Finite Precision Analysis for an FPGA-based NILM Event-Detector

Rubén Nieto, Laura de Diego-Otón, Álvaro Hernández, Jesús Ureña
Electronics Department, University of Alcalá
Alcalá de Henares (Madrid), Spain
{ruben.nieto, alvaro.hernandez, jesus.ureña}@uah.es
laura.diego@edu.uah.es

ABSTRACT

Most Non-Intrusive Load Monitoring (NILM) techniques often require a correct detection of the events that occur in the mains, in order to carry out a correct identification of the corresponding appliances. For that performance, event-detectors are normally based on signals, such as voltages and currents, acquired by a smart meter at the entrance of the household. In this work, a finite precision analysis is performed for an event detector implemented on a System-on-Chip (SoC) based on a Field-Programmable Gate Array (FPGA) for NILM applications. The proposal employs an integrated circuit (ADE9153A) to perform the signal acquisition at 4 ksamples/s, higher than those used in common smart meters installed at homes nowadays. The purpose of the finite precision analysis is to reduce the hardware resource consumption of the FPGA, while the precision of the event detector is maintained. To validate the design, a comparison is carried out in terms of accuracy and specificity against previous works, as well as a comparison of the event detector using finite precision (fixed-point) and single-precision floating-point resolution. Finally, the hardware resource consumption obtained for the proposed architecture is discussed.

KEYWORDS

Non-Intrusive Load Monitoring, Event Detector, System-on-Chip, Finite Precision

ACM Reference Format:
Rubén Nieto, Laura de Diego-Otón, Álvaro Hernández, Jesús Ureña. 2020. Finite Precision Analysis for an FPGA-based NILM Event-Detector. In *The 5th International Workshop on Non-Intrusive Load Monitoring (NILM '20), November 18, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3427771.3427849

1 INTRODUCTION

Non-Intrusive Load Monitoring (NILM) techniques provide energy disaggregation in different applications [16] , mainly to manage the energy demand or to develop monitoring systems for the independent life of elderly. The aim of these systems is to detect and identify the most significant loads in the energy consumption of a household or building.

The aspect that has significantly boosted NILM techniques is the development and deployment of Smart Meters (SM) in many countries, also with a relevant expected expansion in the coming years [7, 17], as well as the possible combined use together with gas and water smart meters [10, 13]. This brings up different applications for power demand management, improving energy efficiency or other monitoring utilities. However, NILM techniques based on SM have the disadvantage that the signal processing is not performed locally, but, instead, they often transfer the complex processing to the cloud. This reduces the cost of the SM and decreases its complexity. Nevertheless, the raw electrical signals (voltage and current) are not transmitted at high sampling rates, reducing the sampling rate to the range of Hz [9]. Whereas this sampling rate may be enough to identify heavy-duty appliances, such as ovens, heaters or fridges [11, 15], the performance of NILM techniques would be increased with the use of higher sampling rates (in the range of kHz or even MHz) in order to identify appliances that have lower energy consumption [5, 6].

NILM techniques often involve an event detector, which is designed to identify any change in the energy consumption registered in a household. These changes are related to the on/off switching of appliances. If an event is detected on the SM, just a reduced window of analysis before and after the event can be enough to identify the corresponding appliance. From the event detection point of view, the signal processing is usually divided into three different stages. At first, a pre-processing is implemented in order to filter and amplify the voltage and current signals to improve event detection. In general terms, the current signal is used for event detection. Secondly, an event detector is required to detect any change in the electrical status of the mains, which is the result from the switching of appliances [4]. Finally, a classification algorithm may be included at this point to identify the appliance associated with the event, based on diverse methods, such as probabilistic ones [14], Principal Component Analysis (PCA) [2] or artificial neural networks (ANN) [8].

In order to perform the different stages described above, some proposals based on FPGA (Field-Programmable Gate Array) devices have been presented to reach a real-time processing, providing flexibility to the design. On the other hand, the recent System-on-Chip (SoC) architectures include a processor along with programmable logic, thus involving a greater versatility to these architectures. These devices, more complex and with higher costs than other alternatives, may allow in future to integrate in the same architecture other stages often involved in NILM techniques, such as load

NILM '20, November 18, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8191-8/20/11…$15.00
https://doi.org/10.1145/3427771.3427849

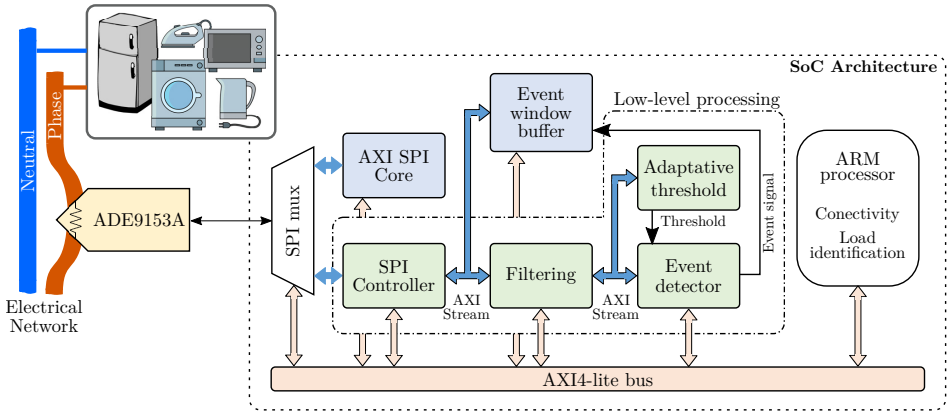

Figure 1: Block diagram of the SoC architecture proposed for the event detection in NILM techniques, based on the ADE9153A.

identification or behavioural pattern recognition, so they can be implemented locally.

This work presents a finite-precision analysis for an event detector [12] implemented in an FPGA-based SoC architecture, based on the Zynq® 7000 family by Xilinx, Inc. The integrated circuit ADE9153A is used as an analog front-end (AFE) to collect the instantaneous voltage and current samples. The AFE is managed by the processor and the programmable logic through a dedicated low-level processing peripheral that processes the incoming samples and detect events at 4 ksamples/s, which is a significant improvement over commercial smart meters and allows developing low-cost prototypes with high sampling rates in order to improve energy disaggregation. This may be key in the later identification of behavioural patterns for enhancing the independent living for elderly. The rest of the manuscript is organized as follows: Section II provides an overview of the SoC architecture proposed for the event detector; Section III deals with the finite-precision analysis for the pre-processing and event detection implemented in the architecture; Section IV presents some experimental results; and, finally, conclusions are discussed in Section V.

2 DESCRIPTION OF THE SOC ARCHITECTURE

Fig. 1 shows a block diagram of the global architecture. The AFE (ADE9153A) is connected to the power grid, acquiring the instantaneous values of voltage and current at 4 ksamples/s. The SoC architecture is connected to the AFE through the SPI link. A first configuration is done with the AXI SPI core by the ARM processor, in which the AFE is self-calibrated and its acquisition module gets ready. Once the device has been configured, the SPI bus control is given to the low-level processing peripheral, where samples are acquired at 4 ksamples/s through the SPI controller. The ADE9153A could be configured to work at lower sampling frequencies, although this option has not been considered here, in order to enable a better load identification capability later. The event detection is based on the current samples collected by the AFE, which are pre-processed and used by the event detector [1].

In (1) and (2) the pre-processing of the instantaneous current signal is described mathematically. In (1) the mean squared $i_{ms}[k]$ is calculated for a interval of W samples from the acquired current signal $i[n]$ without any overlap. The length of the interval W can be configured; in this case a value of 256 samples has been determined, which is a power of two in order to reduce the consumption of hardware resources. Subsequently, in (2) the derivative signal $i_d[k]$ from the previously obtained value $i_{ms}[k]$ is calculated. Note that the resulting value must be squared after the difference has been computed, so that all the values are positive.

$$i_{ms}[k] = \frac{1}{W} \cdot \sum_{j=0}^{W-1} i^2[n-j] \tag{1}$$

$$\sqrt{i_d[k]} = i_{ms}[k] - i_{ms}[k-1] \tag{2}$$

The signal $i_{ms}[k]$ is used to detect events. For this purpose, an adaptive threshold has been defined and is shown in (3), which is calculated for a W_{th} window and applied to the signal $i_d[k]$. The mean squared of the derivative signal is calculated and multiplied by a configurable factor κ, used to adjust the sensitivity of the detector by modifying the threshold value. The final value of k is actually a tradeoff: a high value could discard true events with small amplitude, whereas a low one could generate false events coming from noise. In general terms, it can be experimentally fixed at $\frac{1}{9} \cdot \sqrt{2}$. However, in order to use less logical resources, it may be implemented with additions and displacements.

$$th_{adap}[m] = \kappa \cdot \frac{1}{W_{th}} \cdot \sum_{l=0}^{W_{th}-1} i_{ms}^2[k-l] \tag{3}$$

Note that the threshold is updated every W_{th} samples, which has been defined to 32 samples of $i_d[k]$, which is downsampled by a factor of W. An event is detected when the value $i_d[k]$ exceeds the adaptive threshold $th_{adap}[m]$, what will be notified to the processor via an interrupt line. The event detector has a buffer where it stores a sample window before and after the event that will be used for the appliance identification in the processor.

Table 1: Fixed-point representation applied to the input, the output and the intermediate operations in the event detector.

Current signal pre-processing					Adaptative threshold				
	Range	Word Length	Integer Length	Fraction Length		Range	Word Length	Integer Length	Fraction Length
Input ($i[n]$)	$[-1, 1]$	32	2	30	Input ($i_{ms}[n]$)	$[0, 1]$	32	3	29
Square ($i^2[n]$ and $\sqrt{i_d[k]}$)	$[0, 1]$	32	4	28	Square ($i^2_{ms}[k]$)	$[0, 1]$	32	4	28
Accumulation ($\sum_{j=0}^{W-1}$)	$[0, W]$	32	11	21	Accumulation ($\sum_{l=0}^{W_{th}-1}$)	$[0, W_{th}]$	32	9	23
Right shift ($\frac{1}{W}$)	$[0, 1]$	32	3	29	Right shift $\frac{1}{W_{th}}$)	$[0, 1]$	32	4	28
Difference ($i_{ms}[k]$)	$[-1, 1]$	32	4	28	Right shift (κ)	$[0, 1]$	32	3	29
Output ($i_{ms}[k]$ and $i_d[k]$)	$[0, 1]$	32	4	28	Output ($th_{adap}[m]$)	$[0, 1]$	32	5	27

3 FINITE-PRECISION ANALYSIS FOR THE EVENT DETECTOR

The aim is to design the most efficient architecture possible, taking into account that, apart from the typical fixed-point representation often used in programmable logic, it is possible to implement single-precision floating-point operators at the expense of a larger number of Digital Signal Processing (DSP) cells (basically, a multiplier plus an accumulator). Thus, the final accuracy of the system will be determined by the type of accuracy or finite representation involved, particularly for the input signals. In [17] it is already possible to find an estimation about the resource consumption for common floating-point operators when implemented in FPGAs.

The finite-precision analysis has been carried out by taking into account the features of the used SoC, as well as the features of the AFE previously mentioned. Firstly, The Zynq-7000 SoC incorporates DSP48E1 cells [18] in the FPGA with a word width of 25x18 bits for the multipliers; thus, this is an important consideration for the multiplications carried out in (1) and in (3). Note that divisions are implemented with a right shifting to consume less resources. On the other hand, the ADE9153A circuit provides 32-bit sample via the SPI bus. It has an automatic gain control to adjust the span of the ADC to the maximum range. Therefore, a 32-bit word width is considered, with 1 bit for the integer part and 31 for the fractional part, so the current signal is normalized between ±1.

In order to obtain the maximum accuracy, despite using more resources, in this case the word width of the operations has been adjusted according to the word width of the input signal of 32 bits. Table 1 shows the quantification applied to each of the operations carried out, according with the operations of the event detector described before. It defines the Word Length, as well as the bits dedicated to the Integer Length and Fractional Length, and how they have been distributed for each operation defined in (1) and (2). It is worth noting that, to avoid overflow, the size dedicated to the product is determined by adding the Word Length from both multiplicands (32+32), whereas the length of the sum is one bit longer than the two input operands' length (32+1). It should be mentioned that the number of bits added in the accumulation corresponds to $log_2(W)$, which in this case is 8 for $W = 256$ and 5 for $W_{th} = 32$. Finally, it is necessary to remark that the values shown in Table 1 imply a truncation after the operations.

4 EXPERIMENTAL RESULTS

To validate the finite-precision analysis of the event detector, it has been evaluated using the BLUED database [3] for the current signals in the phases A and B. The accuracy results have been obtained based on the events labelled in the dataset. In this case, since time windows of W samples are evaluated, the time window where an event is labelled in the database as positive has been considered in the same way, whereas the remaining windows where there are no events are considered as negative. Table 2 shows the results obtained in terms of precision, sensitivity and specificity, among others. In the same Table there is a comparison between the proposed finite precision and a simple-precision floating-point solution. In addition, it has been compared with other previous related works.

It is possible to observe that, when using finite precision for event detection, the accuracy values are not directly affected in comparison with the values of accuracy obtained in the simple-precision floating-point proposal. Besides, comparing the results with other previous works, these presented here are very close in terms of precision or F1-score. Nevertheless, in the case of the phase B from the database, the results are slightly worse due to the rapid variations that occur in the current signal, which produce false positives, thus worsening the results slightly compared to phase A.

On the other hand, the specification of the low-level peripheral has been carried out by using the high-level synthesis Vivado HLS tool [19] for the implementation of the pre-processing stage and the event detector, as well as the adaptive threshold shown in (1), (2) and (3). Subsequently, it has been included in the SoC architecture with the rest of the peripherals, such as the SPI controllers and the event windows buffers. With regard to the architecture resource consumption in the Xilinx Zynq-7000 device (xc7z010), the results are shown in Table 3. It is worth noting that the filtering stage and the event detector have a reduced resource consumption. The implementation of the event detector requires the 10 % of the available DSP48e1 cells, whereas it consumes about 3.5 % of the available LUT slices and registers. However, the rest of the architecture mainly requires memory blocks, since there are different buffers to perform the subsequent classification when an event is detected by storing the voltage and current samples captured by the AFE.

Finally, the use of finite-precision operands to carry out the calculations involved in the event detection does not imply a loss in accuracy, whereas it is possible to reduce the consumption of

Table 2: Event detection results for the BLUED database using phase A and B.

	BLUED Phase	Events	Detections	True Positives (TP)	False Positives (FP)	False Negatives (FN)	Threat Score (TS)	Recall (TPR)	F1-Score
Fixed-Point Proposal	A	907	825	809	16	78	0.8959	0.9121	0.9451
Fixed-Point Proposal	B	1578	1521	812	709	850	0.3425	0.4886	0.5102
Floating-Point Proposal	A	907	763	751	12	130	0.841	0.8524	0.9136
Floating-Point Proposal	B	1578	968	777	208	825	0.429	0.485	0.6005
Alcalá [1] Event Detector	A	907	1085	796	289	61	0.6946	0.9288	0.8198
Alcalá [1] Event Detector	B	1578	8458	1170	7288	351	0.1328	0.7682	0.2345

Table 3: Resource consumption of the proposed architecture implemented in a Xilinx xc7x010 FPGA.

Module	Slice LUT	Slice Register	BRAM 36KB	DSP48e1 Cells
Global system	**7860 (45.2%)**	**11123 (32.6%)**	**17 (28%)**	**8 (10%)**
AXI SPI Core	359 (2%)	633 (2%)	0 (0%)	0 (0%)
SPI Multiplexer	62 (1%)	169 (1%)	0 (0%)	0 (0%)
SPI Controller	206 (1%)	406 (1%)	1 (2%)	0 (0%)
Event Detector + Adaptative Threshold	617 (3.5%)	1070 (3.1%)	0 (0%)	8 (10%)
Event Window Buffer	200 (1%)	342 (1%)	16 (26%)	0 (0%)

DSP48E1 cells, compared to using simple-precision floating-point operands. Therefore, this makes feasible to foresee the integration of other stages in the NILM techniques, such as load identification or behaviour recognition, in the the same hardware architecture in future works.

5 CONCLUSIONS

In this work the finite-precision analysis for an event detector in NILM techniques has been described. It has be verified that the results in terms of precision and accuracy for the fixed-point proposal are similar to those obtained for an equivalent the simple-precision floating-point solution. Furthermore, the resource consumption of the architecture is reduced in the fixed-point approach, thus allowing to save hardware resources for further processing or load identification stages to be tackled locally at high sampling frequencies in the same hardware architecture in future works.

ACKNOWLEDGMENTS

This work has been funded by Spanish Ministry of Science, Innovation and Universities (project POM, ref. PID2019-105470RA-C33), Comunidad de Madrid (project CODEUS, ref. CM/JIN/2019-043, and grant ref. PEJD-2019-POST/TIC-16729), and Junta de Comunidades de Castilla-La Mancha (project FrailCheck, ref. SBPLY/17/180501/000392).

REFERENCES

[1] José Manuel Alcalá. 2017. *Non-intrusive load monitoring techniques for activity of daily living recognition.* Ph.D. Dissertation. Universidad de Alcalá.

[2] Jose Manuel Alcala, Jesus Urena, Alvaro Hernandez, and David Gualda. 2017. Sustainable Homecare Monitoring System by Sensing Electricity Data. *IEEE Sensors Journal* 17, 23 (dec 2017), 7741–7749.

[3] Kyle Anderson, Adrian Ocneanu, Diego Benitez, Derrick Carlson, Anthony Rowe, and Mario Berges. 2012. BLUED: A Fully Labeled Public Dataset for Event-Based Non-Intrusive Load Monitoring Research. In *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD).* Beijing, China.

[4] Kyle D. Anderson, Mario E. Berges, Adrian Ocneanu, Diego Benitez, and Jose M.F. Moura. 2012. Event detection for Non Intrusive load monitoring. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society.* IEEE.

[5] Aggelos Bouhouras, Paschalis Gkaidatzis, Konstantinos Chatzisavvas, Evangelos Panagiotou, Nikolaos Poulakis, and Georgios Christoforidis. 2017. Load Signature Formulation for Non-Intrusive Load Monitoring Based on Current Measurements. *Energies* 10, 4 (apr 2017), 538.

[6] Aggelos S. Bouhouras, Paschalis A. Gkaidatzis, Evangelos Panagiotou, Nikolaos Poulakis, and Georgios C. Christoforidis. 2019. A NILM algorithm with enhanced disaggregation scheme under harmonic current vectors. *Energy and Buildings* 183 (jan 2019), 392–407.

[7] European Commission Joint Research Centre. [n.d.]. *European Commission Joint Research Centre.* https://ses.jrc.ec.europa.eu/smart-metering-deployment-european-union

[8] Hsueh-Hsien Chang, Kuo-Lung Lian, Yi-Ching Su, and Wei-Jen Lee. 2014. Power-Spectrum-Based Wavelet Transform for Nonintrusive Demand Monitoring and Load Identification. *IEEE Transactions on Industry Applications* 50, 3 (may 2014), 2081–2089.

[9] Michelle Susan Clark. 2015. *Improving the feasibility of energy disaggregation in very high- and low-rate sampling scenarios.* Ph.D. Dissertation. University of British Columbia.

[10] A. Cominola, K. Nguyen, M. Giuliani, R. A. Stewart, H. R. Maier, and A. Castelletti. 2019. Data Mining to Uncover Heterogeneous Water Use Behaviors From Smart Meter Data. *Water Resources Research* 55, 11 (nov 2019), 9315–9333.

[11] Anwar Haq and Hans-Arno Jacobsen. 2018. Prospects of Appliance-Level Load Monitoring in Off-the-Shelf Energy Monitors: A Technical Review. *Energies* 11, 1 (jan 2018), 189.

[12] Á. Hernández, R. Nieto, D. Fuentes, and J. Ure na. 2020. Design of SoC Architecture for the Local Implementation of NILM Techniques. In *2020 35rd Conference on Design of Circuits and Integrated Systems (DCIS).* 1–5.

[13] William Hurst, Casimiro Aday Curbelo Montanez, Nathan Shone, and Dhiya Al-Jumeily. 2020. An Ensemble Detection Model Using Multinomial Classification of Stochastic Gas Smart Meter Data to Improve Wellbeing Monitoring in Smart Cities. *IEEE Access* 8 (2020), 7877–7898.

[14] Jacob A. Mueller and Jonathan W. Kimball. 2018. Accurate Energy Use Estimation for Nonintrusive Load Monitoring in Systems of Known Devices. *IEEE Transactions on Smart Grid* 9, 4 (jul 2018), 2797–2808.

[15] Hamed Nabizadeh Rafsanjani, Changbum R. Ahn, and Jiayu Chen. 2018. Linking building energy consumption with occupants' energy-consuming behaviors in commercial buildings: Non-intrusive occupant load monitoring (NIOLM). *Energy and Buildings* 172 (aug 2018), 317–327.

[16] Antonio Ruano, Alvaro Hernandez, Jesus Ureña, Maria Ruano, and Juan Garcia. 2019. NILM Techniques for Intelligent Home Energy Management and Ambient Assisted Living: A Review. *Energies* 12, 11 (jun 2019), 2203. https://doi.org/10.3390/en12112203

[17] Noelia Uribe-Pérez, Luis Hernández, David de la Vega, and Itziar Angulo. 2016. State of the Art and Trends Review of Smart Metering in Electricity Grids. *Applied Sciences* 6, 3 (feb 2016), 68.

[18] Xilinx. [n.d.]. *7 Series DSP48E1 Slice (UG479).* https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf

[19] Xilinx. [n.d.]. *Vivado High-level Synthesis (UG902).* https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug902-vivado-high-level-synthesis.pdf