Hardware Modeling of Sorting Mechanism for Finding First Maxima/Minima Values from a Set Greater Than 2

1 Kante Raj Kumar & 2Dr.V.Thrimurthulu.
1 II M.Tech VLSI SD Student, CR Engineering College, Tirupati, Chittoor(Dist) A.P, India,
2 Professor, Head of ECE Dept., CR Engg., College, Tirupati, Chittoor(Dist)A.P, India,
EMAIL: 1rk.kante@gmail.com, 2vtmurthy.v@gmail.com

Abstract:

with this document, we have proposed a complete VLSI model of sorting mechanism to find first maxima/minima values using (BWA) Bit Wise And architecture. In this effort a parallel radix-sort-based VLSI architecture for finding the first W maximum (or minimum) values is proposed. The described architecture, named Bit-Wise-And (BWA) architecture with an added advantage of high level scalability depends on analyzing input data from the most significant bit to the least significant one, the proposed scheme along with parallel structure has been modelled successfully on Xilinx Virtex -5 FPGA with a simple verilog RTL description.

Key Words:

Sorting; Bit Wise and Operation; verilog; RTL description and Xilinx Virtex -5 Fpga

Introduction:

Sorting is a deep-rooted problem in computer science and is a mean of operation in quite a lot of applications. Along with hardware implementation of sorting networks has been identified as well in the history. On the other side, VLSI architectures for partial sorting, which can also be derived from assortment networks (AN), are part of different algorithms in the communication field. Partial sorting is employed, for instance, in and for the decoding of turbo and binary Low-Density-Parity-Check (LDPC) codes, in for maximum-likelihood decoding of arithmetic codes and in for K-best MIMO detectors, non-binary LDPC decoders and turbo product codes respectively. Designs for finding the first two minimum values are used in binary LDPC decoder architectures to implement minterm Sum approximations and recently they have also been proposed for the case of non-binary LDPC decoders. However, very a small number of works, probe the general problem of implementing parallel architectures for finding the first two maximum/minimum values with a logical approach. Similarly, architectures for finding the first W which is greater than maximum/minimum values in a set of M elements are of now a days interest. It has been observed by a review of literature survey that the sorting mechanism implemented with a variety of applications such as comparator based, selection based, partial based and radix based. Radix sorting algorithms depends and works on the bit-wise scrutiny of the data to be sorted and can be absolute to selection and partial sorting problems. Unfortunately all these algorithms are not up to the bottle neck related to the sorting job whose data size is greater than 2. The proposed design was suggested with Radix based sorting along with parallel architecture can overcome afore mentioned problem. The anticipated elucidation, referred to as Bit-Wise-And(BWA) architecture, works by analyzing the set
of candidates beginning from the Most-Significant-Bit (MSB) to the Least-Significant-Bit (LSB).

The problem statement can be identified as described below, let for a given set of data \( D(k) = \{d_0, \ldots, d_{k-1}\} \) prepared of k elements, now if we wish to notice the first \( X \) number of maximum/minimum values, namely \( x(k) = \{x(k)_0; x(k)_1; \ldots; x(k)_{X-1}\} \) where \( x(k)_0 = \max(X(X)) \), 

\( x(k)_{X-1} = \max(X(X) \setminus \{x(X)_0\}), \ldots, x(k)_{l} = \max(X(X) \setminus \{x(X)_0, \ldots, x(X)_l\}), \ldots, x(X)_{X-1} = \max(X(X) \setminus \{x(X)_0, \ldots, x(X)_{X-2}\}) \).

Similarly the same statement can be applied for minimum number of data sorting procedure.

**Design development:**

Earlier Bit Wise And operation works with the analogy of Radix sorting which depends on the analysis of \( x(k) \) values bit by bit from the MSB to the LSB along with the assumption that the values in \( x(k) \) are all positive. And for negative or non-positive data the assumptions will be acceptable with two’s compliment notations preserving the order relation, by flipping the MSB bit and entire operation works bit by bit, where as the proposed architecture relays on array of data operation works in parallel.

The proposed architecture also overcomes the problem of zero-row condition.

**TABLE 1**

**SUMMARY OF DESIGN CONSIDERATIONS**

<table>
<thead>
<tr>
<th>Sno</th>
<th>Design consideration</th>
<th>Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compiler</td>
<td>Xilinx14.4Vivado</td>
</tr>
<tr>
<td>2</td>
<td>Programming Language</td>
<td>Verilog</td>
</tr>
<tr>
<td>3</td>
<td>Architecture</td>
<td>Parallel BWA</td>
</tr>
<tr>
<td>4</td>
<td>FPGA</td>
<td>Virtex -5</td>
</tr>
<tr>
<td>5</td>
<td>Interface</td>
<td>USB</td>
</tr>
</tbody>
</table>

**Modeling of proposed architecture:**

As mentioned earlier, the (BWA) Bit Wise And architecture along with parallel Radix sort having the two methods: applying data from monotonic sequences and implementing proposed logic operation in parallel. The first method is used to implement more data values and the second method is used to unite both having the following principle of operations and to enhance its physical parameters such as speed and utilization power. The falconer cannot hear the falconer

**Bit Wise AND Operation:**

Computer programming applications, the most popular bit wise operations operates on one or more bit patterns or binary numerals at the level of their entity bits. It is a fast, primitive action directly supported by the processor, and is used to influence values for comparisons and calculations. On simple low-cost processors, typically, bitwise operations are substantially faster than division, quite a lot of times faster than multiplication, and every so often significantly faster than addition. While modern processors typically perform addition and multiplication just as fast as bitwise operations due to their longer instruction pipelines and other architectural design choices, bitwise operations do commonly use less power because of the reduced use of resources.

A bitwise AND takes two equal-quantity of binary representations and evaluates the logical AND operation on each pair of the analogous bits, by multiplying them. Thus, if both bits in the compared position are 1, the bit in the resulting binary representation is 1 (1 × 1 = 1); otherwise, the result is 0 (1 × 0 = 0). For example:

1101 (decimal 13)
AND 1011 (decimal 11) = 1001 (decimal 9)

**Parallel Radix Sort Operation:**

Radix sort is one of the linear sorting algorithms for integers. It works with sorting the input numbers on each digit, for each of the digits in the numbers. However, the process adopted by this sort method is somewhat counterintuitive, in the sense that the numbers are sorted on the least-significant digit first, followed by the second-least significant digit and so on till the most significant digit. The time complexity of the algorithm is as follows: Suppose that the n input numbers have maximum k digits. Then the Counting Sort procedure is called a total of k times. Counting Sort is a linear or $O(n)$ algorithm. So the entire Radix Sort procedure takes $O(kn)$ time. If the numbers are of finite size, the algorithm runs in $O(n)$ asymptotic time. Radix Sort $(v, k)$ includes the following two stages of work:

1. For i - 1 to k
2. Do use a stable sort to sort Array $v$ on digit k.

The parallelization and its implementation can be done as shown in the following procedure:

**Parallelization:**

- MSB digit is at first
- There May be Large or Small Pieces of Data

- It is not an easy process to make all the pieces to be sorted fast and efficiently.

**Implementation:**

The parallelization of Radix sort can be implemented as described below stages:

- The most interesting part is the counting
- At first a structure tag is used to count the elements
- Tag is a dyadic array unsigned integer of \([K + 1][n]\)
- $K$ is the number of threads and $n$ is the number of possible values of $K^i$.
- Parallel stable sort the array on $Kt$.
- Step 1: Every thread goes through its piece of data, and counts Elements with $K^i$= (possible value 1), Elements with $K^i$= (possible value 2), Elements with $K^i$= (possible value 3), …….. Elements with $K^i$= (possible value n).
- Step 2: After all the threads finish their counting, get all the numbers and generate some start positions for these pieces.
- Step 3: With these start positions, every thread goes through its piece of data again and move the elements.
to an extra array, and make the extra array sorted on digit $K^t$.

- When thread $k$ come to an element with $K^t = p.v. e$,
  It increase tag $[k+1][e]$.

The process repeats up to the final stage and is as shown in following figure

![Figure3: Final stage of implementation](image)

The parallel Radix sort evaluation is predicted in the following graph

![Figure4: Radix Sort evaluation](image)

Front-End Modeling:

This phase of implementation contains the following stages simulation using Xilinx 14.4 Vivado suite, synthesis using Xilinx 14.4 XST and verifying on Virtex – 5 FPGA board.

VLSI Structural design for proposed method:

The intention of parallel Radix Sort based BWA technique is to create high speed VLSI architecture for sorting applications. The combined architecture is shown in the figure5. The proposed scheme is described using Verilog HDL with RTL coding style and is simulated and synthesized with Xilinx tools. The design is successfully verified on Virtex -5 FPGA board.

I. Simulation and Synthesis Results

The Verilog RTL Description of the above article is simulated and synthesized using Xilinx14.4 (ISE-Simulator), it is noticed that at some instance of simulation time the set of data is sorted in parallel and various results are shown in the following figures.

![Figure5: Proposed Architecture](image)

![Figure6. Simulation output](image)
The Synthesized report is summarized in the following Table.

<table>
<thead>
<tr>
<th>Sno</th>
<th>Parameter</th>
<th>Quantification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Target Device</td>
<td>xc5vlx50t-2-ff1136</td>
</tr>
<tr>
<td>2</td>
<td>Slice Logic utilization</td>
<td>Less than 5%</td>
</tr>
<tr>
<td>3</td>
<td>Slice logic distribution</td>
<td>100%</td>
</tr>
<tr>
<td>4</td>
<td>IO utilization</td>
<td>6%</td>
</tr>
<tr>
<td>5</td>
<td>Specific feature factor</td>
<td>30%</td>
</tr>
<tr>
<td>6</td>
<td>Total logic delay</td>
<td>11.035 nS</td>
</tr>
<tr>
<td>7</td>
<td>Total offset delay</td>
<td>11.035 nS</td>
</tr>
<tr>
<td>8</td>
<td>Total path delay</td>
<td>6.945 nS</td>
</tr>
<tr>
<td>9</td>
<td>Real time compilation</td>
<td>7Sec</td>
</tr>
<tr>
<td>10</td>
<td>Total memory usage</td>
<td>282028 Kb</td>
</tr>
</tbody>
</table>

**Results of on Xilinx Virtex-5 FPGA board verification**

**Conclusion:** This paper analyzes the VLSI hardware modeling of parallel Radix Sorting using BWA architecture, the modified architecture relies on bitwise illusion of input data depends on finite number of multistage where each stage is designed with simple unit, the proposed architecture is described using Verilog HDL and successfully verified on Xilinx Virtex – 5 FPGA.

**References:**

[1] A parallel radix-sort-based VLSI architecture for finding the first $W$ maximum/minimum values

Guoping Xiao, Maurizio Martina, Member IEEE, Guido Masera, Senior Member IEEE, Gianluca Piccinini


decoding,” *IET Communications*, vol. 6, no. 7, pp. 702–707, Jul 2012.


**AUTHORS**

**KANTE RAJ KUMAR** received his BTech degree in Electronics & Comm. Engineering from Kandula Obula Reddy Memorial engineering college, Kadapa (A.P), India, in the year 2013. Currently pursuing his M.Tech degree in VLSI System Design at Chadalawada Ramanamma Engineering College, Tirupathi (A.P), India. His area of research includes low power VLSI design.

**Dr.V.THRIMURTHU LU** M.E., Ph.D., MIETE, MISTE Professor & Head of ECE Dept. He received his Graduation in Electronics & Communication Engineering AMIETE in 1994 from Institute of Electronics & Telecommunication Engineering, New Delhi, Post Graduation in Engineering M.E specialization in Microwaves and Radar Engineering in the year Feb, 2003, from University College of Engineering, Osmania University, Hyderabad, and his Doctorate in philosophy Ph.D from central University, in the year 2012. He has done his research work on Ad-Hoc Networks.