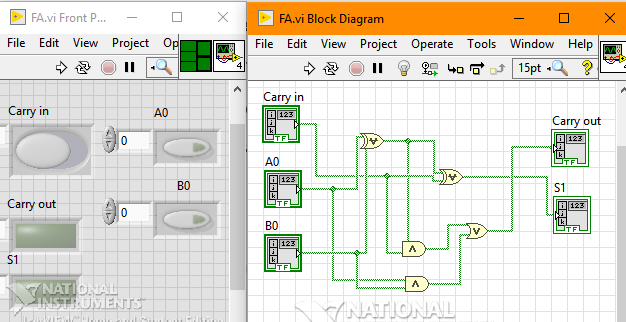
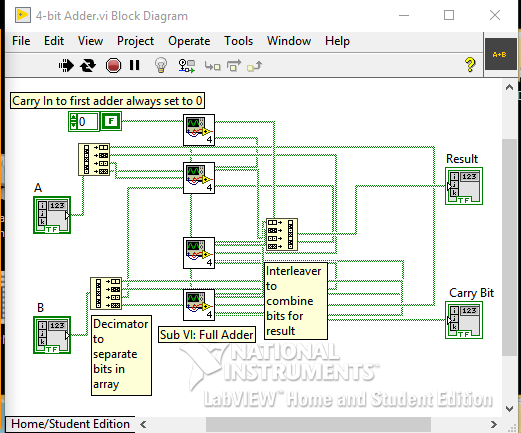
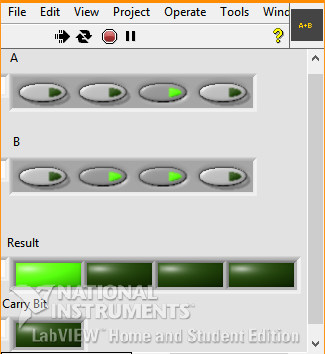
**Addition:**

2-half adders are combined, to form a full adder for each pair of th significant bits to be added.



Carry out of one full-adder is used as an input carry for the next full-adder in cascade.

Carry-in is set to 0 [False Array Constant] for the LSB (1st pair bit) adder.



The front panel snap illustrates that:

*Four Bit Adder Boolean Expression:*

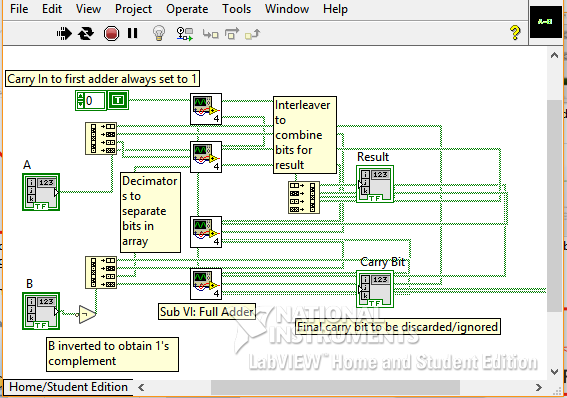
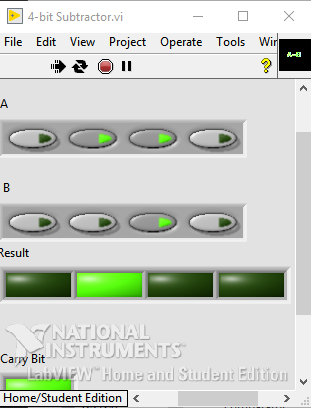
Sum bit =

Carry-out =

Note: The final (4th pair) Carry-out bit is disregarded as it represents an ‘overflow’ condition for a 4-bit ALU.

**Subtraction:**

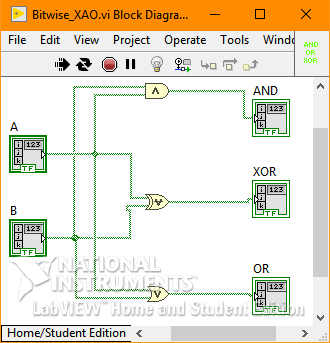
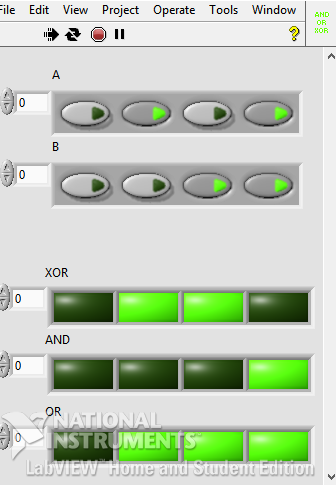
Requires Subtrahend to be converted to 2’s complement and added using same cascaded full adders. So we invert B (to obtain 1’s complement) and set the carry-in of the LSB adder to 1[True Array Constant] (for adding ) to convert to 2’s complement. Any final carry out bit is disregarded.



The output illustrates that:

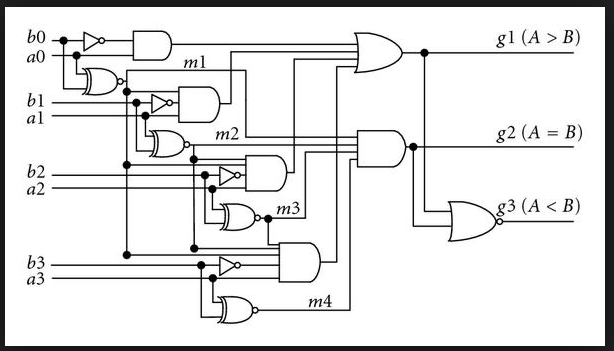
**Bitwise-AND,OR & XOR**

Each corresponding significant bit pair is used for the desired operation. The following result on the VI demonstrates all possible outputs for 2-input logic gates by using an array of 4 bits on each input.



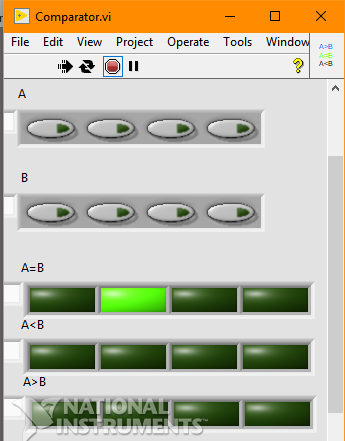
**Comparator:**

The following circuit has been used for implementing the comparator functions



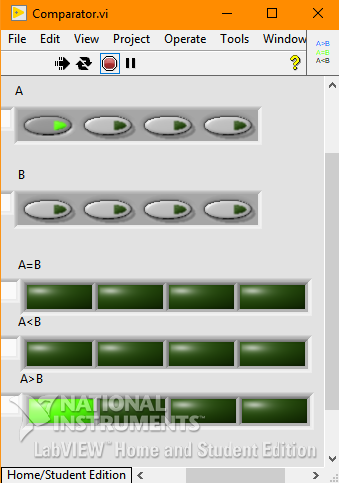
For , an XNOR operation is applied on each pair of corresponding significant bits, and finally an AND opeartion is used on each XNOR output.

The Boolean Expression for this output is:



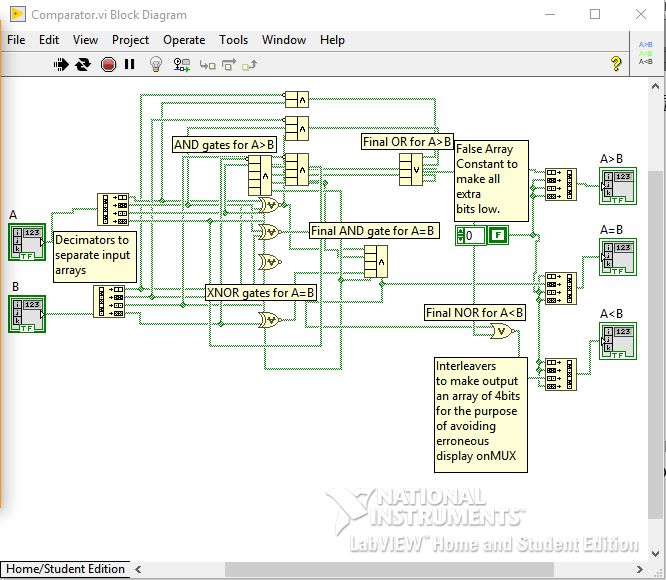
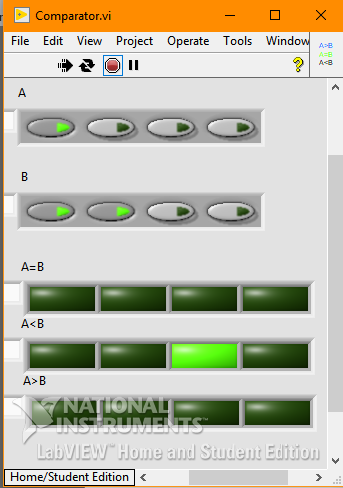
For , an AND opeartion is performed on , , and all preceeding XNOR outputs; for a pair of corresponding significant bits. Finally an OR opeartion is used on each AND gate output.

The Boolean Expression for this output is:



For , a NOR operation is performed on the outputs for the aforementioned comparator operations; if neither of the opeartions are true/high, then this logic must be true.

The Boolean Expression for this output is:

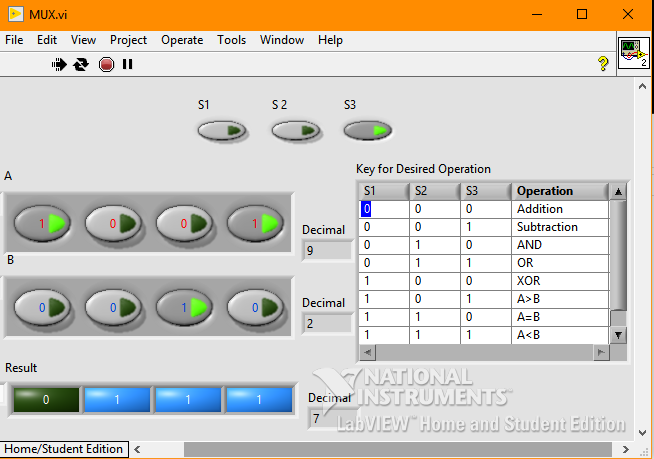


**Multiplexer:**

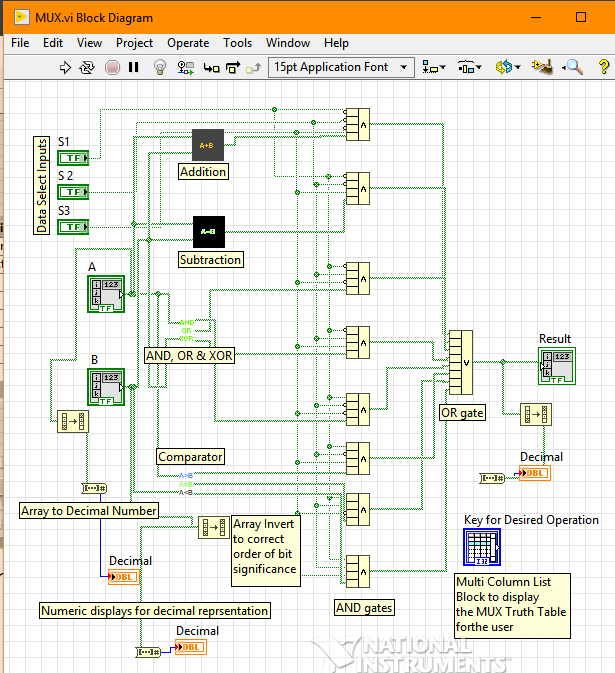
For 8 operations, 3 bit Data-Select Input codes need to be defined for each operation.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

An AND operation is performed on all Data-Select Input and Operation combinations, and finally an OR operation is performed on these outputs to give the result of the operation which is called by the Data-Select Input.



This snip illustrates a ‘Subtraction’ operation.



*NOTE: All VIs are run continuously  which gives us loop-like functionality; without loops.*

IBRAHIM PATEL

me151095 [BE-ME-7B]