose Calculation for Radiotherapy Treatment Planning Using Monte Carlo Methods on FPGA Based Hardware,” Proc. 16th IEEE-NPSS Real Time Conf. (RT ’09), pp. 415-419, May 2009.

[10]. [10] M. Gebremichael, W.F. Krajewski, M. Morrissey, D. Langerud, G.J. Huffman, and R. Adler, “Error Uncertainty Analysis of GPCP Monthly Rainfall Products: A Data-Based Simulation Study,” J. Applied Meteorology, vol. 12, pp. 1837-1848, 2003.