IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 3, Issue 3 (Sep. – Oct. 2013), PP 53-60
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

An Efficient Median Filter in a Robot Sensor Soft IP-Core

Liberty Mutauranwa [1], Golden Kapungu [2]
[1](*Department of Electrical Engineering, University of Zimbabwe, Zimbabwe)*
[2](*Department of Electrical Engineering, University of Zimbabwe, Zimbabwe)*

Abstract: *The design and development of a digital median filter is presented as part of a soft ip-core for an autonomous mobile robot. The median filter filters the output of the measurement module of a line sensor ip-core for the robot. The bubble-sort network architecture is adopted for the median filter design. The effectiveness of the algorithm is verified by Matlab programming. The algorithm is implemented in hardware on a Xilinx Spartan-3 FPGA device as part of the robot's line-sensor ip-core.*
Keywords : *Bubble-sort, FPGA, IP-Core, Median filter, System-on-Chip, VHDL*

I. INTRODUCTION

The work described here is part of a System on Chip solution to the development of an autonomous mobile robot. An important part of the autonomous robot is the sensor network used for line following applications. It is imperative that the data obtained from such sensors is accurate and noise free. As such, an appropriate filtering scheme is relevant for data obtained from such sensor networks. One such scheme is median filtering.

A median filter is a non-linear digital filter widely used in image and signal processing for smoothing of signals, suppression of impulse noise and edge preservation [1]. In median filtering a window of size N = 2M + 1 is slid along the entire signal sequence and the centre value of each window is replaced by the median of the values in the window. In 1 – D filtering the window Wi = {xi-M…xi…xi +M,}; is centred on the ith input value; the filter output is yi = median (Wi). The resulting filtered sequence tends to follow the polynomial trends in the original sample sequence while sharp discontinuities of short duration are filtered out [2].

Three possible architectures for median filtering are Array, Bubble – Sort and Stack – Based architectures [3]. In Array architectures each element of the window is associated with the rank, and with each window shift, ranks are updated. Array architectures therefore consist of N processors but have a long sampling period due to word level comparison. In Bubble – Sort architectures samples are first ranked and then the sample of middle rank is selected. They are pipelined implying a higher throughput. Stack – Based Architectures translate filtering into binary domain. Stack – Based architectures provide reasonable performance with a limited modularity.

The major problem of the median filter is its high computational cost. Therefore, real-time median filters are traditionally implemented in hardware as in [4], [5], [6], [7] and [8]. The complexity and computational processing required makes application of median filters to a real-time robot sensor system a challenge.

In this paper, we present an efficient design of median-filter hardware as part of a soft IP-core for optical reflective sensors on a robot. The median filter designed is capable of delivering the median value in real-time under stringent requirements on area and latency.

II. THE GECKO3 ROBOT SYSTEM ON CHIP PROJECT

1.1 The Gecko3 Robot

The Gecko3 is a general purpose hardware and software co-design platform for System on Chip and real-time information processing applications [9]. Fig. 1 shows the Gecko3 robot.

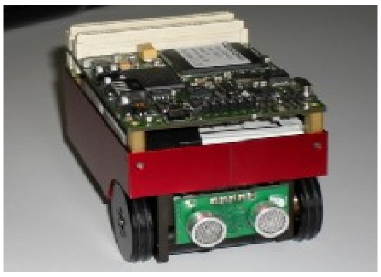

Fig 1. The Gecko3 Robot

It is in the form of a credit card size robot that is rich with sensors and actuators. An on-board FPGA performs all the real-time processing and control tasks.

1.2 The System-on-Chip Gecko3 Application

The broader goal of the project was the development of the Gecko3 into an autonomous mobile robot as a System on Chip solution on the on-board Xilinx Spartan-3 FPGA. The Microblaze RISC processor was instantiated as the microprocessor core for the application and soft ip cores were developed for the sensor and actuator hardware for the robot. Fig. 2 illustrates the Gecko3 System on Chip solution.

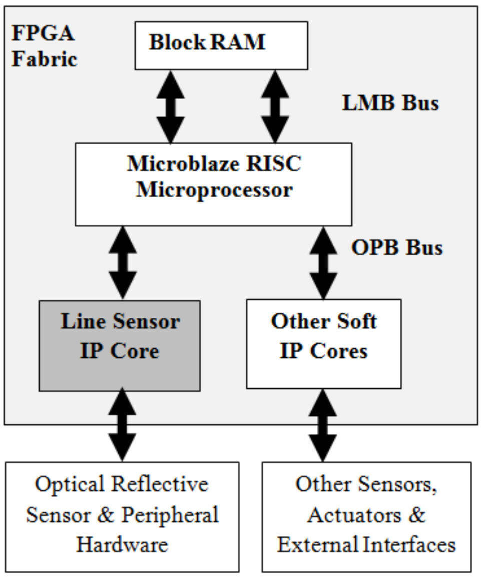
FPGA Fabric
Block RAM
LMB Bus
Microblaze RISC Microprocessor
OPB Bus
Line Sensor IP Core
Other Soft IP Cores
Optical Reflective Sensor & Peripheral Hardware
Other Sensors, Actuators & External Interfaces

Fig 2. The Gecko3 System on Chip System showing the Line Sensor IP- Core

1.3 The Line Sensor IP-Core

Two of the robot's sensors are the Optical Reflective Sensors – a pair of sensors underneath the robot used for line following applications. Each sensor comprises of an infra-red emitter and pin photodiode. The emitter emits infra-red radiation onto the surface over which it is placed, whilst the photodiode detects the infra-red reflection off it. The photodiode outputs a current signal that is a measure of the colour or shade of the surface [10]. A Delta Sigma ADC converts the current signal into a signal that can be processed digitally [11]. Fig.3 shows the picture of the Optical Reflective Sensors underneath the robot.

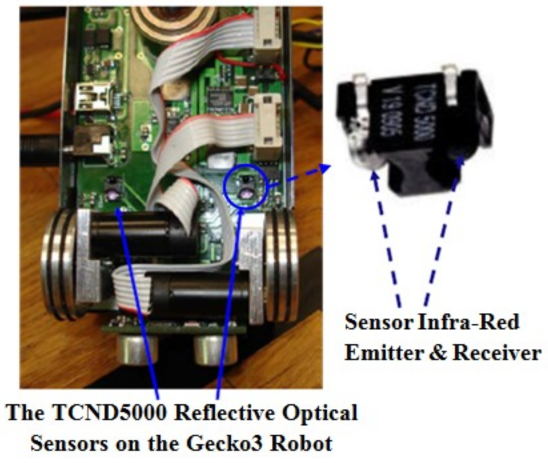
Sensor Infra-Red Emitter & Receiver
The TCND5000 Reflective Optical Sensors on the Gecko3 Robot

Fig 3. The Reflective Optical Sensors underneath the Robot

The IP – Core developed converts the output of the ADC into a decimal value that is a representation of the shade of the surface over which it is placed. The requirement for the developed soft ip-core for these sensors was to enable the use of the sensors to accurately distinguish different grayscale shades [12]. To achieve this, the design of the IP Core, here referred to as the Line Sensor IP-Core, was divided into two modules – a Measurement Module (that would detect the shade over which the sensor is placed) and a Filter Module that would filter the output of the Measurement Module. Fig. 4 shows the concept of the Line Sensor IP-Core and its connection to external sensor hardware.

This paper discusses the design, development and implementation of a median filter as the filter module of the Line Sensor IP Core.

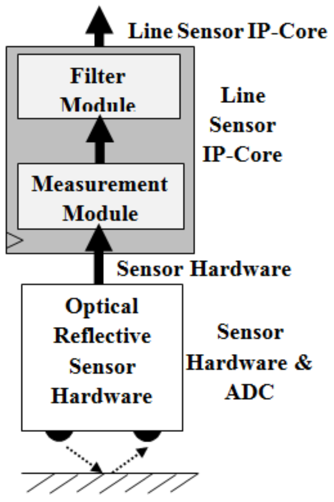

Fig 4. The Line Sensor IP Core Design Concept

III. The Filter Module

3.1 Filtering Requirements

The measurement module outputs an unsigned decimal value that relates to the shade of the surface over which the sensor is positioned. The module is configurable for its bit resolution output with bit resolution configuration varying from 4 bit to 10 bit. The requirement was for the module to enable the sensor ip-core to distinguish different grayscale shades accurately for all bit resolution configurations. Outputs of the measurement module displayed that although it distinguished different grayscale shades, there was still evidence of impulse noise for all the bit resolution configurations.

An appropriate filter was required to remove this noise. The selected filter had to adequately filter the data for all bit resolutions without imposing excessive timing and hardware constraints on the IP-Core. The median filter was selected as the appropriate filter. A software algorithm was developed in Matlab to verify this. Simulations on data obtained from the measurement module gave various values for the median filter's minimum order for the different bit resolution configurations. The minimum median filter order requirements for all the bit resolution configurations are given in Table I.

TABLE I.

TABLE II. MEDIAN FILTER MINIMUM ORDER REQUIREMENTS (MATLAB)

Bit Resolution	4	5	6	7	8	9	10
Minimum Median Filter Order	9	7	7	5	3	3	3

IV. The Median Filter Design & Implementation

4.1 The Algorithm Architecture

The Bubble-Sort Architecture was adopted as it offers a highly modular reusable design structure, limited hardware design complexity and a high throughput. The adopted Bubble-Sort Architecture structure is illustrated in Fig. 5.

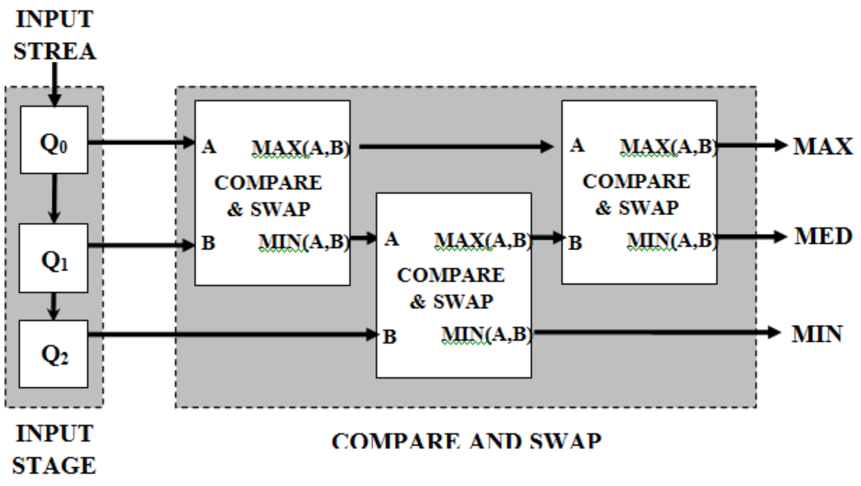

Fig 5. The Basic Structure of a Bubble-Sort Median Filter Architecture (Order 3)

4.2 Median Filter Design

A Finite State Machine – Datapath [13] design approach was adopted for the realization of the median filter. The high level interface block diagram for this design is illustrated in Fig. 6.

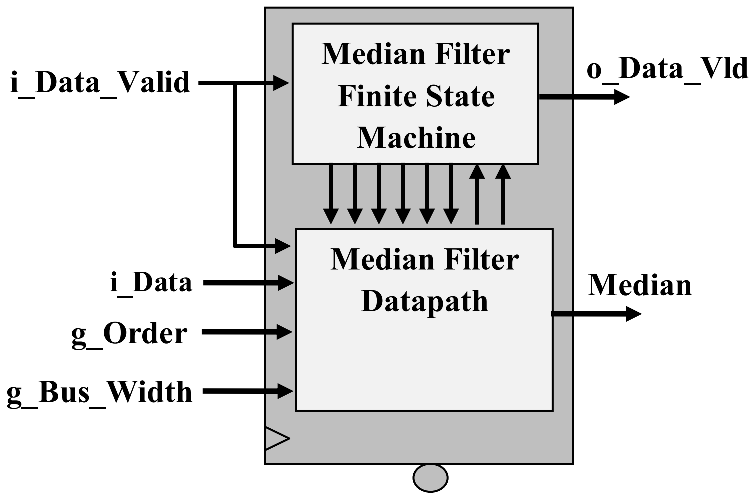

Fig 6. The Median Filter Design Structure

Table II explains the meanings of the inputs and outputs of the median filters.

TABLE III. THE MEDIAN FILTER INPUTS AND OUTPUTS

Port	Direction	Description
i_Data (g_Bus_Width:0)	in	Input Data Sample
i_Data_Valid	in	Input Data Valid Flag
g_Order	generic	Filter Order (N)
g_Bus_Width	generic	Sample Bus Width
Median	out	Filter Output
o_Data_Vld	out	Output Data Valid Flag

i_Data is the data input port of the filter whilst ***Median*** is the data output port. ***i_Data_Valid*** and ***o_Data_Valid*** are flags that signify when a new sample of data is coming into the filter and a new median processed output is coming out respectively. ***g_Bus_Width*** and ***g_Order*** are generic settings that the user sets for the filter data bus width and the filter order according to their requirements respectively.

4.2.1 The Datapath:

The design concept for the datapath is illustrated in Fig. 7. The design employs only two Compare and Swap stages for implementing the median filter algorithm. Each Compare and Swap stage is made up of comparators and multiplexers that compare and swap the input data. The number of these comparators depends on the number of data samples passing through each stage. This number is determined by the filter order, N, requirement of the user. The filter order is set by setting g_Order. The illustration in Fig. 7 shows a filter order setting of five, which requires two comparators in each Compare & Swap stage. Fig. 8 shows a more generic illustration of the Compare and Swap stages CS1 and CS2 where the required filter order is N.

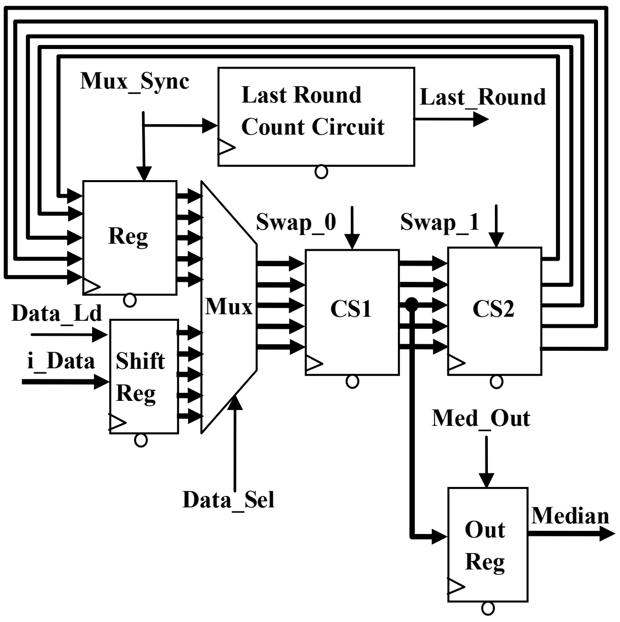

Fig 7. The Median Filter Datapath Design

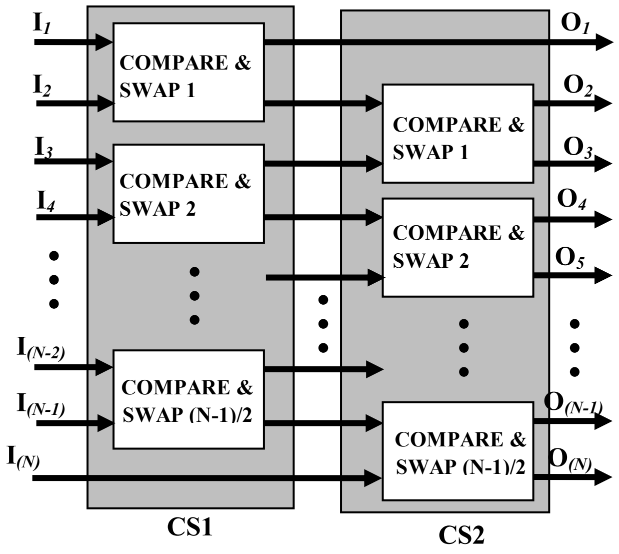

Fig 8. A Generic Illustration of the Operation of the CS1 and CS2 Blocks

4.2.2 The Finite State Machine

The datapath is controlled by signals from the finite state machine. The finite state machine controls the number of times the compare and swap stages are repeated based on the status of the Last_Round flag. The status of the Last_Round flag also depends on the filter order setting (g_Order). The interface of the finite state machine is shown in Fig. 9.

4.2.3 Median Filter Implementation

The median filter design was implemented in VHDL and integrated with the Measurement Module to make the Line Sensor IP-Core as part of the Gecko3 Robot System on Chip Application.

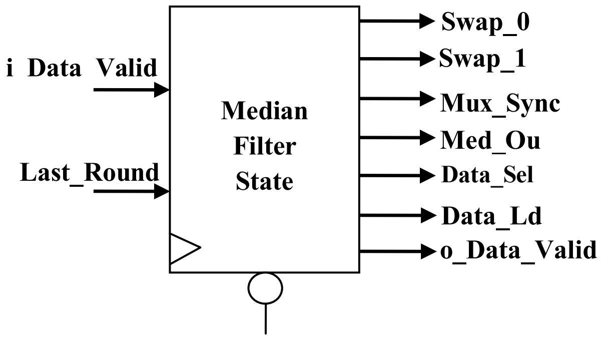

Fig 9. The Median Filter Finite State Machine Interface

V. Results

For all the bit resolution configurations varying from 4 to 10 bits, the two sensors were moved over a quantized grayscalesurface as illustrated in Fig. 10.

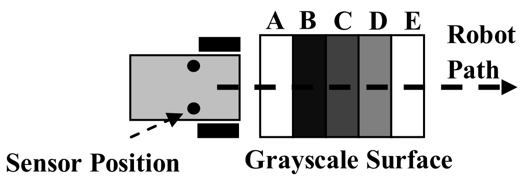

Fig 10. The Experimental Procedure – Sensor Motion over a Grayscale Surface

The outputs of the Line Sensor IP-Core for the motion over the grayscale surface were analysed for all bit resolution configurations. First, the output of the Measurement Module without filtering was noted. The Measurement Module's unfiltered data was filtered using a Matlab software median filter algorithm. The two sets of data were compared with the outputs of the complete Line Sensor IP-Core that included the median filter implementation in hardware. Fig. 11 illustrates the noisy output of the Measurement Module.

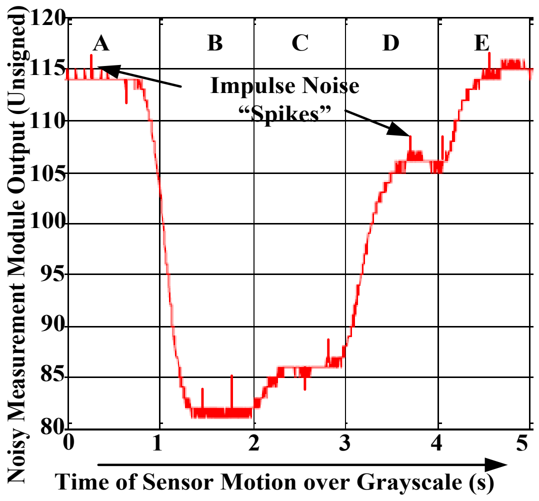

Fig 11. The output of the Measurement Module before filtering showing the corruption by impulse noise (Bit Resolution Configuration of 7)

The spiky impulse noise is evident across all grayscale levels. Fig. 12 illustrates the filtering results obtained from software (Matlab). Fig. 13 illustrates the filtering results obtained from the hardware (FPGA) implementation of the median filter.

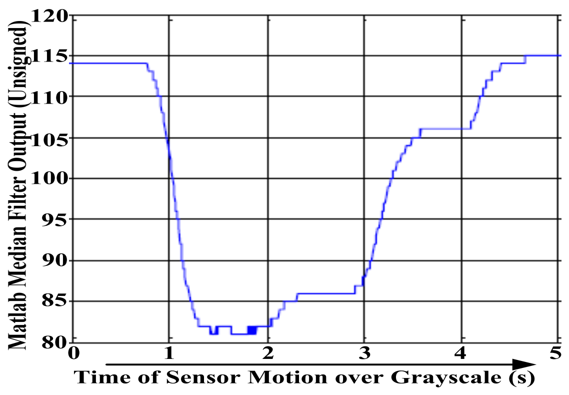

Fig 12. Results of filtering simulations with Matlab algorithm

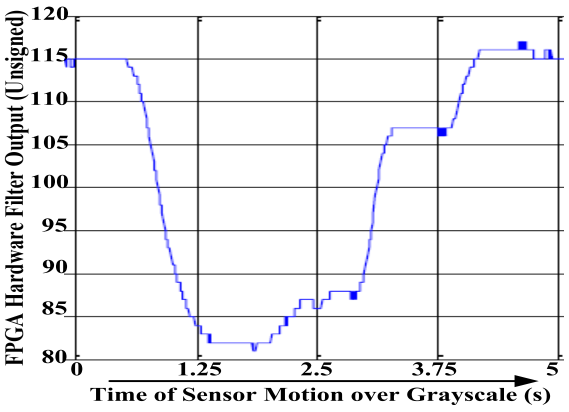

Fig 13. Results of filtering in Hardware (FPGA)

The constraints of the hardware filter circuit designed are as given in Table III. The constraints values relate to the filter of order 9, the minimum order required to effectively filter the measurement module's data across all bit resolution configurations.

TABLE IV. MEDIAN FILTER CIRCUIT DESIGN PARAMETERS

Filter Delay	570ns
Filter Frequency	2.2MHz
Hardware Utilization	277 slices (1% of Spartan-3 FPGA)

VI. CONCLUSIONS

A hardware median filter has been designed, developed and implemented as part of an ip-core for an autonomous mobile robot. The developed filter adopts a customized Bubble-Sort architecture. The architecture applies two Compare and Swap blocks whose operation is determined by the filter order and data bus width set by the user.

The use of only two compare and swap blocks to implement the median filter for various different orders removes the complexity dogma usually associated with median filter design. Such an approach also significantly reduces the latency challenges usually associated with median filtering. A latency in the order of nanoseconds makes the filter efficient and useful in real-time sensor systems for robot and other applications. The reconfigurability of the design allows even lower filter latencies and area utilization for lower order implementations. This is verified by simulations in Matlab and implementation in FPGA hardware.

REFERENCES

[1] Rajul Maheswari, S.S.S.P.Rao and P.G.Poonacha, "FPGA Implementation of Median Filter", Proceedings of International Conference on VLSI Design'97, January 4-7, 1997, pp. 523-524.

[2] V.G.Moshnyaga and K.Hashimoto, "An Efficient Implementation of 1-D Median Filter", Proceedings of the 52-th IEEE Midwest Symposium on Circuits and Systems (MWS2009), Cancun, Mexico, August 2-5, 2009, pp.451-454.

[3] D. S. Richards, "VLSI Median Filters," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 38, pp. 1, Jan 1990.

[4] Pingjun Wei; Liang Zhang; Changzheng Ma; Tat Soon Yeo, "Fast median filtering algorithm based on FPGA," Signal Processing (ICSP), 2010 IEEE 10th International Conference on, pp.426,429, 24-28 Oct. 2010.

[5] Z. Vasicek and L. Sekanina, "Novel Hardware Implementation of Adaptive Median Filters," Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on , vol., no., pp.1,6, 16-18 April 2008

[6] S. A. Fahmy, P. Y. K. Cheung, and W. Luk, "Novel fpga-based implementation of median and weighted median filters for image processing," in FPL, T. Rissa, S. J. E. Wilton, and P. H. W. Leong, Eds. IEEE, 2005, pp. 142–147.

[7] D. Caban, "FPGA implementation of positional filters," in Design of Embedded Control Systems. Springer-Verlag 2005,

[8] C. Chakrabarti, "Sorting network based architectures for median filters," Transaction on Signal Processing, 1994.

[9] The Gecko3 Wiki, http://labs.ti.bfh.ch/gecko/wiki/start

[10] The TCND5000 Reflective Optical Sensor Datasheet, Vishay Semiconductors, April 2005.

[11] B. Mahomva. Multisensor Educational Robot as a System on Chip Research Platform. Master's thesis, NUST, 2008.

[12] L. Mutauranwa and M. Collier, "A soft IP Core for a Reflective Optical Sensor in a robot system on chip," Wireless Communications and Signal Processing (WCSP), 2011 International Conference on , pp.1-4, 9-11 Nov. 2011.

[13] M. Jacomet, VLSI System Design - FSM-D Architecture Model, MicroLab-I3S, Berner Fachhochschule, January 2003