Fast Radio Wave Propagation Prediction by Heterogeneous Parallel Architectures with Incoherent Memory

Florian Schröder, Michael Reyer, Rudolf Mathar
Institute for Theoretical Information Technology
RWTH Aachen University
D-52074 Aachen, Germany
Email: {schroeder, reyer, mathar}@ti.rwth-aachen.de

***Abstract*—The present correspondence deals with radio wave propagation for urban scenarios on the cell broadband engine. Binary space partitioning trees are used to split the building data into manageable size for the parallel units of the cell. By choosing the size of the tree leafs both data transfer between units and runtime of the algorithm is significantly improved. The implementation of these techniques allowing for reflection demonstrates promising results for speeding up radio wave propagation.**

I. Introduction

Fast radio wave propagation plays an essential role in planning, analysis, and optimization of radio networks. A huge variety of prediction scenarios have to be evaluated. Consequently the predictions need to be fast. For the design of fast algorithms it has to be taken into account that the performance improvement of processors has shifted from higher clock speed to more cores. The paradigms of programming need to consider this hardware development resulting in many heterogeneous cores for general purpose calculations, e.g., multi core CPU, high performance graphics hardware (GPU), and so forth.

An overview of radio wave propagation models is given in [1] and [2]. Models proposed in the literature can basically be divided into (semi) empirical and ray optical models. Semi empirical models calculate the received power on the basis of frequency, distance, and an empirical part mainly describing the obstacle influence. The strength of such approaches is the speed of prediction. However, the prediction quality is low if the influence of deflection effects like diffraction, reflection, and transmission is high. This leads to ray optical approaches which identify ray paths through the scene to combat the lack of prediction quality at the cost of higher computation times.

In ray optical models the environment, e.g., buildings, is usually described by polyhedrons, formed of surface sections, called *facets* in the following. Several ray paths between the transmitter and receiver point are searched, regarding deflection effects as reflection on, transmission through, and diffraction at edges of the given facets. Ray optical models are classified as ray tracing and ray launching, depending on the way the ray paths are determined.

In ray tracing models all possible ray paths starting from a receiver point to the transmitter are searched. The set of possible ray paths is limited by a maximum number of deflection points, i.e., points where deflection effects occur. For each receiver point the possible ray paths have to be recalculated, as there might be complete different ray paths. This leads to multiple calculation of nearly identical ray path pieces, particularly, if receiver points are nearby located. Therefore, in [3] an extensive preprocessing is proposed which computes visibility of facets in advance. Hereby faster predictions are achieved.

Ray launching methods emit a finite set of rays from the transmitter in predetermined directions, cf. [4] and [5]. If rays hit a facet, possible deflection effects are performed. A receiver point is hit if the ray path crosses its proximity. As the rays disperse, important deflection points or even receiver points may not be hit. Alternatively, in [6] 3D cones are used instead of single rays. Beyond this work, mixed models have been investigated which follow partly rays and partly use empirical parameters, cf. [7]. Additional work on prediction algorithms, which is based on ray optical approaches, can be found for example in [8].

The high potential of parallel architectures is well known, especially for graphics cards it is discussed in [9], [10]. Using graphics cards for non graphical purposes is called GPGPU (General Purpose computations on Graphics Processing Unit), see [11]. There are numerous applications, e.g., physical simulation in [9] and sorting in [10]. With the introduction of the Playstation 3, for short PS3, in March 2007 the cell is available for research at low cost. The high potential of this architecture is described in [12].

Radio wave propagation has been applied on parallel architectures. Recent results for graphics cards may be found in [13], [14] and for the cell in [15].

The paper is organized as follows. We start with a description of radio wave propagation in Section II. After a brief introduction to parallel architectures in Section III, important principles of the implementation are presented in Section IV. This includes binary space partitioning of the data and a smart method of calculating intersections. In Section V results are

presented. Finally, we conclude this paper in Section VI.

II. Radio Wave Propagation

In this paper we use CORLA (Cube Oriented Ray Launching Algorithm) from [16] for radio wave propagation. The urban environment is described by a simple representation of three dimensional objects, particularly, buildings as polygons with one height, i.e., roof styles are neglected, see Figure 1. The model for the field strength prediction considers 1. line-

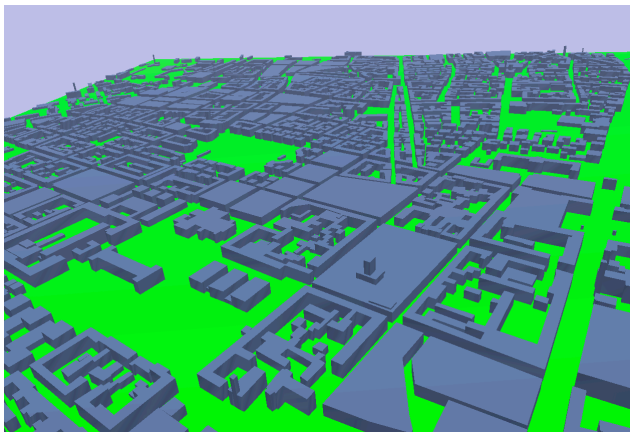

Fig. 1. Example of building representation in Munich, [1]

of-sight, 2. reflection, 3. horizontal diffraction, and 4. vertical diffraction as depicted in Figure 2. As building part of the

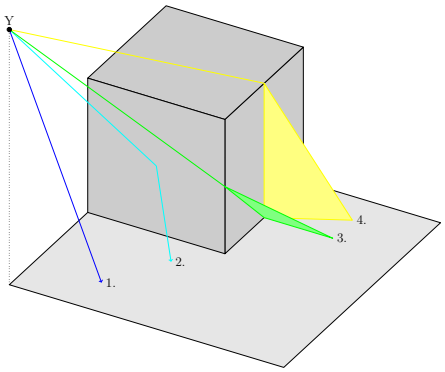

Fig. 2. Overview of ray optical effects

path attenuation we use a modification of the well known free space propagation formula. Using the overall distance $d(p)$ of the path p, neglecting the antenna gains – this may be easily considered in a post processing –, adapting the path loss exponent γ according to the environment, and introducing an estimator z_A mitigating imprecise information about transmit powers leads to

$$L_0^{dB}(p) = 20\lg(4\pi/\lambda) + z_A + 10\gamma \lg d(p)\,,$$

where λ denotes the wavelength. The impact of the effects is modeled by polynomial terms of a low degree with the change of angle of each effect as input. This results in the overall attenuation of a path p

$$L^{dB}(p) = L_0^{dB}(p) + \sum_{i=1}^{n_R(p)} L_R^{dB}(\alpha_{R,i}(p)) \quad (1)$$

$$+ \sum_{i=1}^{n_V(p)} L_V^{dB}(\alpha_{V,i}(p)) \quad (2)$$

$$+ \sum_{i=1}^{n_H(p)} \sum_{j=0}^{k} z_{H,j}\alpha_{H,i}^{j}(p)\,, \quad (3)$$

where $n_X(p)$ is the counter of the effect $X \in \{R, V, H\}$ and $\alpha_{X,i}(p)$ are the changes of angle of the i-th occurrence of the effect. Note, the functions L_X^{dB} given in (1) and (2) have the same structure as shown in (3). If multiple paths arrive at the same receiver point r, we do not add the paths but take the strongest path which is a reasonable approximation, see [8].

III. Parallel Architectures

Parallel architectures are widely spread nowadays. For a long time processing power was improved by increasing clock speed. But due to limitations in cooling and power consumption this is no longer applicable. Consequently, for further improvements processors move towards multi core layouts. Another trend is using specialized computational power for general purpose task. The most popular example is the GPU.

Generally speaking, systems with heterogeneous processor architectures and incoherent memory will be found more often and gain in importance.

A. Heterogeneous Architectures

For heterogeneous architectures the task division is one of the challenging tasks. The CPU, a homogeneous architecture, is designed for all kinds of tasks, it is very flexible, and built for fast calculation and branching, while most other processing units are usually specialized for certain task. Regarding these tasks they are way faster than the CPU itself but on the price of being slow on other task or even incapable of executing them. On a homogeneous system your main concern is to supply each unit with the same work load, because the overall runtime is given by the slowest – in terms of runtime – unit. However, on heterogeneous systems it is also important to designate tasks with respect to the specialties of the hardware.

B. Incoherent Memory

Many vendors make a great effort in offering systems with coherent memory. In such systems the hardware assures that all caches are at sync. Hence, if two units read a value from the memory, store it in their local caches, and one unit changes its value, without a coherency mechanism the second unit would work with an outdated value with undefined consequences. It is most likely that heterogeneous architectures have no coherent memory such that the software needs to take over the validity check of data from hardware. This has to be carefully considered when designing the software.

C. Cell Broadband Engine Architecture

We use the the Cell Broadband Engine Architecture, for short cell or CBEA. It is designed amongst others to speed up memory processing and defying the memory wall, cf. [17], while keeping power consumption at low level as described in [18]. The cell processor is a chip with ten cores. These are a dual-core 64-bit IBM PowerPC for organizing tasks and running the operating system and eight additional cores called Synergistic Processing Elements, short SPE. It is a heterogeneous system on one chip with incoherent memory, the main memory accessible by all cores and the 256kB local memory of each SPE. This local memory is located between the main memory and the processor caches. It is very fast but of limited size compared to todays main memory. The SPEs use their own instruction set architecture for memory transfer and many optimized calculation operations. Its 128 registers are 128-bit wide supporting vector instructions – applying operations not on single data but on whole vectors in one step (SIMD - single instruction multiple data).

Our choice fell on this new type of processor, because it comprises the previous described attributes and has a high computational power on a low cost level. In order to exploit the processor potential the algorithms need to be parallelized and data structures must be designed to handle multiple access and keep their validity when handled by several processors. Particularly, the memory handling is very challenging due to the local SPE memories.

IV. Data structures and algorithms

The data structure of a program has a great impact on its performance. The importance of good data structure design gets more pronounced as the gain on speed for processors is much higher than for memory. This phenomenon, when a processor is burning cycles waiting for data to arrive, is called memory wall, see [17]. A careful designed data structure will reduce this effect. Good data structures should fulfill the following constraints: i) Partition the input data due to hardware constraints, like transfer block sizes etc. ii) Put the data in some order beneficial to the algorithms, e.g., data needed at consecutive steps should be grouped. iii) Enable fast access to subsets of data.

Applying those constraints on radio wave propagation for urban scenarios on a cell leads to binary space partitioning (BSP) trees with ropes. With BSP a set of data will be repeatedly divided into two subsets and saved into a tree structure. Each node of the tree represents a partitioning and each leaf denotes a subset of the final partition. In the following leaf is a synonym for subset. The leafs contain a set of facets and therefore the leaf size represents the number of facets within the leaf. Figure 3 shows a 2D representation of a tree, where each rectangle denotes a leaf and each line – excluding the outer frame lines – depicts the nodes. For example the vertical line roughly in the middle, splitting the image into two, is the root node followed by two horizontal lines splitting these halves again and so forth. Since the tree has no information about the neighborhood

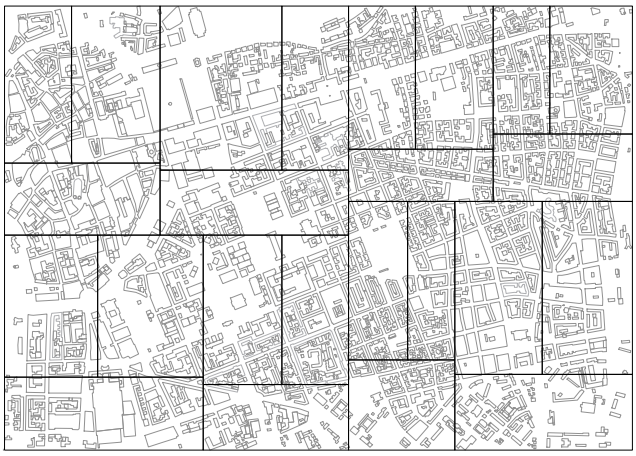

Fig. 3. Graphical representation of a BSP tree

of leafs, an additional structure is set on top of the tree, ropes. Ropes add to each leaf the information about their neighbors such that quick navigation among leafs is possible and constraint iii) is fulfilled. While constructing the BSP tree, it is ensured that the leafs need approximative the same size in memory. This size is set to a multiple of the optimal memory block transfer size for the hardware bus system of the PS3. It fits optimal into the limited local memory of the SPEs and it allows static memory allocation which is much more convenient then dynamic memory allocation on SPEs – not using dynamic allocation reduces memory management problems. Consequently constraint i) is fulfilled.

For explanation of constraint ii), the benefit to the algorithm, we need to explain the algorithm first. One of the most important parts of radio wave propagation is to determine which ray intersects with which facet. With a BSP tree it is very efficient to determine a subset (leaf) belonging to some coordinate or point; the complexity is at maximum the depth of the tree in binary decisions which is logarithmic for balanced trees. BSP trees are by design close to balance, i.e., there is very small variance of the leaf depth. For our purposes this needs to be generalized to the determination of all subsets lying on a path from transmitter to receiver. This can easily be performed by using the ropes providing the beneficial ordering of the data mentioned in constraint ii).

Hence, we have excluded a lot of subsets and all of its facets while determining the intersections. A detailed description of the algorithm to create such BSP trees can be found in [15].

For evaluation of the attenuation at receiver points paths to those points need to be calculated. Therefore, a sufficiently large amount of rays is launched from the transmitter into all directions. For each ray it is calculated, if the ray hits a i) receiver point, ii) a facet within the subset, or iii) the border of the subset. So either the attenuation is calculated or a ray optical effect has to be processed or the ray is followed in the neighboring subset.

As the calculation of the intersection between rays and

facets is an often repeated core component of this algorithm, its runtime should be minimized. Obviously the size of the leafs in the data structure may be chosen to optimize the number of candidates on which the intersection test is executed. A second measure is to carefully chose the number of processed rays. So the goal is to minimize the amount of rays while providing a certain level of accuracy for the resulting prediction. And thirdly, the number of calculations may be reduced by exploiting redundancy. In our case it is most likely that rays just differing in the vertical angle either all will hit a facet or pass over it. Meaning, in a full 3D calculation a bunch of rays will intersect the facet at same (similar) x and y coordinates but differing in the z coordinate, where z represents the height. By splitting up this 3D calculation into two 2D calculation, as suggested in [19], much of the redundancy is taken away. In Figure 4 the first phase is depicted where vertical overlaying rays are put together. The

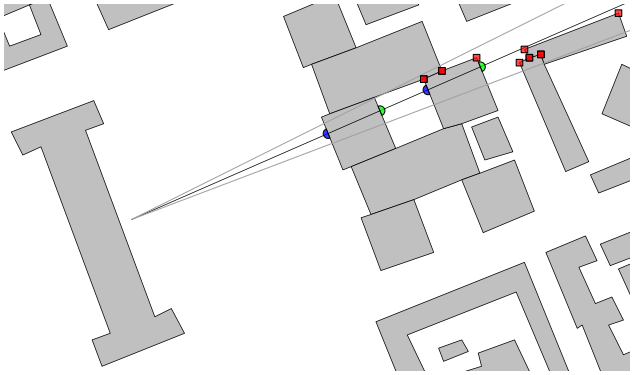

Fig. 4. First phase: determining intersections in xy-plane

green and blue half circles mark the position where a ray hits or drops out of a building. The red squares mark edges in a close cone around the ray, indicating candidates for horizontal diffraction. In the second phase the vertical overlaying rays are separated again to process the different effects. Figure 5 depicts the vertical diffraction in the second phase, where the numbers of vertical diffractions a path undergoes on its way to the receiving plane is given. So naturally a recursive

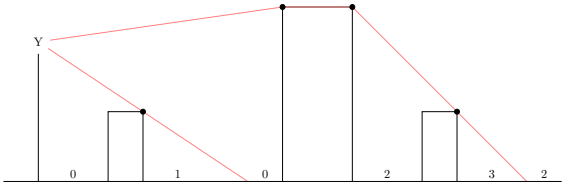

Fig. 5. Second phase: vertical diffraction

process is defined in which nodes can be discarded whenever all children are processed. The children are independent and therefore allow for easy parallelization.

V. Results

As we have seen in the last section, the leaf size of a BSP tree has strong impact on the runtime and is therefore be studied. Recall, the smaller the leaf size the fewer intersection calculations between rays and facets have to be performed per leaf, but at the cost of a higher complexity which is explained in the following. With a lower number of facets per leaf the number of leafs increases and therefore more ray transfers between leafs have to be executed. Additionally, the leafs cover a smaller area and consequently the probability that a facet crosses a leaf border grows. In such cases these facets have to be split up and stored in each leaf which increases the total amount of stored facets. In summary, the lower the leaf size the more memory is used, the more management in terms of transfers is needed and the more redundant calculations due to the doubling of cut facets have to be executed. Thus, it is to be expected that runtime will decrease with a decreasing leaf size, but that it will increase again for very low leaf sizes. Furthermore, it is to be expected that the number of leafs times the leaf size will not be constant – as it is in the optimal case – but increase with decreasing leaf size due to the doubling of facets.

The above observations can be clearly seen in Figure 6 comparing the solid red line representing the realized number of leafs and the dashed blue line displaying its lower bound.The solid red line is well above the blue one because on one hand the leaf size is a maximal and not an average value and on the other hand the number of leafs needs to be a natural number. The runtime is relative to the runtime with leaf size 2500 in which the number of leafs size is slightly greater than the number of SPEs; beyond this level side effects distort the measurement. Most interestingly, for big

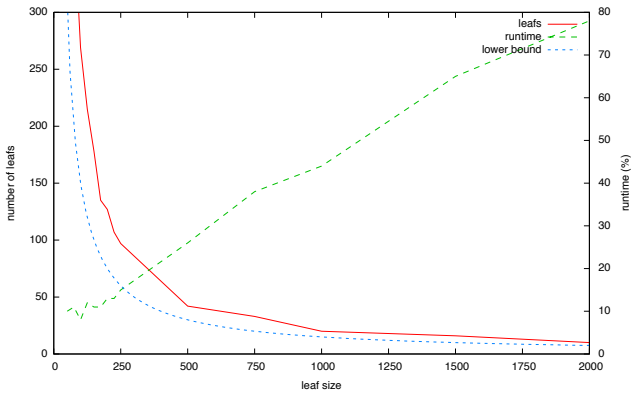

Fig. 6. Runtime and number of leafs over the leaf size

leaf sizes the relation to the runtime is approximative linear. For small leaf sizes the runtime is not monotonous. This effect is mainly motivated by the algorithm used for the BSP tree generation. To find the optimal partition is hard as the amount of facets within each subtree cannot be precisely determined with the information of the current set of facets because of the duplication of facets which cut a border line. Instead a greedy algorithm is used which in principle works as follows. Starting with all facets at the root the greedy algorithm divides the current set of facets generating two new children for the current node. This procedure is repeated until the desired leaf

size is reached. So a suboptimal solution with a relatively high degree of variation in the leaf size is attained. It is a topic for further research to optimize the generation of BSP trees which might be worthwhile taking into consideration that the BSP tree generation needs to be executed once for a given set of building data, whereas the radio wave propagation will be run quite often to get predictions for all base stations and their configurations within this area. Note, that those curves are strongly dependent on the evaluated scenario and therefore do not provide optimal values.

Ray generation has also an impact on runtime. While developing strategies for ray generation the trade off between prediction coverage and accuracy as well as speed needs to be considered. The prototype uses a uniform distribution of rays.

The receiver points are evaluated according to the path loss model of Section II. In Figure 7 an image representation of

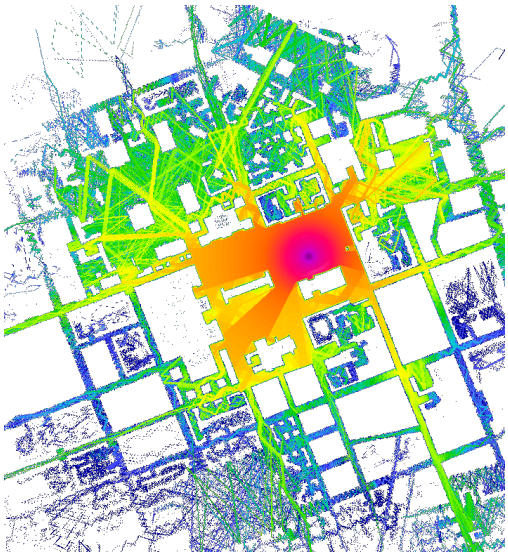

Fig. 7. Prediction with reflection in Munich, [1]

the result is shown. The receiver points are equally distributed on that area, i.e., each pixel represents a receiver point. Furthermore, only the reflection effect is activated.

The current implementation is comprised of a full 3D approach of ray evaluation and of a uniform distribution of rays. This results in a runtime of about 40 seconds for the Munich scenario which sounds slow at a first glance. However, the runtime per followed ray is comparable between cell and CPU implementations. Taking into account that more complex rays are followed on the cell, it is quite promising and should be significantly faster, when replacing the full 3D approach with the two times 2D version and introducing a smart ray generation and duplication. Note that both features are included in the CPU implementation already.

VI. Conclusion

In this work a promising basis for speeding up radio wave propagation by parallel architectures is presented. The use of binary space partitioning has shown two major advantages. Firstly, it enables for providing small independent data sets which can be processed in parallel. And secondly, the runtime of the algorithm can be sped up significantly as this partitioning also allows for a preselection of relevant data to be processed. First results show that the application on the cell broadband architecture will be significantly faster than a classical CPU implementation. This goal will be achieved after replacing the 3D ray launching approach by a two times 2D approach and a smart ray generation and duplication.

Acknowledgments

This research was partly supported by the UMIC excellence cluster of RWTH Aachen University

References

[1] E. Damosso, Ed., *COST Action 231: Digital mobile radio towards future generation systems, Final Report*. Luxembourg: Office for Official Publications of the European Communities, 1999.

[2] N. Geng and W. Wiesbeck, *Planungsmethoden für die Mobilkommunikation*. Springer, 1998.

[3] G. Wölfle, R. Hoppe, and F. Landstorfer, "A fast and enhanced ray optical propagation model for indoor and urban scenarios, based on an intelligent preprocessing of the database," in *Proceedings PIMRC*, Osaka, Japan, 1999.

[4] G. Durgin, N. Patwari, and T. S. Rappaport, "An advanced 3D ray launching method for wireless propagation prediction," in *Proceedings IEEE VTC Spring*, Phoenix, AZ, 1997, pp. 785 – 789.

[5] M. Schmeink and R. Mathar, "Preprocessed indirect 3D-ray launching for urban microcell field strength prediction," in *Proceedings IEEE AP*, Davos, Switzerland, 2000.

[6] T. Frach, "Adaptives hierarchisches Ray Tracing Verfahren zur parallelen Berechnung der Wellenausbreitung in Funknetzen," Ph.D. dissertation, RWTH Aachen University, 2003.

[7] J. Beyer, "Ausbreitungsmodelle und rechenzeiteffiziente Methoden für die Feldstärkeprognose in städtischen Mikrozellen," Ph.D. dissertation, Universität-Gesamthochschule Siegen, 1997.

[8] R. Wahl, G. Wölfle, P. Wertz, P. Wildbolz, and F. Landstorfer, "Dominant path prediction model for urban scenarios," *14th IST Mobile and Wireless Communications Summit, Dresden (Germany)*, 2005.

[9] M. Harris, *GPU Gems*. Addison-Wesley, 2004.

[10] P. Kipfer and R. Westermann, *GPU Gems 2*. Addison-Wesley, 2005.

[11] "GPGPU." [Online]. Available: http://www.gpgpu.org

[12] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick, "The potential of the cell processor for scientific computing," in *CF '06: Proceedings of the 3rd conference on Computing frontiers*. New York, NY, USA: ACM, 2006, pp. 9–20.

[13] M. Reyer, T. Rick, and R. Mathar, "Graphics hardware accelerated field strength prediction for rural and urban environments," in *Proceedings: European Conference on Antennas and Propagation (EuCAP)*, Edinburgh, Scotland, UK, November 2007, pp. 1–5.

[14] A. Schmitz, T. Rick, T. Karolski, L. Kobbelt, and T. Kuhlen, "Simulation of radio wave propagation by beam tracing," in *Eurographics Symposium on Parallel Graphics and Visualization*, 2009.

[15] F. Schröder, "Konzepte zur effizienten Umsetzung von Algorithmen für die IBM Cell-Architektur - Anwendung auf Verfahren der Feldstärkeprädiktion im Mobilfunk," Master's thesis, RWTH Aachen, 2008.

[16] R. Mathar, M. Reyer, and M. Schmeink, "A cube oriented ray launching algorithm for 3D urban field strength prediction," in *Proc. IEEE International Conference on Communications ICC '07*, 2007, pp. 5034–5039.

[17] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, 1995.

[18] H. Hofstee, "Power efficient processor architecture and the cell processor," in *Proc. HPCA-11 High-Performance Computer Architecture 11th International Symposium on*, 2005, pp. 258–262.

[19] J.-P. Rossi and Y. Gabillet, "A mixed ray launching/tracing method for full 3-D UHF propagation modeling and comparison with wideband measurements," *Antennas and Propagation, IEEE Transactions on*, vol. 50, no. 4, pp. 517–523, Apr 2002.