ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 3, March 2016

Area And Power Optimized One-Dimensional Median Filter

P. Premalatha,
PG Scholar,
PA College of Engineering and Technology,
Tamilnadu, India.

Ms. P. Karthika Rani, M.E.,
Assistant Professor,
PA College of Engineering and Technology,
Tamilnadu, India.

***Abstract*— A median filter is a nonlinear filter widely used in digital signal and image processing for the smoothing of signals, suppression of impulse noise, and edge preservation. The proposed work presents a low-power architecture for the design of a one-dimension median filter. It is a word-level two-stage pipelined filter, receiving an input sample and generating a median output at each machine cycle. It is particularly effective at removing 'salt and pepper' type noise. The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value. The power consumption is reduced by decreasing the number of signal transitions in the circuit. This can be done by keeping the stored samples immobile in the window through the use of a rank generation module. The additional power consumption can be reduced with the help of token ring architecture. This can increase the area of the architecture to certain extent. So Mux logic is employed to reduce area in the Proposed architecture, which replaces the logic gates, which utilizing the resources effectively. The simulations are obtained using ISIM simulator in XILINX 12.1 software. The power and area report are also obtained using the XILINX 12.1 software.**

***Index Terms* — Low-power, median filter, one-dimensional (1-D), token ring, Mux logic.**

I. INTRODUCTION

Lowering the dynamic power of a very large scale integrated circuit (VLSI) is an effective way to reduce the total power consumption [1]. One of the best ways to reduce the dynamic power dissipation is to minimize the switching activities, i.e., the number of signal transitions[6] in the circuit.

The token-ring architecture, adopted in the design of a mixed-timing First-In–First-Out (FIFO)[5] interface offers the potential for low power consumption since data are immobile in the FIFO.

In the designs, once data is queued, it will not be moved and is simply dequeued in place. A token, which is represented as logic state 1 in the token register, is used to control the dequeuing of old data and queuing of new data at the same time. All the token registers form a token ring, which is a succession of nodes interconnected in a circular manner. The token-ring architecture with exactly one token will be adopted in the design for lowering the power consumption of a 1D median filter.[1]

The window by moving some of the stored samples to the left. In the second step, the incoming sample is compared with the already sorted samples and then inserted in the right place by moving some of them to the right.[7] In both of their methods, however, some of the stored samples have to be shifted left or right, depending on their values when a new input sample enters the window[3]. For some applications that require a larger sample width, more signal transitions The median of a set of samples in the word-level sorting network is often computed by first sorting the input samples and then selecting the middle value. In those methods, the input samples are sequentially processed word by word, and the incoming sample is inserted into the correct rank in two steps.[7] In the first step, the oldest sample is removed from in the circuit are needed; i.e., more dynamic power will be consumed.

II. EXISTING TECHNIQUE

A. Architecture Overview

An overview of the low-power median filter architecture with window size N is shown in Figure 1. It consists of a circular array of N identical cells and three auxiliary modules: Rank Calculation (RankCal), Rank Selection (RankSel),

Median Selection (MedianSel). All the cells are also connected to a global input register X, through which they receive the incoming sample, and the median is stored in the output register Y. [1]

The architecture is implemented as a two-stage pipeline, where the registers in all the cells serve as the internal pipeline registers. All the registers in the architecture are synchronized by the rising edge of a global clock. Each cell block c_i is composed of a rank generation (RankGen) module, a comparator module "==," and three registers: an m-bit (m = log2 N), rank register (P_i), a data register (R_i), and a 1-bit token register (T_i). Register R_i stores the value of the sample in cell c_i, register P_i keeps the rank of this sample, and the enable signal (en) of R_i is stored in register T_i.[1]

All the samples in the window are ranked according to their values, regardless of their physical locations in the window. In the design, a cell with a greater sample value will be associated with a greater rank. However, for two cells c_i and c_j , whose sample values are equal, c_i will be given a greater rank if R_i is newer than R_j (or R_j is older than R_i); i.e., the sample in c_j enters the window earlier than the sample in c_i. The rank is hence unique for each cell. For a window with size N, the rank starts from 1 for a cell with the least sample value, and ends with N for a cell with the greatest sample value.[1]

The median of the window can then be obtained from the sample value R_i of a cell c_i whose rank P_i is equal to (N + 1)/2, assuming N is an odd number. In the architecture, the input sample enters the window in a FIFO manner. After it is queued, it will not be moved and is simply dequeued in place. A token, which is represented as logic state 1 in the token register of some cell, is used to control the dequeuing of old sample and queuing of new input sample at the same time. After the token is used, it will be passed to the next cell at a new cycle. [1]

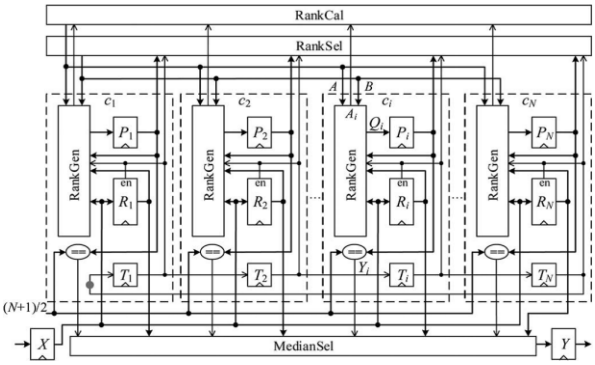


Figure 1: Architecture of the Median Filter.

At the initial state of the architecture, to make the first incoming sample be stored in the first cell c_1, the token is assumed to exist in the last cell c_N, shown as the shadowed circle at the output of register T_N. Whenever an input sample enters the window at a new cycle, the rank of each cell has to be updated. It may have to be recalculated, or may be the old rank decremented by 1, incremented by 1, or kept unchanged.[1]

The new rank of each cell, which is denoted by signal Q_i in the cell, is generated by the RankGen module. Each RankGen module receives signal A from the RankCal module and signal B from the RankSel module. Signal A is the recalculated rank of a cell c_i that contains the token and signal B is the old rank of c_i. Moreover, signal Y_i is the output of a comparator module "==," which compares the value of rank P_i with a constant value (N + 1)/2 so that Y_i = 1 if P_i is equal to (N + 1)/2, else Y_i = 0. This signal is used to indicate if the corresponding cell c_i contains the median in R_i. Similar to the RankSel module, the MedianSel module transfers the value of R_i to the output register Y if Y_i = 1; i.e., if the median is stored in R_i.[1]

B. Circuit Behaviour

At each machine cycle t_i, the first stage of the two-stage pipelined filter performs the following operations for the input sample X: calculate the new rank of each cell, insert X in a cell that contains the token, and pass the token to the next cell. This means

that for all P_i, R_i, and T_i registers, their new values will be calculated and determined at this stage so that they can be updated at the next cycle t_i+1. At the same time, the second pipeline stage calculates the median value for the input sample that enters the window at the previous cycle t_i−1. That is, for the output register Y , its new value will be calculated at this stage so that it can also be updated at the next cycle t_i+1.[1]

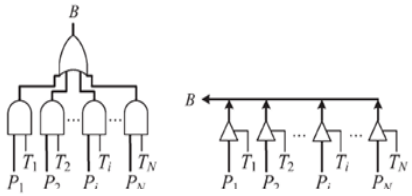
An example illustrating the insertion of nine input samples into a window with five cells is given in Figure 2. Initially, at cycle t_0, to make the first input sample to be stored in the first cell c_1, the last cell c_5 is designed to contain the token (T_5 = 1). The rank and sample values (P_i and R_i) of each cell, along with the values of the two input/output registers X and Y, are all reset to be zero. When the first sample 12 enters the window at cycle t_1, the token has been moved from c_5 to c_1 (T_1 = 1 and T_5 = 0). The value of P_5 has also been updated to be 5 since the initial zero of X (now stored in R_5) is treated as a virtual sample at the initial cycle t_0. To prepare for the next cycle t_2, the new value of R_1 will be determined as 12 to store the input sample since c_1 contains the token. The new value of P_1 will be calculated as 5 since sample 12 (to be stored in R_1) is greater than the sample values of the other four cells. Finally, the new values of T_1 and T_2 will be determined as 0 and 1, respectively, to indicate that the token will be moved from $c_1$to c_2. [1] All the values of P_i, R_i, and T_i will then be updated at the next cycle t_2.

Clk	Input Reg	Cell Registers															Output Reg
	X	T_1	T_2	T_3	T_4	T_5	R_1	R_2	R_3	R_4	R_5	P_1	P_2	P_3	P_4	P_5	Y
t_0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
t_1	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
t_2	59	0	1	0	0	0	12	0	0	0	0	5	0	0	0	4	0
t_3	35	0	0	1	0	0	12	59	0	0	0	4	5	0	0	3	0
t_4	47	0	0	0	1	0	12	59	35	0	0	3	5	4	0	2	0
t_5	66	0	0	0	0	1	12	59	35	47	0	2	5	3	4	1	12
t_6	52	1	0	0	0	0	12	59	35	47	66	1	4	2	3	5	35
t_7	38	0	1	0	0	0	52	59	35	47	66	3	4	1	2	5	47
t_8	18	0	0	1	0	0	52	38	35	47	66	4	2	1	3	5	52
t_9	26	0	0	0	1	0	52	38	18	47	66	4	2	1	3	5	47

Figure 2: Circuit Behaviour.

It can be seen from the example that when an input sample is inserted into the window, the old sample in each cell will not be moved. Instead, the rank of each cell is recalculated so that the new median can be obtained in a cell whose rank is equal to (N + 1)/2.[1]

C. *RankSel And MedianSel Modules*



(a) (b)

Figure 3: Implementation of the RankSel Module (a) Using AND/OR Gates. (b) Using Tristate Buffers.

The RankSel module is responsible for transferring the rank P_i of a cell c_i to its output B if c_i contains the token; i.e., when T_i = 1. Figure 3(a) shows a simple implementation of this module using AND/OR gates. It can also be implemented by tristate buffers in Figure 3(b), where B is the output of a global data bus that collects the output signals of all the tristate buffers. Since there exists exactly one cell that contains the token at any time.[1]

D. RankGen and RankCal Modules

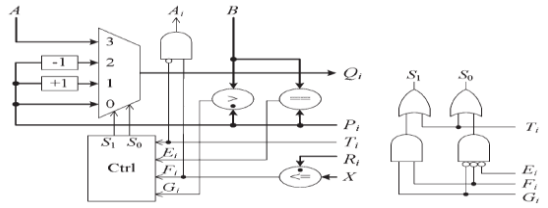
For the RankGen module in a cell c_i, its implementation is given in Figure 4(a). Signal F_i is the output of a comparator module "<=," which compares the value of R_i with that of the input sample X so that F_i = 1 if R_i is less than or equal to X (R_i<= X), else F_i = 0 (R_i> X). Signal A_i is the output of a logic AND gate so that A_i = 1 if T_i = 0 and F_i = 1; i.e., if cell c_i does not contain the token and R_i is less than or equal to X, else A_i = 0. The A_i signal of each cell is connected to the RankCal module, which calculates the new rank of a cell that contains the token. The new rank is calculated as K + 1, where K is the number of cells, which does not contain the token and whose sample value is less than or equal to X. [1]

The RankCal module can be implemented by a multi-input adder that adds all the A_i signals and then increments the sum by 1. If cell c_i contains the token, the output A of the RankCal module will be its new rank at the next cycle. On the contrary, if cell c_i does not contain the token, its new rank will be

determined by the other signals. Signal G_i is the output of a comparator module ">", which compares the value of rank P_i with that of signal B so that G_i= 1 if P_i is greater than B, else G_i = 0.[1]

E. Ctrl Module

For a cell c_i, since there are four possible sources to update its rank, a 4-to-1 multiplexer is used to select one of these sources for signal Q_i in Figure 4(b).The Case 1 (Decremented by 1): P_i>P_j and R_i<= X. Case 2 (Incremented by 1) : P_i<P_j and R_i>X. Case 3 (Kept Unchanged) : P_i<P_j and R_i<= X, Case 5 (Kept Unchanged) : P_i = P_j Case 4 (Kept Unchanged) : P_i>P_j and R_i> X. Then, the value of rank P_i will be updated by the value of Q_i at each cycle. The multiplexer is controlled by two selection signals S_1 and S_0 and these two signals, which are generated by the Ctrl module, are determined by four signals T_i, E_i, F_i, and G_i.[1]



(a) (b)

Figure 4: (a) Implementation of the RankGen Module. (b) Implementation of the Ctrl Module.

If cell c_i contains the token (T_i = 1),its new rank is obtained from the output A of the RankCal module, i.e., the value of S_1 S_0 should be 11 when T_i = 1. If cell c_i does not contain the token (T_i = 0), there are five cases. In Case 1 when P_i>P_j (E_iG_i = 01) and Ri<= X (F_i = 1), the rank of c_i will be decremented by 1; i.e., the value of S_1 S_0 should be 10 when $E_iF_iG_i$= 011. Similarly in Case 2, when P_i <P_j (E_iG_i = 00) and R_i> X (F_i = 0), the rank of ci will be incremented by 1; i.e., the value of S_1 S_0 should be 01 when $E_iF_iG_i$ = 000. Finally, when P_i<P_j (E_iG_i= 00) and R_i<= X (F_i = 1) in Case 3, P_i>P_j (EiGi = 01) and R_i>X (F_i = 0) in Case 4, and P_i = P_j(E_iG_i = 10) in Case 5, the rank of c_i will be kept unchanged; i.e., when $E_iF_iG_i$ = 010,001, the value of S_1 S_0 should be 00. Figure 4 depicts a simple implementation of the Ctrl module.[1]

III. POWER AND AREA OPTIMIZED 1D MEDIAN FILTER

Optimized Area and Power Efficient 1D median filter is having all the features of A Low-Power Architecture for the Design of a One-Dimensional Median Filter. The AND, OR and XOR logic used in the system Is changed into a mux based gates with optimal usage of FPGA resources.[2] This substitution helps to reduce the area and power consumption of the whole system. Also this substitution helps in optimized utilization of the FPGA slices.[2] Thus this 1D median filter can be a good substitute for all the filters and can be used in fast, power and area efficient devices.

A. Modified Cntrl Module

For a cell c_i, since there are four possible sources to update its rank, a 4-to-1 multiplexer is used to select one of these sources for signal Q_i in Figure 4(b). Then, the value of rank P_i will be updated by the value of Q_i at each cycle. The multiplexer is controlled by two selection signals S_1 and S_0 and these two signals, which are generated by the Ctrl module, are determined by four signals T_i, E_i, F_i, and G_i.

Here the AND, XOR and OR gates used are modeled using multiplexer (MUX) in order to optimize the area power and delay of the system by efficient utilization of slices in the FPGA.

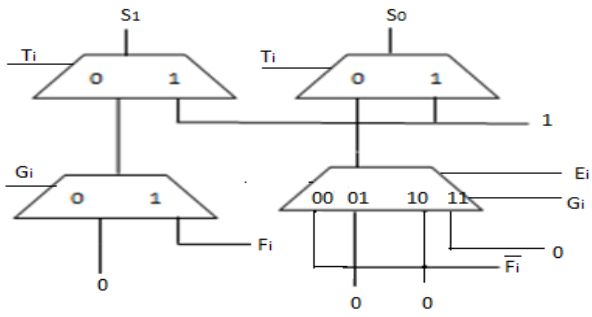


Figure 5: Modified Cntrl Module.

This modified control units as shown in Figure 5 can be used in the RankGen unit, so that it can utilize the FPGA/Tool resources effectively. The area will be reduced by this modified work. Further area can

be also be reduced by converting all the logic gates into Mux.

B. *Modified RankSel Module*

The modified RankSel unit used in the proposed work is shown in which is helpful in area as well as the power reduction.

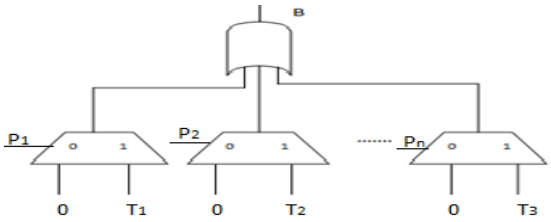


Figure 6: Modified RankSel Module.

Here the AND gates are converted into Equivalent Multiplexers as in Figure 6. The OR is used as it is to reduce the Hardware Complexity of the circuit.

IV. RESULTS AND DISCUSSIONS

A. *Median Filter output*

The output of optimized 1D median filter is shown in Figure 7.

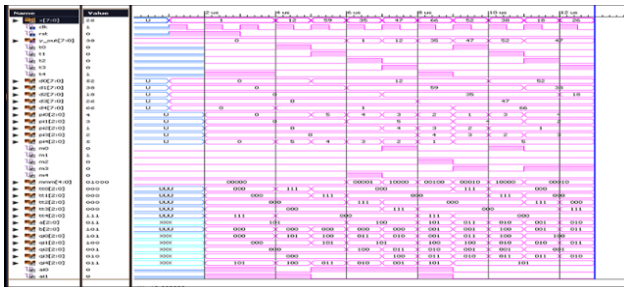


Figure 7: Response of the proposed Filter.

B. *Area Utilization*

By implementing the total system, the area consumed by the system is checked in the design summary. The area consumption is given in the terms of number of slices occupied. Figure 8 shows the area utilization of the proposed system in an FPGA.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	61	1,408	4%	
Number of 4 input LUTs	167	1,408	11%	
Number of occupied Slices	111	704	15%	
Number of Slices containing only related logic	111	111	100%	
Number of Slices containing unrelated logic	0	111	0%	
Total Number of 4 input LUTs	167	1,408	11%	
Number of bonded IOBs	18	108	16%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	3.92			

Figure 8: Area Report of the Proposed Filter.

C. *Power Consumption*

The power consumption is the total power consumed by the system. The power consumption is also reduced. The power consumption is taken as a sum of leakage power and dynamic power. Thus the total consumed power can be obtained. Figure 9 shows the power consumption of the proposed system.

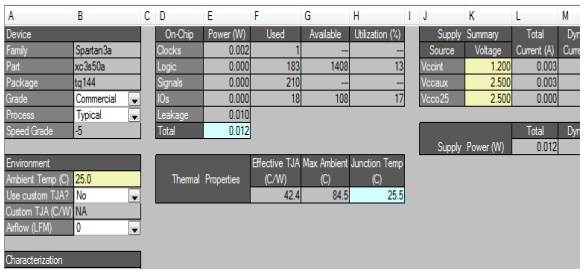


Figure 9: Power Report of the Proposed Filter.

V. PERFORMANCE COMPARISON

Table 1: Comparison Table

DESIGN	AREA(No of LUTs)	Power(mw)
Area And Power Optimized One-Dimensional Median Filter	**167**	**12**
A Low Power Area Efficient One Dimension Median Filter	**183**	**16**

The results shows that the power consumption and area of the system have been reduced comparing to previous systems. Table1 shows the comparison of the area and Power between

existing and proposed carry select adders. Also the simulation of the system using mux based gates instead of logic gates shows a reduction in area and power consumption than the system using logic gates in FPGA. Thus the Median filter proposed is having reduced area and power consumption than existing Filters.

VI. CONCLUSION

Thus, a new median filter structure based on Mux logic is designed in order to reduce the hardware complexity and power consumption, which is beneficial to the filter when the number of stages is increasing. In median filter implementation, sorting the samples to generate rank and comparators consumes more power. In this architecture the power consumption can be reduced by keeping the stored samples immobile in the window through the use of the rank generation unit. The additional power consumption can be done by token ring architecture. The proposed median filter offers comparable search performance, scalability and lower cost than existing median filter. And Mux logic reduces the Area by utilizing the resources of the tool/FPGA slices effectively. Here Area and Power has been Reduced upto 8.74% and 25% respectively.

REFERENCES

[1]. Ren-Der Chen, Pei-Yin Chen, Member, IEEE, and Chun-Hsien Yeh (2015), " A Low-Power Architecture for the Design of a One-Dimensional Median Filter", Ieee Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 62, No. 3, March 2015.

[2]. Mr. MoosaIrshad K. P., Mrs. M. Meenakumari, Ms. S. Sharmila (2015), "Optimized area-delay and power efficient carry select adder", International Advanced Research Journal in Science, Engineering and Technology Vol. 1, Issue 4, December 2014

[3]. Benkrid. K, Crookes. D and Benkrid. A (2002), "Design And Implementation A of Novel Algorithm For General Purpose Median Filtering On FPGAS", IEEE Trans.

[4]. Cadenas. J, Megson. G. M, Sherratt. R. S and Huerta. P (2012), "Fast median calculation method," Electron. Lett., Vol. 48, No. 10, pp. 558–560.

[5]. Chelcea. T and Nowick. S. M (2004), "Robust interfaces for mixed-timing systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Vol. 12, No. 8, pp. 857–873.

[6]. Chen.O. T.C, Wang. S and Wu. Y. W (2003), "Minimization of switching activities of partial products for designing low-power multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Vol. 11, No. 3, pp. 418–433.

[7]. Chen. R. D, Chen. P. Y and Yeh. C. H (2013), "Design of an area-efficient one dimensional median filter," IEEE Trans. Circuits Syst. II, Exp. Briefs, Vol. 60, No. 10, pp. 662–666.

[8]. Choo. C and Verma. P (2008), "A real-time bit-serial rank filter implementation using Xilinx FPGA," in Proc. SPIE Real-Time Image Process.,Vol. 6811, pp. 68110F-1–68110F-8.

[9]. Fahmy. S. A, Cheung. P. Y. K and Luk. W (2009), "High-throughput one-dimensional median and weighted median filters on FPGA," IET Comput.Digit. Tech., Vol. 3, No. 4, pp. 384–394.

[10]. Gonzalez. R. C and Woods. R. E (2001), "Digital Image Processing", 2nd edition.,Englewood Cliffs, NJ: Prentice-Hall.

[11]. Jan Rabaey (2009), "Low Power Design Essentials", Springer Science & Business Media - Technology & Engineering.

[12]. Moshnyaga. V. G and Hashimoto. K (2009), "An efficient implementation of 1-D median filter" in Proc. 52nd IEEE Int. MWSCAS, pp. 451–454.