1. In Fig. 7.12, Qa is true output of gated D latch, Qb is true output of pos. edge-triggered D FF, and Qc "neg".

Hence, Qa follows D whenever CLK=1 and when CLK goes 1→0, Qa holds value of D at that instant. Qb takes and holds value of D at pos. clock edge until next pos. clock edge. Qc takes and holds value of D at neg. clock edge until next neg. clock edge.
2. Note that DFF holds its values for one entire clock cycle. So, if we force the output to toggle at every triggering clock edge, the DFF output will be a clock signal with twice the period of the original clock, hence half its frequency. Do this twice:

```
100 MHz  
|    |    |    |    |    |    |    |
| Q   | Q   | Q   | Q   | Q   | Q   | Q   |
| Q   | Q   | Q   | Q   | Q   | Q   | Q   |
```

4. Set up k-map for output T and inputs J, K, Q

```
D 00 01 11 10
0 0 0 1 1
1 0 0 0 0
```

\[ T = \overline{JQ} + \overline{KQ} \]

Circuit below:

```
J
\rightarrow\rightarrow
Q
K
\rightarrow\rightarrow
\overline{Q}
```

Clock
Note that gates 5 & 6 form a SR latch with P1 being the R input and P2 being the S input. When \( \text{clock} = 1 \), both \( \overline{P_2} \) and \( \overline{P_3} \) (R = 0 and S = 0) are forced to 0, maintaining the values of \( \overline{D} \) and \( \overline{Q} \). With \( P_2 = 0 \), \( P_4 = \overline{D} \), which is one input to gate 1. The other input is \( P_1 = 0 \), so \( P_3 = \overline{D} = D \). When clock changes to 0 (neg edge), \( P_1 \) becomes \( \overline{P_3} = \overline{D} \), the inputs to gate 3 are \( \overline{D}_0, P_0, \overline{D}_0 \), so \( P_2 = D \). At this point we have \( P_1 (R) \) equals \( \overline{D} \) and \( P_2 (S) = D \), so \( \overline{Q} = 0 \) and \( \overline{Q} = \overline{D} \). Must now show that while \( \text{clock} = 0 \), after the neg edge, changes in \( D \) will not result in changes to \( \overline{Q} (\text{and } \overline{Q}) \). Two cases to consider.

If \( D = 1 \) at the negative edge, this forces \( P_2 \) to 1 (as above) which forces \( P_4 \) to 0. This combined with \( P_1 = 0 \) holds \( P_2 = 1 \) as long as \( \text{clock} = 0 \), even if \( D \) changes, so \( P_4 \) remains 0 and \( P_2 \) remains 1. Case 2: If \( D = 0 \) at the neg clock edge, this forces \( P_1 = 1 \) (as above), which forces \( P_2 \) and \( P_3 \) to be 0. If \( D \) then changes to 1, \( P_4 \) is forced to 0, but this does not change \( P_2 \) or \( P_3 \) (both 0), and so \( P_1 \) stays at 1.
Parallel load most easily done with D FFs. For parallel load, connect each $c_i$ bit to $i$ input of mux with select input $= L$, and connect mux output to $D$ input of D FF. For other mux input, we would connect output of D FF to left for right shift ($s=0$) and output of D FF to right for left shift ($s=1$). This is another mux. Let D inputs to DFF's be $d_3, d_2, d_1, d_0$, with $d_0$ on right (LSB).

Equations:

\[
\begin{align*}
d_0 &= L r_0 + I (s q_1 + s \cdot 0) \\
d_1 &= L r_1 + I (s q_2 + s q_0) \\
d_2 &= L r_1 + I (s q_3 + s q_0) \\
d_3 &= L r_1 + I (s q_4 + s q_2)
\end{align*}
\]

\[\text{mux}\]

\[\text{mux}\]
6. If we could be sure that we would never have \( \text{Start} = \text{Stop} = 1 \), we could use a JK FF with \( J = \text{Start} \) and \( K = \text{Stop} \). To prevent having \( J=K=1 \), we could let \( K = \text{Stop} \) and \( J = \text{Start} \cdot \overline{\text{Stop}} \).

Equivalent functionality can be obtained with a D FF by setting \( D = 0 \) if \( \text{Stop} = 1 \) and setting \( D = \text{Start} + \overline{\text{Q}} \) if \( \text{stop} = 0 \). Again, \( \text{Run} = Q \) output of FF.

Other designs are no doubt possible as well.