Count minimum steps to get the given desired array

Consider an array with n elements and value of all the elements is zero. We can perform following operations on the array.

1. Incremental operations: Choose 1 element from the array and increment its value by 1.
2. Doubling operation: Double the values of all the elements of array.

We are given desired array target[] containing n elements. Compute and return the smallest possible number of the operations needed to change the array from all zeros to desired array.

Sample test cases:

```
Input: target[] = {2, 3}
Output: 4
To get the target array from {0, 0}, we
first increment both elements by 1 (2
operations), then double the array (1
operation). Finally increment second
element (1 more operation)

Input: target[] = {2, 1}
Output: 3
One of the optimal solution is to apply the
incremental operation 2 times to first and
once on second element.

Input: target[] = {16, 16, 16}
Output: 7
The output solution looks as follows. First
apply an incremental operation to each element.
Then apply the doubling operation four times.
Total number of operations is 3+4 = 7
```

Source: http://qa.geeksforgeeks.org/7023/create-desired-array-from-zero-array

One important thing to note is that the task is to count the number of steps to get the given target array (not to convert zero array to target array).

The idea is to follow reverse steps, i.e. to convert target to array of zeros. Below are steps.

```
Take the target array first.

Initialize result as 0.

If all are even, divide all elements by 2
and increment result by 1.

Find all odd elements, make them even by
reducing them by 1. and for every reduction,
increment result by 1.

Finally we get all zeros in target array.
```

Below is C++ implementation of above algorithm.

```
/* C++ program to count minimum number of operations
   to get the given target array */
#include <bits/stdc++.h>
using namespace std;

// Returns count of minimum operations to covert a
// zero array to target array with increment and
// doubling operations.
// This function computes count by doing reverse
```

```
// This function computes count by doing reverse
// steps, i.e., convert target to zero array.
int countMinOperations(unsigned int target[], int n)
{
   // Initialize result (Count of minimum moves)
   int result = 0;

   // Keep looping while all elements of target
   // don't become 0.
   while (1)
   {
      // To store count of zeroes in current
      // target array
      int zero_count = 0;

      int i;  // To find first odd element
      for (i=0; i<n; i++)
      {
         // If odd number found
         if (target[i] & 1)
            break;

         // If 0, then increment zero_count
         else if (target[i] == 0)
            zero_count++;
      }

      // All numbers are 0
      if (zero_count == n)
        return result;

      // All numbers are even
      if (i == n)
      {
         // Divide the whole array by 2
         // and increment result
         for (int j=0; j<n; j++)
            target[j] = target[j]/2;
         result++;
      }

      // Make all odd numbers even by subtracting
      // one and increment result.
      for (int j=i; j<n; j++)
      {
        if (target[j] & 1)
        {
          target[j]--;
          result++;
        }
      }
   }
}

/* Driver program to test above functions*/
int main()
{
   unsigned int arr[] = {16, 16, 16};
   int n = sizeof(arr)/sizeof(arr[0]);
   cout << "Minimum number of steps required to "
        "get the given target array is "
       << countMinOperations(arr, n);
   return 0;
}
```

Output :

```
Minimum number of steps required to
get the given target array is 7
```