



**UNIVERSITY
OF MALAYA**

Department of Electrical Engineering

**Virtual Instrumentation
using LabVIEW**

Name: _____

Matric number: _____

LIST OF EXPERIMENTS

No.	Title
1	Basic arithmetic operations: addition, subtraction, multiplication and division
2	Boolean operations: AND, OR, NOT, XOR and NAND
3	Sum of 'n' numbers using 'for' loop
4	Factorial of a given number using 'for' loop
5	Determine square of a given number
6	Factorial of a given number using 'while' loop
7	Sorting even numbers using while loop in an array
8	Finding the array maximum and array minimum

Introduction to LabVIEW

Virtual Instrumentation:

Virtual instrumentation is the use of customizable software and modular measurement hardware to create user-defined measurement systems, called virtual instruments. Traditional hardware instrumentation systems are made up of pre-defined hardware components, such as digital multimeters and oscilloscopes that are completely specific to their stimulus, analysis, or measurement function. Because of their hard-coded function, these systems are more limited in their versatility than virtual instrumentation systems. The primary difference between hardware instrumentation and virtual instrumentation is that software is used to replace a large amount of hardware. The software enables complex and expensive hardware to be replaced by already purchased computer hardware; e. g. analog-to-digital converter can act as a hardware complement of a virtual oscilloscope, a potentiostat enables frequency response acquisition and analysis in electrochemical impedance spectroscopy with virtual instrumentation.

Layers of Virtual Instrumentation

- **Application Software:** Most people think immediately of the application software layer. This is the primary development environment for building an application.
- **Test and Data Management Software:** Above the application software layer the test executive and data management software layer. This layer of software incorporates all of the functionality developed by the application layer and provides system-wide data management.
- **Measurement and Control Services Software:** The last layer is often overlooked, yet critical to maintaining software development productivity.

LabVIEW

Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW) is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution. In LabVIEW, a user interface can be build by using a set of tools and objects. The user interface is known as the front panel. Then code can be added using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

LabVIEW programs are one of the suitable for virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. Every VI uses functions that manipulate input from the user interface or other sources and display that information or move it to other files or other computers.

A VI contains the following three components:

Front panel - Serves as the user interface. Fig 1 shows front panel example.

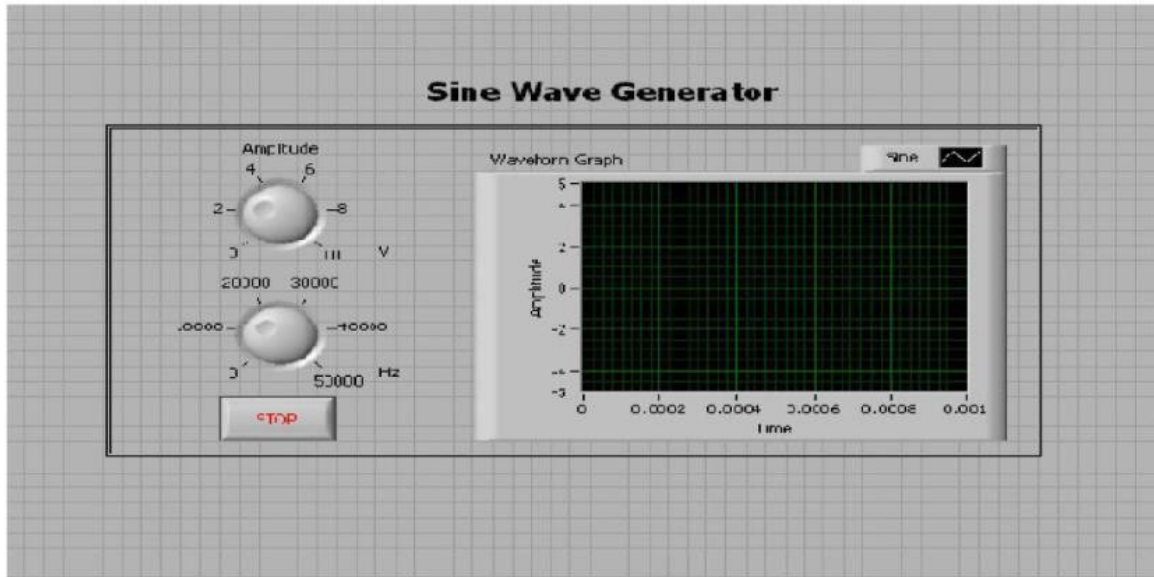


Fig 1. Front Panel of Virtual Instrumentation.vi

Block diagram - Contains the graphical source code that defines the functionality of the VI. Icon and connector panel - Identifies the VI so that the VI can be used in another VI. A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages. Fig 2 shows Block diagram of VI programming.

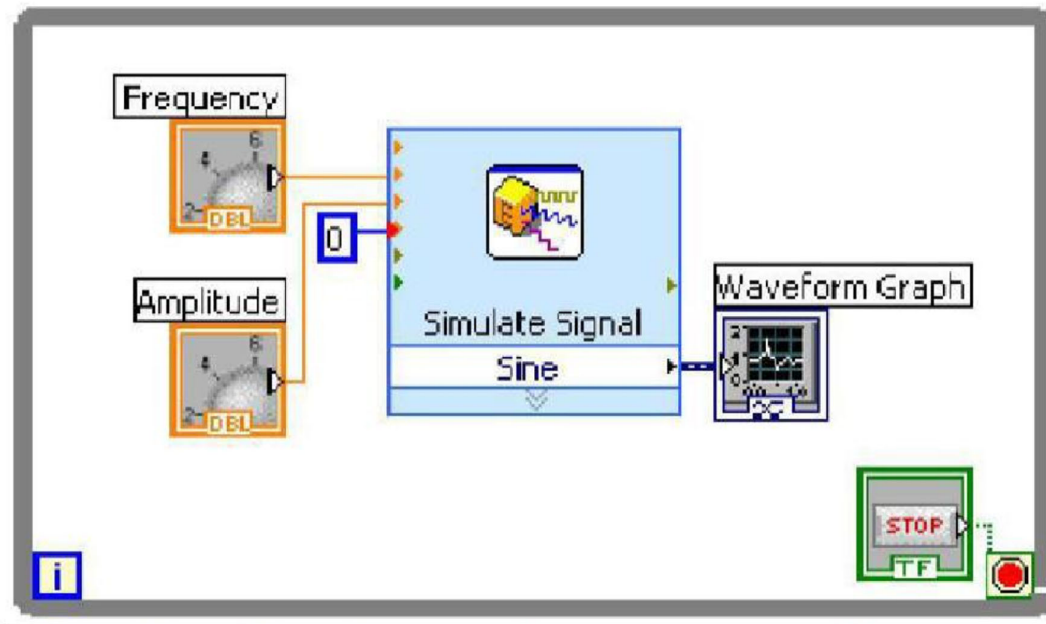


Fig 2. Block Diagram of Virtual Instrumentation.vi

The front panel is the user interface of the VI. The front panel is built with controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, pushbuttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates. After the front panel is built, add code using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code. Front panel objects appear as terminals on the block diagram.

Additionally, the block diagram contains functions and structures from built-in LabVIEW VI libraries. Wires connect each of the nodes on the block diagram, including control and indicator terminals, functions, and structures.

LabVIEW Palettes

LabVIEW palettes give provide the options needed to create and edit the front panel and block diagram. The Tools palette is available on the front panel and the block diagram. A tool is a special operating mode of the mouse cursor. By selecting a tool, the cursor icon changes to the tool icon. Use the tools to operate and modify front panel and block diagram objects.

Select Window » Show Tools Palette to display the Tools palette.

The Tools palette can be placed anywhere on the screen. If automatic tool selection is enabled and as the cursor is moved over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the Tools palette.

The Controls palette

The Controls palette is available only on the front panel. The Controls palette contains the controls and indicators used to create the front panel. to display the Controls palette, Select Window » Show

Controls Palette or right-click the front panel workspace. The Controls palette can be placed anywhere on the screen. Fig 3 shows control palette.

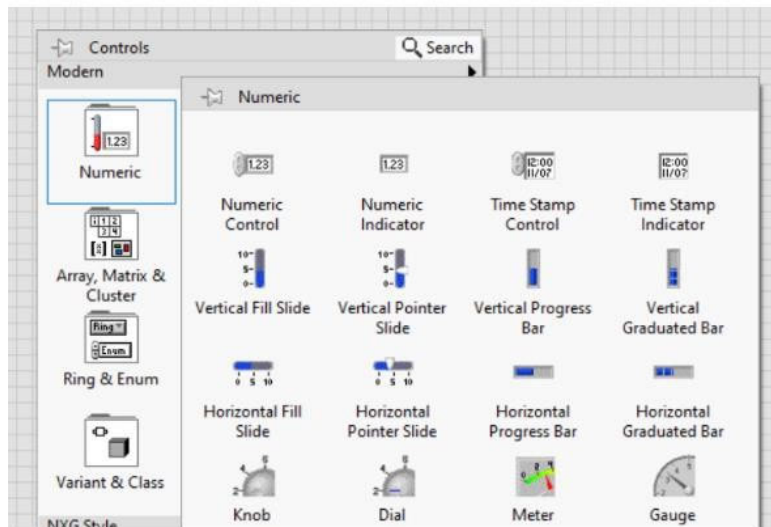


Fig 3: Controls Palette

The Functions palette

The Functions palette is available only on the block diagram. The Functions palette contains the VIs and functions used to build the block diagram. To display the Functions palette, select Window » Show Functions Palette or right-click the block diagram workspace. The Functions palette can be placed anywhere on the screen. Fig 4 shows Function palette of LabVIEW

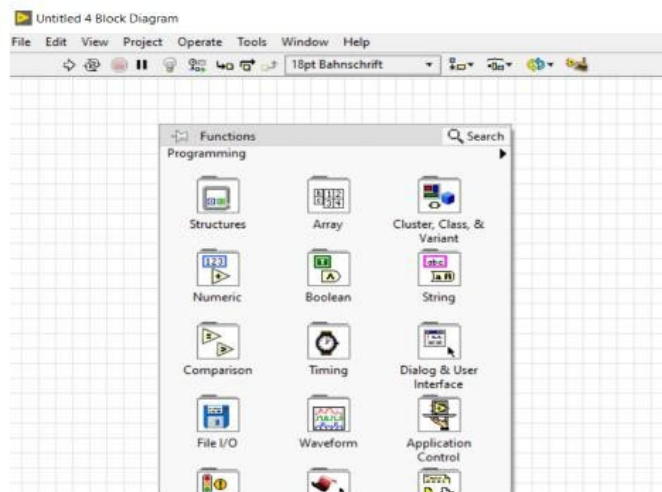


Fig 4: Functions Palette

Tips for Working in LabVIEW

- Keystroke Shortcuts

- <ctrl-H> – Activate/Deactivate Context Help Window
- <ctrl-B> – Remove Broken Wires From Block Diagram
- <ctrl-E> – Toggle Between Front Panel and Block Diagram
- <ctrl-z> – Undo (Also in Edit Menu)
- Tab Key – Toggle Through Tools on Toolbar
- Tools » Options... – Set Preferences in LabVIEW
- VI Properties – Configure VI Appearance, Documentation, etc.

Advantages of LabVIEW

- Graphical User Interface
- Drag-and-Drop built-in functions
- Modular design and hierarchical design
- Multiple high level development tools
- Professional Development Tools
- Multi platforms
- Reduces cost and preserves investment
- Flexibility and scalability
- Connectivity and instrument control

Creating a VI

Launch Labview: Fig 5 shows front and block diagram panel of LabVIEW files.

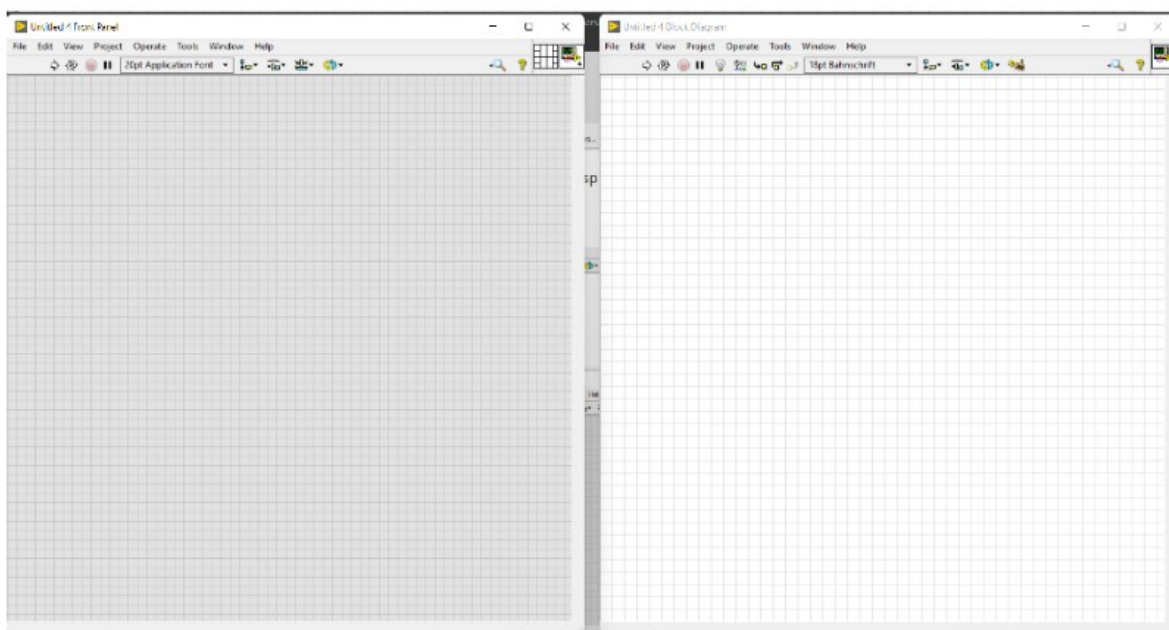


Fig 5: LabVIEW Windows

Front Panel Controls and Indicators

Controls and Indicators

- Controls are knobs, push buttons, dials and other input devices.
- Indicators are graphs, LEDs and other displays.
- Controls simulate instrument input devices and supply data to the block diagram of the VI.
- Indicators simulate instrument output devices and display data the block diagram acquires or generates.
- Every control or indicator has a data type associated with it. They are numeric data type, boolean data type, string data type

Block Diagram

Terminals

- Front panel object appear as terminals on the block diagram.
- Terminals are entry and exit ports that exchange information between the front panel and block diagram.
- Terminals are analogous to parameters and constants in text-based programming languages. Types of terminals include control or indicator terminals and node terminals.
- Control and indicator terminals belong to front panel controls and indicators.
- Data you enter into the front panel controls enter the block diagram through the control terminals.
- The terminals represent the data type of the control or indicator.
- You can configure front panel controls or indicators to appear as icon or data type terminals on the block diagram.
- By default, front panel objects appear as icon terminals.
- To display a terminal as a data type on the block diagram, right-click the terminal and select *View As Icon* from the shortcut menu

Wires

- You can transfer data among block diagram objects through wires.
- Each wire has a single data source, but you can wire it to many VIs and functions that read the data.
- Wires are different colors, styles and thicknesses, depending on their data types.
- A broken wire appears as a dashed black line with a red X in the middle.

- Broken wires occur for a variety of reasons, such as when you try to wire two objects with incompatible data types.
- You must connect the wires to inputs and outputs that are compatible with the data that is transferred with the wire.
- You cannot wire an array output to a numeric input.
- In addition, the direction of the wires must be correct. You must connect the wires to only one input and at least one output.
- You cannot wire two indicators together
- <Ctrl+B> to delete all broken wires or right click and select Clean Up Wire to reroute the wire.

Data Flow Program

- LabVIEW follows a dataflow model for running VIs.
- A block diagram node executes when all its inputs are available.
- When a node completes execution, it supplies data to its output terminals and passes the output data to the next node in the dataflow path.
- Visual Basic, C++, JAVA, and most other text-based programming languages follow a control flow model of program execution.

Review Questions.

What is virtual instrumentation? What are the different palettes in VI? What is sub VI?

What are the advantages of VI?

Compare hardware instrumentation and VI

-

EXPERIMENT 1

ARITHMETIC OPERATIONS

AIM:

To perform basic arithmetic operations using LabVIEW. Addition, Subtraction, Multiplication and division.

THEORY:

To give the 2 inputs numeric controls are selected in Front panel. And all arithmetic operators can be selected from Functions palette. Addition, subtraction, multiplication and division are the basic arithmetic operations.

PROCEDURE:

Step 1: Start the LabVIEW and create new project and select the blank VI. And click finish button.

Step 2: Press `ctrl -T` button to align Create front and block diagram panel.

Step 3: Numeric controls are given as inputs and numeric indicators are given as output they are selected by right clicking on the front panel.

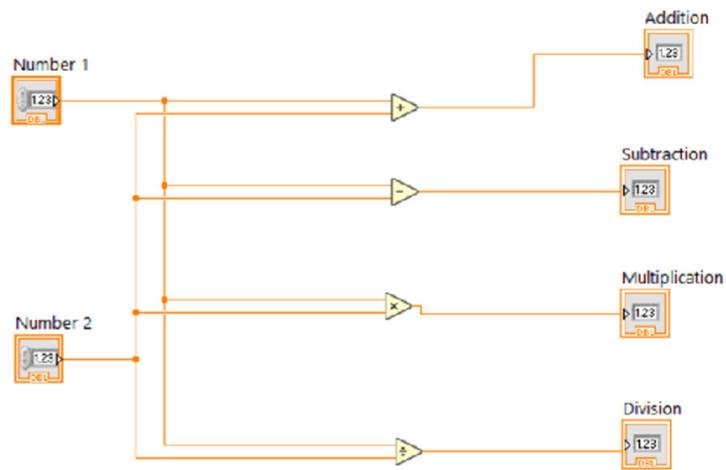
Step 4: Different arithmetic operators such as addition, subtraction, multiplication and division are generated in block diagram panel.

Step 5: Using wiring operation inputs and outputs are connected to the respective operators in the block diagram panel.

Step 6: Input values are given in the front panel and the program is executed. Hence the output is generated.

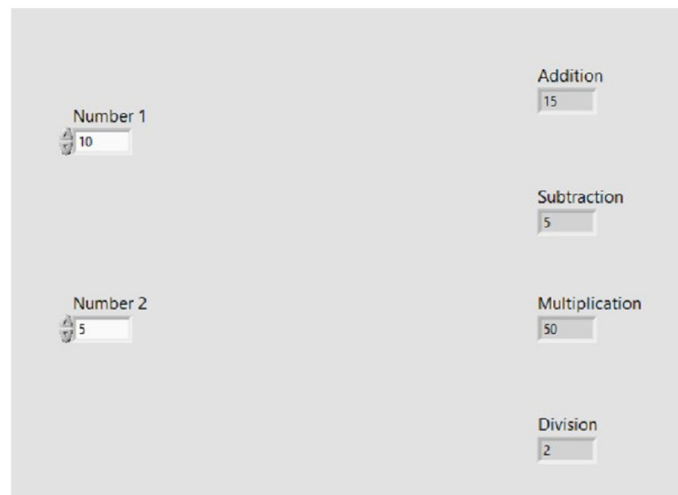
BLOCK DIAGRAM:

The block diagram for basic arithmetic operations using LabVIEW is shown below in Fig.1.1.



OUTPUT/Front PANEL:

Output of basic arithmetic operations is shown in below fig 1.2.



RESULT:

The arithmetic operations were performed and the result is verified using LabVIEW.

Viva questions:

1. What are the advantages of LabVIEW?
2. What are the two panels used in LabVIEW programming?
3. What is the difference between local variable and global variable in LabVIEW?

EXPERIMENT 2

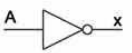





BOOLEAN OPERATIONS

AIM: To perform Boolean operations using LabVIEW. AND,OR,NOT,XOR , NAND

THEORY:

The truth table of Boolean operations like AND, OR,NOT,XOR ,and NAND is shown below in Fig 2.1.The Boolean inputs are given by push buttons in the control palette .All the Boolean operators can be taken from the functions palette.

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR																																																																																	
Alg. Expr.	\bar{A}	AB	\overline{AB}	$A+B$	$\overline{A+B}$	$A\oplus B$																																																																																	
Symbol																																																																																							
Truth Table	<table border="1"><thead><tr><th>A</th><th>X</th></tr></thead><tbody><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></tbody></table>	A	X	0	1	1	0	<table border="1"><thead><tr><th>B</th><th>A</th><th>X</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1"><thead><tr><th>B</th><th>A</th><th>X</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></tbody></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1"><thead><tr><th>B</th><th>A</th><th>X</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table border="1"><thead><tr><th>B</th><th>A</th><th>X</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></tbody></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table border="1"><thead><tr><th>B</th><th>A</th><th>X</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></tbody></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0
A	X																																																																																						
0	1																																																																																						
1	0																																																																																						
B	A	X																																																																																					
0	0	0																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	1																																																																																					
B	A	X																																																																																					
0	0	1																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					
B	A	X																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	1																																																																																					
B	A	X																																																																																					
0	0	1																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	0																																																																																					
B	A	X																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					

PROCEDURE:

Step 1: Start the LabVIEW and select the blank VI. Step 2: Create front and block diagram panel.

Step 3: To perform Boolean operation push buttons are taken as inputs and round LED as output.

Step 4: Different Boolean operations such as AND, OR, XOR, NOT, NAND are selected from the block diagram panel.

Step 5: Boolean inputs and outputs are wired in the block diagram panel.

Step 6: Logic values 0 & 1 are given in the front panel and the program is executed.

BLOCK DIAGRAM:

The block diagram of Boolean operations is shown below in Fig.2.2.

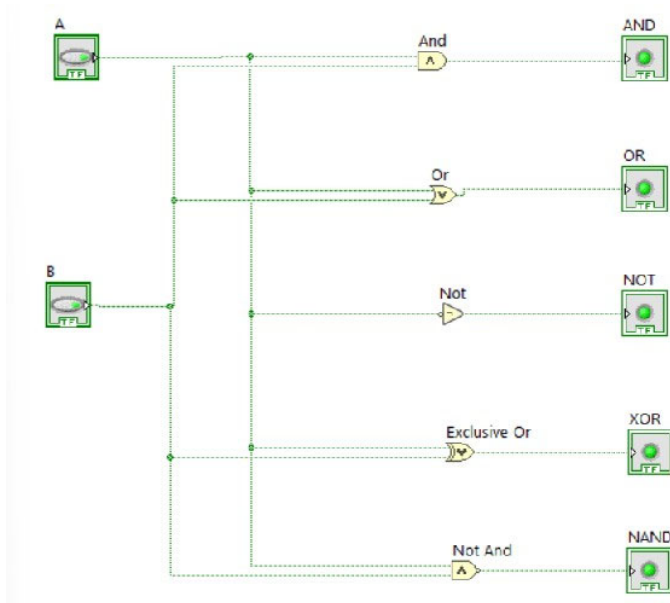
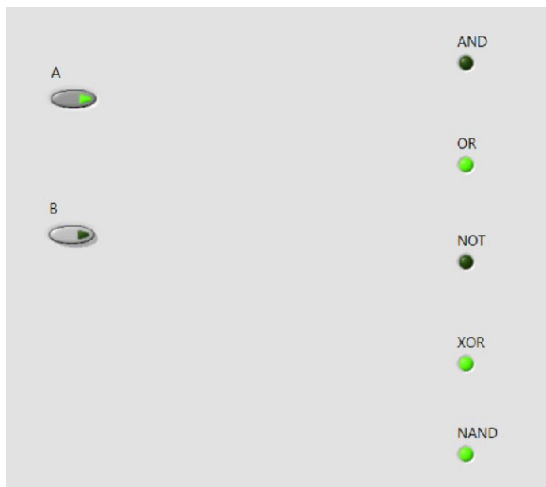


Fig 2.2:Block diagram of Boolean operations

OUTPUT/FRONT PANEL:

The output of Boolean operations is shown below in Fig.2.3.



RESULT: The truth table of Boolean operations AND, OR, NOT, XOR, and NAND gates are verified.

VIVA QUESTIONS:

1. Is LabVIEW a compiled programming language?
2. Can LabVIEW be integrated into existing software engineering practices?
- 3.3.What is LabVIEW?

EXPERIMENT 3

SUM OF N NUMBERS USING FOR LOOP

AIM:

To find the sum of 'n' numbers using FOR loop.

THEORY:

Usually sum of n numbers, we take for natural numbers. The equation to calculate sum of n numbers is $n(n+1)/2$. So we want to take sum of first 3 numbers, $n=3$. Then as per the equation it becomes $3(4)/2=6$. i.e $3+2+1=6$.

FOR loop in LabVIEW is shown below in Fig 3.1.

In FOR loop, shift registers are used if we make use of previous results. N is the count and 'i' is the iteration terminal and initial value is zero.

PROEDURE:

Step 1: Create blank VI.

Step 2: First of all move to the front panel and press right then control palette and choose numeric option place in the front panel and its correspondent will shown block diagram.

Step 3: Right click on the block diagram panel, select program , go to structures and select a FOR loop.

Step 4: Right click on the border of the FOR loop and select add shift register, borders are converted into shift register.

Step 5: In block diagrams select the numeric button and choose constant and connect it to register.

Step 6: Select the add button from the numeric tab. Then select the increment option from the numeric tab .Connect the output of this increment block to the remaining input of the add block.

Step 7: At the right shift register click right and from the drop down select create and then select indicator .

Step 8: Inputs are given in the front panel and the program is executed.

BLOCK DIAGRAM:

The block diagram of sum of n numbers using FOR loop is shown in Fig 3.2 below.

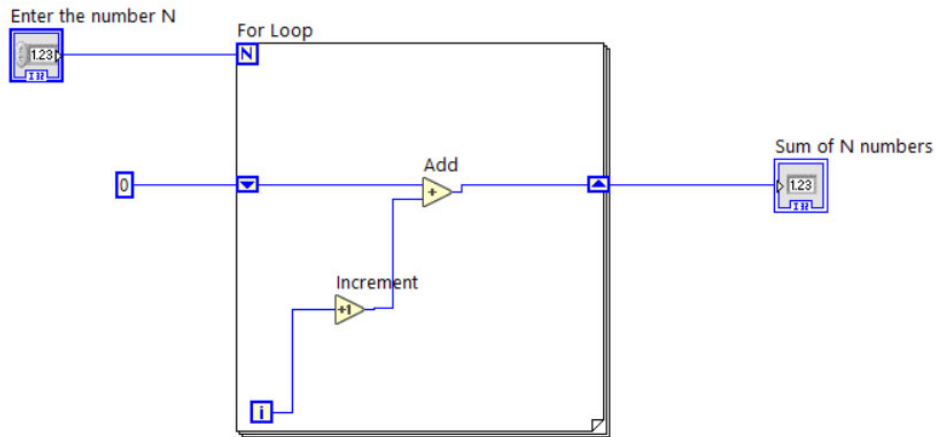
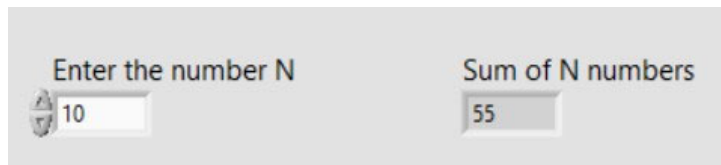


Fig 3.2: Block diagram of sum of n numbers using FOR loop

OUTPUT/Front PANEL:

The output or the front panel view of sum of n numbers using FOR loop is shown below in figure.



RESULT:

Thus the sum of 'n' natural numbers using FOR loop is performed in LabVIEW

For N = 10, Sum of 10 integers = 55

VIVA QUESTIONS:

1. What is shift register? How is it implemented in LabVIEW?
2. Can LabVIEW Vis be merged?
3. What is LabVIEW signal express?
4. What is Formula node in LabVIEW?
5. In which palette, FOR loop is available?

EXPERIMENT 4

FACTORIAL OF A NUMBER

AIM: To perform the factorial of a given number using FOR loop.

THEORY:

Factorial is an important function, which is used to find how many ways things can be arranged or the ordered set of numbers. The factorial concept is used in many mathematical concepts such as probability, permutations and combinations, sequences and series, etc. In short, a factorial is a function that multiplies a number by every number below it till 1. For example, the factorial of 3 represents the multiplication of numbers 3, 2, 1, i.e. $3! = 3 \times 2 \times 1$ and is equal to 6.

A [For Loop](#) is a structure you use to execute a block of code a set number of times. When the VI runs, the iteration count is evaluated, and then the code is executed. A For Loop can be configured to [conditionally stop code execution](#) in addition to its iteration-based exit. In these cases, the code will execute until the count terminal setting is reached *or* the condition is met

– whichever happens first. The structure of FOR loop in LabVIEW is shown below in fig.

PROCEDURE

Step 1: Create blank VI.

Step 2: Click right on the front panel from the control palette selects numeric and then selects control.

Step 3: Right click on the block diagram panel, select program, go to structures and select a FOR loop.

Step 4: Connect the numeric control in which the user input will be stored to the N. Right click on the border of the FOR loop and select add shift register, borders are converted into shift register.

Step 5: To initialize the shift register, from the function palette select numeric and then select constant. Place this numeric with the shift register and set its value to 1.

Step 6: For the multiplication task place a multiply block, from the function palette select numeric and then select multiply. At one of the two inputs of the multiply block connect the right shift register, and the output of this block connect the right shift register.

Step 7: On the other input of the multiply block we have to connect the incremented iteration because the number of iterations start from 0 but we want the factorial from 1 onwards, because the factorial will turn out to be zero otherwise. From the function palette select numeric and then select increment.

Step 8: At the input of this increment block connect the iterative index of the for loop, and connect the output of this increment block to the remaining input of the multiply block.

Step 9: At the right shift register click right and from the drop down select create and then select indicator.

Step 10: Inputs are given in the front panel and the program is executed.

BLOCK DIAGRAM:

Block diagram of factorial of a number using FOR loop is given below in Fig.4.2.

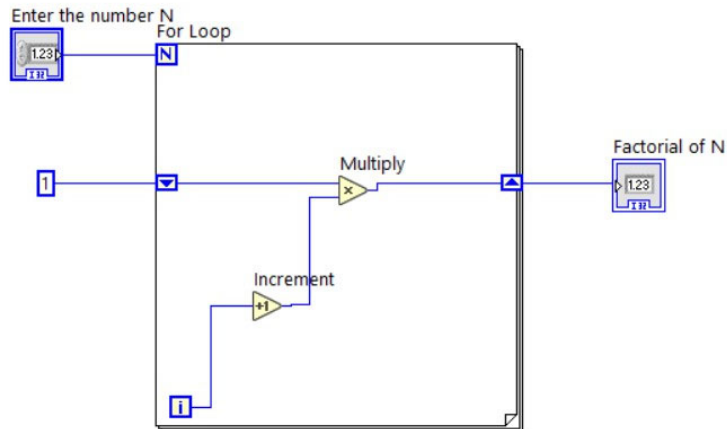
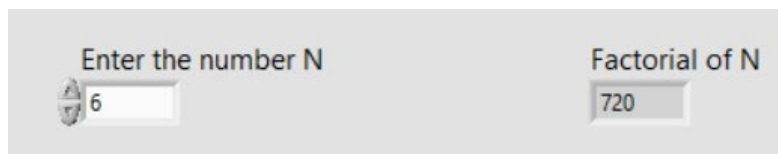


Fig.4.2: Block diagram of Factorial of a number using FOR loop

OUTPUT/Front PANEL:

The output of factorial of a number using FOR loop is shown below in figure below



RESULT:

The factorial of a given number is using FOR loop is performed in LabVIEW.

VIVA QUESTIONS:

1. How mixed data types are combined and passed from one file or place to the other in LabVIEW?
2. What is the initial value of iteration count?
3. What is subVI?
4. What is block diagram panel?
5. What is front panel?

EXPERIMENT 5

DETERMINE SQUARE, CUBE & SQUARE ROOT OF A GIVEN NUMBER

AIM: To determine Square, Cube & Square root of a given number using LabVIEW.

THEORY: The Square root VI computes the square root of the input value. If x is negative, the square root is NaN unless x is complex. If x is a matrix, this function takes the matrix square root of x . The connector pane displays the default data types for this polymorphic function. x can be a scalar number, array or cluster of numbers and array of clusters of numbers. $\text{sqrt}(x)$ is a double-precision, floating-point number if x is an integer. If x is less than 0, $\text{sqrt}(x)$ is not a number (NaN).

PROCEDURE:

Step 1: Create blank VI.

Step 2: Right click on the block diagram window, select numeric, add numeric constant to represent the input.

Step 3: Right click on the block diagram window, select numeric, add numeric indicator to represent output

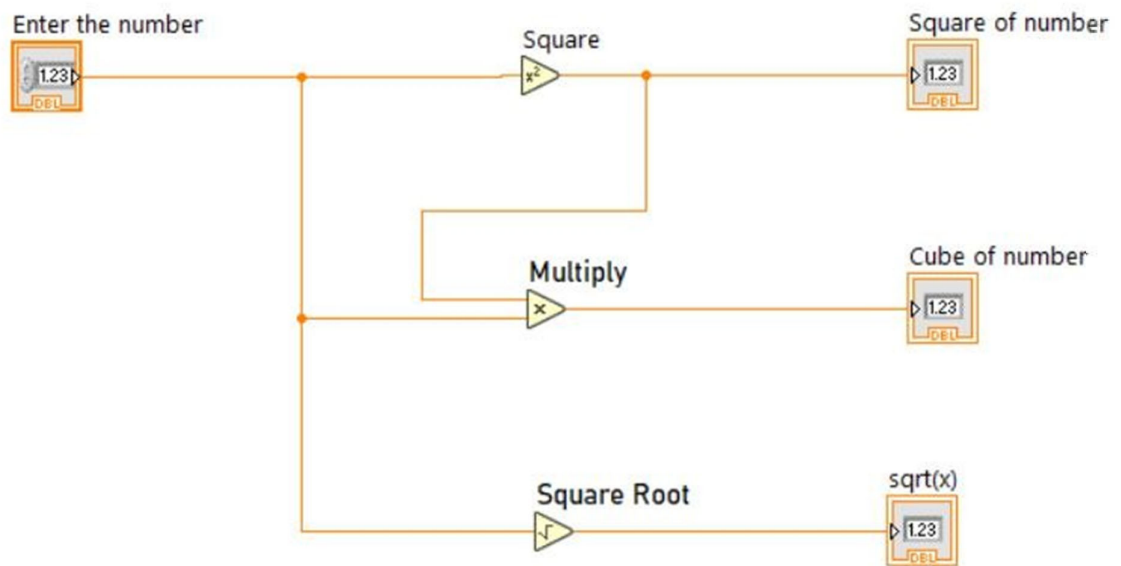
Step 3: Right Click on the block diagram window, select numeric, add the square, Square root and multiply blocks

Step 4: Using wiring operations required connections are given in the block diagram.

Step 5: Inputs are given in the front panel and the program is executed.

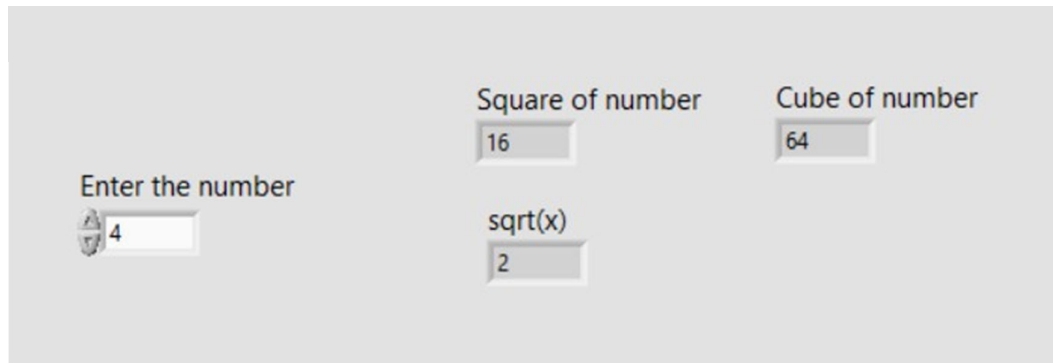
BLOCK DIAGRAM:

Block diagram to calculate square and square root of a number is shown below in figure.



OUTPUT/FRONT PANEL:

The output or the front panel window is shown below in figure



RESULT:

Square of a given number is determined using LabVIEW.

VIVA QUESTIONS:

1. What is the output of square root block if input is negative number?
2. What is the default data type for input for square root block?
3. Can square root block help in finding roots of a complex number?
4. What is the datatype for integer or floating-point number?

EXPERIMENT 6

FACTORIAL OF A NUMBER USING WHILE LOOP

AIM:

To find factorial of a number using while Loop in LabVIEW.

THEORY:

Use shift registers when you want to pass values from previous iterations through a loop to the next iteration. A shift register appears as a pair of terminals, shown as follows, directly opposite each other on the vertical sides of the loop border. The terminal on the right side of the loop contains an up arrow and stores data on the completion of an iteration. LabVIEW transfers the data stored in the right terminal of the shift register to the left terminal. The loop then uses the data from the left terminal as the initial values for the next iteration. This process continues until all iterations of the loop execute. After the loop executes, the terminal on the right side of the loop returns the last value stored in the shift register. A shift register transfers any data type and automatically changes to the data type of the first object wired to the shift register. The data you wire to the terminals of each shift register must be the same type. If you have multiple operations that use previous iteration values within a loop, use multiple shift registers to store the data values from those different processes in the structure, as shown in the following block diagram in Fig 6.1

PROCEDURE:

Step 1: Create blank VI.

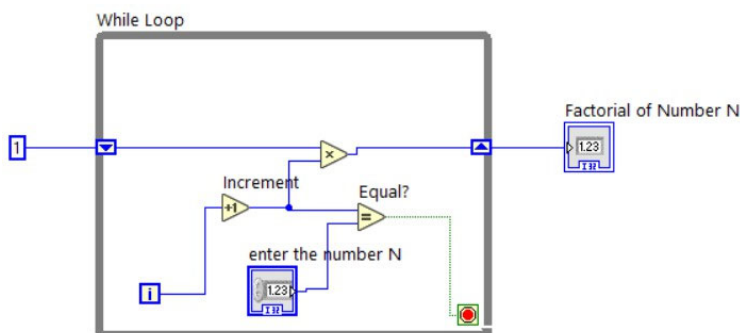
Step 2: Right click on the block diagram panel, select program , go to structures and select a WHILE loop.

Step 3: Right click on the border of the WHILE loop and select add shift register,borders are converted into shift register.

Step 4: Using wiring operations required connections are given in the block diagram.

Step 5: Inputs are given in the front panel and the program is executed

BLOCK DIAGRAM:



Block diagram using while loop

OUTPUT/FRONT PANEL:

The output or the front panel window view is shown below in figure



RESULT:

Factorial of a number is determined using while loop in LabVIEW

VIVA QUESTIONS:

1. What is the role of shift register in the while loop?
2. What is the default data type for factorial block?
3. How to find factorial of integer numbers?
4. Differentiate for loop and while loop.

EXPERIMENT 7

SORTING EVEN NUMBERS USING WHILE LOOP IN AN ARRAY

AIM:

To sort even numbers using WHILE loop in an array.

THEORY:

While Loop

While Loop executes a subdiagram until a condition is met. The While Loop executes the subdiagram until the conditional terminal, an input terminal, receives a specific Boolean value. The default behavior and appearance of the conditional terminal is Continue If True. When a conditional terminal is Continue If True, the While Loop executes its subdiagram until the conditional terminal receives a FALSE value. To change the behavior and appearance of the conditional terminal, right-click on the terminal and select Stop If True. When a conditional terminal is Stop If True, the While Loop executes its subdiagram until the conditional terminal receives a TRUE value. Because the VI checks the conditional terminal at the end of each iteration, the While Loop always executes at least one time. The VI is broken if the conditional terminal is not wired.

PROCEDURE:

Step 1: Create blank VI.

Step 2: Right click on the Block diagram panel →structures→ while loop

Step 3: Right click on the Block diagram panel (inside while loop) → array→ index array Step 4: Right click on the index array (input side) →create →control

Step 5: Right click on the Block diagram panel →array →array size

Step 6: Right click on the Block diagram panel →numeric→ Quotient & Remainder Step 7: Right click on the Block diagram panel → Comparison→ Equal to 0?

Step 8: Right click on the Block diagram panel →numeric→ numeric constant Step 9: Right click on the Block diagram panel →numeric→ decrement

Step 10: Right click on the Block diagram panel →numeric→ numeric constant Step 11: Right click on the Block diagram panel → Comparison→ Equal?

Step 12: Using wiring operations required connections are made as given in the block diagram inside loop.

Step 13: Make a connection to while loop from index array

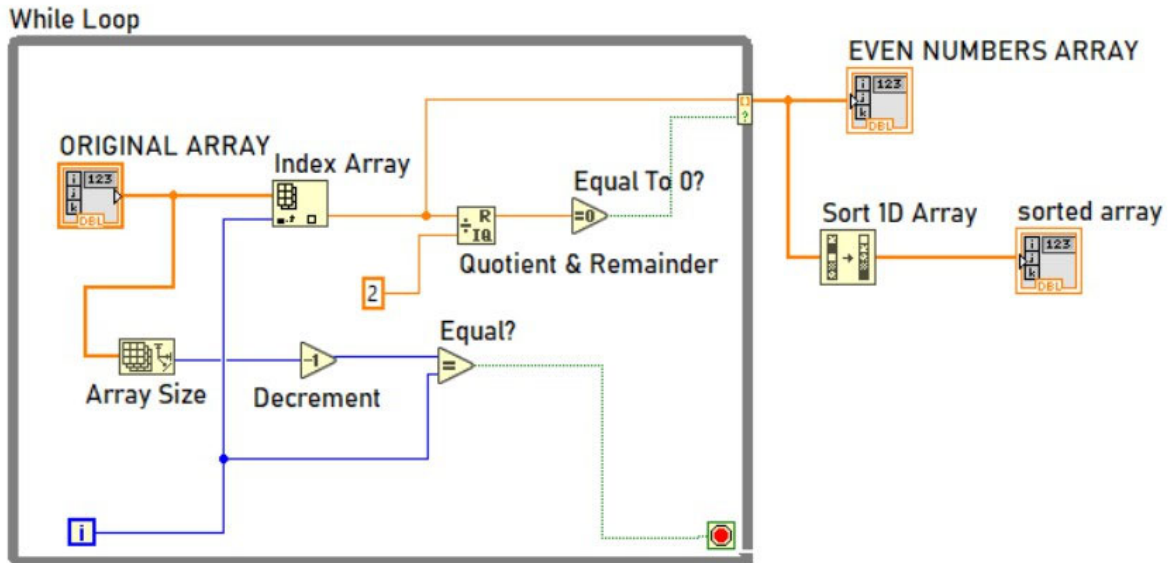
Step 14: Right click step 13 connection (on loop) →Tunnel mode→ Indexing Step 15: Right click step 14 connection (on loop) →Tunnel mode→Conditional Step 16: Right click on the Block diagram panel →array→ sort 1D array

Step 17: Right click on the sort 1D array (input side) →create→ control [Even array] Step 18: Right click on the sort 1D array (output side) →create→ control [Sorted array]

Step 19: Using wiring operations required connections are made as given in the block diagram outside loop.

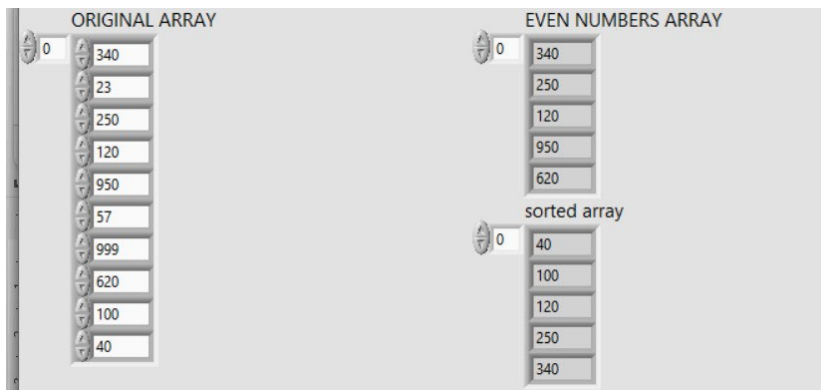
BLOCK DIAGRAM:

Below Figure shows the block diagram of sorting even numbers using while loop. The array takes the input values. Using indexing values are accessed, and checked for even or not. To check for even number the element value is divided by two and resulting remainder if zero considered as even. This data is stored in output array. Using sort function, array is sorted.



Block diagram Sorting even Number in LabVIEW

OUTPUT/ FRONT PANEL:



RESULT: Front panel shows the original array, even numbers array and sorted even numbers array. Even numbers are sorted using while loop in an array.

VIVA QUESTIONS:

1. What are the different loops available in LabVIEW?
2. Distinguish between for and while loop.
3. What is loop iteration control?
4. What is Multi tracing?

EXPERIMENT 8

FINDING ARRAY MAXIMUM AND ARRAY MINIMUM

AIM:

To find the maximum and minimum variable from an array.

THEORY:

Arrays

- A group of homogeneous elements of a specific data type is known as an *array*.
- Arrays hold a sequence of data elements, usually of the same size and same data type placed in contiguous memory locations.
- Individual elements are accessed by their position in the array.
- The position is given by an index, which is also called as *subscript*
- Some arrays are multi-dimensional, generally one -and two- dimensional arrays are the most common.

PROCEDURE:

Step 1: Create blank VI.

Step 2: Right click on the Block diagram panel →array→ max & min. Step 3: Right click on the max & min (input side) →create →control.

Step 4: Create four numeric indicators in the front panel for maximum variable, index, minimum variable, and index by right clicking on output side of max & min.

Step 5: Using wiring operations required connections are made as given in the block diagram. Step 6: Inputs are given in the front panel and the program is executed

BLOCK DIAGRAM:

Figure shows the block diagram of array maximum and minimum in LabVIEW. Array input is taken and using the array max and min function values are indicated using the indicators

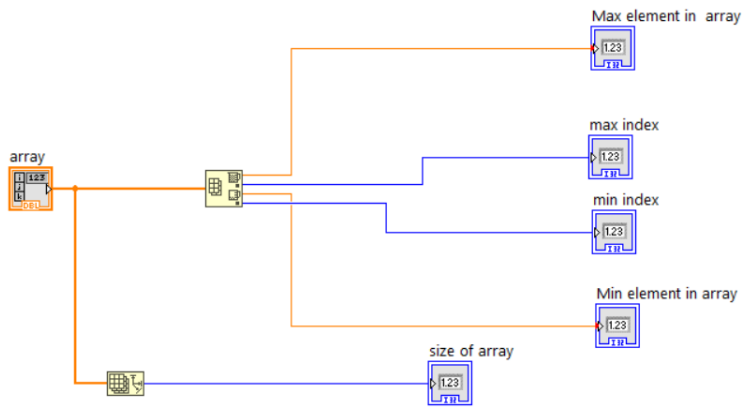
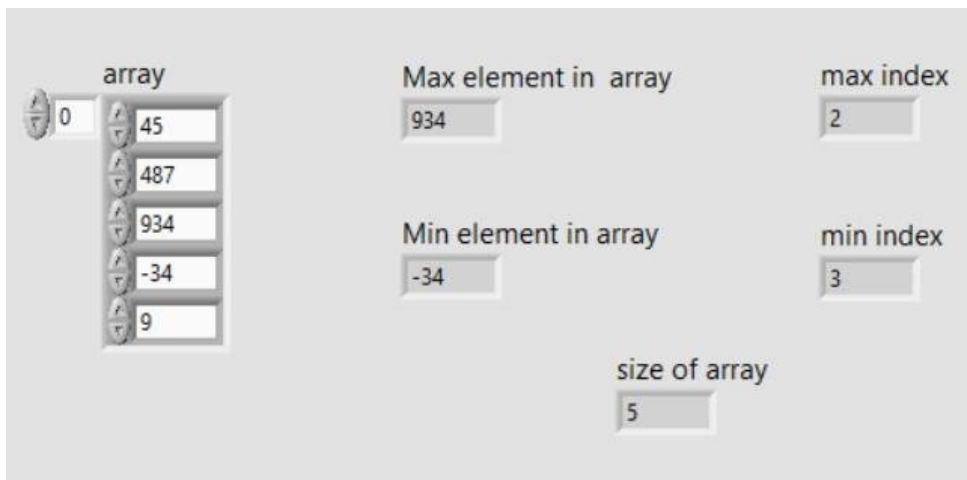


Fig 8.1: Block diagram of array maximum and minimum in LabVIEW

OUTPUT /FRONT PANEL :



RESULT:

Figure shows the maximum and minimum of given input array. The array maximum and minimum is determined using LabVIEW and also its index is displayed. Determine the minimum and maximum value for the array element?