Abstract - There are many established techniques in gray value image processing for noise removal. Median filters are the most popular method for noise extraction to achieve high throughput and low hardware cost. In contrast, energy efficiency architecture remains an untapped area for power improvement though it has become a topic of current technology. This paper proposes efficient low-power architecture for the design of a median filter. This is a pipelined filter architecture which receives input samples and generating a median output for every clock cycle. The number of signal transition in the design is decreased by storing 3 x 3 sample window’s pixel. By this proposed architecture will decrease the power at the expenses of slight increase in area.

Key words: Low-power, median filter, one-dimensional (1-D), token ring.

I. INTRODUCTION

The effective way to reduce the total power consumption of a very-large-scale integrated circuit is to lowering the dynamic power. The dynamic power consumption is due to switching activities in the circuit and the best way to minimize the signal transitions. Using first in first out interface in the design and the data in the FIFO is stationary, the switching activities are reduced and this design offers low power. One of the best ways to reduce the dynamic power dissipation is to minimize the switching activities, i.e., the number of signal transitions in the circuit [1], [2]. The token-ring architecture is adopted in the design first in first out interface which offers low power consumption since data is stationary in the FIFO [3],[4]. In their designs, a status is maintained in the token to indicate whether queuing is required or not. When the token =1 indicates that refresh a new data by removing the old data from the queue. A token ring is formed by the token register, which are circular register interconnecting nodes. In the token-ring architecture for lowering the power consumption we adopt one token in one dimension (1-D) median filter.

Median filtering is a nonlinear smooth technology. Each pixel of the gray value of a neighborhood has its own pixel gray value of the median. That means all pixels within the neighborhood sort by gray value, taking the median of the group as a neighborhood center pixel output value. A median filter is used in image processing for suppression of impulse noise, and edge preservation and smoothing of signals [5]. There are word-level architecture and bit-level architecture for hardware design depending on the number of samples processed at the same cycle. In the word-level architecture, the input samples are sequentially processed word by word, and the bits of the sample are processed in parallel [6], [7], [8]. On the contrary, the bit-level architecture processes the samples in parallel and the bits of the incoming samples are sequentially processed [9], [10], [11], [12]. For low power median filter normally we will adopt the word-level architecture for practical use.

The set of input samples in the word level sorting network is computed in two phases 1) sorting the input samples and 2) selecting the middle value [6], [7]. The authors proposed their methods, the input samples are sequentially processed word by word, and the incoming sample is inserted into the correct rank in two steps. The oldest sample is removed from the window by moving some of the stored samples to the left in the first step and 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. The difference between the two architectures in [6] and [7] is that these two steps are separately performed in two clock cycles in [6], whereas in [7], it takes only one cycle.

However in these methods the stored samples are shifted left or right depending on their values when a new sample is arrived in the window. For few applications require a large sample width, there will be a more number of signal transition can occur and this will increase the dynamic power consumption.

In order to take care of the above problem new median filter architecture is proposed to take care of low power consumption. In this architecture the samples are kept stationary and the rank of each samples are altered and updated. In order to improve the throughput the architecture has two stage pipelines and the median output will be generated at each clock cycle when an input sample enters in the window. The improvement in power consumption is achieved by utilizing a token ring in the proposed architecture. Since the stored samples in the window are stationary, this architecture is suitable for low-power applications.

The paper is organized as follows. Section II gives the overall low-power architecture of the proposed median filter and Section III explains how the rank is updated in this architecture. The implementation of the architecture in the circuit level is discussed in the section IV and the section V gives the simulation and synthesis report the proposed architecture. Finally the conclusion is briefly discussed in section VI.

II. FILTER ARCHITECTURE
The overview of proposed low power median filter architecture is shown in Fig. 1 with window size N. This architecture consists of a N identical cell circular array with three modules namely rank selection (RankSel), rank calculation (RankCal) and median selection (Mediansel). The register cells are connected to the input register X, through which they receive the incoming sample and the median output is stored in the output register Y. The filter architecture is implemented using two-stage pipeline, where the registers in all the cells serve as the internal pipeline registers. The registers in the architecture are synchronised with the rising edge of the global clock. Each cell block c_i is composed of a comparator module (=) and rank generation (RankGen) module, and it consists of three registers namely a data register (R_i), an m-bit (m=log_2^N) rank register (P_i) and a 1-bit token register (T_i). Register R_i is used to stores the value of the sample input data in cell c_i, register P_i keeps the rank of this sample and R_i is stored in register T_i by the enable signal of (R_i). Rank register contains the rank according to the data values which is stored in data register R_i regardless of their physical locations in the window. In the proposed architecture, a cell with a higher sample value will be associated with a greater rank. However, for two cells c_i = c_j, 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_i enters the window earlier than the sample in c_j.

The rank is unique for each cell and window size N, the rank starts from 1 for a cell having smallest value and to N for the cell having highest value. Assuming N is odd and the median value of the window can be obtained from the smallest value R_1 of a cell c_1 whose rank P_1 is equal to \((N+1)/2\).

Whenever an input sample enters the window at a new cycle, the rank of each cell has to be calculated and updated by decrementing the old rank by 1 or increment by 1 or kept unchanged. The new rank of each cell, which is denoted by signal Q_i in the cell, is enerated by the RankGen module. Each RankGen module receives signal A from the RankCal module and signal B from the RankSel module.

The 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

\[
Y_i = \begin{cases} 
1 & \text{if } P_i = \frac{N+1}{2} \\
0 & \text{else} 
\end{cases}
\]

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.

At each machine cycle t, the first stage of the proposed two stage pipeline filter performs the following operations for the input sample data X:

1. Calculate the new rank of each cell
2. Insert input data X
3. The new value of P_i, R_i and T_i are calculated and updated in the next cycle t+1.

The second stage pipeline stage calculates the median value for the input sample that enters the window at the previous cycle t-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+1.

Example for illustration: The insertion of nine input samples into a window with five cells is given in Fig. 2. For each cell c_i, the values of its P_i, R_i and T_i registers are all shown in the figure.

Initially, at cycle t_o:

1. The first input sample be stored in the first cell c_1, the last cell c_5 is designed to contain the token (T_5=1).
2. 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.

At cycle t_1:

1. When the first sample 12 enters the window, the token has been moved from c_5 to c_1 (T_5=1 and T_1=0).
2. The value of P_5 is updated to 5 since the initial zero of X (now stored in R_5) is treated as a virtual sample at the initial cycle t_0.

At cycle t_2:

1. The new value of R_1 will be determined as 12 to store the input sample since c_1 contains the token and the new value of P_1 is 5 since sample 12 > the sample values of the other four cells.

Figure 1. Low-power filter architecture
2 The new values of $T_1=0$ and $T_2=1$ to indicate that the token will be moved from $c_1$ to $c_2$.
3 The values of $P_i$, $R_i$, and $T_i$ updated.

At cycle $t_5$:
1 The window is fully occupied with valid data at cycle $t_6$, cell $c_i$ holds the token again ($T_i=1$).

At cycle $t_7$:
1 The median output $Y$ is determined as the value of $R_k$ since rank $P_j$ is equal to $3$, i.e., $(5 + 1)/2$
2 Proposed architecture is a two-stage pipeline, when $Y$ is updated the values of $P_i$, $R_i$, and $T_i$ will also be updated.

It can be seen from this 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 $(N+1)/2$.

![Image](38x447 to 258x560)

**Figure 2.** Example illustrating the insertion of nine input samples into a window

### III. RANK UPDATING

This section explains how the new rank for each cell is determined. Two types of cells will be separately discussed: a cell with the token and a cell without the token.

Cell with the Token: The cell $c_i$, with the token, its sample value $R_i$ will be replaced by the input sample data $X$, and its new rank $P_i$ is recalculated. The rank for $P_i$ will be obtained by comparing $X$ with the sample values of all the other $N-1$ cells that do not contain the token. Assume $K$ is the number of cells whose sample value is less than or equal to $X$, the new value of $P_i$ will be $K+1$. The new value of rank $P_i$ will be calculated at the next cycle $t_7$.

Cell Without the Token: The cell $c_i$ without the token, its sample value register $R_i$ will not be affected when an input sample $X$ enters the window. However, its rank $P_i$ may be affected by the sample value $R_j$ of another cell $c_j$ that contains the token. The relation between $R_i$ and $R_j$ may change due to change of $R_j$.

The cell which does not contain any token will not affect $P_i$, since the value of the register will not be changed. When the input sample data $X$ is inserted into the window, the new value of $P_i$ may be decremented by 1, incremented by 1, or kept unchanged depending upon the relation between $P_i$ and $P_j$. There are five cases for the change of rank $P_i$, which are given below:

Case 1—(Decremented by 1) $P_i > P_j$ and $R_j <= X$ : $R_i$ will change from a value that is less than or equal to (but older than) $R_i$ to a value that is greater than or equal to (but newer than) $R_i$. Therefore, the number of cells whose sample value is less than or equal to (but older than) $R_i$ will be decremented by 1 at the next cycle.

Case 2—(Incremented by 1) $P_i < P_j$ and $R_i <= X$ : The number of cells whose sample value is less than or equal to (but older than) $R_i$ will be incremented by 1 at the next cycle.

Case 3—(Kept Unchanged) $P_i < P_j$ and $R_i >= X$ : The number of cells whose sample value is less than or equal to (but older than) $R_i$ at the current cycle will be equal to that at the next cycle; i.e., $P_i$ has to be kept unchanged.

Case 4—(Kept Unchanged) $P_i > P_j$ and $R_i >= X$: Similar to Case 3, $P_i$ has to be kept unchanged.

Case 5—(Kept Unchanged) $P_i = P_j$: If rank $P_i$ is equal to rank $P_j$, the values of $P_i$ and $P_j$ are both zero, and neither cell $c_i$ nor cell $c_j$ contains valid data. The new value of $P_i$ at the next cycle will still be zero since $c_i$ does not contain the token; i.e., $P_i$ has to be kept unchanged at zero.

### IV. CIRCUIT IMPLEMENTATION

The hardware implementation of RankGen, RankSel and MedianSel modules are presented in this section.

RankSel and MedianSel Modules: if $c_i$ contains the token (when $T_i=1$) 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 $T_i=1$. There are two ways to implement this logic and the logic diagram is shown in figure 3 and figure 4.

a) AND/OR gate: Since there exists exactly one $T_i$ 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 transfers the value of $R_i$ to the output register $Y$ if $R_i$ is the median; i.e., when $Y_i=1$.

b) Tristate buffer: where $B$ is the output of a global data bus that collects the output signals of all the tristate buffers. The median will be valid only when there exists at least $(N+1)/2$ samples in the window; otherwise, it will remain in a high impedance state.

![Image](317x66 to 505x144)

**Figure 3.** Implementation of the RankSel module using AND/OR gates.
RankGen and RankCal Modules: For the RankGen module in a cell $c_i$, its implementation is given in the following expression and the figure 4.

$$ F_i = \begin{cases} 
1 & \text{if } R_i \leq X \\
0 & \text{otherwise} 
\end{cases} \quad \text{(2)} $$

$$ A_i = \begin{cases} 
1 & \text{if } F_i = 1 \text{ and } T_i = 1 \\
0 & \text{otherwise} 
\end{cases} \quad \text{(3)} $$

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 equal to the number of logic 1’s on the $A_i$ signals of all the cells. 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.

$$ G_i = \begin{cases} 
1 & \text{if } P_i > B \left( P_i > P_j \right) \\
0 & \text{otherwise } \left( P_i \leq P_j \right) 
\end{cases} \quad \text{(4)} $$

$$ E_i = \begin{cases} 
1 & \text{if } P_i = B \left( P_i = P_j \right) \\
0 & \text{otherwise } \right) 
\end{cases} \quad \text{(5)} $$

Combining these two signals $E_i$ and $G_i$, for the three relations between $P_i$ and $P_j$, the corresponding value of $E_iG_i$ are tabulated in Table 1.

Table 1. $E_iG_i$ Generation

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Relation Between $P_i$ and $P_j$</th>
<th>$E_iG_i$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P_i &gt; P_j$</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>$P_i = P_j$</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>$P_i &lt; P_j$</td>
<td>00</td>
</tr>
</tbody>
</table>

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 Fig. 4(a). The following table gives the control signal $S_i$ and $S_0$ depending upon the four signals.

Table 2. Control signal generation

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_i</td>
<td>E_i</td>
<td>F_i</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

V. SIMULATION RESULT

The proposed architecture is implemented and coded using HDL. The simulation result is shown below. By using this approach the switching activity of the sample data is further reduced.

The above code is synthesized using Xilinx EDA tool. The RTL diagram is shown in figure 4.2.

The RTL diagram of the rank selection unit is shown below.
The following table shows the device utilization summary:

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Description</th>
<th>Number device used</th>
<th>Number of device available</th>
<th>% usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of Slices</td>
<td>7</td>
<td>46560</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>Number of Slice Flip Flops</td>
<td>7</td>
<td>46560</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>Number of 4 input LUTs</td>
<td>15</td>
<td>46560</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>Number of bonded IOBs</td>
<td>23</td>
<td>360</td>
<td>6%</td>
</tr>
<tr>
<td>5</td>
<td>Number of GCLKs</td>
<td>1</td>
<td>24</td>
<td>4%</td>
</tr>
<tr>
<td>6</td>
<td>Minimum input arrival time</td>
<td></td>
<td></td>
<td>1.089ns</td>
</tr>
<tr>
<td></td>
<td>before clock</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VI. CONCLUSION

The paper proposed 1-D median filter design using low power architecture. The switching activity of the input data is reduced by adopting token ring in the design. Here the rank of each sample is updated at each new cycle keeping the stored sample data immobile. Using this proposed method the power consumption of the median filter is reduced at the expense of area increase. The future work of the proposed architecture is to incorporate in a dedicated IC chip to further reduction in area and power delay product.

REFERENCES