1. The minimum clock period is the sum of the propagation delay through the first set of registers, the propagation delay through both of the adders and the setup time to the last set of registers. Also, in this case, you want to use the maximum delays. So the result is: $300 + 800 + 800 + 200 = 2100$ ps. Thus the maximum frequency is 476 Mhz.

2. No, the circuit does not work. The fastest path through the circuit is from the input registers, through the adders and into the output registers. Using the minimum delay characteristics, this path is 400 ps. If the clock skew is also 400 ps, this value will arrive at the same time as the clock, violating the hold time of the output registers by 100 ps. (Note that we are right on the edge of failure; that is, there is no margin at all. There is the question of whether it makes a difference if the equation is written $>$ or $\geq$. Mathematically I suppose so, but either way your margin is 0. Whether you put in an extra buffer or not depends on your concept of safety and how good you think the min and max delay numbers are.)

3. To fix this problem, you need to delay the output of the adders long enough to avoid violating the hold time of the output registers. The cheapest solution is to add a buffer to the input of each of the output registers.

4. Start by adding registers to the outputs (sum and carryout) of the bottom adder. This delays the output of that adder by one cycle, so it is also necessary to delay the inputs to the top adder by one cycle. Do this by inserting a register between the top input register and the top adder. The delay between these registers needs to be greater than the sum of the clock skew and the hold time to avoid violating the register hold time. The minimum propagation delay of the registers is 200 ps, so we need to insert at least 100 ps of delay by adding a buffer between them. The same is true between the sum output of the adder and the bottom, rightmost register.
5. If the clock skews right by 2 ns, we need to make sure that it doesn’t violate the setup time of the receiver’s register. If it skews left by 2 ns, we need to make sure it doesn’t violate the hold time of the receiver’s register. The register hold time is 100 ps, while the setup time is 200 ps. Since we want the clock centered, we must use 200 ps on either side, add it to 2 ns and double it to get the clock period. That gives a clock period of 4.4 ns.

6. Start by adding a register to the data line; this register should be clocked by dclk. Starting with time 0, this register will change values at time 0 and time 40. We need to add three registers back-to-back to the dclk line; these registers are clocked by clk. The output of the first of these registers could be out of sync with dclk by as much as 10 ns. It is shown in the figure as changing between time 1 and 11 to indicate the register delay and to make it clear that it will be seen by the next clock, which happens 10 ns later. The output of the second register changes in response to the next clock. The AND gate is inserted so that we only assert the clock enable on the rising edge of dclk. Since the clock enable changes sometime between 11 and 21, the register will be latched by the clock that happens between time 20 and 30, which fits within the 0 to 40 timeframe we established for the data input.
7. For this circuit we have the following quantities:

- \( T_0 = 2\mu s \)
- \( \tau = 300\, ps \)
- \( a = 25\, MHz \)
- \( f = 100\, MHz \)
- \( t_r = t_{\text{clk}} - t_{\text{sw}} - t_{\text{pd}} = 10\, ns - 200\, ps - 300\, ps = 9500\, ps \)

The mean time between failure can be calculated as:

\[
MTBF = \frac{e^{\frac{9500\, ps}{300\, ps}}}{2\mu s \cdot 100\, MHz \cdot 25\, MHz} = \frac{e^\frac{95}{3}}{2 \cdot 100 \cdot 25 \cdot 10^6} = \frac{e^\frac{95}{3}}{5 \cdot 10^9}
\]

This is only 11,316 seconds, which is a little more than 3 hours. Not good!
(Since we use the value from the second register, the third register does not help us at all with synchronization.)

One option is to add another synchronization stage. But this will affect the timing by adding a 10ns delay to the sampling clock, which may not happen as late as time 40, right when the next data value changes. We can fix this by adjusting the relative delay in the two paths. But this increases the MTBF to only a little more than 4 years – which is even worse, because you probably won’t catch it until you’ve sold a bunch.

The second option is to divide the clock to the synchronizers by 2, reducing the sampling rate by a factor of two and the data rate as well. This will complicate the circuit by a lot. If we do this, than we change the above computation to:

\[
MTBF = \frac{e^{\frac{19500\, ps}{300\, ps}}}{2\mu s \cdot 50\, MHz \cdot 25\, MHz} = \frac{e^\frac{195}{3}}{2 \cdot 50 \cdot 25 \cdot 10^6} = \frac{e^\frac{195}{3}}{2.5 \cdot 10^9}
\]

which is more than 2 billion centuries.

8. The circuit from #6 does not work because the data changes at time 0 and 25 now and we sample with a clock somewhere between time 20 and 30. If we add an extra delay to the dclk, then we can move the sample window to a safe region. This adds an extra synchronizer stage, which is the same as option 1 above.