We are given a CBit module shown here.

This module takes corresponding bits of two unsigned binary numbers, A and B, along with two Cin bits from higher-order CBit modules.

Its output bit R, is the appropriate bit of the larger among A and B.

It also has two additional output bits Cout which indicate respectively if A>B or B>A in the bits considered so far.

The propagation delay of each Cbit module is 4ns.

The CBit module is used to create a product MAXC which is a combinational device that determines the maximum of its two 4-bit unsigned binary inputs.

It is constructed using 4 CBit modules as shown here.

The first question we want to consider is what are the propagation delay and throughput of this combinational circuit?

The propagation delay of a combinational circuit is the propagation delay along its longest path from input to output.

In this case, the longest path is through 4 CBit modules.

Since each CBit module has a propagation delay of 4 ns, the propagation delay of this combinational circuit is 4 * 4 ns = 16 ns.

The throughput of a combinational circuit is 1 / Latency, so the throughput of this circuit is 1 / (16 ns).

The next question we want to consider is what two bits would we expect to see coming out of the (unused) Cout outputs from the low-order CBit module if A3:0 and B3:0 are identical numbers.

The Cout[1] bit indicates whether A>B and the Cout[0] bit indicates whether B>A. Since the numbers are identical neither of these inequalities is true, so both Cout bits are 0.

Next, we want to pipeline this circuit for maximum throughput.

We are given ideal pipeline registers for this step.

Recall that ideal registers have a 0 propagation delay and 0 setup time.

In order to pipeline this circuit for maximum throughput, we want to add pipeline registers that isolate each individual CBit to be in its own pipeline stage.

Recall, that when pipelining a circuit, we want to add pipeline registers to all the outputs, so we begin by adding a contour that goes across all 4 outputs.

We now want to isolate the low order CBit module.

To do this we draw an additional contour that crosses the outputs R3-R1, then passes between the two low order CBit modules and finally crosses the A0 and B0 inputs.

Remember that every time a contour crosses a wire, it means that we are adding a pipeline register.

So this first step added 6 pipeline registers, one for each of R3, R2, and R1, another between the two CBit modules and the last 2 on the A0 and B0 inputs.

Notice that regardless of which input to output path we look at the number of pipeline registers along the path so far is 2.

We continue in this manner isolating each Cbit module into its own pipeline stage.

Now that each CBit module has been placed into its own pipeline stage, we can clock this circuit for maximum througput.

Our clock period must allow for enough time for the propagation delay of the pipeline register, plus the propagation delay of the Cbit module, plus the setup time of the next pipeline register.

Since our pipeline registers are ideal, the propagation delay and the setup time of the pipeline registers is 0, so the clock period is equal to the propagation delay of 1 CBit module which is 4 ns.

The latency of this pipelined circuit is equal to the number of pipeline stages (4 in this example) times the clock period time, so the latency is 4 * 4 ns = 16 ns.

The throughput of a pipelined circuit is 1 divided by the clock period, so throughput is 1 / (4 ns).

Our Cbit module can be used to create the MAX4X4 product which is a combinational circuit capable of determining the maximum of four 4-bit binary inputs.

Our first task here is to determine the combinational latency and throughput of this new circuit.

Looking carefully at this circuit, we see that the longest path through the circuit begins at the upper left CBit

module and ends at the bottom right CBit module.

Counting the number of CBit modules along this path, we see that we need to cross 6 CBit modules.

This means that the latency of this circuit is L = 6 * propagation delay of a single CBit module = 6 * 4 ns = 24 ns.

The throughput of a combinational circuit is equal to 1 / Latency = 1 / (24 ns).

Out final task is to pipeline this new circuit for maximum througput.

We begin as we did before by adding a contour that goes across all of our outputs.

Next we want to figure out how to add the remaining contours so that the clock period of our pipelined circuit can be minimized.

The clock period must allow enough time for the propagation delay of the pipeline register, plus the propagation delay of any combinational logic, plus the setup time of the next pipeline register.

Since our pipeline registers are ideal, both the propagation delay and the setup time of the pipeline registers is 0, so our clock period is equal to the propagation delay of the combinational logic between each pair of pipeline registers.

In order to minimize this, we would like the number of CBit modules between each pair of pipeline registers to be at most 1.

This would make our period, T = 4 ns.

To achieve this, we can draw diagonal contours through our circuit.

Notice that these contours result in at most 1 CBit module appearing between any pair of pipeline registers while at the same time including as many CBit modules that can be executed in parallel in a single pipeline stage.

The latter constraint will minimize our latency in addition to maximizing our throughput.

Notice that regardless of which input to output path you follow, at this stage you are crossing 3 pipeline registers.

We continue pipelining our circuit in this manner until we have added enough pipeline stages so that each stage passes through a single CBit module.

We now see that any path from input to output passes through 6 pipeline registers because we have split our circuit into 6 pipeline stages in order to break up the longest path.

We can now clock this circuit with period T = 4 ns.

So the latency is the number of pipeline stages times the clock period which equals 6 * 4 ns = 24 ns.

The throughput of this circuit is 1 / clock period (T) = 1 /(4 ns).