<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1* (after R=1 and S=0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0**</td>
<td>1** (after R=0 and S=1)</td>
</tr>
</tbody>
</table>
2. The basic idea is to add together all the bits in the string, except that we are only concerned with the rightmost bit (which will tell us whether the decimal counterpart of the binary string is even or odd, which in turn tells us whether the number of 1's is even or odd.) Thus:

In this circuit, the output would determine whether the number of 1's is even or odd at each step until the whole binary string is processed. Assuming we have a way of knowing when we are finished with processing the bit string, the final output would answer our question—if it is a 1, then the number of 1's is odd; if the output is a 0, the number of 1's is even.

A further assumption we make is that \( W \) is turned on and back off once for each bit that is processed. Thus \( W \) functions as a sort of clock: each clock “tick” of the circuit is sort both the next bit in the string and the bit from memory that represents whether the number of 1's processed so far is odd or not.

Let \( b_i \) represent the \( i \)th bit in the string, and \( m_i \) the bit that represents whether the number of 1's to that point is odd or even, (1 or 0, respectively).

Also, assume the 1-bit memory register which remembers the feedback from the output is just an “airlock” flip-flop like we saw in class.
3.

\[ B = \text{NOT} \ X \ \text{AND} \ Y \]
\[ D = [(\text{NOT} \ X) \ \text{AND} \ Y] \ \text{OR} \ [(X) \ \text{AND} \ \text{NOT} \ Y] \]

\[ B = (x'.y) \]
\[ D = (x'.y + x.y') \]

(1)
4-bit subtractor =

\[ 0 \quad \text{(B₁ always = 0)} \]

\[ \begin{align*}
X_1 & \quad X_3 & \quad X_2 & \quad X_1 \\
- & \quad Y_4 & \quad Y_3 & \quad Y_2 & \quad Y_1 \\
\text{D}_0 \quad \text{D}_1 \quad \text{D}_2 \quad \text{D}_3
\end{align*} \]

\[ B_i \]

\[ X_i \quad Y_i \]

\[ B(C_{i+1}) \]

\[ B_3 \text{ (should be zero if we're not subtracting big from small)} \]