Geosci. Instrum. Method. Data Syst., 1, 23–31, 2012
www.geosci-instrum-method-data-syst.net/1/23/2012/
doi:10.5194/gi-1-23-2012

Open Access Geoscientific Instrumentation Methods and Data Systems

Martian dust devils detector over FPGA

E. de Lucas[1], M. J. Miguel[1], D. Mozos[1], and L. Vázquez[2]

[1]Universidad Complutense de Madrid, Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Madrid, Spain
[2]Universidad Complutense de Madrid, Departamento de Matemática Aplicada, Facultad de Informática, Madrid, Spain

Correspondence to: E. de Lucas (enriquedelucas@pas.ucm.es)

Received: 1 November 2011 – Published in Geosci. Instrum. Method. Data Syst. Discuss.: 1 December 2011
Revised: 14 March 2012 – Accepted: 23 March 2012 – Published: 20 April 2012

Abstract. Digital applications that must be on-board space missions must comply with a very restrictive set of requirements. These include energy efficiency, small volume and weight, robustness and high performance. Moreover, these circuits cannot be repaired in case of error, so they must be reliable or provide some way to recover from errors. These features make reconfigurable hardware (FPGAs, Field Programmable Gate Arrays) a very suitable technology to be used in space missions. This paper presents a Martian dust devil detector implemented on an FPGA. The results show that a hardware implementation of the algorithm presents very good numbers in terms of performance compared with the software version. Moreover, as the amount of time needed to perform all the computations on the reconfigurable hardware is small, this hardware can be used most of the time to realize other applications.

1 Introduction

Digital systems for space applications have some special requirements not needed on normal systems. These requirements include energy/power efficiency, small volume, robustness and high performance as well as resistance under extreme conditions of pressure, temperature fluctuations, radiation and mechanical shock. Traditionally, this has been achieved using embedded microprocessors and dedicated hardware peripherals. But many times the performance of these devices is not enough for the applications. Moreover, many applications have a high degree of internal parallelism that it is not used to improve the performance.

In general, hardware-based solutions provide better performance and less energy consumption than purely software solutions, since they eliminate the overhead due to instruction decoding and they include optimized hardware for the requested operations instead of carrying out those operations, executing a sequence of predefined generic instructions. However, due to the exigent volume restrictions and the complex and expensive design process, it is not always feasible to implement all the needed functionalities of an embedded system using only a hardware solution based on Application Specific Integrated Circuits (ASICs). Nevertheless, since embedded systems are targeting more and more complex applications, it is not likely that a software-based solution will achieve the requested performance. Hence, hardware accelerators for the most complex tasks are needed.

During the last 20 yr, the hardware reconfigurable technology has evolved from small uniform devices, able to implement small circuits and are statically reconfigurable, to very huge heterogeneous devices with capacity to be dynamically reconfigured. This evolution has produced FPGAs (the most popular type of reconfigurable hardware) which are used in a great number of applications, automotive circuits, digital image processing, video games, etc.

Although FPGAs are very promising to be used in space applications, they have several problems that must be fixed in the future in order to be a really practical solution. The main problem seems to be that the reprogramming feature of the current FPGAs is more sensitive to radiation than the programmed device and might fail at a much lower total radiation dose than stated by the manufacturers for the programmed device. So, more research must be done in this issue in order to get FPGAs totally reliable.

In the field of space applications, one additional requirement is that all the hardware sent in space missions must be certified for space operation. This is because space-based

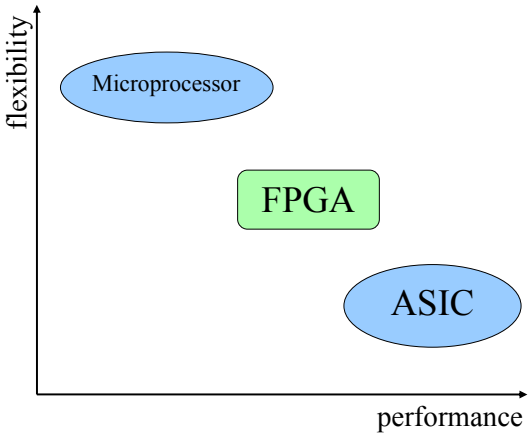


Fig. 1. Trade-off flexibility performance for different technologies.

systems must operate in an environment in which radiation effects have an adverse impact on integrated circuit operation (Thomson, 2005). Ionizing radiation can cause soft errors in the static cells used to hold the configuration data. This will affect the circuit functionality and ultimately result in system failure. This requires special FPGAs that provide on-chip reconfiguration error detection and/or correction circuitry. High-speed, radiation-hardened FPGA chips with million gate densities have recently emerged to support the high throughput requirements for space applications. In fact, radiation-hardened FPGAs are in great demand for military and space applications. For instance, companies such as Actel Corporation or Xilinx have been producing radiation-tolerant anti-fuse FPGAs for several years, intended for high-reliability space-flight systems. Actel FPGAs have been on-board more than 100 launches and Xilinx FPGAs have been used in more than 50 missions (for instance Xilinx, 2003a,b and Graham et al., 1999).

FPGAs are an intermediate solution between software developments (highly flexible, but power consuming and with performance problems) and ASICS (static applications, with good numbers in performance and power consumption), as we can see in Fig. 1.

FPGAs are now fully reconfigurable (DeHon and Wawrzynek, 2009; Hauck and DeHon, 2008), a technological feature that allows a control station on Earth to adaptively select a data processing algorithm (out of a pool of available algorithms implemented on the FPGA) to be applied on-board. The idea is that FPGAs can be reconfigured on the fly. This approach is called run-time reconfiguration (Resano et al., 2008). Basically, the FPGA (or a region of the FPGA) executes a series of tasks one after another by reconfiguring itself between tasks. The reconfiguration process updates the functionality implemented in the FPGA, and a new task can then be executed. This time-multiplexing approach supports the reduction of hardware components on-board since one single reconfigurable module can substitute several hardware peripherals, carrying out different functions during different phases of the mission.

The flexibility provided by reconfigurable hardware can also be used to modify the functionality of the satellite instrumentation during the flight, or to automatically recover the system from malfunction. Moreover, the hardware design-cycle for FPGAs is much shorter than the one for custom integrated circuits, mainly because the design can be tested on the target platform since the first steps of the design process, thus, avoiding a complex chip fabrication process. Nevertheless, in space applications always exists an additional cost of the designed circuits dedicated to verify the quality requirements of the instrumentation.

In this paper we are going to present a space application that shows that the use of FPGAs is a good alternative in this kind of situations. The application is used to detect dust devils on the Martian surface. A dust devil is a hot whirlwind generated by a huge contrast between the atmospheric air and the surface in contact. This phenomenon has been studied on Earth and Mars (Renno et al., 2000) and it is well-known. We have to be able to detect dust devils before they happen due to their destructive potential. This weather phenomenon can cause great damage to instrumentation and human beings. It is known that some big dust devils, about 1 km of height, have been reported. On Mars some dust devils have been identified with a height over 10 km. Apart from that, the friction caused by suspended particles brings a huge static load over dangerous levels.

The dust devils are detected as sudden changes in temperature and pressure. The need to have an automatic system that detects these phenomenon is due to the distance between Earth and Mars. The key to propose a dust devil detector over an FPGA is that (1) an application like this is cheap in terms of space in an FPGA, (2) we can implement a lot of different applications in the FPGA at the same time, (3) we can reprogram it if we need to and (4) there are FPGAs tested for space for other applications.

2 System description

To develop the design many ideas have been studied. We have studied several approaches to the STA/LTA algorithm and have implemented a software version in order to obtain a first approach to the solution. STA and LTA mean Short Time Average and Long Time Average, respectively. The idea about having STA and LTA is that STA represents the recent changes and LTA represents the long time changes in some parameters. If we detect changes between the behavior of the temperature and pressure parameters in the short and large time, we will declare an event that could be a dust devil. In order to achieve a better realism, we used real data from Mars Pathfinder (PF) mission to test our design. Once the algorithm has been validated in software, we implemented a

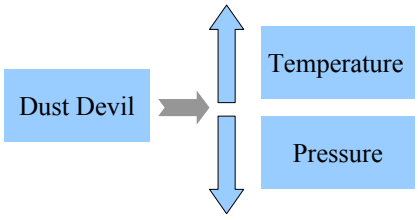


Fig. 2. A dust devil implies a temperature rise and a pressure drop.

hardware design in an FPGA and a communication strategy between PC and FPGA.

2.1 Input data

In order to recreate appropriately the Martian environment, we have used real data from the experiments in Mars PF mission (Schofield, 1997). In 1997 this mission, which included the PF lander and a surface rover, the Sojourner completed a successful landing on Mars. This mission recovered data from the red planet for more than 3 months.

The sensors of the PF lander (Seiffet et al., 1997) measured pressure at one height and temperature at three different heights (0.25 m, 0.5 m and 1 m over the surface). The PF lander also had a wind sensor (1.1 m over the surface) but it experienced problems and its data are not available in the Planetary Data System web page. The nominal period between two samples was 4 s and the minimum period was 1 s for periods no longer than 1 h.

Data from this mission can be accessed by anybody in NASA Planetary Data System web page NASA PDS (1997), where we can find Mars PF mission data in a tabulated text plain format containing sensor lectures and date information. During the mission was detected certain number of dust devils (Renno et al., 2000) that we used to test our system.

For our interest and with the purpose of dust devil detection, we needed to study data from temperature and atmospheric pressure, because these are the variables involved when a dust devil runs over a surface. Specifically, a dust devil produces a temperature rise and a pressure drop (see Fig. 2). In order to minimize the temperature error induced for the heat of the PF lander, we only take into account the temperature data of the higher temperature sensor.

2.2 Algorithm

STA/LTA algorithms were developed for the first time by Lee and Stewart (1981) and therefore were improved for seismic detection by Allen (1982). These algorithms are based on two sample sets: a short time set (STA) and long time set (LTA).

With these two sample sets, we can do different operations to achieve the current value (CV) and predicted value (PV) results of our algorithm, which are compared with each other and if we get significant changes between them with respect to a threshold value (THR), we will report an event (see Fig. 3).

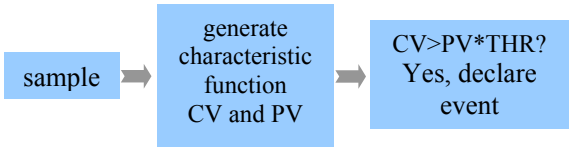


Fig. 3. Data flow diagram STA/LTA algorithm.

2.3 Software approximation

Different possibilities have been studied with software built around STA/LTA algorithms and their configuration parameters (number of STA and LTA samples and THR value). Once tested, they were implemented on an FPGA. To execute the variants of these algorithms, these are needed (1) a suitable storage structure for STA and LTA sets, (2) a specific implementation of primary characteristic function (PCF) and (3) the definition of a THR value and its associated characteristic function.

The STA and LTA sets have a finite and static number of samples. We have decided to use a circular array to handle the refreshing of the STA and LTA sets efficiently. That circular array, which stores the samples as we see in Fig. 4, is a contiguous memory zone with two more variables to manage it: (a) an auxiliary pointer, that points to a position in memory where is stored the oldest element of the array; and (b) an integer, which represents the occupation level of the array.

The PCF represents the operations made with the samples. Usually, the most used operation is the arithmetic average of each set, but we can make more complex calculations. Other usual functions are root mean square and dispersion.

Finally, we have the THR concept and its associated characteristic function (ACF). In our case we want to detect rise and descent events on temperature and pressure values. For that we have a rising THR (THR_R) and a descending THR (THR_D). To detect variations between STA and LTA sets, we realize a division between STA average (STA_{AVG}) and LTA average (LTA_{AVG}) and then we compare its result with THR_U and THR_D, as we show in Fig. 5. STA_{AVG} is equivalent to CV and LTA_{AVG} is equivalent to PV.

2.4 Hardware

Once we tested the different variants of STA/LTA algorithms, we implemented the hardware version. We used a Virtex II Pro to develop our designs and all the logic was implemented with CLBs (Configurable Logic Blocks)(see in Fig. 6 the board used to realize this implementation). Each CLB contains four slices (Xilinx, 2007). In a Virtex II the time used to reconfigure one fifth of the total reconfigurable area is around 4 ms.

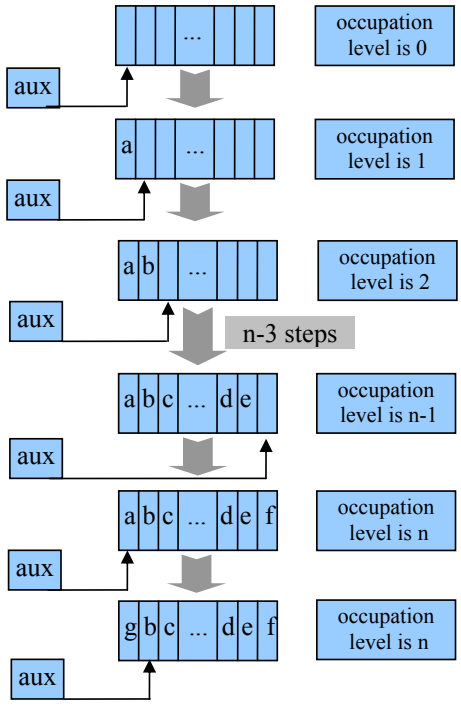


Fig. 4. Circular array occupation.

2.4.1 Hardware improvements

With a hardware design we can achieve a better performance than with a software design for several reasons.

A hardware design in an FPGA allows a high degree of parallelism. We can implement a system with a module for each magnitude without overhead on time execution. Also, with this alternative, we can have different modules with arithmetic average, root mean square, dispersion or any other functions as we want.

For all hardware algorithms we have taken the following design decisions:

1. We work with fixed point numbers. The reason is that the hardware required to execute integer division is not as expensive as float point division hardware. So we send data to the FPGA in fixed point format. Also, the studied magnitudes such as pressure and temperature are presented in millibar and kelvin units, respectively, which allows us to work with unsigned integers. The use of integer arithmetic is not always possible; in most situations it will be necessary to work with floating point numbers.

2. We have defined a four byte-wide data for each in-sample and return value.

3. In order to achieve a better performance, we have implemented a circular FIFO RAM to access and compute data samples. This FIFO RAM has an integer occupation variable and a least recently used pointer to run.

4. As we said we have two sample sets, STA and LTA, being STA the set used to compute STA_{AVG} (STA average) and LTA used to compute LTA_{AVG} (LTA average). In order to calculate efficiently the arithmetic average and the root mean square, we introduce an integer variable for each set, which represents the sum of all values in each set. So when we introduce a new sample, we do not have to sum up all samples again; we only have to subtract the least recently used sample from this integer variable and add the new sample obtaining the sum of all samples at O(1) time,

$$\begin{aligned} &\frac{a_1 + a_2 + \ldots + a_n}{n} \\ &\frac{a_1 + a_2 + \ldots + a_n}{n} + \frac{a_{n+1} - a_1}{n} \\ &\frac{a_2 + \ldots + a_n + a_{n+1}}{n} \end{aligned} \tag{1}$$

5. Eventually, we studied the viability of change integer division by binary right shift. Each right shift of an integer represents a division by two; so if we can utilize this replacement, the division would be done in a single clock cycle instead of several. This produces an important reduction in area and execution time.

2.4.2 Hardware communication

The data to be analyzed was sent from a PC to the FPGA. This communication was performed through a serial port using RS232 protocol at 115 200 baud rate without hardware flow control, one stop bit and no parity. The reason to use this method is that we do not need a system that computes a huge amount of data in the minimum possible time; we need a system that computes a single piece of data in the minimum number of cycles, because the maximum frequency of new data is 1 Hz. So this strategy (see Fig. 7) is enough for our requirements. We have defined not only a sample four bytes frame to send data to the FPGA, but also for test purposes we have defined a four bytes return frame that contains execution information.

2.4.3 Arithmetic average version

This algorithm calculates STA_{AVG} and LTA_{AVG} (see Fig. 9), being the PCF the arithmetic average, and then executes the ACF (Associated Characteristic Function), which obtains the result of the algorithm, i.e. declares or does not declare an event. The core ACF is composed by a division between $STA_{AVG} \cdot 100$ and LTA_{AVG} (see Fig. 8). Then, we compare this result with THR_R and THR_D and we declare the event as appropriate or not. The reason to multiply STA_{AVG} by one hundred is that we are using fixed point arithmetic and we want two digit precision to compare with the THRs.

About cycles of execution we have to comment that when we receive a new sample until we finish the execution for that

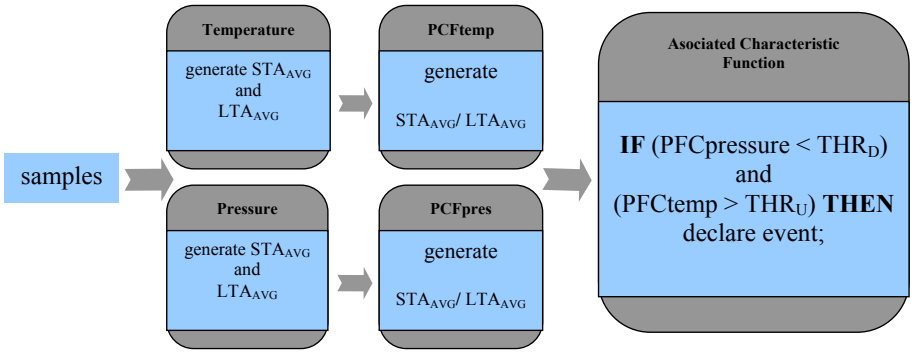


Fig. 5. Data flow diagram dust devils detector.

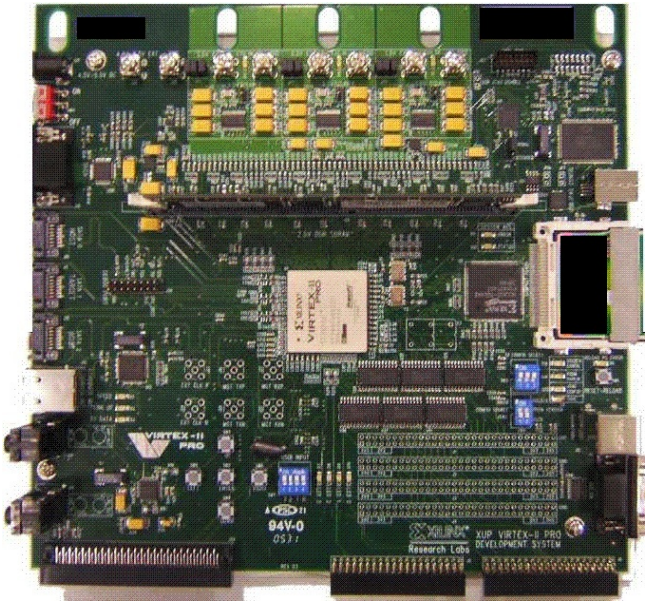


Fig. 6. XUPV2P30 Development System.

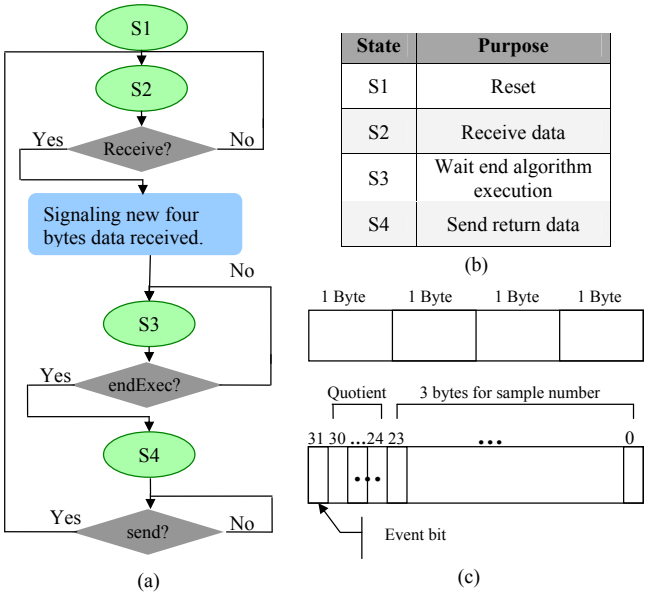


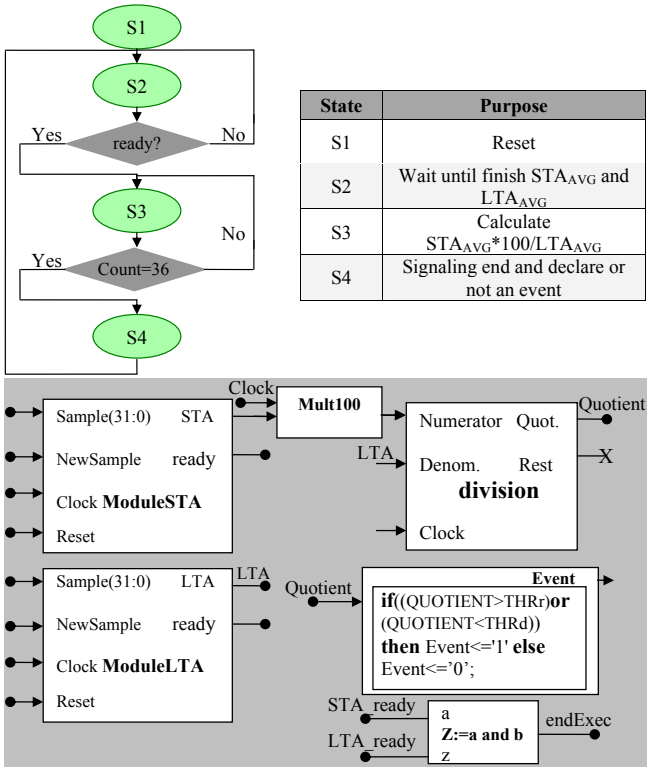
Fig. 7. (a) Data flow diagram of system's communication. **(b)** Table of states. **(c)** Frame sample and return value format.

sample, we consume 79 clock cycles. A big amount of those cycles are due to the two divisions of the algorithm (36 clock cycles per division): one division to calculate STA and LTA averages and other division to calculate STA average and LTA average quotient. So if we are able to eliminate these divisions, we would accelerate rather the whole algorithm. We cannot eliminate the second division, because we cannot guarantee that LTA_{AVG} is a power of two, but the number of samples of STA and LTA sets can be a power of two; it depends on if we are able to achieve configuration parameters with this characteristic and if that fulfills our requirements.

Eventually, we have to speak about FPGA utilization. The final report provided by Xilinx shows us an utilization of 29 % of slices, with a number of samples of 64 for the STA set and 128 for the LTA set.

2.4.4 Arithmetic average optimized version

As we said if we can achieve a configuration with a number of samples power of two by each sample set, we can replace the division by a right shift. This shift will be of x positions, being $x = \log_2$(number of samples). The algorithm is similar to the original algorithm, being the only difference the substitution of the divider by a right shifter. Due to this right shift, it will be necessary a new initial stage to initialize the content of the FIFO RAMs before we start to divide. Only when we have filled the FIFO RAMs, we can start to divide; otherwise, we will obtain several incorrect initial values (up to the number of LTA samples). With this variation we save 36 clock cycles. The final report provided by Xilinx shows us an utilization of 11 % (instead 29 %) of slices, with a number of samples of 64 for the STA set and 128 for the LTA set.



State	Purpose
S1	Reset
S2	Wait until finish STA_{AVG} and LTA_{AVG}
S3	Calculate $STA_{AVG}*100/LTA_{AVG}$
S4	Signaling end and declare or not an event

Fig. 8. (a) Data flow diagram of arithmetic average version. **(b)** Table of states. **(c)** Interconnection.

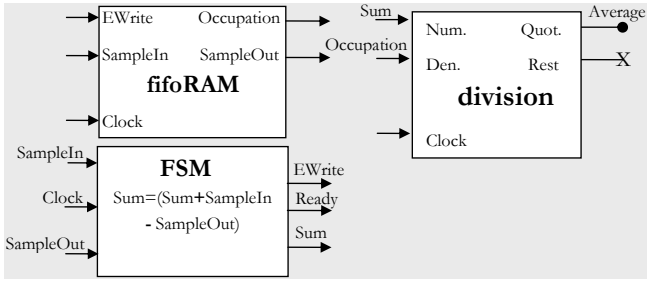


Fig. 9. Core arithmetic average version.

2.4.5 Root mean square version

This variation of the algorithm calculates the root mean square of the samples. When we receive the new sample, we calculate the square of it and this is what we send to the FIFO RAM of STA and LTA modules (see Fig. 10). The rest of the algorithm is the same as the arithmetic average algorithm. As we have to calculate the square of each sample, we have introduced a multiplier in the design.

Each multiplier adds 5 clock cycles in parallel with execution time and due to that we need to store the new sample; to calculate the square we add one more clock cycle. Then, we have an algorithm that can compute a new sample each $79 + 5 + 1 = 85$ clock cycles.

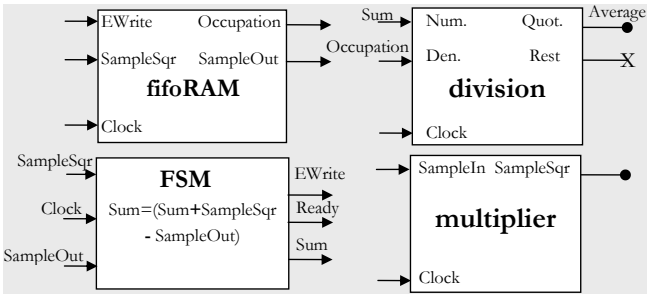


Fig. 10. Core root mean square version.

Table 1. Algorithm parameters.

Parameter	Pressure	Temperature
No STA samples	8	4
No LTA samples	1024	32
THR_R	0.997	–
THR_D	–	1.01

Regarding the FPGA occupation, now, we have two new 32-bit multipliers so the size of the design grows significantly. Each multiplier occupies 1088 LUTs of the FPGA. Due to the fact that we have similar results with this algorithm and the first algorithm, we have decided not to implement it in hardware.

2.4.6 Dust devils detector version

This algorithm consists of two copies of the first algorithm (arithmetic average algorithm), one for each magnitude to analyze, in our case a module for pressure and the other for temperature. The difference to the first algorithm is the final part of the ACF, which is the condition to determine if we signal an event or not. In our study of the data with the software version of this algorithm, we obtained the configuration parameters contained in Table 1. These parameters are the number of samples for each set for pressure and temperature and the values for THR_R and THR_D.

As we can see we achieved, for a number of samples, power of two for each sample set, so the dust devil detector will be implemented with the optimized version of the first algorithm. So we have the same number of clock cycles as the first algorithm: 79 for normal algorithm and 43 for the optimized version. Regarding the FPGA occupation, we obtained a rate of 59 % of slices for the normal version and 23 % for the optimized version. The number of samples of each set could change dynamically depending on the evolution of the algorithm, using larger sets when it is necessary more sensitivity.

Table 2. Hardware and Software comparative.

Arithmetic Average			
	Software	Hardware*	Optimized Hardware*
Samples freq. (Hz)	94 931.0070	1 327 311.9241	2 383 127.2727
Gain	1.0	13.9819	25.1037
Dust Devils Detector			
	Software	Hardware*	Optimized Hardware*
Samples freq. (Hz)	91 819.5936	1 327 311.9241	2 383 127.2727
Gain	1.0	14.4562	25.9543

* Results obtained supposing that we have a new data every time we need it.

Table 3. Test 1. Dust devils detected each sol.

Algorithm parameters Test 1		
Parameter	Pressure	Temperature
No STA samples	8	4
No LTA samples	1024	32
THR_R	0.997	–
THR_D	–	1.01
Results for Test 1		
Sol	Local Hour	Detected
25	13:10	Yes
25	13:53	Yes
34	09:52	No
34	11:32	Yes
34	11:38	Yes
38	12:32	Yes
39	11:31	No
39	13:47	Yes
49	11:02	Yes
52	12:03	No
55	14:19	Yes
60	10:09	No
62	12:31	No
62	12:34	Yes
62	14:06	Yes
68	11:42	No
68	13:29	Yes
69	12:54	Yes
70	14:25	Yes

Table 4. Test 2. Dust devils detected each sol.

Algorithm parameters Test 2		
Parameter	Pressure	Temperature
No STA samples	4	4
No LTA samples	1024	32
THR_R	0.998	–
THR_D	–	1.01
Results Test 2		
Sol	Local Hour	Detected
25	13:10	Yes
25	13:53	Yes
34	09:52	No
34	11:32	Yes
34	11:38	Yes
38	12:32	Yes
39	11:31	Yes
39	13:47	Yes
49	11:02	Yes
52	12:03	No
55	14:19	Yes
60	10:09	No
62	12:31	Yes
62	12:34	Yes
62	14:06	Yes
68	11:42	No
68	13:29	Yes
69	12:54	Yes
70	14:25	Yes

3 Results

In this section we show the performance of the dust devil detector with the data provided and also we comment on the pros of the hardware system versus a software system.

The data provided from Mars PF mission is a suitable data set to test the algorithms implemented in this work since in that mission were detected several dust devils. Using that data we have tuned our dust devils detector. The tuning process was done with the software alternative and once some suitable parameters were achieved, we incorporated these in

Table 5. Dust devils detected.

Test	Number of dust devils detected
Test 1	13 (of 19)
Test 2	15 (of 19)

the hardware system, obtaining the results shown in Tables 3 and 4.

We can see that when we improve the pressure variations sensitivity, we have an increment in the number of dust devil detections (Table 5), but we also increment the number of possible detections that are not reported in the original data as dust devils.

The software algorithm was run over an Intel Q8200 Core2 Quad, with 2.33 Ghz frequency, 4 GB RAM at 1333 Mhz and Ubuntu 10.04LTS operative system. The FPGA clock used in this comparative has a frequency of 100 Mhz. Table 2 shows us the number of samples per second that the hardware and software algorithms can run. We observed that the performance of the hardware of the not optimized version is about 15 times higher than the software algorithm performance and the performance of the hardware of the optimized version is about 25 times higher. Also, we obtain a decrease of about 61 % in area with the hardware optimized version.

4 Conclusions

As we said, many applications have a high degree of internal parallelism, which cannot be used by traditional software solutions to improve the performance. A hardware design in an FPGA allows us to use that high degree of internal parallelism, given that we can implement an algorithm with an independent module for each magnitude; even more, we can implement different algorithms and all without overhead on execution time. Another relevant aspect of the use of FPGAs is that the flexibility provided by reconfigurable hardware can be used to modify the functionality of the satellite instrumentation during the flight or to automatically recover the system from malfunction.

The results show that the hardware implementation of the algorithm presents very good numbers in terms of performance compared with the software version (up to 25 times higher). Moreover, as the amount of time needed to perform all the computations on the reconfigurable hardware is small, this hardware can be used most of the time to realize other applications. With these results and keeping in mind the difference between FPGA and PC clocks frequency, we can conclude that the use of FPGAs to implement these algorithms is a better alternative.

Future work

In a real scenario the dust devils detector should be used to increment or decrease the measuring frequency of sensors. This would allow a system to not lose events and to preserve power consumption. We would not need to have a high measuring frequency all the time to record all the events, but we would only need a minimum initial measuring frequency and capacity to modify that frequency, in order to avoid the loss of events. Also, in order to minimize the false positives in the detection of events, we could add to the system information about the normal values of the parameters involved in the detection. This will be possible as soon as we have an accurate characterization of the Martian surface layer (Martínez et al., 2008).

Acknowledgements. This research was supported by "Participación en la Misión a Marte Meiga-Metnet-Precursor AYA 2009-14212-C05-05/ESP", TIN2009-09806 and AYA2009-13300. We want to thank Germán Martínez and Carlos González for their scientific and technic support.

Edited by: M. Genzer

References

Allen, R.: Automatic phase pickers: their present use and future prospects, B. Seismol. Soc. Am., 72, 225–242, 1982.

DeHon, A. and Wawrzynek, J.: Reconfigurable computing: what, why, and implications for design automation. Proceedings of the IEEE/ACM Design Automation Conference, 610–615, 2009.

Graham, C., Petkovik, M., Russell, S., Seumahu, E. S., and Vesely, M.: The FedSat Microsatellite, Proceedings. 2nd Intl. Conf. on Information, Communications & Signal Processing (ICICS'99), Singapore, December 1999.

Hauck, S. and DeHon, A. (Eds.): Reconfigurable computing: The theory and practice of FPGA-Based computation, Morgan Kaufmann, 2008.

Lee, W. and Stewart, S.: Principles and applications of microearthquake networks, Adv. Geophys., Supplement 2, Academic Press, New York, USA, 293 pp., 1981.

Martinez, G., Valero, F., and Vazquez, L.: Characterization of the Martian Surface Layer, J. Atmos. Sci., 66, 187–198, 2008.

Meiga-Metnet: The next generation lander mission for martian atmosphere science, http://meiga-metnet.org, last access: 30 June 2010, 2000.

NASA PDS: The planetary Atmospheres Data Node: Mars Pathfinder Mission, http://pds-atmospheres.nmsu.edu/cgi-bin/getdir.pl?dir=index\&volume=mpam_0001, last access: 24 February 2010, 1997.

Renno, N. O., Nash, A. A., Lunine, J., and Murphy, J.: Martian and terrestrial dust devils: Test of a scaling theory using Pathfinder data, J. Geophys. Res., 105, 1859–1865, 2000.

Resano, J., Clemente, J. A., Gonzalez, C., Mozos, D., and Catthoor, F.: Efficiently scheduling runtime reconfigurations, ACM T. Des. Automat. El., 13, 58–69, 2008.

Schofield, J. T., Barnes, J. R., Crisp, D., Haberle, R. M., Larsen, S., Magalhaes, J. A., Murphy, J. R., Seiff, A., and Wilson,

G.: The Mars Pathfinder Atmospheric Structure Investigation/Meteorology (ASI/MET) Experiment, Science, 278, 1752, doi:10.1126/science.278.5344.1752, 1997.

Seiff, A., Tillman, J. E., Murphy, J. R., Schofield, J. T., Crisp, D., Barnes, J. R., LaBaw, C., Mahoney, C., Mihalov, J. D., Wilson, G. R., and Haberle, R.: The atmosphere structure and meteorology instrument on the Mars Pathfinder lander, J. Geophys. Res., 102, 4045–4056, doi:10.1029/96JE03320, 1997.

Thomson, J. T.: Rad Hard FPGAs, available at: http://esl.eng.ohio-state.edu/~rstheory/iip/RadHardFPGA.doc, 2005.

Xilinx: FPGAs aboard Mars 2003 Exploration Mission, http://www.xilinx.com/prs_rls/design_win/03104mars.htm, last access: 21 January 2010, 2003a.

Xilinx: Raytheon OPTUS space program, http://www.xilinx.com/prs_rls/design_win/03135optus.htm, last access: 25 January 2010, 2003b.

Xilinx: Virtex II Pro and Virtex II Pro X Platform FPGAs: Complete Data Sheet, http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf, last access: 24 January 2010, 2007.