Efficient Data Sorting: Introducing a new Method.

Raghavendra Devidas[1] , Aishwarya Kulkarni S[2]

[1]ITT/QIM, Mercedes Benz Research & Development India Pvt ltd, Bengaluru-560066, India, Telephone No: +91 9538676171, Email: Raghavendra.devidas@daimler.com

[2]ITT/QIM, Mercedes Benz Research & Development India Pvt ltd, Bengaluru-560066, India, Telephone No: +91 8310820879, Email: aishwarya.s.kulkarni@daimler.com

Abstract: The efficiency of data sorting algorithms is the key aspect which determines the speed of data processing and searching. The best known efficiency of sorting algorithm has been Log (N) if there are N terms. All of the well-known sorting algorithms use various techniques to sort data. The basis for most of these are comparing the data terms with each other. In this manuscript, we are introducing a new approach for sorting data. This method is postulated to have the highest efficiency ever achieved by any of the sorting algorithms. We achieve this by sorting data without comparing the data terms. Or achieving results of data comparison without comparing the terms explicitly!

Keywords: Efficient data sorting, New techniques for sorting, Sorting through hardware re-configuration, new computer architecture for sorting, Sorting by Hardware optimization, quantum inspired computer architecture, parallelism for Grover's algorithm.

1. Introduction

Existing sorting algorithms in the field of computer science have complexities varying from O Nlog(N) to O(N^2), from the best case to the worst case. These algorithms belong to a class called "comparison sorts" where the sorted order is determined based only on comparisons between the input elements. It has also been proven that any comparison sort algorithm requires Ω(NlgN) comparisons in the worst case. Thus, the comparison sorting algorithm like merge sort is optimal, and no comparison sorting algorithm exists that is faster by more than a constant factor. There are also non-comparison based sorting algorithms like counting sort, bucket sort etc. which can run in linear time, but with a prior assumption. In order to sort a given set of data array/list, comparison of terms in inevitable. The complexity of the sorting algorithm increases with more & more comparisons needed to sort the data completely. Our approach here is to sort the data in a specialized memory hardware. Such that the data gets placed in an ordered array as soon as the new value/term is entered. The terms of an array are stored in their corresponding weighted order. As if the terms float if they are lighter and dip down if they are heavier. And each addition of a term gets weighted and settles down in the special hardware at its right position. i.e the sorting doesn't happen by explicit comparison. However by implicit ordering of terms. There are various alternate approaches such as by Hayes et al., [1], Gowtham et al., [5] and Abdel-Hafeez et al., [4]. Similalry the wonderful approach adopted by Jmaa et al., [3] gives great inspiration for further innovation mainly due to the application of such sorting methods in aviation industries.

The detailed survey of sorting algorithms along with their efficienices by Karunanithi et al., [6] and Mishra et al., [7] guided the opportunities for further innovation in finding sorting algorithms with greater efficiencies.

We wanted to leverage the speed with which the electreomagnetic waves travels, for sorting data in the foeld of computer science. In this paper we have used the electric signal as the medium of data transfer and storage.

2. Methods

While we look for possibilities of tuning the efficiency of the widely used existing sorting methods, the survey by Estivill-Castro et al., [8] points at clear opportunities which we can target in our solution approaches. Though such approaches offer some efficiencies, there could be several challenges for their applications in use cases, involving huge dataset. Hence with the aim to bring in an exponential speed in the data sorting & searching methods, we have adapted an unconventional method for sorting data. There are various focus areas to improve efficiency of the existing algorithms. One of them is from Idrizi etal.,

[9]. i.e either we can try to identify the areas of improvement in the well established sorting methods or we can lok for completely new methods.

The sorting methods and search capabilities as methods as given by Wilkes et al., [10], have given quantitative measures to check the feasibility for applying a specific sorting method. Also it is clearly highlighted that the efficiency is affected with increasing volume of data. Hence we have found a new method, whose efficiency is nearly independent from the volume of data. It is limited only by the extent to which the infrastructure can scale. Insights regarding types of memories present in the current computer hardware as mentioned by Meena et al., [11], have helped in exploring further possibilities for memory storage and opportunities for further optimizations. The details mentioned by Arge et al., [12] have given sufficient guidance on internal & external memory units and their role in achieving sorting efficiencies.

For our method, one of the key circuit element, the OPAMP and its capabilities as mentioned by Solomon et al., [14] were very useful. The operating conditions of the OPAMP under various circumstances as mentioned by [15] were key inputs for our method. The challenges of existing methods and future directions for further innovations as mentioned by Mutlu et al., [16] were instrumental in the realization of the new method, we have built. The ideas of parallel sorting as mentioned by Bitton et al., [17] were the methods which could offer a clear advantage in performance. We saw few challenges with respect to scaling for huge data processing requirements. The hardware acceleration methods as mentioned by Zurek et al., [18] were useful. However we saw that such possibilities are always attached to increased cost. And unless there is a fundamental change to the sorting method, an exponential speed-up could not be achieved.

As we have used the circuit element OPAMP in our methods. The key details outlined by Waltari et al., [19] for this circuit element were very useful. A hybrid approach as mentioned by Jan et al., [20] gave key insights to address challenges in sorting upfront.

A clear scope for improvement in the area of sorting & searching and the components of a computer which determine how efficiently this can be made as mentioned by Li et al., [13] shows the evolution in approaches for challenges in the field of computer science.

Methods to imrove sorting & searching efficiency via Hardware design & optimization methods [21] – [25] helped narrow down the approach for solving the challenges in sorting & searching.

In this paper we demonstrate the sorting method by using the properties of electric current flow in a customized circuit. The flow of current adhering to the existing laws, is used to simulate data sorting without comparison. We are using the fundamental property of flow of electricity and the conditions in which it will flow. i.e a potential difference or presence of potential gradeint. We demonstrate the working using simulator tools.

Having seen the evolutions in the field of computer science, and after verifying various methos applied for imroving the data sorting efficiencies. We have chosen the "DataSorting" as a special case. And we are focussing on solving the efficieny problems by significantly improving the sort operations.

The key aspects as mentioned by Chen et al [2], are useful in order to come up with new sorting methods which can improve efficiency by a large factor.

Approach :

Unlike the typical sorting where in the terms are compared and stored in the memory. We are introduing a specialised optimization by adding Sorting memory. This memory sorts data as soon as the data is entered. i.e if we enter 100 terms in a sorting memory of capacity 1000 terms. The 100 data terms will be already in a sorted order. And the remaining 900 data terms space is empty. Though this may not be the best approach with respect to memory utilization perspective. This will bring exponential speed in terms of performance of the application for the use cases where sorting is inevitable. It is always done instantly, with no additional sorting operation.

The potential difference is mapped to constituent terms. The descrete values of volt are assigned to data terms when they are entered. i.e We demonstrate the sorting of numbers by mapping each of these with a specific value such that there exists a potential difference, among them. This potential difference is chosen such that we have the satisfying conditions for the demonstration of the data sorting of these numbers in real time. We have demonstrated the sorting operations for a the numbers 1, 2 & 3(could be entered in any order).This principle can be extended for sorting text data(non-numeric) as well. Thereby it is possible to sort a large amount of data. This is limited only by the scale of this new hardware setup. In temporal terms the sorting happens at the speed of light. Since the data automatically knows its sorted position due to its corelation with the potential difference. As the current flow from the lower potential to its higher potential. The curent passage is blocked to any other memory paths by the elelctrical proprty of current and not by any explict logic / check gate. Thus, each element undergoes a single task of storing once, leading to a time complexity of O(1) or constant time because the operation only happens once, and they do not depend on the size of the input as they run.

Unlike typical data sorting, the sorting is carried out in a dedicated **SortingMemory**. The design of the SortingRAM differs from other types of memory disks, in its design and functionality. The Memory registers are arranged with potential grader. The inputs are statically separated to give higher current/potential multipliers. The mapping is done in the potential difference grader.

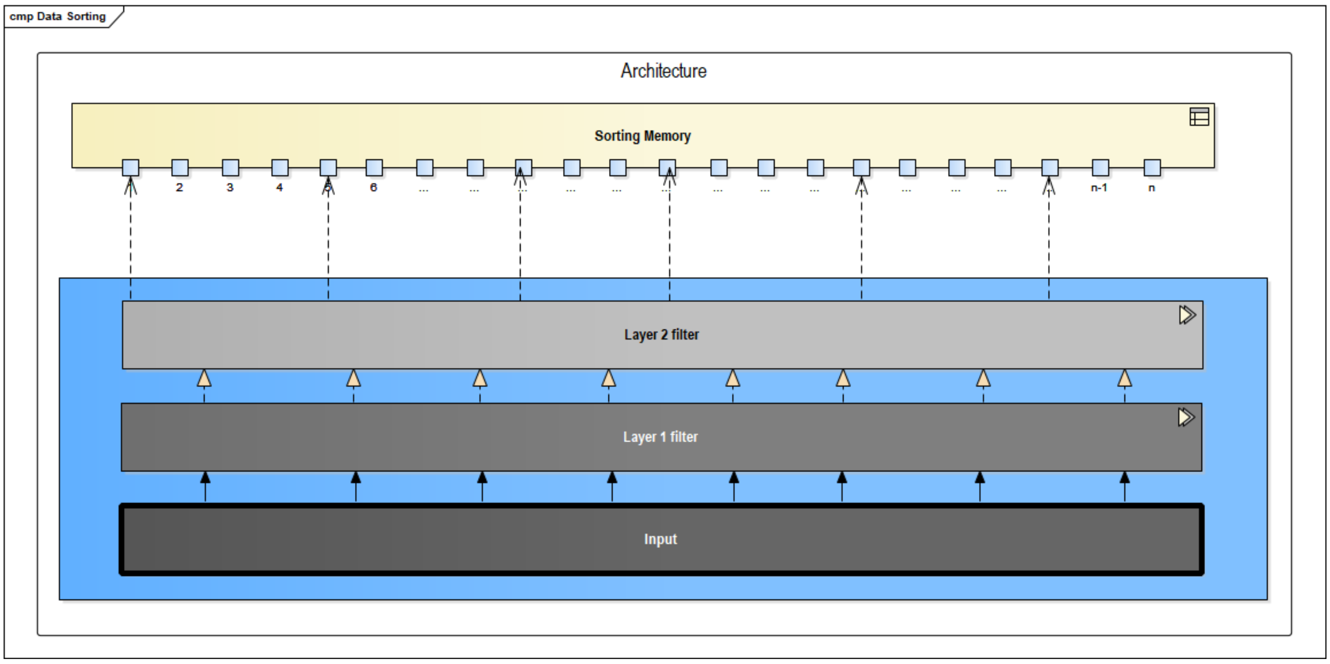


Figure 1 - Data sorting design template.

The circuit design considers sorting of the numbers 1, 2 & 3 for the demonstration of the sorting method. The circuit contains OPAMP and inverted OPAMP elements. The sorting is demonstrated by providing input corresponding to the nos 1, 2 & 3 numbers. We have taken the voltage at the input as 1V, 2V & 3V respectively. This is demonstrated by lighting the LED corresponding to the inputted no. The lighting of the LED confirms that the data has reached that part of the circuit. The Figures 5, 6 & 7 demonstrate the sorting operations for the numbers 1,2 & 3 by lighting the LEDs corresponding to these numbers.

The potential difference in terms of volt is mapped to constituent terms. The descrete values of volt are assigned to data terms when they are entered. i.e for the number n the **volt(Vin)** value will be n Volts and the destination **volt(Vend)** value will also be n Volts. A circuit component called OPAMP is introduced in between Vin and Vend which will amplify Vin to create the required potential difference. Due to the amplified voltage being more than **Vend**, potential difference is created which leads to current flowing through this channel and automtically the memory unit representing this location storing the value n.

Note: For the current to flow or for an LED to light up indicating the storage, a minimum of 3V potential difference is necessary.

The core principle here is to have the current flow from one end of the circuit (input) to the other end (memory storage). And we would like to establish the potential difference so that the current flows as per our planned path/route.

Take into account that all Vend voltages for the 3 numbers are present with 1 common Vin voltage which will vary.

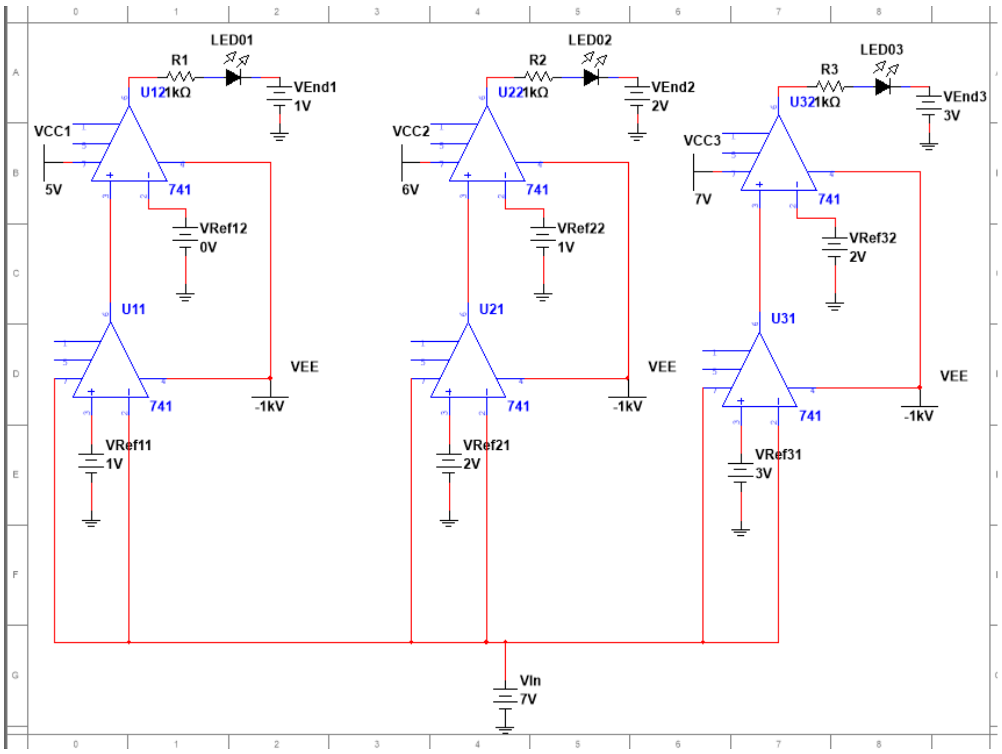


Figure 2 - Data sorting design template.

List of basic components used:

Board, Clock, Mode Register, Memory Banks, Burst Counter etc

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 September 2021 doi:10.20944/preprints202109.0174.v1

Additional components:

DC Voltage source and Ground	: Input and end potentials of each number
LEDs	: Representing successful storage of each number
Resistor	: A series resistor makes sure any small differences in voltage have just a negligible effect on the LED's current
Cables	: Connections

Inverting OPAMP :

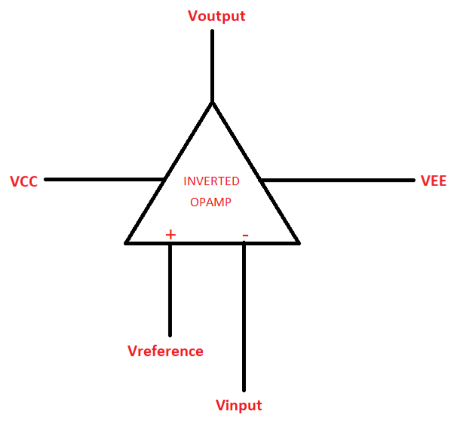


Figure 3 - Inverting OPAMP.

An operational amplifier has a property of conditional voltage flow which is used in our circuit. In an inverting OPAMP, Vin is connected to negative terminal and Vref is connected to positive terminal. Internally the inverting OPAMP performs an additional operation of these values : (+Vref) + (-Vin). If the result is 0 or +ve, value of VCC will be the new output voltage sent through the Vout path. Likewise, if the result is –ve, VEE is considered which is connected to the GND(ground).

Therefore, only if Vin<=Vref flow of voltage and current in the circuit continues.

For instance

1. If Vref=1V and Vin=1V , result =(+1)+(-1) = 0 Therefore VCC is triggered and Vout will be the value of VCC
2. If Vref=1V and Vin=3V , result =(+1)+(-3) = -2 Therefore VEE is triggered and there is no flow of further voltage.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 September 2021 doi:10.20944/preprints202109.0174.v1

3. If Vref=2V and Vin=1V , result =(+2)+(-1) = 1 Therefore VCC is triggered and Vout will be the value of VCC

Non-Inverting OPAMP :

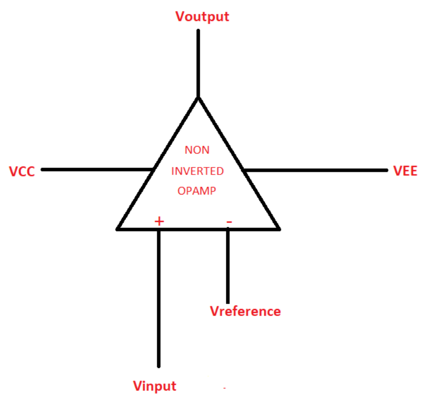


Figure 4 - Non-Inverting OPAMP.

An operational amplifier also has a property of amplifying the voltage along with conditional voltage flow which is used in our circuit. In a non-inverting OPAMP, Vin is connected to positive terminal and Vref is connected to negative terminal. Internally the inverting OPAMP performs an additional operation of these values : (+Vin) + (-Vref). If the result is +ve, value of VCC will be the new output voltage sent through the Vout path. Likewise, if the result is 0 or –ve, VEE is considered which is connected to the GND(ground).

Therefore, only if Vin>Vref flow of voltage and current in the circuit continues

For instance

1. If Vin=1V and Vref=0V, result =(+1)+(-0) = 1 Therefore VCC is triggered and Vout will be the value of VCC
2. If Vin=1V and Vref=1V, result =(+1)+(-1) = 0 Therefore VEE is triggered and there is no flow of further voltage.
3. If Vin=1V and Vref=2V, result =(+1)+(-2) = -1 Therefore VEE is triggered and there is no flow of further voltage.

Let’s consider 3 numbers - 1, 2 and 3.

2.1. First Scenario – Sorting the numeric 1.

Beginning with the number 1 i.e , to store number 1 which is represented by lighting up of LED 1. Considering left section of the circuit in *Figure 5*, Vin will be 1V and Vend1 will also be 1V. As yellow path represents the flow of current, it first encounters an inverting OPAMP first . Here, Vin = 1V , Vref is also set to 1V and VCC is also connected to Vin which makes VCC=1V as well . Since Vin<=Vref, condition(Vin<=Vref) is satisfied, value of VCC will be sent as Vout i.e, 1V. Next, the path encounters a non-inverting OPAMP. Here, the Vin value will be the output from previous OPAMP i.e new Vin=1V, Vref is set to 0V and VCC is set to 5V. Since Vin>Vref, condition(Vin>Vref) is satisfied, value of VCC will be sent as Vout i.e, 5V, but the OPAMP offers an internal resistance due to which Vout will be around 1V less than VCC which makes Vout=4V. Finally, the path leads to an LED and an end potential Vend1 of 1V. The output of non-inverting OPAMP will be the new start potential which is 4V. As discussed in the beginning, a minimum potential difference of 3V is required between the start and end potential for proper amount of current to flow and light up the LED (indicating storage of the number). The current potential difference will be (start potential – end potential) i.e, 4V-1V =3V which satisfies the requirement, considering the nature of current, it flows until the end lighting up LED1.

However, considering the middle section of the circuit in *Figure 5*, Vin will be 1V and Vend2 will be 2V. As yellow path represents the flow of current, it first encounters an inverted OPAMP first . Here, Vin = 1V , Vref is set to 2V and VCC is also connected to Vin which makes VCC=1V as well . Since Vin<=Vref, condition(Vin<=Vref) is satisfied, value of VCC will be sent as Vout i.e, 1V. Next, the path encounters a non-inverting OPAMP. Here, the Vin value will be the output from previous OPAMP i.e new Vin=1V, Vref is set to 1V. Since Vin=Vref, the condition(Vin>Vref) fails and VEE is considered which is connected to the GND(ground).

Similarly, considering the right section of the circuit in *Figure 5*, Vin will be 1V and Vend3 will be 3V. As yellow path represents the flow of current, it first encounters an inverted OPAMP first . Here, Vin = 1V , Vref is set to 3V and VCC is also connected to Vin which makes VCC=1V as well . Since Vin<=Vref, condition(Vin<=Vref) is satisfied, value of VCC will be sent as Vout i.e, 1V. Next, the path encounters a non-inverting OPAMP. Here, the Vin value will be the output from previous OPAMP i.e new Vin=1V, Vref is set to 2V. Since Vin<Vref, the condition(Vin>Vref) fails and VEE is considered which is connected to the GND(ground).

Thus, when Vin=1V, LED1 lights up indicating memory storage of the number 1.

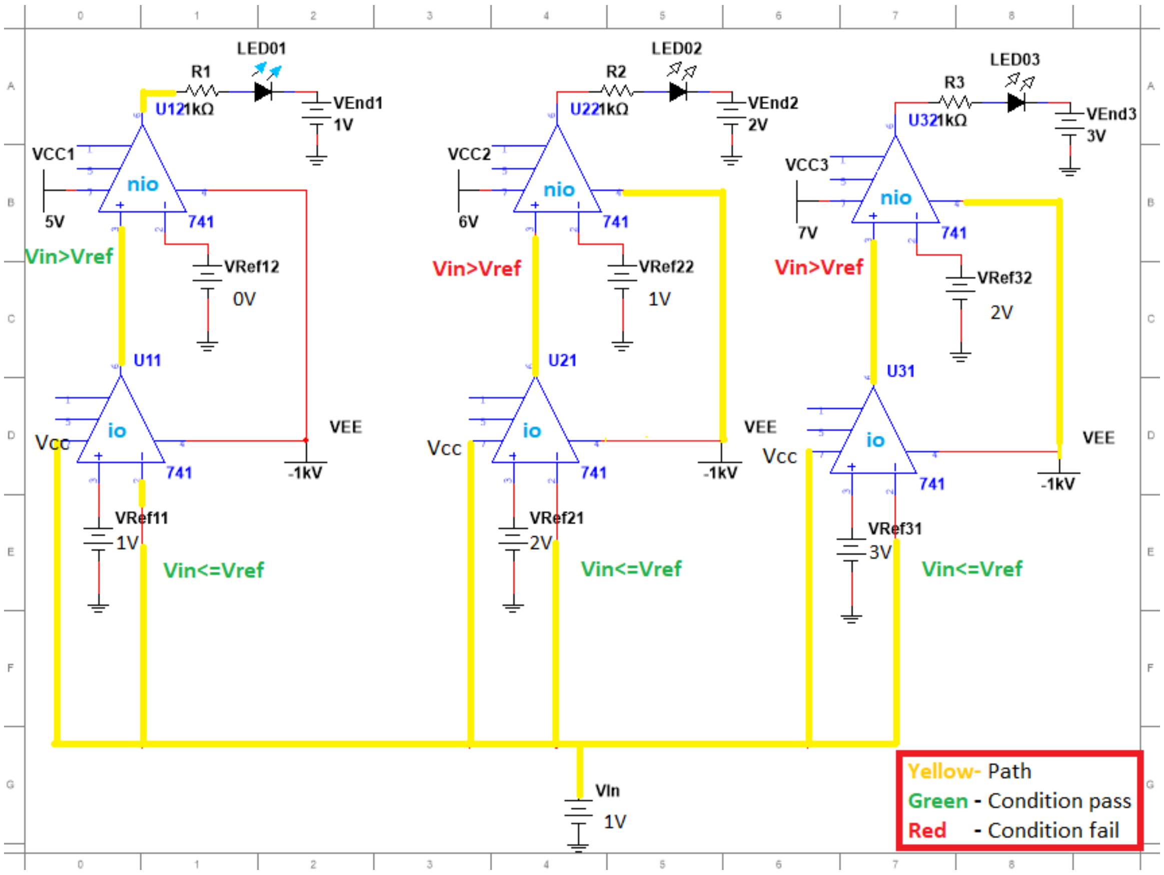


Figure 5 - Circuit representing the sorting of no 1.

No	Vin (Volts)	OPAMP -1 (inverting)					OPAMP-2 (non-inverting)					Vstart (Volts)	Vend (Volts)
		Vin	Vref	Vcc	Vee	Vout	Vin	Vref	Vcc	Vee	Vout		
1	1	1	1	1	X	1	1	0	5	X	4	4	1
	1	1	2	1	X	1	1	1	X	GND			2
	1	1	3	1	X	1	1	2	X	GND			3

Table 1 - Shows the Values for voltage at various points in the circuit. Finally storing no 1.

2.2. Second Scenario – Sorting the numeric 2.

Next for the number 2 i.e , to store number 2 which is represented by lighting up of LED 2. Considering middle section of the circuit in *Figure 6*, Vin will be 2V and Vend2 will also be 2V. As yellow path represents the flow of current, it first encounters an inverting OPAMP first . Here, Vin = 2V , Vref is also set to 2V and VCC is also connected to Vin which makes VCC=2V as well . Since Vin<=Vref, condition(Vin<=Vref) is satisfied, value of VCC will be sent as Vout i.e, 2V. Next, the path encounters a non-inverting OPAMP. Here, the Vin value will be the output from previous OPAMP i.e new Vin=2V, Vref is set to 1V and VCC is set to 6V. Since Vin>Vref, condition(Vin>Vref) is satisfied, value of VCC will be sent as Vout i.e, 6V, but the OPAMP offers an internal resistance due to which Vout will be around 1V less than VCC which makes Vout=5V. Lastly, the path leads to an LED and an end potential Vend2 of 2V. The output of non-inverting OPAMP will be the new start potential which is 5V. As discussed in the beginning, a minimum potential difference of 3V is required between the start and end potential for proper amount of current to flow and light up the LED indicating storage of the number. The current potential difference will be (start potential – end potential) i.e, 5V-2V =3V which satisfies the requirement, considering the nature of current, it flows until the end thereby lighting up LED2.

However, considering the left section of the circuit in *Figure 6*, Vin will be 2V and Vend1 will be 1V. As yellow path represents the flow of current, it first encounters an inverted OPAMP first . Here, Vin = 2V , Vref is set to 1V. Since Vin>Vref, the condition(Vin<=Vref) fails and VEE is considered which is connected to the GND(ground).

Similarly, considering the right section of the circuit in *Figure 6*, Vin will be 2V and Vend3 will be 3V. As yellow path represents the flow of current, it first encounters an inverted OPAMP first . Here, Vin = 2V , Vref is set to 3V and VCC is also connected to Vin which makes VCC=2V as well . Since Vin<=Vref, condition(Vin<=Vref) is satisfied, value of VCC will be sent as Vout i.e, 2V. Next, the path encounters a non-inverting OPAMP. Here, the Vin value will be the output from previous OPAMP i.e new Vin=2V, Vref is set to 2V. Since Vin=Vref, the condition(Vin>Vref) fails and VEE is considered which is connected to the GND(ground).

Thus, when Vin=2V LED2 lights up indicating memory storage of the number 2

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 September 2021 doi:10.20944/preprints202109.0174.v1

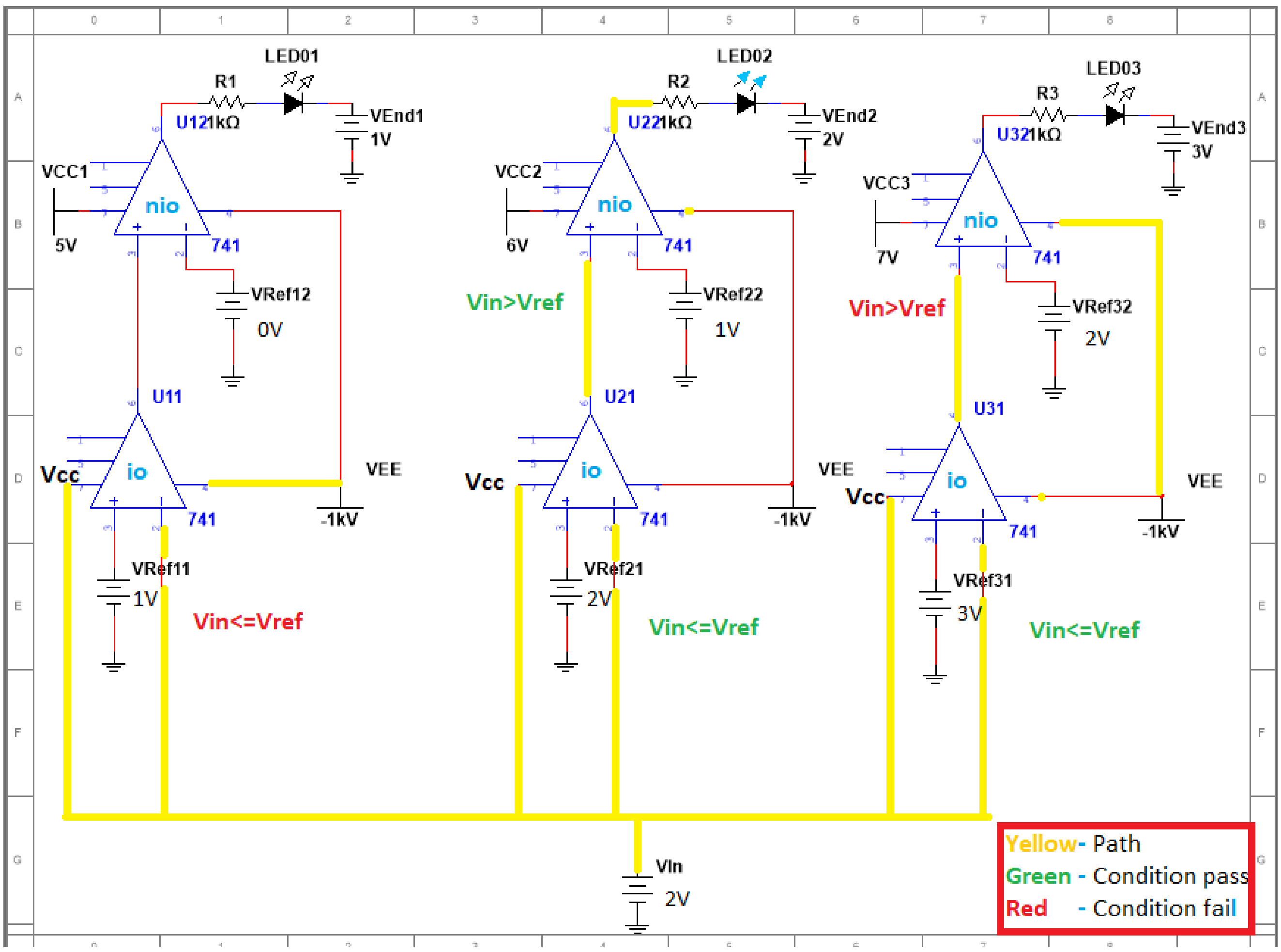


Figure 6 - Circuit representing the sorting of no 2.

No	Vin (Volts)	OPAMP -1 (inverting)					OPAMP-2 (non-inverting)					Vstart (Volts)	Vend (Volts)
		Vin	Vref	Vcc	Vee	Vout	Vin	Vref	Vcc	Vee	Vout		
2	2	2	1	X	GND			0					1
	2	2	2	2	X	2	2	1	6	X	5	5	**2**
	2	2	3	2	X	2	2	2	X	GND			3

Table 1 - Shows the Values for voltage at various points in the circuit. Finally storing no 2.

2.3. Third Scenario – Sorting the numeric 3.

Lastly for the number 3 i.e , to store number 3 which is represented by lighting up of LED 3. Considering right section of the circuit in *Figure 7*, Vin will be 3V and Vend3 will also be 3V. As yellow path represents the flow of current, it first encounters an inverting OPAMP first . Here, Vin = 3V , Vref is also set to 3V and VCC is also connected to Vin which makes VCC=3V as well . Since Vin<=Vref, condition(Vin<=Vref) is satisfied, value of VCC will be sent as Vout i.e, 3V. Next, the path encounters a non-inverting OPAMP. Here, the Vin value will be the output from previous OPAMP i.e new Vin=3V, Vref is set to 2V and VCC is set to 7V. Since Vin>Vref, condition(Vin>Vref) is satisfied, value of VCC will be sent as Vout i.e, 7V, but the OPAMP offers an internal resistance due to which Vout will be around 1V less than VCC which makes Vout=6V. Lastly, the path leads to an LED and an end potential Vend3 of 3V. The output of non-inverting OPAMP will be the new start potential which is 6V. As discussed in the beginning, a minimum potential difference of 3V is required between the start and end potential for proper amount of current to flow and light up the LED indicating storage of the number. The current potential difference will be (start potential – end potential) i.e, 6V-3V =3V which satisfies the requirement, considering the nature of current, it flows until the end lighting up LED3.

However, considering the left section of the circuit in *Figure 7*, Vin will be 3V and Vend1 will be 1V. As yellow path represents the flow of current, it first encounters an inverted OPAMP first . Here, Vin = 3V , Vref is set to 1V. Since Vin>Vref, the condition(Vin<=Vref) fails and VEE is considered which is connected to the GND(ground).

Similarly, considering the middle section of the circuit in *Figure 7*, Vin will be 3V and Vend2 will be 2V. As yellow path represents the flow of current, it first encounters an inverted OPAMP first . Here, Vin = 3V , Vref is set to 2V. Since Vin>Vref, the condition(Vin<=Vref) fails and VEE is considered which is connected to the GND(ground).

Thus, when Vin=3V LED3 lights up indicating memory storage of the number 3

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 September 2021 doi:10.20944/preprints202109.0174.v1

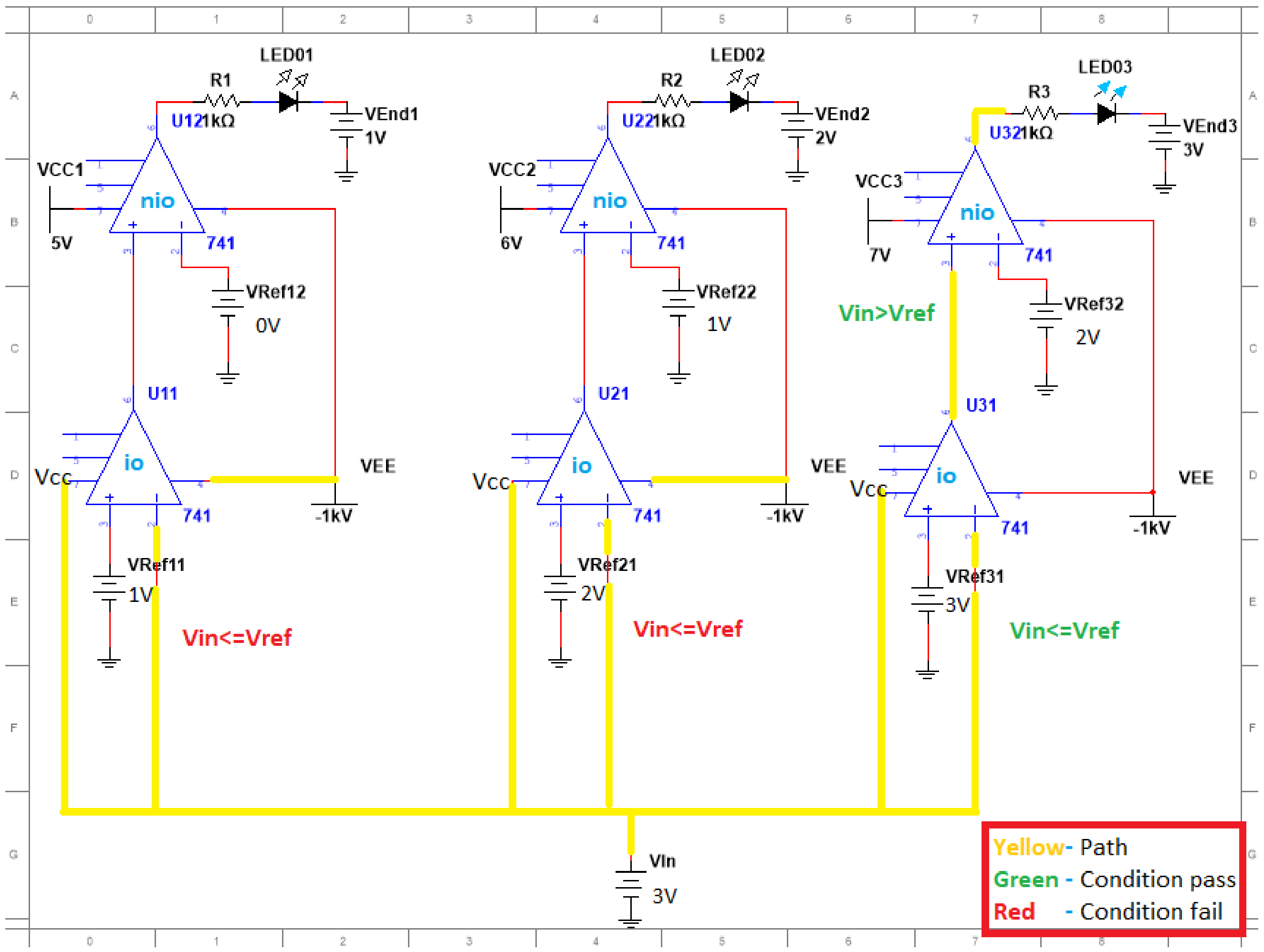


Figure 7 - Circuit representing the sorting of no 3.

o	Vin (Volts)	OPAMP -1 (inverting)					OPAMP-2 (non-inverting)					Vstart (Volts)	Vend (Volts)
		Vin	Vref	Vcc	Vee	Vout	Vin	Vref	Vcc	Vee	Vout		
3	3	3	1	X	GND			0					1
	3	3	2	X	GND			1					2
	3	3	3	3	X	3	3	2	7	X	6	6	3

Table 1 - Shows the Values for voltage at various points in the circuit. Finally storing no 3.

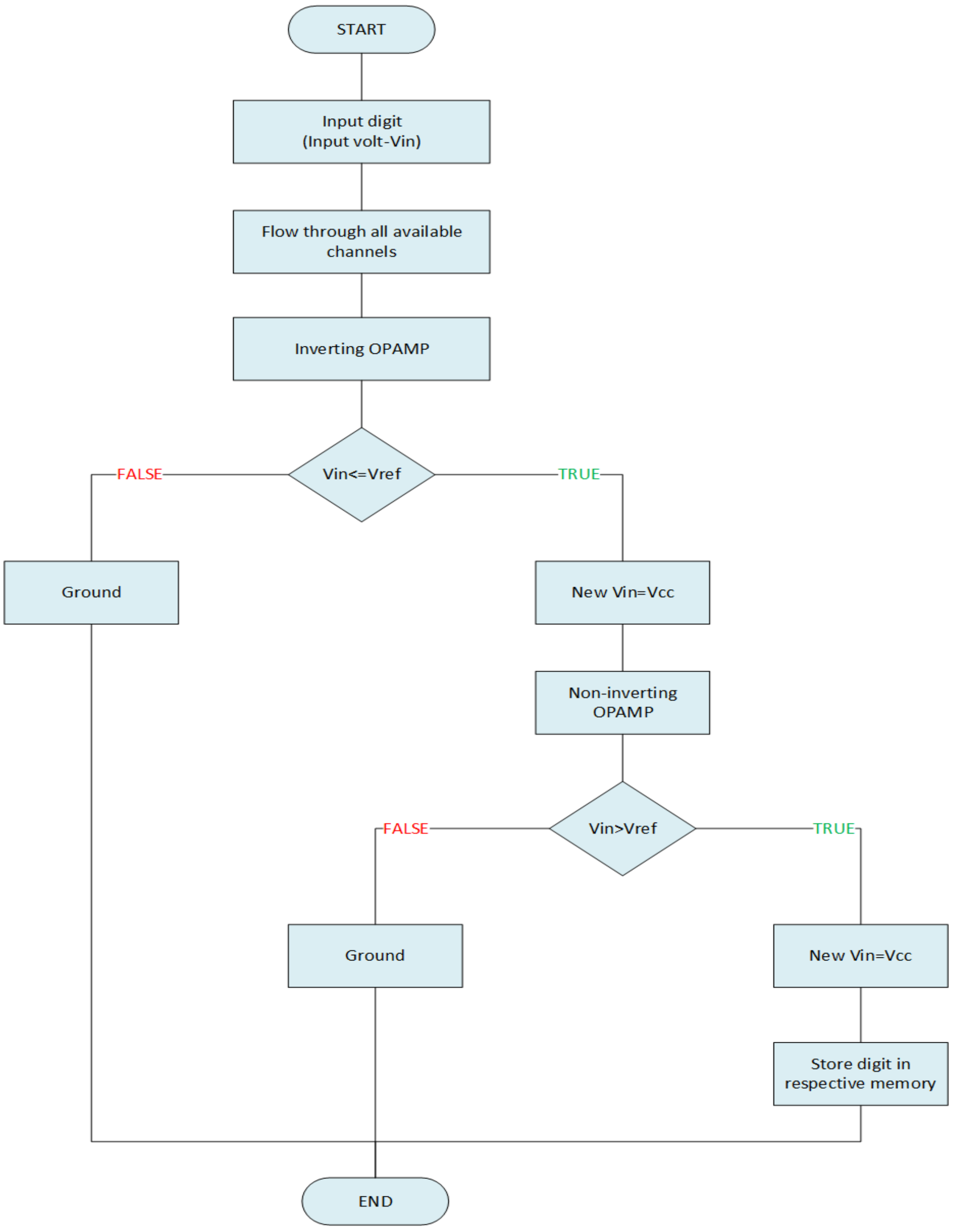
Note:

1. Only Vin needs to be changed according to the input number.
2. Each location or LED indicates storage for a number. And n will have two OPAMPs (inverting and non-inverting) associated with fixed Vref and Vee

3. Vend will be n Volts.

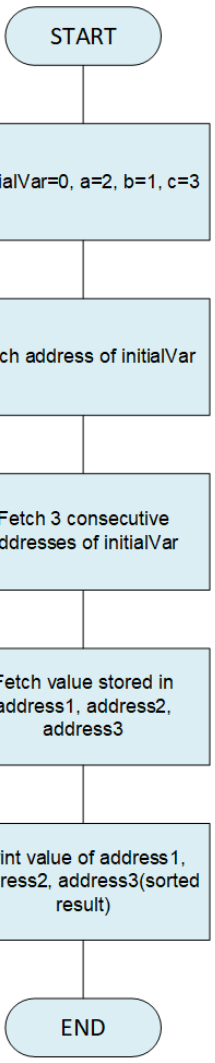
FLOWCHART:

Overview-



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 September 2021 doi:10.20944/preprints202109.0174.v1

Algorithm-


START
initialVar=0, a=2, b=1, c=3
Fetch address of initialVar
Fetch 3 consecutive addresses of initialVar
Fetch value stored in address1, address2, address3
Print value of address1, address2, address3(sorted result)
END

ALGORITHM (With 3 numbers):

Step 1: Start

Step 2: initialVar=0, a=2, b=1, c=3

Step 3: Fetch address of initialVar (* initialVar =& initialVar)

Step 4: Fetch 3 consecutive addresses of initialVar (address1= * initialVar ++, address2= * initialVar +2, address3= * initialVar +3)

Step 5: Fetch value stored in address1, address2, address3

Step 6: Print value of address1, address2, address3(sorted result)

Step 7: Stop

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 September 2021 doi:10.20944/preprints202109.0174.v1

PROGRAM (With 3 numbers- integer sorting RAM):

```
main()

{

int initialVar=0, a=2, b=1, c=3;

int *initialVarAddress;

initialVarAddress=& initialVar;

printf("Fetch address of first variable- initialVar = %d", initialVarAddress);

int address1= (int)initialVarAddress + 4;

int address2= (int)initialVarAddress + 8;

int address3= (int)initialVarAddress +12;

int *p1,*p2,*p3;

p1=(int *)address1;

p2=(int *)address2;

p3=(int *)address3;

int lowest = *p1;

int mid= *p2;

int highest = *p3;

printf("Sorted result= %d , %d, %d",    lowest, mid, highest);

}
```

Time Complexity:

Since the procedure does not involve comparsion or looping, all the actions take place just once. Hence, the time complexity will be equal to $\Omega(1)$ for Best case, $\Theta(1)$ for average case and O(1) for worst case.

Space complexity:

S(p) = A + Sp(I)

= fixed part + variable part

= 1 + n (Consider n=3 as in current example)

= 4

(4) * data type memory storage

= (4) * 4 (Integer- 2 or 4. Let's assume 4)

= (4) * 4 =16 bytes

Since the procedure too does not involve comparsion or looping, all the actions take place just once. Hence, the space complexity will be equal to O(1).

COMPARISON:

Algorithm	**Time Complexity**			**Space Complexity**
	Best Case	**Average Case**	**Worst Case**	**Worst Case**
Bubble Sort	Ω(N)	$\Theta(N^2)$	$O(N^2)$	O(1)
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$	O(1)
Insertion Sort	Ω(N)	$\Theta(N^2)$	$O(N^2)$	O(1)
Merge Sort	Ω(N log N)	Θ(N log N)	O(N log N)	O(N)
Heap Sort	Ω(N log N)	Θ(N log N)	O(N log N)	O(1)
Quick Sort	Ω(N log N)	Θ(N log N)	$O(N^2)$	O(N log N)

Algorithm	**Time Complexity**			**Space Complexity**
	Best Case	Average Case	Worst Case	Worst Case
Proposed algorithm	Ω(1)	Θ(1)	O(1)	O(1)

2.4. Method for searching a data term.

As it might be clear now, though this isn't mentioned explicitly. That, this method is useful for searching a data item inside the already stored dataset. i.e the search item would also pass through the circuit and would follow it's applicable path. And this could be acknowledged back by the receiving end, by sending the signal back to the input element confirming that the search item is present.

In the proposed sorting RAM, since each element or digit has a designated memory address, one can search for the value present in respective address and if the result is not empty or null, the element is declared present and the search is successful.

Example:

Consider we have a sorting RAM for 10 designated integers and initialVar=0 to be the element stored in the very first memory address.

Corresponsing memory addresses will belong to the remaining numbers in order.

Thus respective memory addresses for each number will be known by incrementing each address by 4 bytes since they involve integers.

To search for a number, its designated memory address can be checked for a value and if not empty can be declared as present.

To search for the number: 1

```
main()
{ int initialVar=0;
int *initialVarAddress;
initialVarAddress=& initialVar;
printf("Fetch address of first variable- initialVar = %d", initialVarAddress);
int address1= (int)initialVarAddress + 4;
p1=(int *)address1;
int lowest = *p1;
if(lowest!=null){
printf("Number is present");
}else{
printf("Number is not present"); }
}
```

3. Results

The sorting or positioning of the inputted value which happens instantaneously is inspired by the Quantum computing, the computing realm offering exponential speed. While the quantum computers offer great advantages over the classical computers. The quest for quantum supremacy is still actively pursued. We are yet to evidence a useful application from the quantum computer. While there are challenges in building a quantum computer with sufficnet no of qubits. We could try to get a method for improved speed compared to a classical method for one of the applications of the computer.

As there are numerous applications in the areas such as Optimization problems and several applications in Big data. And since conventional approaches have practical limitation of efficiencies in seraching a given data term. This method offers exponential speed in sorting & searching data. Though there is additional cost involved to have dedicated memory for specific data. Such approach will be more suitable for some specific use cases if not for all scenarios.

We see that, the method demonstrated here can be used to prepare an input state for a complex quantum problem with greater speed.

4. References

Future oppurtunities:

1. Hayes et al., 2015 T. Hayes, O. Palomar, O. Unsal, A. Cristal and M. Valero, "VSR sort: A novel vectorised sorting algorithm & architecture extensions for future microprocessors,"

2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), 2015, pp. 26-38, doi: 10.1109/HPCA.2015.7056019.

2. Chen et al., 2020 Wenhan Chen, Yang Liu , Zhiguang Chen, Fang Liu , Nong Xiao, ”External Sorting Algorithm: State-of-the-Art and Future Directions” ,IOP Conf. Series: Materials Science and Engineering 806 (2020) 012040 IOP Publishing doi:10.1088/1757-899X/806/1/012040

Similar papers:

3. Jmaa et al., 2017 Y. B. Jmaa, K. M. A. Ali, D. Duvivier, M. B. Jemaa and R. B. Atitallah, "An Efficient Hardware Implementation of TimSort and MergeSort Algorithms Using High Level Synthesis," 2017 International Conference on High Performance Computing & Simulation (HPCS), 2017, pp. 580-587, doi: 10.1109/HPCS.2017.92.

4. Abdel-Hafeez et al., 2017 S. Abdel-Hafeez and A. Gordon-Ross, "An Efficient O(N) Comparison-Free Sorting Algorithm," in IEEE Transactions on Very Large Scale

Integration (VLSI) Systems, vol. 25, no. 6, pp. 1930-1942, June 2017, doi: 10.1109/TVLSI.2017.2661746.

5. Gowtham et al., 2020 A.S. Gowtham , Dr. D. Jaya Kumar,MTech., Ph.D. (2020) IMPLEMENTATION OF SORTING OF ONE-DIMENSIONAL ARRAY USING RAM BASED SORTING ALGORITHM. *JCR*, 7 (15), 5904-5914. doi:10.31838/jcr.07.15.763

Trigger for innovation:

6. Karunanithi et al., 2014 Karunanithi, Ashok Kumar. "A survey, discussion and comparison of sorting algorithms." *Department of Computing Science, Umea University* (2014).doi: 10.1.1.570.2322

7. Mishra et al., 2008 Mishra, Aditya Dev, and Deepak Garg. "Selection of best sorting algorithm." *International Journal of intelligent information Processing* 2.2 (2008): 363-368.

8. Estivill-Castro wtal., 1992 Vladmir Estivill-Castro and Derick Wood. 1992. A survey of adaptive sorting algorithms. ACM Comput. Surv. 24, 4 (Dec. 1992), 441–476. DOI:https://doi.org/10.1145/146370.146381

9. Idrizi etal., 2017 F. Idrizi, A. Rustemi and F. Dalipi, "A new modified sorting algorithm: A comparison with state of the art," 2017 6th Mediterranean Conference on Embedded Computing (MECO), 2017, pp. 1-6, doi: 10.1109/MECO.2017.7977252.

10. Wilkes et al., 1974 M. V. Wilkes, The Art of Computer Programming, Volume 3, Sorting and Searching, *The Computer Journal*, Volume 17, Issue 4, November 1974, Page 324, https://doi.org/10.1093/comjnl/17.4.324

11. Meena et al., 2014 Meena, J.S., Sze, S.M., Chand, U. *et al.* Overview of emerging non-volatile memory technologies. *Nanoscale Res Lett* **9,** 526 (2014). https://doi.org/10.1186/1556-276X-9-526

12. Arge et al., 2013 Arge L., Thorup M. (2013) RAM-Efficient External Memory Sorting. In: Cai L., Cheng SW., Lam TW. (eds) Algorithms and Computation. ISAAC 2013. Lecture Notes in Computer Science, vol 8283. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-45030-3_46

13. Li et al., 2010 H. Li and Y. Chen, "Emerging non-volatile memory technologies: From materials, to device, circuit, and architecture," 2010 53rd IEEE International Midwest Symposium on Circuits and Systems, 2010, pp. 1-4, doi: 10.1109/MWSCAS.2010.5548590.

14. Solomon et al., 1974 J. E. Solomon, "The monolithic op amp: a tutorial study," in IEEE Journal of Solid-State Circuits, vol. 9, no. 6, pp. 314-332, Dec. 1974, doi: 10.1109/JSSC.1974.1050524.

15. Karthikeyan et al., 2000 S. Karthikeyan, S. Mortezapour, A. Tammineedi and E. K. F. Lee, "Low-voltage analog circuit design based on biased inverting opamp configuration," in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 47, no. 3, pp. 176-184, March 2000, doi: 10.1109/82.826743.

16. Mutlu et al., 2015 Mutlu O. (2015) Main Memory Scaling: Challenges and Solution Directions. In: Topaloglu R. (eds) More than Moore Technologies for Next Generation

Computer Design. Springer, New York, NY. https://doi.org/10.1007/978-1-4939-2163-8_6

17. Bitton et al., 1984 Dina Bitton, David J. DeWitt, David K. Hsaio, and Jaishankar Menon. 1984. A taxonomy of parallel sorting. ACM Comput. Surv. 16, 3 (Sept. 1984), 287–318. DOI:https://doi.org/10.1145/2514.2516

18. Zurek et al., 2013 Dominik Zurek, Marcin Pietron, Maciej Wielgosz, Kazimierz Wiatr, " Comparison of Hybrid Sorting Algorithms Implemented on Different Parallel Hardware Platforms"*ACC AGH Cyfronet, Krakow, Poland*Computer Science, 14 (4), 2013DOI:10.7494/csci.2013.14.4.679

19. Waltari et al., 1999 M. Waltari and K. Halonen, "A switched-opamp with fast common mode feedback," ICECS'99. Proceedings of ICECS '99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.99EX357), 1999, pp. 1523-1525 vol.3, doi: 10.1109/ICECS.1999.814460.

20. Jan et al., 2012 Bilal Jan, Bartolomeo Montrucchio , Carlo Ragusa , Fiaz Gul Khan and Omar Khan, "FAST PARALLEL SORTING ALGORITHMS ON GPUS" , DOI : 10.5121/ijdps.2012.3609

21. Alaparthi et al., 2009 S. Alaparthi, K. Gulati and S. P. Khatri, "Sorting binary numbers in hardware - A novel algorithm and its implementation," 2009 IEEE International Symposium on Circuits and Systems, 2009, pp. 2225-2228, doi: 10.1109/ISCAS.2009.5118240.

22. Matai et al., 2016 Janarbek Matai, Dustin Richmond, Dajung Lee, Zac Blair, Qiongzhi Wu, Amin Abazari, and Ryan Kastner. 2016. Resolve: Generation of High-Performance Sorting Architectures from High-Level Synthesis. In Proceedings of the 2016 ACM/SIGDA Interna-tional Symposium on Field-Programmable Gate Arrays (FPGA '16). Association for Com-puting Machinery, New York, NY, USA, 195–204. DOI:https://doi.org/10.1145/2847263.2847268

23. Bentley et al., 1997 Bentley, J.L. and Sedgewick, R., 1997, January. Fast algorithms for sorting and searching strings. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms* (pp. 360-369).
24. Xiao et al., 2000 Li Xiao, Xiaodong Zhang, and Stefan A. Kubricht. 2001. Improving memory performance of sorting algorithms. <i>ACM J. Exp. Algorithmics</i> 5 (2000), 3–es. DOI:https://doi.org/10.1145/351827.384245
25. Széll et al., 2006 A. Széll and B. Fehér, "Efficient sorting architectures in FPGA", *Proc. Int. Carpathian Control Conf. (ICCC)*, pp. 1-4, May 2006.