International Journal of Current Research in Multidisciplinary (IJCRM)
ISSN: 2456-0979
www.ijcrm.com ‖ Volume 1 Issue 5 ‖

The Design of Median Filter to Reduce the Power Consumption

M. Banumathi[1], V. Praveen Jeba Selvan[2]

[1]*Student, Applied Electronics Department, Infant Jesus College of Engineering, TN, India*
[2]*Associate Professor, ECE Department, , Infant Jesus College of Engineering, TN, India*

Abstract

A Low power architecture for the design of 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. The power consumption is reduced by decreasing the number of signal transition in the circuit .This is done by keeping the stored samples are immobile in the window through the use of token ring in our architecture. The result have shown that, at the expense of some additional area cost, the power consumption can be successfully reduced

Keywords: *Low power, median filter, one-dimensional (1-D), token ring*

I. Introduction

Lowering the dynamic power of a very-large –scale integrated circuit is an effective way to reduce the total power consumption. One of the best way to reduce the dynamic power dissipation is to minimize the switching activities[3], [9]. The token ring architecture, adopted in the design of a mixed –timing first-in first out interface [2], [7], offer the potential for low power consumption since data are immobile in the FIFO. In their designs, once the data 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 register form one token ring.

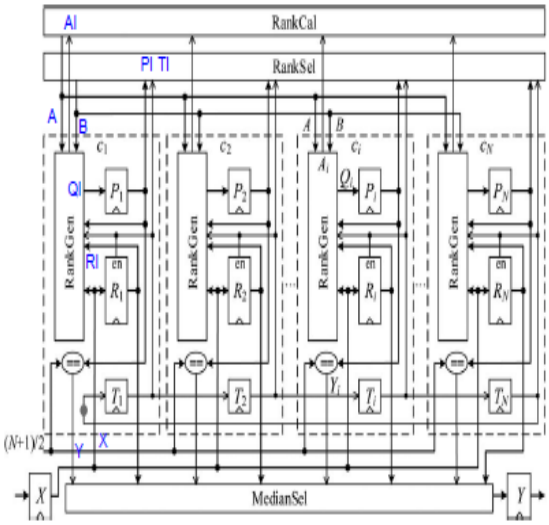
A median filter is a non linear filter widely used in digital signal and image processing for smoothing of signal, suppression of noise , and edge preservation[11].The median filter replaces a samples with the middle-ranged value among all the samples inside the sample window, centered around the sample. Depending on the number of samples processed at the same cycle. there are two types of architecture for hardware design, i.e., word level architecture and bit level architecture. In the word level architecture, the input samples are sequentially processed word by word, and the bits of the sample are processed in parallel[4], [6], [8]. On the contrary, the bit level architecture process the samples in parallel and the bits of the incoming samples are sequentially processed [1], [5] [10], [12] .in our project , the word level architecture will be adopted in the design of low power median filter for practical use.The median of set of samples in the word level sorting the network is often computed by first sorting the input samples and then selecting the middle value .in their method the input samples are sequentially processed word by word, and the incoming samples is inserted into the correct rank in two steps. In the first step, the oldest sample is removed from the window by moving some of stored samples to the left. In the second step, the incoming samples is compared with the already stored samples and then inserted in the right place by moving some of them to the right. The difference between two architecture in [4] and [8] is that these two steps are separately performed in two clock cycle in [8], Where as in [4], it takes only one cycle. In both of their method ,however , some of the stored samples have to be shifted left or right, depending upon their values when a new input samples enter a window.

To conquer this problem, a new median filter architecture targeting low power consumption. Instead of sorting the samples physically in the window, the stored samples are kept immobile there. Only the rank of each sample, which uses fewer bits, has to be updated at each new cycle when an input samples enter a window. Since our architecture is implemented as a two stage pipeline the median output, Which is the median rank, will be generated at each cycle. The improvement of power consumption is achieved by utilizing a token ring in our architecture. The rest of this brief is organized as follows. Section II gives the low power architecture of our median filter ,and section III explain rank updating method of our architecture.

The circuit implementation is discussed in section IV. section V includes the result and discussions. And finally section VI conclude this brief.

II. Filter Architecture

A .Architecture : Fig. 1 gives an over view of low power median filter architecture with window size N. It consist of a circular array of N identical cells and three auxiliary modules: rank calculation(RANKCAL), rank selection(RANKSEL), and median selection(MEDIANSEL). All the cells are also connected to a global input register X, through which they receive the incoming sample, and median output is stored in the output register Y.



.Fig 1. Low power filter architecture

The architecture is implemented as a two stage pipeline, where the register in all the cells serve as the internal pipelined registers. Each cell block composed ci is composed of a rank generation (RANKGEN) module, a. comparator module "==" and three register: an rank register (Pi), a data register(Ri), and a one bit token register (Ti). Register store the value of sample in cell ci, register Pi keeps the rank of this sample, and the enable signal (en) of Ri is stored in register Ti. All the samples in the window are ranked according to their values, regardless of their physical location in the

window. In our design, a cell with a greater sample value will be associated with greater rank. In the architecture, the input samples enter the window in a FIFO manner. After it is queued, it will not be moved 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. All the token register form one token ring .whenever an input samples enter the window at a new cycle, the rank of each cell has to be updated. it may have to recalculated, or may be old rank decremented by 1, incremented by 1, or kept unchanged. Each RankGen module receive signal A from the RangCal module and signal B from the RankSel module. Signal A is the recalculated rank of a cell ci that contain the token and signal B is the old rank of a comparator module, which compares the value of rank Pi is equal to (N+1)/2, else Yi=0, This is used to indicate if the corresponding cell ci contains the median in Ri. Similar to the RankSel module, the MedianSel module transfer the value of Ri to the output register Yi =1; i.e., if the median is stored in Ri.

B. Circuit Behavior : At each machine cycle ti, the first stage of our 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 contain the token, and pass the token to the next cell. This means that for all Pi, Ti, and Ri register, their new values will be calculated and determined at this stage so that they can be updated at the next cycle ti+1. At the same time, the second pipeline stage calculates the median value for the input samples that enter the window at the previous cycle ti-1. The insertion of nine input samples in to a window with five cell in fig.2. for each cell ci, the value of Pi, Ri and Ti registers are also shown in the figure. Initially at cycle t0, to make the first input samples be stored in the first cell C1 , last cell C5 is designed to contain the token (T5=1). The rank and sample values (Pi and Ri) of each cell, along

with the values of the two input /output registers X and Y , are all rest to zero. When the first input sample 12 enters the window at a cycle t1, the token has been moved from c5 to c1 (Ti =1and T5=0). The value of p5 has also been updated to be 5 since the initial zero of X (now stored in R5) is treated as a virtual sample at the initial cycle t0. To prepare for the next cycle t2, the new value of R1 will be determined as 12 to store the input sample since c1 contains the token. The new value of P1will be calculated as 5 since the sample 12 (to be stored in R1) is greater than the sample values

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

Fig. 2. Example illustrating the insertion of nine input samples in to a window

of the other four cells. Finally, the new value of Ti and T2 will be determined as 0 and 1, respectively, to indicate that the token will be moved from c1 to c2. All the values of Pi, Ri and Ti will be updated at the next cycle t2. When the window fully occupied with the valid data at cycle t6, cell ci1 hold the token again (T1=1). The new value of the median output Y for the next cycle t7 will determined as the value of R4 (47) since the rank of P4 equal to 3.since the architecture is a two-stage pipeline, when Y is updated to be 47 at cycle t7 for the input sample 66, the value of all Pi , Ri, and Ti will be updated at this cycle for the next input sample 52.

III. Rank updating

This section explain how to determine the new rank for each cell. Two types of cell will be separate ly discussed: a cell with a token and a cell without token.

A .Cell With the Token : A cell ci with the token, its sample value Ri will be replaced by the input sample X, and its rank Pi has to be recalculated. The new value of pi can be obtained by comparing X with the sample values of all the other N-1 cells that do not contain the token. For these cells, if K is the number of cells Whose sample value is less than or equal to X, the new value of Pi will be K+1.

B. Cell Without the Token : For a cell ci without token, its sample value Ri will not be affected when an input sample X enter the window. However its rank Pi may be affected by the sample value Rj of another cell cj that contains the token. Since the value of Rj replaced by X, the relation between Ri and Rj may change. However, the sample value Rk of any other cell ck that does not contain the token will not affect Pi since the value of Pi and Pj, and relation between Ri and X, the new value of Pi may be decremented by 1, incremented by 1, or kept unchanged when an input sample X is inserted in to the window the changes of rank Pi can be explained as the following five cases, where the cell ci does not contain the token, but cell cj does. Case 1—(Decremented by 1) Pi > Pj and Ri <=X: If the rank Pi is greater than rank Pj, Ri is greater than or equal to (but newer then) Rj at the current cycle. If Ri is less than or equal to the input sample X, Ri will be less than or equal to (but older than)Ri to the value that is greater than or equal to (but newer than) Ri. Therefore, the number of cells whose sample value is less than or equal to (but older than) Ri will be decremented by 1at the next cycle; i.e., Pi has to be decremented by 1. Take rank P4 at cycle t6 , its value will be decremented by 1(3 to 1) at the next cycle t7.

Case 2—(Incremented by 1) Pi < Pj and Ri > X: similar to case 1, if rank Pi is less than rank Pj, Ri is less than or equal to (but older than) Rj at the current cycle. If Ri is greater than the input sample X, Ri will be greater than the input sample X, Ri will be greater than the new value of Rj at the next cycle. Therefore, the number of cells whose sample value is less than or equal to (but older than) Ri will incremented by 1 at the next cycle; Pi has to be incremented by 1.
Casse 3—(Kept Unchanged) Pi < Pj and Ri < = X: If Pi is less than Pj, Ri is less than or equal to(but older than) Rj will also be less than or equal to (but older than)Ri at the next cycle . Therefore, the number of cells whose sample values less than or equal to (but older than)Ri at the current cycle will be equal to that at the next cycle; i.e., Pi has to be kept unchanged .for example, at cycle t7 of fig.2 the value of rank P3 has to be kept unchanged at one at the next cycle t8.

Case 4__(Kept Unchanged) Pi > Pj and Ri >X: similar to case #, If Pi is greater than Pj and Ri is greater than X , the number of cells whose sample value is less than or equal to (but older than) Ri at the current cycle will also be equal to that at the next cycle; i.e., Pi has to be unchanged.

Case 5__(kept Unchanged) Pi = Pj : This case occur when the window is not fully occupied with valid data . At the initial state, the rank of each cell is reset to be zero. After the window is fully occupied, each cell will be assigned a nonzero and unique rank. If rank Pi is equal to rank Pj, the values of Pi and Pj are both zero, and neither cell ci nor cell cj contains valid data. The new value of Pi at the next cycle will still be zero since ci does not contain the token; i.e., Pi has to be kept unchanged at zero. At cycle t3 of fig. 2, for example, the value of rank P4 will still be zero at the next cycle t4. That It can be obtained from the above discussion that for a cell ci that contains the token , its rank Pi has to be recalculated at a new cycle . If ci does not contain the token, its rank Pi be decremented by 1, incremented by 1, or kept unchanged. Therefore, there are four sources are for ci to update its rank.

IV. Circuit Implementation

The implementation of the Ranksel, Mediansel, RankGen, and Rankcal modules in the architecture will be discussed.

A.RankSel and MedianSel Modules : The RankSel module is responsible for transferring the rank Pi of a cell to its output B if ci that contain the token; i.e. when Ti = 1. Fig. 3(a) shows a simple implementation of this module using AND/OR gate. It can also be implemented by tristate buffers in Fig. 3(b), where B is the output of a global data bus that collect the output signals of all the tristate buffers. Since there exists exactly one cell that contains the token at any time; i.e., there exist exactly one Ti signal whose value is equal to 1, the value of B will always be valid. The Mediansel module can also be implemented in a similar way. It transfer the value of Ri to the output register Y if Ri is the median; i.e., when Yi=1. If it is implemented by tristate buffer . the median will be valid when there exist at least (N+1)/2 samples in the window; otherwise it will remain in a high impedance state.

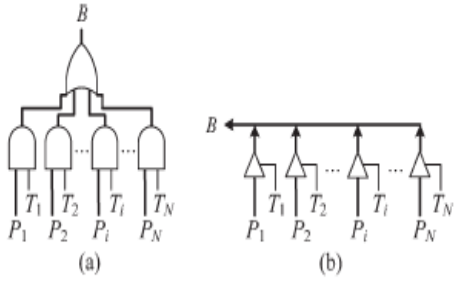


Fig. 3. Implementation of the RankSel module. (a) using AND/OR gates. (b) using tristate buffers

B. RankGen and RankCal Modules : For the RankGen module in a cell ci, its implementation is given in Fig.4(a). Signal Fi compare the values of Ri with that of the input sample X so that Fi = 1 if Ri is less than or equal to X (Ri <=X), else Fi = 0(Ri > X). Signal Ai is the output of a logic And gate so that Ai = 1 if Ti = 0 and Fi = 1; i.e., if a cell ci does not contain the token and Ri is less or equal to X, else Ai = 0. The Ai signal of each cell is connected to the RankCal module, which calculates the new rank of a cell that contains the token. As mentioned in Section III-A, 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; i.e., K is the number of logic 1's on the Ai signals of all the cells. The RankCal

module can be implemented by multi-input adder that adds all the Ai signal and then increments by the sum by 1. If cell ci contains the token, the output A of the RankCal module will be its new rank at the next cycle.

On the contrary, if cell ci does not contain the token, its new rank will be determined by the other signals. Signal Gi is the output of comparator module ">,"
which compares the values of rank Pi with that of signal B so that Gi = 1 if Pi is greater than B, else Gi = 0. Since the value of B is the rank Pj of another cell cj that contain the token, the meaning of gi can also be described as Gi = 1 if Pi is greater than Pj (Pi > Pj), else Gi = 0 (Pi <= Pj). Signal Ei is the output of a comparator module "==," which also compares the values of Pi with that of B so that Ei = 1 if Pi is equal To B, else Ei = 0 . Likewise, the meaning of Ei can be described as Ei = 1 if Pi is equal to Pj (Pi = Pj), else Ei= 0. Combining these two signals Ei and Gi, for the three relations between Pi and Pj (Pi > Pj, Pi = Pj,and Pi < Pj), the corresponding value of EiGi will be 01,10,and 00, respectively.

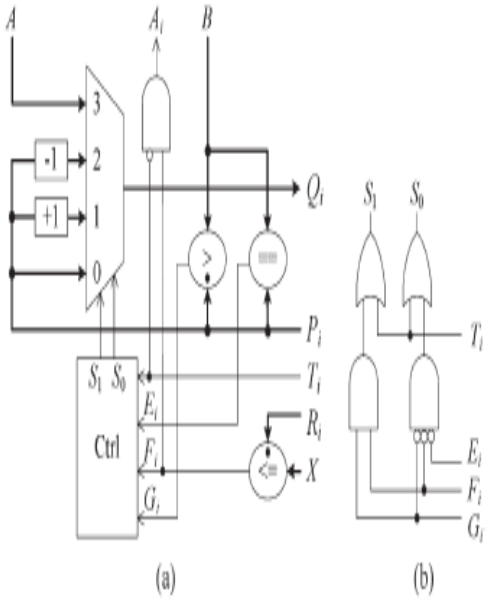


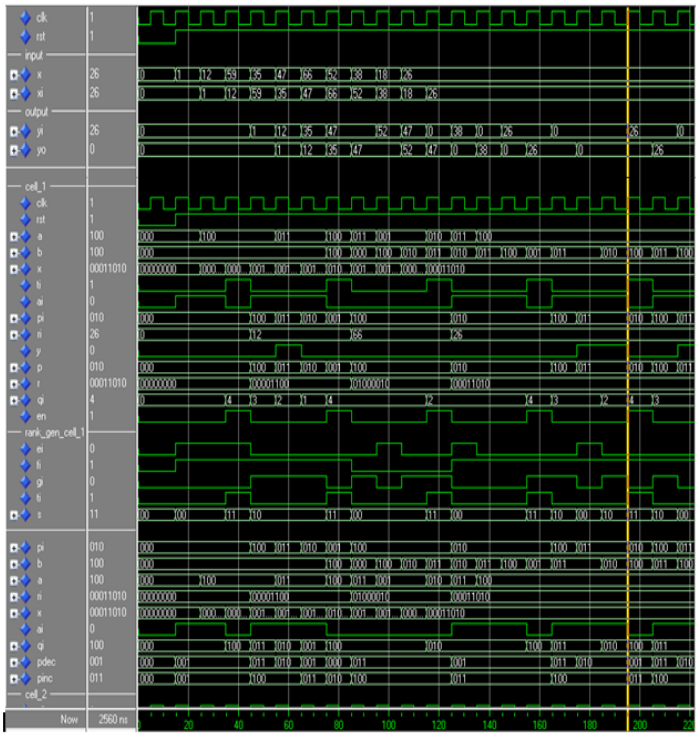
Fig. 4. (a) Implementation of the RankGen module. (b) Implementation of the Ctrl module.

Fig. 4. (a) Implementation of the RangGen module. (b) Implementation of the control module.

C. Control Module : For a cell ci, there are five possible sources to update its rank, a 4- to-1 multiplexer is used to select one of these sources for the signal Qi in Fig. 4(a). Then, the value of rank Pi will be updated by the value of Qi at each cycle. The multiplexer is controlled by two selection signal S1 and S0; and these two signals, which are generated by the Ctrl module, are determined by four signals Ti, Ei, and Gi. If cell ci contain the token (Ti = 1), its new rank is obtained from the output A of the RankCal module; i.e., the value of S1S0 should be 11 When Ti=1. If cell ci does not contain the token (Ti = 1), its new rank is obtained from the output A of the RankCal module; i.e., the value of S1S0 should be 11 when Ti=1. If cell ci does not contain the token (Ti=0) there are five cases for this, as described in section III-B. In case 1 When Pi > Pj (EiGi =01) and Ri <=X (Fi =1), the rank of ci will be decremented by 1; i.e., the value of S1S0 should be 10 when EiFiGi = 011. Similarly in case 2, when Pi <Pj (EiGi = 00) and Ri <=X (Fi = 0), the rank of ci will be incremented by 1; i.e., the value of S1S0 should be 01 When EiFiGi = 000. Finally, when Pi < Pj (EiGi = 00) and Ri <= X (Fi = 1) in Case 3, Pi > Pj (EiGi = 01) and Ri <= X (Fi = 0) in case 4, and Pi = Pj (EiGi =10) in case 5, the rank of ci will be kept unchanged; i.e., when EiFiGi = 010, 001, or 1 – 0, the value of S1S0 should be 00. Fig. 4(b) depicts a simple implementation of the ctrl module.

V. Simulation Results

The simulation results obtained for median filter. First we are giving input values(x, x_i) for each cell and the input values stored in register(a, b) of the cell. Then it will be enter into the Rank calculation module. The RankCal module can be implemented by a full adder that adds all the Ai signals and then increments the sum by 1. The output of RankCal values are given to the input of Rank generation. Rank generation output will be stored in register P1 then P1 and token register(T1) values are move on to the Rank selection module. Median filter operates over a window by selecting middle value. The outputs of the median selection are given to the output register.



VI. CONCLUSION

On this paper, we have reported The power saving is achieved by adopting a token ring in our design, where the stored samples are kept immobile; only the rank of each sample has to be updated at each new cycle. As can be seen from the experimental results, the power consumption for median filters in practical use has been successfully reduced at the expense of some area overhead.

.

References

[1] Ren- Der Chen, Pei- Yin Chen, Member, IEEE, and Chun- Hsein Yeh, " A Low Power Architecture For The Design Of One Dimentional Median Filter,"IEEE Transaction On Circuits And Systems – II; Express briefs, vol 62, No.3, March 2015.

[2] J. Cadenas, g. M. Megson, R.S.Sheratt, and P.Huerta, "Fast median calculation method," Electron Lett, vol. 48, no. 10,pp.558-560 May 2012.

[3] T.Cholcca and S. M. Nowick, "Robust interfaces for mixed-timing system systems, "IEEE Trans. Very Large scale integr. (VLSI) Syst., vol.12, no. 8, pp.859-873, Aug. 2004..

[4] R.-D. Chen, P.-Y. Chen, and C.-H. Yeh, "Design of an area-efficient one dimensional median filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*,
vol. 60, no. 10, pp. 662–666, Oct. 2013.

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

[6] S. A. Fahmy, P. Y. K. Cheung, and W. Luk, "High-throughput one dimensional median and weighted median filters on FPGA," *IET Comput.*
Digit. Tech., vol. 3, no. 4, pp. 384–394, Jul. 2009.

[7] I. M. Panades and A. Greiner, "Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in GALS architectures," in *Proc. 1st Int. Symp. NOCS*, 2007, pp. 83–94.

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

[9] M. Mottaghi-Dastjerdi, A. Afzali-Kusha, and M. Pedram, "BZ-FAD: A low-power low-area multiplier based on shift-and-add architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 2, pp. 302–306, Feb. 2009.

[10] D. Prokin and M. Prokin, "Low hardware complexity pipelined rank filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 6, pp. 446– 450, Jun. 2010.

[11] D. S. Richards, "VLSI median filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 1, pp. 145–153, Jan. 1990.

[12] Z. Vasicek and L. Sekanina, "Novel hardware implementation of adaptive median filters," in *Proc. 11th IEEE Workshop DDECS*, 2008, pp. 1–6.