INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 4, PP. 363–368
Manuscript received October 1, 2012; revised November, 2012. DOI: 10.2478/v10177-012-0050-2

Hardware Accelerated Simulation of Crest Factor Reduction Block for Mobile Telecommunications

Maciej Nikodem and Krzysztof Kępa

***Abstract*—This paper reports results of the hardware accelerated simulations of the crest factor reduction (CFR) block which is a common element of the radio signal processing path in base stations for mobile telecommunications. Presented approach increases productivity of radio system architects by shortening the time of model architecture evaluation. This enables unprecedented scale of CFR parameter optimization which requires thousands of simulation runs. We use FPGA device and Xilinx System Generator for DSP technology in order to model CFR block in MATLAB/Simulink environment, implement the accelerator and use it for mixed hardware-software simulation. Reported approach reduces simulation time by 70%, provides straightforward use of fixed-point arithmetic and lowers power consumption by 73% at the cost of constant and relatively low overhead on model development.**

***Keywords*—Crest factor reduction, configurable hardware, hardware acceleration, FPGA, telecommunications.**

I. Introduction

DEVELOPMENT of mobile network Base Stations (BSs), and BS Radio Modules (RMs) in particular, require number of hardware and software issues to be solved. Also, in order to achieve the best in-field performance, the time-consuming configuration parameter optimization is required. RMs are required (e.g. [1]) to achieve flexibility through simultaneous support for diverse radio-access technologies (e.g. GSM, UMTS, LTE, WiMAX), in-the-field reliability in harsh and unsupervised environment, and high efficiency in terms of power consumption per Watts transmitted. Also, the RM products are required to be compliant with 3GPP specification and Federal Communication Commission (FCC) regulations. At the same time RM production must be cost-effective, thus use of less expensive components must often be compensated with additional hardware and software processing in order to meet the requirements. Power amplifiers (PAs) are typical example as they may account for up to 30% of the total cost of the RM. Important requirement metrics for PA is its high power-efficiency, linearity and low distortion. Class A PA meets requirements for linearity and low distortion, however, it is built with expensive high-power transistors and is inefficient in transforming DC power to radio frequency power. Therefore, the cost-attractive alternative is to use less-expensive AB class PA and compensate for its nonlinearities, e.g. by using digital pre-distortion (DPD). DPD offers high efficiency and flexibility at relatively low costs. The idea behind DPD is to introduce inverse-distortion to the input signal that will compensate for distortions introduced by PA. Such linearization is easier when dynamic range of the input signal is small. Unfortunately today's wireless communication systems use sophisticated, non-constant envelope signals and multicarrier transmission that often yield signals of high Peak-to-Average ratio (PAR) and large dynamic range. Therefore, to achieve cost-effective and efficient PA linearization PAR reduction is required.

Crest Factor Reduction (CFR) is a technique to reduce PAR. CFR blocks are part of radio frequency base band processing, usually situated after the digital up conversion (DUC) and prior to DPD (Fig. 1). The idea behind CFR is to detect signal peaks that exceed acceptable power level and to cancel these peaks. The straightforward cancellation through signal saturation has an adverse impact on frequency spectrum and cause interferences to neighboring bands that are measured with adjacent channel power ratio (ACPR) parameter. These interferences can be filtered out to meet the requirements of spectral mask, however, in-band distortions, which are expressed in Error Vector Magnitude (EVM) need additional processing in order to ensure EVM is on acceptable level and within limits specified by 3rd Generation Partnership Project (3GPP). Consequently, CFR is a complex signal processing algorithm composed of peak detection, clipping, in-band and out-of-band processing blocks (Fig. 1).

The CFR algorithm must be flexible in order to properly clip carrier signals in different radio access technologies, multi-carrier solutions and configurable frequency band assignment. Therefore, in real life applications the operation of CFR is controlled by a number of configurable parameters that are verified using a 3GPP test signals. An ideal solution to select proper parameters is to develop a universal CFR module which can be parameterized and verified. Unfortunately, developing such module is time-consuming, costly, inefficient and impractical as different CFR architectures are still investigated and radio access technologies evolve quickly (they may change faster than time required to design and fabricate a dedicated application specific integrated circuit – ASIC).

Faster and less expensive solution is to model architecture of the CFR block in software and simulate its operation for various sets of parameters and possible input reference signal combinations. Such exploration is not only faster but also more universal as it is relatively easy to investigate different

This publication was prepared as a part of the project of the City of Wroclaw, entitled – "Green Transfer" – academia-to-business knowledge transfer project co-financed by the European Union under the European Social Fund, under the Operational Programme Human Capital (OP HC): sub-measure 8.2.1.

M. Nikodem is with the Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wroclaw, Poland (e-mail: maciej.nikodem@pwr.wroc.pl).

K. Kępa is with Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, VA, USA (e-mail: kepa@vt.edu).

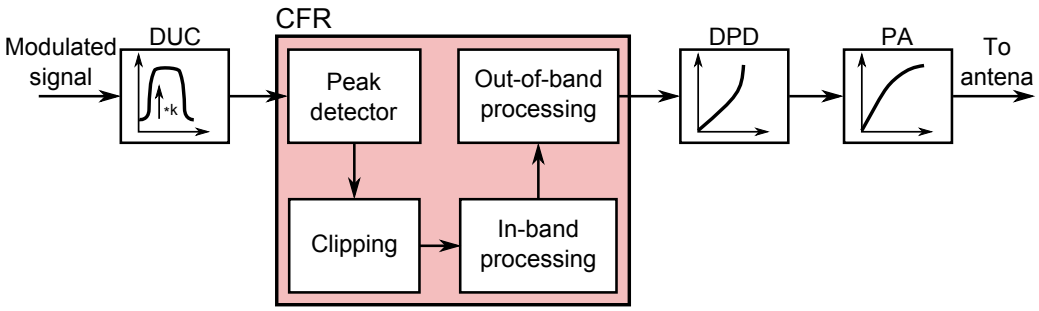

Fig. 1. General structure of signal processing in BS radio module.

structures of CFR block. Chosen architecture and parameters can be later on used to develop a hardware prototype, to verify operation and possible additional adjustments of the CFR block. In the last step, when architecture and parameters are decided the dedicated CFR module (e.g. ASIC), is developed.

Although software simulation has indisputable advantages (just to mention shorter development time and lower costs) there are several issues that need to be addressed to make it efficient and more practical. This paper addresses three of them:

- speed – software simulations take more time when compared to processing time of the same amount of data in hardware module,
- data representation and precision – software simulations use floating-point arithmetic while fixed-point arithmetic is used in hardware. Different arithmetic yields differences in signal representation, calculations and resulting cumulative error (e.g. due to rounding),
- clarity of the CFR algorithm – CFR is a digital signal processing (DSP) algorithm that is easier to analyse and understand when represented using signal processing primitives (e.g. filters, coordinate rotation digital computers – CORDICs, modulators, demodulators, multirate operations, etc.) as they conceal algorithm details. Although this is also possible for programming languages (e.g. MATLAB) developing such software is difficult.

Our approach is to solve above simulation shortcomings through reuse of configurable devices (FPGAs in particular) for mixed hardware-software accelerated simulation. Such method improves the speed of signal processing, shortens simulation time and gives results closer related to what is expected from CFR blocks running in the real BSs (as fixed-point arithmetic is used). Our approach extends productivity of radio system architects, by increasing number of possible architectural evaluations within a fixed project time-period, while maintaining the software-like configurability of the prototype. Additionally we use MATLAB/Simulink environment for modeling the CFR block. This gives an easy to understand and analyse representation of the algorithm that can be also used to generate hardware structure and to run CFR module simulation using configurable devices.

II. Related Work

Since multicarrier transmissions and non-constant envelope modulations become popular in mobile networks a number of papers and technical solutions were proposed to efficiently combine them and reduce PAR. One of the early paper by Väänänen et al. [2] focused on GSM and EDGE signals and analysed different windowing methods and algorithms to prevent signal over clipping. In [3] the same authors presented that CFR can be successfully applied to GSM and WCDMA carriers. Swaroop and Gard [4] proposed to separate clipped signal into correlated and uncorrelated components using autocorrelation function with the input signal (prior clipping). The correlated component is a valuable desired signal, while uncorrelated component is unwanted signal composed of both in-band and out-of-band noises that are later removed. Similar approach for OFDM modulation was presented by Zhao [5].

Hardware accelerated simulation using configurable devices attracts more attention since they surpassed performance of digital signal processors. Paper by Lin et al. [6] presents a dedicated HAST tool that use high level MATLAB code (m-files) for implementation of signal processing algorithm that is later run on a Nallatech Xilinx FPGA board. The HAST tool uses MATLAB code to generate hardware description which is likely to output results larger and slower from what can be achieved using dedicated synthesis tools. Simulations of various FIR filters show a peak improvement of 69 times when compared to software implementation in MATLAB. This shows a great potential of hardware accelerated simulation but was only verified for simple circuits. Contrary results were presented in [7] where authors analysed efficiency of mixed hardware/software (HW/SW) simulation. Due to long latencies for accessing the FPGA, system driver overhead and simplified implementation, mixed HW/SW simulation times exacerbated software simulation times.

Number of authors also analysed MATLAB/Simulink based design flows. Zoss et al. [8] used two reference design and compared Synopsys' tools, Mathworks' Simulink HDL Coder and Xilinx's System Generator (SG) for DSP. SG generated faster hardware requiring lower number of resources. Zoss claims limitation to Xilinx's devices is a disadvantage of SG, however, favors SG for high level hardware description and library of specialised, efficient IP cores. MATLAB/Simulink was also used by Chugh at al. [9] who implemented a WCDMA rake receiver and compared required resources, operating frequency, and throughput for different architectures of the receiver.

This paper presents results of HW accelerated simulation of CFR algorithm used in mobile network BS. Since we

chose to use Xilinx devices for HW acceleration, we follow MATLAB/Simulink based design flow and use Xilinx System Generator for DSP. Another reason to do so is the fact that Simulink became a standard, flexible and convenient tool for developing, modifying and analysing of DSP algorithms. Developed SG models are compared with three different simulation approaches: floating-point MATLAB; fixed-point MATLAB; floating-point MATLAB/Simulink.

III. Why to Use Mixed HW/SW Simulation

Despite the mentioned advantages, software simulation of CFR algorithms has several drawbacks that cannot be efficiently and easily solved. The software simulation speed and high memory requirements (for long test signals) are two of the most important disadvantages. Both shortcomings are result of large amount of IQ samples that need to be processed, (number of samples is multiplied by digital up conversion that precedes CFR module). Since memory constrain is crucial thus the solution is to split signal into smaller blocks (chunks) and process them sequentially. Unfortunately, such method requires additional management, increases overhead and consequently affects overall execution time. Chunk processing makes computation parallelisation difficult, thus having an unfavourable impact on simulation time. The amount of data to process increases additionally in multicarrier simulation extending execution time further. Multitasking and multithreading of today's computers and operating systems has also an adverse effect on software simulation time, as processor time is consumed on running other applications, managing memory, disk access, etc. As a result software simulation of the CFR block for 2-carrier reference scenario, using a 10 ms long test signal, takes 54 s on average.

Another issue, when using software simulation, is the floating-point arithmetic that is by default used in the MATLAB simulation environment. This is in contrast to hardware devices that use fixed-point arithmetic instead. Fixed-point arithmetic implementation is typically more resource-efficient and has fixed precision for all the numbers in the representation range (in floating-point arithmetic precision decreases as absolute value of the number grows). Consequently, running floating-point software simulation may output signals that are different from output of hardware implementing the same algorithm, but using fixed-point arithmetic. For complex CFR algorithms of practical interest, this may lead to situations when selected set of parameters cause the resulting hardware to fail 3GPP requirements (e.g. EVM, ACPR). This is unacceptable due to extremely high cost of redesign. The obvious solution is to emulate fixed-point arithmetic in MATLAB simulation. Unfortunately, due to floating-point nature of the software this requires additional operations that increase overhead and extend simulation time of the reference scenario by the factor of 2.8, up to 150 s on average. This slow down is unacceptable if the large number of simulations is required, e.g. in architecture/parameter space exploration. Using floating-point simulator for vast exploration and fixed-point for only some sets of parameters seems like a solution, but such approach requires two versions of the simulator being developed and managed simultaneously which is impractical.

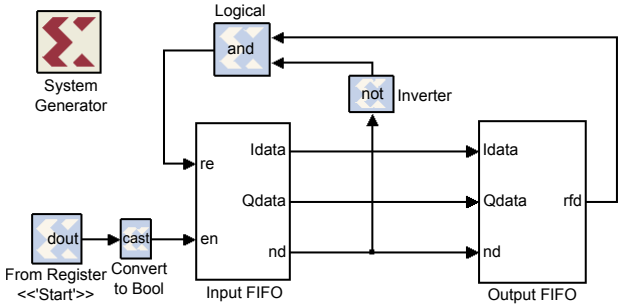

Fig. 2. Simple SG model for running in FR mode. This model was used to verify throughput of JTAG and Ethernet communication interfaces.

Above mentioned issues can be solved by moving simulation to configurable accelerator (e.g. FPGA device) and use fixed-point arithmetic. Although moving may yield additional overhead during model development this cost is incurred only once. After hardware model is developed, simulations in fixed-point arithmetic yield high throughput and give results that are closer related to what is expected from the final CFR module. Such approach also enables detection of situations mentioned earlier, when the proposed CFR architecture and set of parameters misses the requirements. This allows algorithm redesign in early stage, when the cost of architecture modification is still low. Additionally, both Altera [10] and Xilinx [11] provide dedicated signal processing primitives that can be used in MATLAB/Simulink environment to easily create and simulate signal processing algorithms. Since all such blocks have underlying IP cores it is possible to use MATLAB/Simulink models to synthesise hardware that may be loaded to configurable device and run. Appropriate functions allow communicating with hardware model directly from MATLAB code as well as sending data to/from the hardware. Although hardware constructed as a composition of predefined IP cores is not necessarily as efficient as dedicated circuit, it can still improve simulation and shorten time required to verify sets of parameter values. As a result these technologies allow for easy and straightforward development of efficient hardware accelerated simulations.

IV. Hardware Model

We developed models for HW accelerated simulation using Xilinx System Generator for DSP [11] that is high-level tool for developing digital signal processing algorithms using FPGAs. The advantage of SG is that it integrates with MATLAB/Simulink environment providing simple system modeling, code generation and mixed HW/SW simulation. We developed three different types of models for three signal processing blocks: DUC, CFR and both DUC+CFR. All types of models were developed in Simulink environment – one was composed of Simulink blocks; two other were build using SG blocks for two different modes of operation: single-step (SS) and free-running (FR). Simulink model was developed as an intermediate step between transformation from MATLAB reference simulation and target SG models.

A. Single Step and Free-Running Modes

SS and FR models were developed to compare these two modes. In SS mode hardware is kept in step lock with the simulation and is provided a fixed number of clock pulses per each simulation step. In this mode performance of hardware accelerated simulation may be significantly limited due to the overhead associated with the Simulink simulation and communication. This is not the case in the FR mode as hardware accelerator runs asynchronously to the Simulink simulation. This ensures faster simulation times but also requires an extra synchronization mechanism to be inserted between the SG model and Simulink.

The need for synchronization also introduces different methods of interfacing signal to/from hardware model. Although both methods are independent of the actually used communication interface (either JTAG or Ethernet) the hardware model changes. Dedicated Gatway In and Gatway Out blocks can be used in SS mode. These blocks are responsible for converting signal representation (float to fixed and vice versa) and synchronizing software and hardware parts of the simulation so that signals are correctly transferred – they simply do whatever is needed to transmit signals to/from hardware simulation. In FR mode gateways are useless as hardware part runs asynchronously to the software and there is a need to use shared memories (FIFO buffers, and registers) and additional signalling to indicate data is available for processing (cf. Fig. 2). Shared memories are accessible from both hardware and software thus allowing data exchange between both parts of the simulation.

Another consequence of using shared memories is the requirement for converting signal representation before writing the data to input shared memory (sending data to hardware model) and after reading the data from shared memory (receiving from hardware). This is due to the fact that hardware interprets content of the shared memories as fixed point values, while MATLAB interprets the same content as floating-point representation. Consequently, to make sure data is correctly interpreted we need to convert representation before writing to shared memory. For the same reason data received from hardware needs to be represented in floating-point format to make sure it is correctly interpreted by MATLAB. Although representation can be also converted in hardware, such approach requires additional hardware resources and increases amount of data transmitted. Therefore, we decided to implement it in software.

B. Programming and Communication Interfaces

Xilinx's System Generator for DSP offers two programming and communication interfaces – JTAG and Ethernet. JTAG interface is primarily used for FPGA programming but can be also used to transmit data between hardware and software part of the simulation. Some development boards (e.g. ML 506) also support programming over the Ethernet connection but such possibility is restricted to the small number of boards and requires the FPGA device to be programmed with dedicated initial design beforehand. For other types of development boards Ethernet interface can be only used as a communication interface for data transmission, while programming of the FPGA is done over the JTAG. SG model is independent of the used interface, but it has to be decided before synthesis.

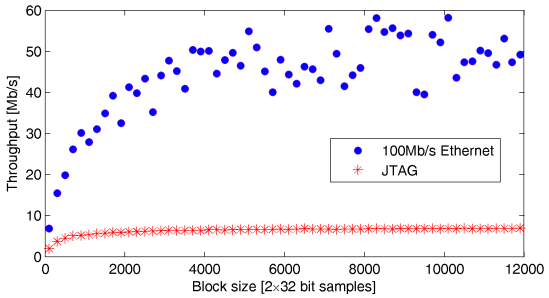

Fig. 3. Throughput for Ethernet and JTAG interfaces as a function of block size.

Figure 3 presents a throughput of both interfaces when used for data transmission to a simple hardware model presented in Fig. 2, as a function of the block size. As expected Ethernet interface is more than 8 times faster comparing to JTAG and hits 59 Mb/s for a 100 Mb/s Ethernet interface (we used the ML 507 board which supports 1 Gb/s Ethernet but it was unattainable with 100 Mb/s Ethernet card in the computer). For all the tests presented in this paper Ethernet interface and 8192 samples shared FIFOs were used in hardware models.

V. Simulations, Measurements and Results

We simulated Simulink and SG models for each block (DUC, CFR, DUC+CFR), for operation with two 10 ms long carrier signals. We measured time required to perform simulations and compared it with time required to run MATLAB simulation (both floating and fixed point). We also compared the resulting output signal with the output of corresponding software simulation to estimate its quality. The difference between signals is expressed in absolute values, root mean square value (RMS), average and peak relative difference between absolute values.

To run hardware accelerated simulations efficiently we developed a set of MATLAB functions that are responsible for loading the model into the FPGA, converting input and output signals, and supervising the simulation process. The simulation procedure is composed of six steps: (1) model loading; (2) CFR/DUC configuration and setup; (3) input data conversion; (4) processing; (5) output data conversion; and (6) disconnection.

Programming the FPGA device is the first step of the simulation. When loaded the hardware model can be parameterized and shared memories can be associated with software objects. Sample parametrisation may include definition of clipping threshold, allowable EVM or number of input carrier signals. Third step is responsible for converting the floating-point to fixed-point representation, and is implemented in software. Due to relatively large overhead of this operation the conversion is done prior to actual simulation. This approach allows 4th step to only consist of data transmission to/from hardware model and simulation execution with no additional delays on data conversion. Due to the same reason output conversion

TABLE I
AVERAGE SIMULATION TIME FOR DIFFERENT SOFTWARE AND HARDWARE SIMULATIONS OF DUC, CFR AND DUC+CFR MODELS

Model (signal length [ms])	HW / SW	Time [s] DUC	CFR	DUC+CFR
Floating-point MATLAB (10)	SW	44.24	10.10	54.34
Fixed-point MATLAB (10)	SW	117.54	16.23	133.77
MATLAB/Simulink (10)	SW	23.26	45.28	75.69
SG in FR mode (10)	HW	47.35	23.58	16.06
SG in SS mode (0.4)	HW	3 220.3	857.5	4 499.3

TABLE II
DETAILS OF SIMULATION TIME FOR DUC+CFR MODEL SIMULATED IN FREE-RUNNING MODE WITH 10 MS SIGNAL

Step	Time [s]	Percentage
Loading	12.163	75.75%
Configuration and setup	0.005	0.03 %
Input conversion	0.159	0.99 %
Processing	3.632	22.62 %
Output processing	0.034	0.21 %
Disconnection	0.001	0.01 %

is also carried out in software after the whole simulation is finished. The last step ensures that FPGA disconnects properly and all the resources, that are associated with shared memories, are released.

It follows from results presented in Tab. I that SG model of DUC+CFR, running in FR mode, outperforms MATLAB simulators by a factor of 3.3 and 8.8 with respect to floating and fixed point simulation. For SG model running in FR mode simulation times of DUC and CFR block are slightly longer than corresponding MATLAB simulation but differences are small. For Simulink and SG model running in SS mode results are worse as both models introduce additional overhead due to step lock simulation mode. Note also that SS mode has been simulated only for 0.4 ms input signal, in contrast to 10 ms input signal used in remaining simulations. Since simulation time for SS mode is proportional to signal length thus simulation for 10 ms long inputs would take approximately 22, 6 and 31 hours for DUC, CFR and DUC+CFR models respectively.

Worse performance of SG models running in FR mode is a consequence of an up-sampling factor that yields large amount of data (approx. $6\text{x}10^6$ samples) to be transmitted to/from the FPGA. This is not an issue for DUC+CFR model as internal signals are not transferred from the model back to PC (approximately only $0.45\text{x}10^6$ output samples are transmitted). Second explanation for relatively long simulation times for SG in FR mode is that total time results from completion time of all six simulation steps. Table II compares average time required to execute each step of the simulation procedure for a simulation of DUC+CFR block in FR mode. It follows that loading accounts for three quarters of the overall simulation time. This time is constant for different models and independent of signal length. Negligible times for configuration, setup and sample representation conversion suggest that, in practical simulations, hardware model should be loaded once and then run for a large number of test signals and parameters settings (configured through shared registers). In such approach FR mode will outperform MATLAB simulation as loading time can be discarded – cf. Tab. III.

TABLE III
TIMING AND SPEED COMPARISON FOR SOFTWARE AND FREE-RUNNING HARDWARE SIMULATION FOR A 10 MS LONG SIGNAL. FOR FREE-RUNNING SIMULATION LOADING TIME IS DISCARDED

Model	Time [s] DUC	CFR	DUC+CFR
Floating-point MATLAB (SW)	44.24	10.10	54.34
Fixed-point MATLAB (SW)	117.54	16.23	133.77
SG in FR mode (HW)	35.06	11.48	3.83
Speed improvement for SG in FR mode (HW)			
vs. float-point MATLAB	1.2	0.8	14.2
vs. fixed-point MATLAB	3.3	1.4	34.9

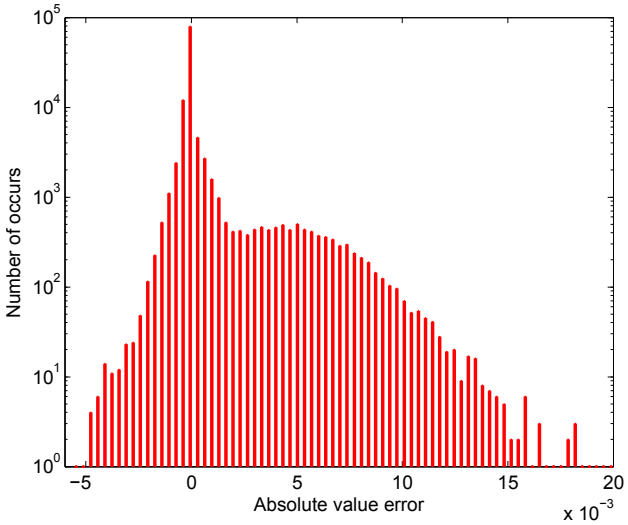

Fig. 4. Histogram of absolute error for output signals from software and hardware accelerated simulations of DUC+CFR block.

The signal produced by the hardware models was verified against the signal which is obtained from the software simulation. For each model and output signal we calculated difference in absolute value, mean and peak relative amplitude error and root mean square error (RMS). Figure 4 presents a histogram of absolute value difference. It follows that for 89.5% of samples the difference between referential and actual signal is below 10^{-3} and for 99.6% – below 10^{-2}. Relative error measures (Tab. IV) also confirm that outputs of software and hardware models are consistent. The highest relative amplitude difference occurs for input samples of a small value (close to zero). The reason is that these values cannot be represented precisely using fixed-point format and consequently, samples are truncated to zero. The EVM value for output of hardware accelerated DUC+CFR simulation, relative to unclipped signal, equals 5.658%. It is slightly more pessimistic (by 0.15%) when compared to the EVM values for output signal of floating-point MATLAB simulation which yields EVM of 5.508%.

Another benefit from using hardware accelerated simulation over pure software simulation is reduced power consumption per test (Fig. 5). We measured how much energy is consumed by a simulation setup (i.e. laptop and LCD monitor for software simulation; and laptop, LCD monitor and ML 507 FPGA board for hardware simulations). Software floating and fixed point simulations consume around 51 W (compared to

TABLE IV
MEASURED RMS AND RELATIVE ERRORS BETWEEN OUTPUT SIGNAL AMPLITUDES FOR SOFTWARE AND HARDWARE ACCELERATED SIMULATION

Error measure	Error value [%]		
	DUC	CFR	DUC+CFR
RMS	0.097	0.034	0.246
Mean relative absolute error	0.083	0.005	0.143
Peak relative absolute error	0.383	1.508	4.015

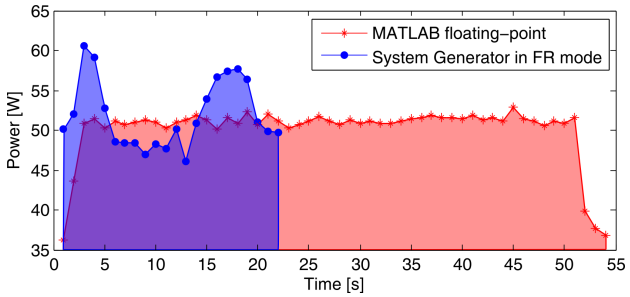

Fig. 5. Power consumption during hardware (blue) and software floating-point (red) simulation of DUC+CFR blocks.. Note different simulation times for both models.

37 W when laptop is idle). Using FPGA introduces additional power cost of approximately 12 W yielding a total power consumption of 49 W when no simulation runs. During hardware accelerated simulation average power consumption varies between 58 and 60 W with instantaneous peak power consumption of about 62 W. This is both a consequence of higher consumption from hardware running as well as from laptop that consumes energy on managing and transmitting the data. Output conversion introduces another increase in power consumption, however, it is insignificant as time required to perform this operation is short. Nevertheless increased power consumption, total energy consumed per each HW accelerated simulation is almost 73% lower when compared to the software floating-point and 92% lower compared to software fixed-point simulation (0.29 Wh vs. 1.11 Wh and 3.90 Wh respectively).

VI. CONCLUSIONS

System Generator for DSP is an intuitive and straightforward environment for modeling, software simulation, verification, hardware synthesis and hardware accelerated simulation of DSP algorithms. Ease of modeling and software simulation does not however go in line with hardware implementation. This is caused by the implicit requirement to synchronise model's elements. This requirement does not exist in MATLAB/Simulink simulations but is crucial in hardware models. Consequently, it is easy to develop, simulate and verify SG model in MATLAB/Simulink but it is much more difficult to develop model that may run in hardware. Single-step simulation is a solution that overcomes synchronization problem but at the cost of lower performance. Thus in order to fully benefit from hardware capabilities it is necessary to develop more complex SG models that can run in free-running mode.

Our SG models does not supports resource sharing between different SG blocks. It is therefore impossible to tradeoff throughput and speed for complexity or lower resource requirements. Consequently FPGA utilisation grows quickly with model complexity. In our case dedicated multiplier blocks (DSP48E) were limitation as only 128 were available. This is escalated by standard settings for some SG blocks (e.g. multiplier, FIR filters, ect.) as they use DSP48E by default. If multiplication is a common operation you may easily exceed available resources.

Despite limitations using SG and FPGA devices for hardware accelerated simulation is easy. It allows for significant improvement in simulation speed, reduces overall energy consumption and simplifies DSP modeling. As presented, it can improve simulation speed by a factor of 14 for a complex model. Further improvement are possible either by limiting amount of data transmitted or improving model architecture (i.e. building dedicated circuits from simple elements instead of using standard blocks).

ACKNOWLEDGMENT

Authors would like to thank Maciej Klemensowicz and Łukasz Skomra from the Nokia Siemens Networks European Software and Engineering Center in Wrocław, Poland for their significant support and engagement in this project.

REFERENCES

[1] "3rd generation partnership project; technical specification group radio access network; E-UTRA, UTRA and GSM/EDGE; multi-standard radio (MSR) base station (BS) radio transmission and reception," 3rd Generation Partnership Project, Tech. Rep. version 11.0.0, release 11, March 2012.

[2] O. Vaananen, J. Vankka, and K. Halonen, "Reducing the peak to average ratio of multicarrier GSM and EDGE signals," *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, pp. 115–119, 2002.

[3] ——, "Simple algorithm for peak windowing and its application in GSM, EDGE and WCDMA systems," *Communications, IEE Proceedings-*, vol. 152, no. 3, pp. 357–362, 2005, DOI: 10.1049/ip-com:20059014.

[4] P. Swaroop and K. Gard, "Crest factor reduction through in-band and out-of-band distortion optimization," in *Radio and Wireless Symposium, 2008 IEEE*, jan. 2008, pp. 759–762, DOI: 10.1109/RWS.2008.4463603.

[5] C. Zhao, "Distortion-based crest factor reduction algorithms in multicarrier transmission systems," Ph.D. dissertation, Grorgia Institute of Technology, 2007.

[6] V. Lin, R. Speelman, C. Daniels, E. Grayver, and P. Dafesh, "Hardware accelerated simulation tool (HAST)," in *Aerospace Conference, 2005 IEEE*, march 2005, pp. 1475–1483, DOI: 10.1109/AERO.2005.1559437.

[7] T. Suh and H.-h. S. Lee, "Initial Observations of Hardware/Software Co-Simulation using FPGA," in *in Architecture Research, 2nd Workshop on Architecture Research using FPGA Platforms*, 2006, DOI: 10.1.1.84.3965.

[8] R. Zoss, A. Habegger, V. Bandi, J. Goette, and M. Jacomet, "Comparing signal processing hardware-synthesis methods based on the Matlab tool-chain," in *Electronic Design, Test and Application (DELTA), 2011 Sixth IEEE International Symposium on*, jan. 2011, pp. 281–286, DOI: 10.1109/DELTA.2011.58.

[9] M. Chugh, D. Bhatia, and P. T. Balsara, "Design and implementation of configurable W-CDMA rake receiver architectures on FPGA," *Parallel and Distributed Processing Symposium, International*, vol. 4, p. 145b, 2005, DOI: 10.1109/IPDPS.2005.162.

[10] "DSP builder handbook - volume 2: DSP builder standard blockset," Altera Corporation, Tech. Rep. 12.0, June 2012.

[11] "System Generator for Digital Signal Processing - User Guide," Xilinx Inc., Tech. Rep. 14.1, April 2012.