The Computational Performance and Power Consumption of the Parallel FDTD on a Smartphone Platform

Robert G. Ilgner and David B. Davidson

Department of Electrical and Electronic Engineering
University of Stellenbosch, Matieland, Private Bag X1, Western Cape, South Africa
bobilgner@gmail.com, davidson@sun.ac.za

Abstract ─ The use of the FDTD in Android applications heralds the use of mobile phone platforms for performing electromagnetic modeling tasks. The Samsung S4 and Alpha smartphones computations are powered by a pair of multi-core Advanced RISC Machines (ARM) processors, supported by the Android operating system, which comprises a self-contained platform, which can be exploited for numerical simulation applications. In this paper, the parallelized two dimensional FDTD is implemented on the Samsung Smartphone using threading and SIMD techniques. The computational efficiency and power consumption of the parallelized FDTD on this platform are compared to that for other systems, such as Intel's i5 processor, and Nvidia's GTX 480 GPU. A comparison is made of the power consumption of the different techniques that can be used to parallelize the FDTD on a conventional multicore processor. In addition to parallelizing the FDTD using threading, the feasibility of accelerating the FDTD with the SIMD registers inherent in the phone's ARM processor is also examined.

Index Terms ─ ARM, EXYNOS, FDTD, NEON.

I. INTRODUCTION

The viability of using Smartphones for High Performance Computing (HPC) is currently being examined for a variety of scientific disciplines [1], [2]. The Smartphone is commonly described as being a low power processing device and the work described in this article will quantify this assertion in the context of electromagnetic modeling using the FDTD method.

The finite difference time domain technique has been used for the modelling of electromagnetic scenarios since the late 1960s and is a popular choice for parallelization in HPC owing to its simple deployment on a wide variety of hardware architectures [3].

Power consumption of processors is a topic of interest owing to the cost implications of supplying electricity to HPC processing centers [4]. Koomey's law [5] states that the power being consumed by modern processors is halved every 18 months for an equivalent computational output. Here, the two dimensional parallel FDTD implemented on the Samsung S4's ARM based platform will be used to gauge how the Samsung S4 compares to the power consumption of the FDTD on other HPC platforms. The power consumption of different parallelization techniques will be compared to a contemporary multi-core processor, Intel's i5-4440k.

This paper is structured into three main sections. The first section describes the parallelization of the two dimensional finite differences time domain (FDTD) method on the Samsung S4 and Alpha smartphones using generic threading. The second section presents measurements of computational performance and power consumption of the parallel FDTD on the Samsung S4. Comparison is made with different implementations of the parallel FDTD on an Intel i5 multi core processor. In the third section, the computational/power performance ratio of the parallel FDTD is also compared to the parallel FDTD implemented on other contemporary computing platforms. Several practical real world scenarios that can be computed on the Samsung S4 and Alpha are given as examples, including computing the RFI screening of a large soil berm; results are compared with computations on the Intel i5-4440k processor.

The main contributions of this paper are firstly, a description of parallelizing the FDTD algorithm onto an Android platform, and secondly, an evaluation of the power efficiency of the Android based parallel FDTD implementation compared to the parallel FDTD implemented on a number of other more conventional computing-platforms.

II. PROCESSING THE FDTD ON THE SAMSUNG S4

A. Maxwell's equations and the parallel FDTD method

The FDTD is a numerical approximation to Maxwell's equations [6] and is based on the finite differences formulation of the scalar computations as originally proposed by Yee [7]. For most of the work in this article the peripheral boundaries of the computational

Submitted On: July 2, 2015
Accepted On: November 3, 2015

space are defined using the split field perfectly matched layers (PML). A computational stencil, incorporating some values of adjacent cells, is applied to every field value in the calculation space and repeated in an iterative leap-frog manner [6], [7]. The implications of this stencil processing are two-fold:

i) The vector processing of data has to accommodate non-consecutive array reads [8] for two and three dimensional data.

ii) Threads must communicate fringe data between adjacent data chunks as the values contributing to the computational stencil may be contributed by different data chunks. This inter-chunk communication results in the parallelization of the FDTD being more sophisticated than for an embarrassingly parallel algorithm.

Parallelisation technologies such as MPI, OpenMP and generic threading are generally used to speed up the FDTD in parallel form on multi-core microprocessors [9], [10]. The openMP and generic threading method for the parallelization of the FDTD requires a shared memory architecture, such as a multi-core microprocessor, as the representative sections of the FDTD code are divided amongst several threads and these are each computed on dedicated cores. The FDTD was parallelised for the ARM processors using generic threading, as neither the openMP or the MPI facilities were readily available for the Android platform.

Results for the acceleration of the FDTD on the GPU were attained using the techniques described by [11]. Further acceleration of the parallel FDTD for the Intel processors using SIMD engines such as the SSE and AVX were implemented according to the descriptions by [12].

B. Hardware architecture of the ARM platform

The ARM platform used for the processing of the results shown in this article is part of Exynos System [13] on a Chip (SoC) processors. The Exynos 5410 or Exynos 5430 processor provides the processing power of the Samsung S4 phone. The Exynos 5410 and the 5430 SoCs shown in Figs. 1 and 2, both have an embedded ARM 7 and an ARM 15 processor, a GPU, and embedded storage, albeit in different configurations. The Exynos 5410 resident ARM cores are configured such that low power tasks are processed on the cores of the ARM 7 processor and more computationally intensive tasks are processed on the ARM 15 cores. Although there are eight cores available in total, the tasks are managed between corresponding ARM 7 and ARM 15 cores, as shown in Fig. 1, in a power saving strategy that ARM refers to as a "big.Little" configuration. The switching of the tasks between the ARM 7 and ARM 15 cores is controlled by the firmware and is not available to the programmer.

The Exynos 5430 SoC on the other hand, uses a more conventional arrangement of the ARM 7 and the ARM 15 cores, where the scheduler has direct access to both the ARM 15 and ARM 7 cores. This configuration will allow the processing of eight tasks simultaneously, but the ARM 7 processor cores will run at a lower operating frequency than the ARM 15 cores. Although the availability of the larger number of cores is appealing at first impression, one has to note that the speed of processing of equal sized computational loads will be limited by the capacity of the slowest core. It is not programmatically trivial to assign computational threads to specific cores in the Exynos 5430 SoC.

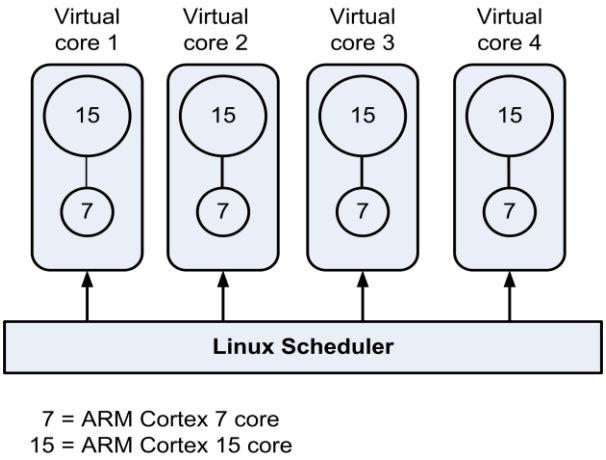

Fig. 1. Schematic arrangement of ARM cores in the Exynos 5410 SoC.

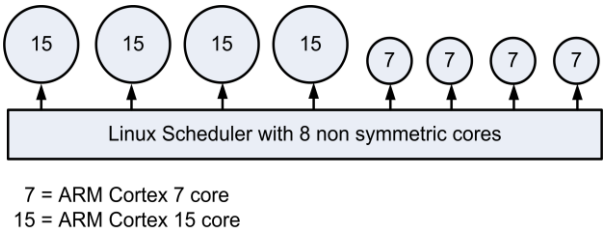

Fig. 2. Schematic arrangement of ARM 7 and ARM 15 cores in the Exynos 5430 SoC.

The 5430 SoC has better computational performance than the 5410 SoC but will consume more power, as is described in the subsequent discussion.

C. Programming on the Exynos System on a Chip

The serial form of the FDTD was parallelised on the Samsung S4's Android v4.4.2 platform using generic threading techniques. The program was built on the phone using the Android-specific gcc compiler. The editing of the program code itself was performed on a

desktop computer and the code was transferred to the phone using local area wireless technology (WiFi). Timing markers were embedded in the code to provide profiling and performance statistics. All computational performance and power consumption measurements for this work were made with programs written in single floating point precision. Graphical results were written to the phone's storage in the VTK [14] format and transferred to the desktop computer for visualisation using Paraview [15].

Android v4.4.2 uses the Dalvik runtime system for the interpretation of the FDTD code. The newer Android Run Time (ART) system is claimed to process Android application code quicker and more power efficiently. The ART was not available for the development platforms used for this work and will be superseded by the Dalvik runtime system in subsequent versions of the Android generations.

For the purpose of comparison with the ARM platform, the two dimensional serial FDTD was also parallelised for the i5-4440k processor using the openMP, MPI, generic threading and various SIMD techniques.

D. The parallel FDTD on multicore platforms using generic threading

The serial form of the FDTD was parallelised using the generic threading for the Exynos 5410 SoC, Exynos 5430 SoC, and the Intel i5-4440k processor.

The computational throughput of the serial FDTD and the parallel FDTD on the four core i5-4440k is shown in Fig. 3. The performance of the serial FDTD on the i5-4440k is shown as a performance reference. The computational throughput of the parallel FDTD on the i5 results shown in Fig. 3 was achieved by using four threads, so that each core is occupied by one thread. The results shown in Fig. 3 do not include the optimisation using the SIMD registers in the i5 processor, although these throughput results are used in Table 1 and shown in Fig. 4.

The computational throughput of the 2D parallel FDTD on the 5410 and 5430 Exynos SoCs is also shown in Fig. 3. The computational throughput was achieved using four threads on the Exynos 5410, although eight cores are made available when one combines the number of ARM 7 and ARM 15 cores, as shown in Fig. 1. The Exynos 5410 did not achieve greater throughput by doubling the number of threads owing to the exclusive core scheduling strategy of the Exynos 5410, as described in section II.B. The computational performance of the 2D FDTD implemented on the Exynos 5430 was not improved either when using more than four threads, very probable because the smaller processing capacity of the ARM 7 core is a limiting factor.

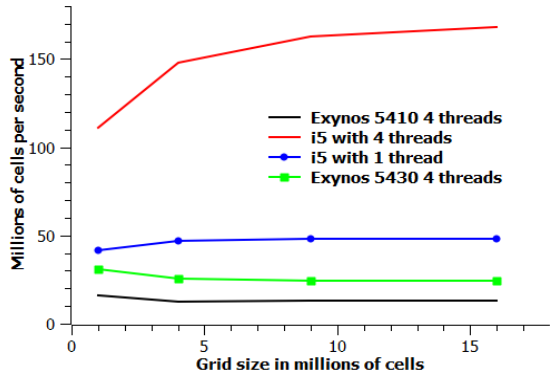

Fig. 3. The processing throughput of the two dimensional FDTD with an increase in grid size.

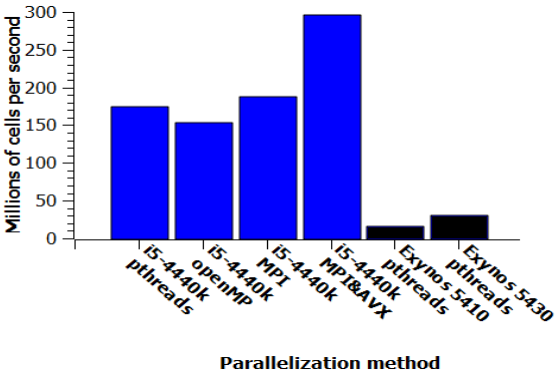

Fig. 4. The performance of the parallel FDTD on the Exynos SoC and the Intel i5-4440k processor. Peak throughput for a grid of 16 million cells.

E. Further parallelisation using the NEON SIMD registers on the Exynos processor

The ARM 7 and ARM 15 processors both have an inherently SIMD engine called NEON. The width of the NEON registers is 128 bits and the NEON operations are capable of handling four floating point operations simultaneously. FDTD implementations making use of the SIMD engines on other processors have shown acceleration of two to three times that of the serial versions.

The implementation of the serial FDTD when using the NEON compiler intrinsics in the Exynos processor did not provide any improvement in computational performance over the serial FDTD. As a test, the FDTD array data was substituted with constant data values, which did indeed provide a marked speedup of the SIMD based acceleration. Whilst the computed results in this case are clearly of no value from a modelling perspective, it showed that the NEON registers are indeed processing the data values effectively, but moving the FDTD data to the NEON registers from the data array to the SIMD engine is

inefficient. Attempts to improve the performance of the array data movement to the SIMD engine by explicitly prefetching the FDTD array data into the cache memory did not enhance the efficiency of the FDTD program when making use of the NEON functionality.

It can also be noted that making use of the automated implementation of the NEON functionality provided by the gcc compiler did not provide any further acceleration of the FDTD either.

By contrast with the lack of acceleration improvement obtained using the SIMD facility for the ARM processors on the Exynos SoCs, further acceleration of the FDTD by using the SIMD facilities inherent in Intel processors provided very good speedups. The speedup in the computational throughput of the two dimensional FDTD on the i5-4440k is shown in Fig. 5. The FDTD was implemented using compiler intrinsic and single precision floating point arithmetic for the AVX registers. The computational performance of the i5-4440k processor is limited when supplying data using only one channel to the physical memory chip, a situation which is alleviated when using two memory channels to the physical memory chips. The use of the auto-vectorisation options on different compilers resulted in a small improvement in computational performance of the FDTD on one core only. Auto-vectorization did not accelerate the parallel FDTD when implemented for more than one core.

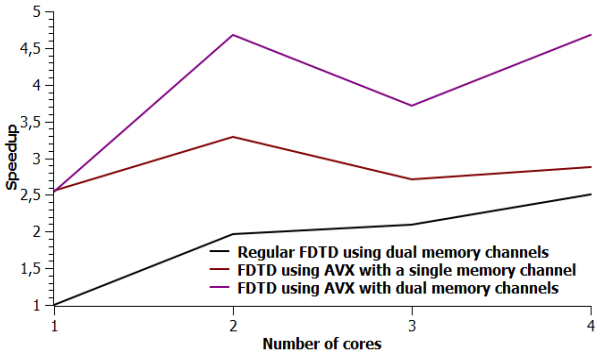

Fig. 5. The data bottleneck effect of using the SIMD registers on the Intel i5-4440k.

III. FDTD COMPUTATIONAL PERFORMANCE/POWER COMPARISON

A. A note on the use of power consumption values on different processors and systems

Although power consumption values are readily given by manufacturers, it is often unclear what these values refer to or how to apply this data to calculate the power consumed by a specific program or process. The peak power consumed for different processes performing at the their maximum capacity is varied [4] and even relatively minor variations within one algorithm can affect the power consumption, as is shown by the results of the parallel FDTD on the i5-4440k, shown in Fig. 6.

The power consumption measurements made for the Intel i5 processor results used in this article were made using a dedicated power meter between the desktop computer and the mains power supply. The baseline power consumption of the desktop computer was measured before the testing and readings were taken while the program was running. The difference in these two power values was used to identify the amount of power consumed by the FDTD itself. Readings were also taken to determine what the effect of the AVX is on the power consumption of the parallel FDTD, as is shown in Fig. 6.

The power consumed by the FDTD on the Samsung phone was determined by calculating the drop of the battery capacity and relating that to the Watt Hour rating of the phone's battery. As it was not possible to exclude the functioning of the myriad of other devices operating in the phone, such as the screen, this power consumption calculation should be assumed to be the worst case scenario for the FDTD. WiFi, Bluetooth, and other unrequired phone features, were switched off during the performance testing.

Another variable affecting the power consumption comparison is the effect of the release age of the processor. According to Koomey's law, new generations of processors are becoming twice as efficient every 18 months. It is therefore implied that newer processors will be more power efficient than older processors. When comparing performance across platforms, programming implementation can also impact on performance evaluation; however, other than the data obtained from Simon [9], all implementations were done by the present authors using very similar code. The power measurements to be presented later on in the article are taken from manufacturers' specification sheets.

B. The performance/power ratio of different parallel algorithms

When comparing the power consumption of the parallelized FDTD for different platforms, it has to be noted that the FDTD may have been parallelized using a different technique to the method with generic threading, and that these techniques all consume power at different rates. This can be demonstrated by examining the techniques used to parallelize the FDTD on the Intel i5-4440k four core processor, where each technique consumes different amounts of power, as shown in Fig. 6. The technique consuming the least amount of power on the Intel four core i5-4440k platform is the FDTD implemented with generic threads, and this has been implemented on the Exynos 5410 SoC, as is shown in Fig. 6. As expected, the FDTD implementation using the SIMD capability of

the i5-4440k also consumes more power than the non-vectorized version. The relative throughput of the parallelized 2D FDTD shown in Fig. 4, measured in millions of cells per second, is as one would expect. The throughput from the Intel i5 four core processor far outstrips the eight cores on the Exynos processor. To achieve this computational throughput advantage though, the i5-4440k processor consumes much more power.

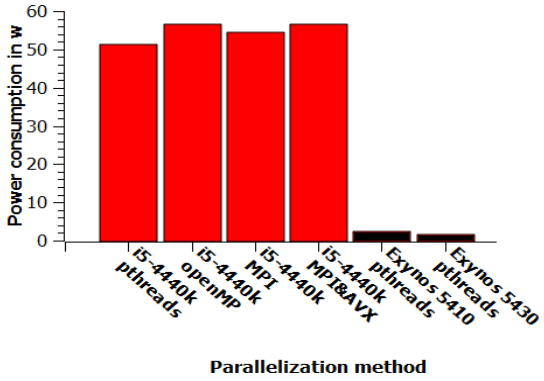

Fig. 6. Power consumption of the parallel FDTD on the Exynos and Intel i5-4440k processors.

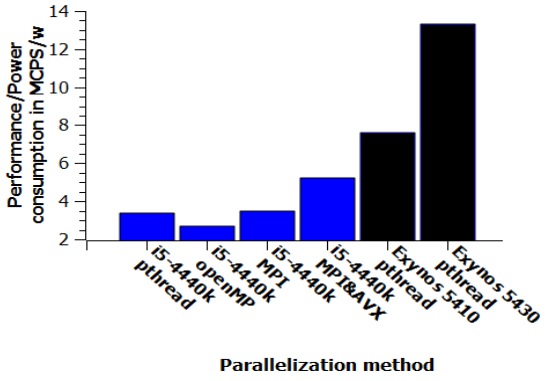

Fig. 7. FDTD performance/power ratio of the Exynos and the Intel i5-4440k.

C. The performance/power consumption of the parallel FDTD compared to other popular computing platforms

One of the features of ARM processors that is constantly emphasized in the popular literature is that the ARM platform consumes very little power. As shown in Fig. 7, the values of FDTD cell throughput normalized by the power consumption seems to verify this for the i5-4440k and Exynos 5410 at least. Table 1 is a comparison of the performance/power ratio of the parallelized FDTD on a variety of computing platforms and illustrates how the FDTD on the Exynos platform relates to these systems. Apart from the FDTD results for the i7 processor [9], all of the parallel FDTD implementations on these platforms originated from the same serial two dimensional FDTD program and was coded by the present authors. It is obvious from this comparison that the processing of the FDTD on the ARM platform does not provide any considerable saving in power consumption. Although the Nvidia GTX 480 appears to have the lowest performance/power rating, it should be noted that this is probably owing to the age of the processor's generation, as predicted by Koomey's law.

The Power PC A2 processor is of particular interest as it processing building block used by IBM's Blue Gene/Q [16] supercomputer. Although released as early as 2011, the Blue Gene/Q still occupies a large proportion of the rankings in Green500 [17] list of energy efficient computers.

The FDTD power efficiency ratings (MCPS per watt) shown in Table 1 agree with the ratings of the top energy efficient processors shown in the Green500 list of November 2014, in that the top 10 positions in the Green 500 list all use Xeon processors similar to the Intel Xeon E5-2640 featuring in Table 1. Although the ARM processors deployed for the FDTD implementations in this work feature near the top of the evaluation in Table 1, they do not appear to feature highly in the current Green500 list.

Table 1: A comparison of the power efficiency of the parallel FDTD on different platforms

Platform	Processor Type	Cores	Release Date	Peak Power (Watt)	Peak MCPS	MCPS Per Core	MCPS Per Watt
GTX 480 Nvidia	GPU	480	2010	320	680	1,4	2,1
IBM Power PC A2	CPU	16 + 2	2010	55	176	11	3,2
C2070 Nvidia	GPU	448	2009	238	780	3,2	3,3
i5-4440 Intel	CPU+AVX	4	2013	56,5*	296	75	5,2
Exynos 5410	CPU	4	2013	2,3*	17,5	4,4	7,6
E5-2640 Intel	CPU+AVX	6	2012	95	1153	192	12,1
Exynos 5430	CPU	4	2014	1,7*	22,7	5,7	13,3
i7-3960x Intel [9]	CPU+AVX	6	2011	130	1800	300	13,8

*Power measurements made manually

IV. FDTD APPLICATIONS ON THE SAMSUNG S4

Despite the limitations of Android platforms on contemporary smartphones as noted in this paper, some quite useful, albeit limited, 2D FDTD simulations can be performed. A commercial application is available on the Android platform to calculate the most suitable position of a WiFi router in an apartment [18]. The FDTD is used to calculate the propagation of the WiFi router's radio transmission throughout the residence so as to determine areas of good and poor WiFi reception.

As a proof of concept, the authors also made computations on the Samsung S4 Smartphone using the two dimensional parallelized FDTD described in this paper. The objective of the computation was to quantify the radio frequency interference shielding provided by a berm (a large earth mound) on a sensitive radio astronomy site in Southern Africa [19]. The results from the FDTD process on the Samsung S4 agreed with those derived from a similar FDTD process modelled on an Intel i5-4440k processor and are shown in Fig. 8. The agreement is satisfactory, given that the application focused on screening, and a 2D model of the 3D berm was used in the simulations, so precise agreement between simulations and measurements was neither expected nor required.

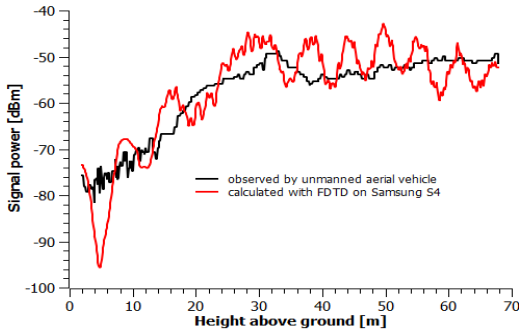

Fig. 8. The signal strength as calculated by the parallel 2D FDTD on the Samsung 4 compared to the signal observed by the unmanned aerial vehicle.

V. CONCLUSION

A comparison of the computational efficiency with the 2D FDTD on other HPC platforms reveals that the ARM processors do not afford a large power saving when computing the FDTD in terms of power-normalized performance. This result may appear surprising, given the claims surrounding low-power processors, but is of course a consequence of their limited performance. For the FDTD at least, most contemporary high performance processors achieve a similar computational efficiency. For applications in large HPC systems, it is the fabric of the system - in particular, the interconnect technology and access to memory - which differentiate systems on the basis of computational efficiency [20]. Nonetheless, as has been described here, the parallel FDTD can be easily deployed in parallel on a Smartphone and used for small-scale rudimentary electromagnetic modeling.

ACKNOWLEDGMENT

DBD and RGI acknowledge the support of SKA South Africa, the South African Research Chairs Initiative of the Department of Science and Technology (DST) and National Research Foundation (NRF), and the Centre for High Performance Computing. The loan of the Alpha Smartphone containing the Exynos 5430 SoC by Samsung is greatly appreciated.

REFERENCES

[1] M. H. Tandel and V. S. Venkitachalam, "Cloud computing in Smartphone: is offloading a better bet?," *Wichita State University*, Wireless Networking and Energy Systems Research Lab., [Online]: http://webs.wichita.edu/?u=WINES&p=/Publications/.

[2] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaersan, and S. V. Krishnamurthy, "Computing while charging: building a distributed computing infrastructure using Smartphones," *8th International Conference on Networking Experiments and Technologies*, Nice, France, 2012.

[3] R. Ilgner and D. B Davidson, "Price-performance aspects of accelerating the FDTD method using the vector processing programming paradigm on GPU and multi-core clusters," *Applied Computational Electromagnetics Society (ACES) Journal*, vol. 29, no. 5, pp. 351-359, Apr. 2014.

[4] D. Hackenberg, R. Schöne, D. Molka, M. Müller, and A. Knüpfer, "Quantifying power consumption variations of HPC systems using SPEC MPI benchmarks," *Computer Science–Research and Development*, vol. 25, pp. 155-163, Sep. 2010.

[5] J. G. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, pp. 46-54, 2011.

[6] A. Taflove and S. C. Hagness, *Computational Electrodynamics, The Finite-Difference Time-Domain Method*, Third Edition, Artech House, Chapters 3 to 7, 2005.

[7] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas Propagation*, vol. AP-14, pp. 302-307, 1966.

[8] C. Yuan and Y. Canqun, "Optimizing SIMD parallel computation with non-consecutive array access in inline SSE assembly language," *Fifth International Conference on Intelligent Computation Technology and Automation*, Zhangjiajie, Hunan,

China, pp. 254-257, 2012.
[9] W. Simon, A. Lauer, and A. Wien, "FDTD simulations with 10^{11} unknowns using AVX and SSD on a consumer PC," *IEEE Antennas and Propagation Society International Symposium (APSURSI)*, Chicago, IL, USA, pp. 1-2, July 2012.
[10] A. Asaduzzaman, F. Sibai, and H. El-Sayed, "Performance and power comparisons of MPI Vs Pthread implementations on multi-core systems," *9th International Conference on Innovations in Information Technology*, Abu-Dhabi, 2013.
[11] D. K. Price, A. L. Paolini, K. E. Spagnoli, and J. P. Durbano, "An accelerated GPU FDTD solver using CUDA," *24th Annual Review of Progress in Applied Electromagnetics*, Niagra Falls, Apr. 2008.
[12] L. Zhang, X. Yang, and W. Yu, "Acceleration study for the FDTD method using SSE and AVX instructions," *Conference on Consumer Electronics, Communications and Networks*, Yichang, China, pp. 2342-2344, Apr. 2012.
[13] Exynos Processor Family, [Online]: available at http://en.wikipedia.org/wiki/Exynos.
[14] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit*, 4th edition, Kitware Inc., 2006.
[15] A. Henderson, *Paraview Guide, A Parallel Visualization Application*, Kitware Inc., 2007.
[16] IBM, *Introduction to Blue Gene/Q*. 2011. [Online]: available at: http://public.dhe.ibm.com/commonlss ilecm/enldcI12345 usen/ DCL l2345USEN.PDF.
[17] Green500 list November 2014, [Online]: available at http://www.Green500.org.
[18] *Wifi Solver FDTD*. [Online]: available Google Android Playstore.
[19] H. Reader and H. Pienaar, "Model and full scale study of soil berm for Karoo array telescope shielding," *International Symposium on Electromag. Compatibility*, Raleigh, North Carolina, Aug. 2014.
[20] R. G. Ilgner, "A comparative analysis of the performance and deployment overhead of parallelized finite difference time domain (FDTD) algorithms on a selection of high performance multiprocessor computing systems," *Ph.D. Thesis*, Dept. of Electronic and Electrical Eng., Stellenbosch Univ., Stellenbosch, South Africa, 2013.

Robert G. Ilgner obtained his B.Sc. and B.Sc. (Hons) degrees in Geophysics from the University of Witwatersrand in 1982 and 1983 respectively. He received a M.Sc. in Telematics in 1991 from the University of Surrey in Guildford, UK, and was awarded a Ph.D. from the University of Stellenbosch in 2013.

As a Geophysicist he conducted geophysical exploration surveys for mining houses in Southern Africa. He then worked for Siemens (UK) in the Information Technology Industry designing large database systems. He was subsequently employed by Schlumberger in their Seismic Division creating parallel processing applications used for seismic data reduction and modeling. He has built software for the creation of imaging applications and advertising on the internet.

His research interests are in HPC and CEM with the FDTD. He currently manages a variety of commercial IT projects and is a Member of ACES and the South African Geophysical Association.

David B. Davidson received the B.Eng., B.Eng. (Hons), and M.Eng. degrees (all cum laude) from the University of Pretoria, South Africa, in 1982, 1983, and 1986 respectively, and the Ph.D. degree from the University of Stellenbosch, Stellenbosch, South Africa, in 1991.

In 1988, he joined the University of Stellenbosch. He currently holds the South African Research Chair in Electromagnetic Systems and EMI Mitigation for SKA, there in 2014, he was promoted to Distinguished Professor. He has held a number of visiting appointments in Europe, the UK and the USA.

His main research interest through most of his career has been computational electromagnetics (CEM), and he has published extensively on this topic. He is the author "Computational Electromagnetics for RF and Microwave Engineering" (Cambridge, U.K.: Cambridge Univ. Press, 1st ed, 2005, 2nd ed., 2011). Recently, his interests have expanded to include engineering electromagnetics for radio astronomy.

Davidson is a Fellow of the IEEE, a Member of the South African Institute of Electrical Engineers and the Applied Computational Electromagnetic Society, and is a registered professional engineer. He was a recipient of the South African FRD (now NRF) President's Award in 1996. He received the Rector's Award for Excellent Research from the University of Stellenbosch in 2005. He is the Editor of the "EM Programmer's Notebook" column of the IEEE Antennas and Propagation Magazine. He was Chair of the local organizing committee of ICEAA'12-IEEE APWC-EEIS'12, held in Cape Town in September 2012. He received the inaugural IEEE/SAIEE Distinguished Volunteer Award in 2015. He currently serves on South Africa's national Astronomy Advisory Council.