A. Briefly, by evaluating S on the standard basis in the usual way, we deduce that

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

B. The basic concept is that $f : \{0,1\}^n \to \{0,1\}^m$ converts into a permutation matrix F such that $F \cdot (|\alpha\rangle \otimes |\beta\rangle) = |\alpha\rangle \otimes |\beta \oplus f(\alpha)\rangle$. By comparing this construction to the given F, we deduce that $f : \{0,1\}^4 \to \{0,1\}^2$ is defined by

$$f(\alpha_5, \alpha_4, \alpha_3, \alpha_2) = (\alpha_5 \odot \alpha_3, \alpha_4 \odot \alpha_2).$$

In slightly different notation, we could instead write

$$f(\alpha) = f(\alpha_3\alpha_2\alpha_1\alpha_0) = (\alpha_3 \odot \alpha_1)(\alpha_2 \odot \alpha_0).$$

(By the way, the meaning of f is that it takes two two-bit strings as input, and returns their bitwise product as output.)

C. The state passing through F is now a superposition of $|\beta\rangle \otimes |+\rangle$. By a homework problem that we did on Day 13, such states are eigenvectors of F with eigenvalue 1. So F has no effect; it might as well not be there. Then the Hadamard layers cancel each other. So the measurement outputs $|0 \cdots 0\rangle$ with probability 1. The circuit does nothing of value.

D. The column without a leading 1 corresponds to δ_2 in $\delta = \delta_8\delta_7 \cdots \delta_2\delta_1\delta_0$. So we set $\delta_2 = 1$. Then the other columns force us to choose the other δ_j so that $\delta = 011011100$.

E. The idea is that the new algorithm runs the old algorithm repeatedly, on $k, k^2, k^4, k^8, \ldots$, until it gets the information that it needs. Let's flesh out this idea.

1. The new algorithm runs the old algorithm on k and m to obtain a putative period q. Either $p = q$ or p is even and $q = 1$. So it computes $k^q \mod m$. If $k^q \equiv 1 \pmod{m}$, then it must be true that $p = q$.

2. Otherwise, it runs the old algorithm on k^2 and m to obtain a new q. If $(k^2)^q \equiv 1 \pmod{m}$, then $k^{2q} \equiv 1 \pmod{m}$, and it must be true that $p = 2q$.

3. Otherwise, it runs the old algorithm on k^4 and m to obtain a new q. If $(k^4)^q \equiv 1 \pmod{m}$, then $p = 4q$.

4. Otherwise, it continues...

By continuing in this fashion, the new algorithm eventually discovers the period p of k.

To see so, notice that the period p can be written as $p = 2^\ell j$, where $\ell \geq 0$ and j is odd. The new algorithm eventually computes the period of k^{2^ℓ}, correctly finds that it is j, and correctly outputs $2^\ell j$ for the period of k. Is it possible that the algorithm never reaches this step, because it stops at an earlier step? No. For it could only stop at an earlier $k^{2^{\ell'}}$ with putative period q' if it found that $(k^{2^{\ell'}})^{q'} \equiv 1 \pmod{m}$. But that q' would have to be 1. So we would have $k^{2^{\ell'}} \equiv 1 \pmod{m}$, in contradiction of the fact that $p = e^\ell j$ is the least positive power, to which we can raise k modulo m, to obtain 1.

The new algorithm has to invoke the old algorithm ℓ times. And ℓ is (at most) logarithmic in p, and $p \leq \phi(m) < m$. So ℓ is logarithmic in m and hence linear in the number of bits needed to represent m. So the new algorithm is only slightly slower than the old algorithm. For example, if the old algorithm is $\mathcal{O}(n^2)$, where n is the number of bits needed to represent m, then the new algorithm is $\mathcal{O}(n^3)$.