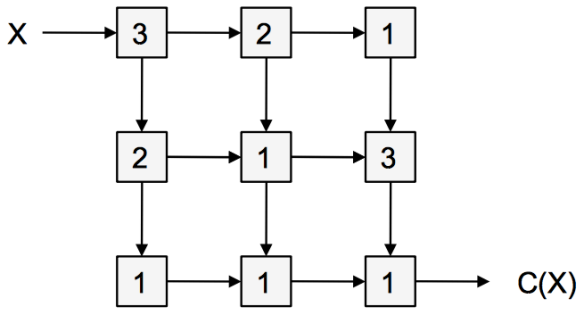
Video explanation of solution is provided below the problem.

Pipelining

6/7 points (ungraded)

The RIAA has come up with a new media encryption engine called the Piper, consisting of nine combinational modules connected as shown to the right.



The device takes a music sample X and computes an encrypted version C(X). In the diagram to the right each combinational component is marked with its propagation delay in microseconds; contamination delays are zero for each component. Unfortunately, it is too slow.

(A) What are the latency and throughput of the Piper device?

Latency (microseconds): 10 ✔ Answer: 10

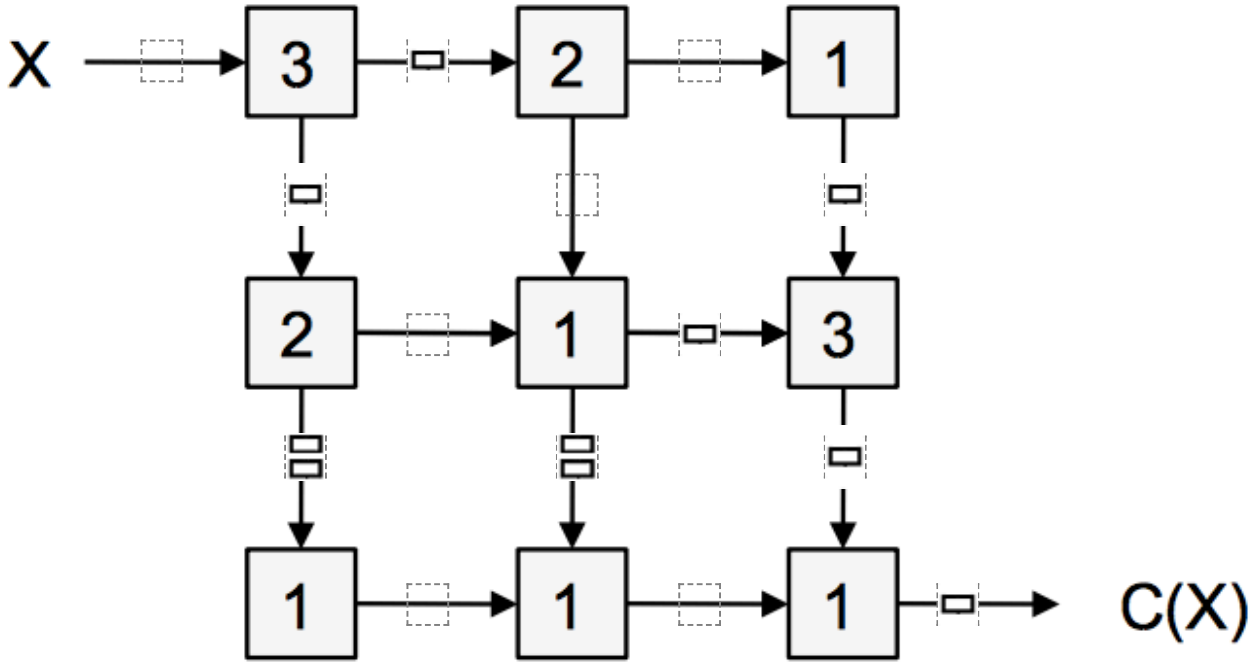
Throughput (1/microseconds): 1/10 ✔ Answer: 1/10

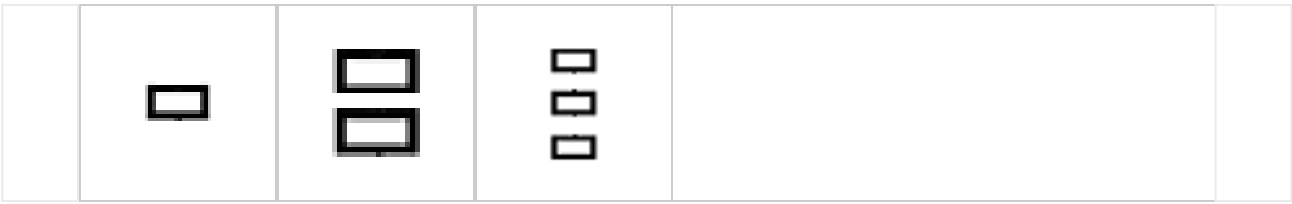
Explanation

The latency of the Piper device is the propagation delay along the longest path from input to output. The longest path for this circuit is a path that goes through two 3 microsecond units, one 2 microsecond unit, and two 1 microsecond unit. So the latency L = 2(3) + 1(2) + 2(1) = 10 microseconds.

The throughput of a combinational circuit is 1/Latency, so throughput = 1/(10 microseconds).

(B) Show the RIAA how to pipeline the Piper by adding registers to maximize throughput, but achieve the smallest latency that meets the maximum throughput constraint. Using the diagram below indicate the locations for ideal (zero-delay) registers to create a pipelined implementation that meets these goals. Remember that your answer should have a register on the output signal.

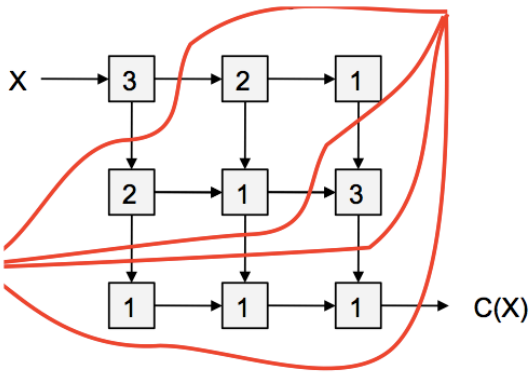




Explanation

In order to pipeline the circuit for maximum throughput, we want to make each pipeline stage include at most 3 microseconds of combinational logic because that is the propagation delay of the slowest component. Recalling that all outputs must have a pipeline register on them, and that each path from input to output must cross the same number of pipeline registers, we can begin drawing our contours. The first goes through all of our outputs. The next wants to isolate the 3 microsecond components so that there is no additional combinational logic that must run before or after the 3 unit component within a single pipeline stage. Note that there are multiple ways that the contours can be drawn to achieve this goal. All of them are correct provided that you end up with 4 pipeline stages each with a propagation delay of 3 microseconds, and that along all input to output paths you have 4 registers.

An example of one such solution is shown here.



Note that the bottom right 1 unit module ends up in its own pipelining stage because it is being isolated from the 3 unit component just above it.
(C) What is the latency and throughput of your pipelined implementation?

Latency (microseconds): 12 ✔ Answer: 12

Throughput (provide your answer in the form 1/X) (1/microseconds):

1/3 ✔ Answer: 1/3

Explanation
The latency of the Piper device pipelined for maximum throughput has 4 pipeline stages with a maximum propagation delay of 3 microseconds. Since ideal pipeline registers are provided, the clock period equals that propagation delay, or T = 3 microseconds. The latency of this pipelined circuit is equal to 4 * clock period = 4 * 3 microseconds = 12 microseconds.
The throughput of this circuit is 1/T = 1/(3 microseconds).

(D) Suppose you found pipelined replacements for the components marked 3 and 2 that had 3 and 2 stages, respectively, and could be clocked at a 1 microsecond period. Using these replacements and pipelining for maximum throughput, what is the best achievable performance?

Latency (microseconds): 9 ✖ Answer: 10

Throughput (provide your answer in the form 1/X) (1/microseconds):

1/1 ✔ Answer: 1

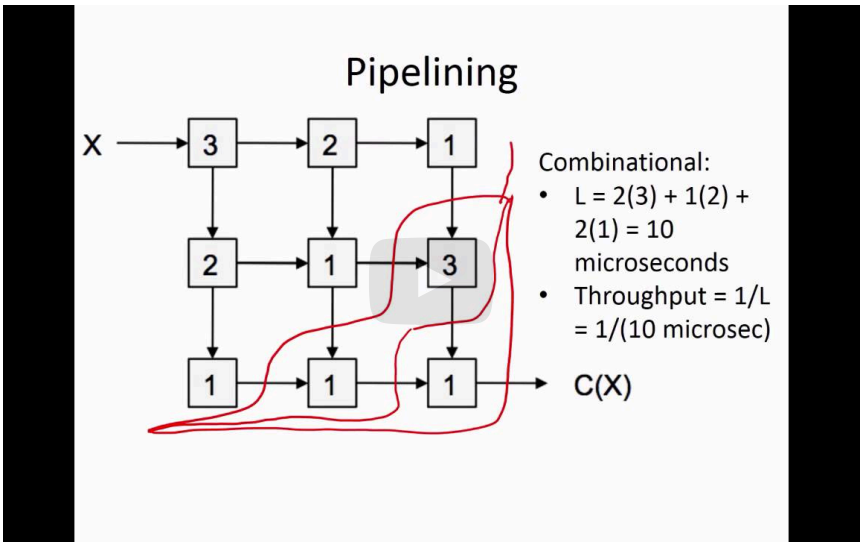
Explanation

Using the new components, the circuit can be clocked with a period of 1 microsecond. However, each 3 unit component now introduces 3 pipeline stages, and each 2 unit component introduces 2 pipeline stages. So the total number of pipeline stages in this design is 10. That means that the latency = 10 * T = 10 microseconds.
The throughput of this circuit is 1/T = 1/(1 microseconds).

Submit

Answers are displayed within the problem

Pipelining 2



0:00 / 0:00 1.0x

stage can be clocked at the rate of our slowest component which is 3 microseconds.

This means that within a single pipeline stage all combinational paths must have at most a propagation delay of 3 microseconds.

Recall, that when pipelining a circuit, all outputs must have a pipeline register on them, so our first contour is drawn so as to cross the single output C(X).

Each pipeline stage should have at most a latency of 3 microseconds.

This means that the bottom right 1 microsecond module and the 3 microsecond module above it must be in separate

Video

Download video file

Transcripts

Download SubRip (.srt) file
Download Text (.txt) file

Discussion

Hide Discussion

Topic: 7. Performance Measures / WE7.2

Add a Post

All Posts

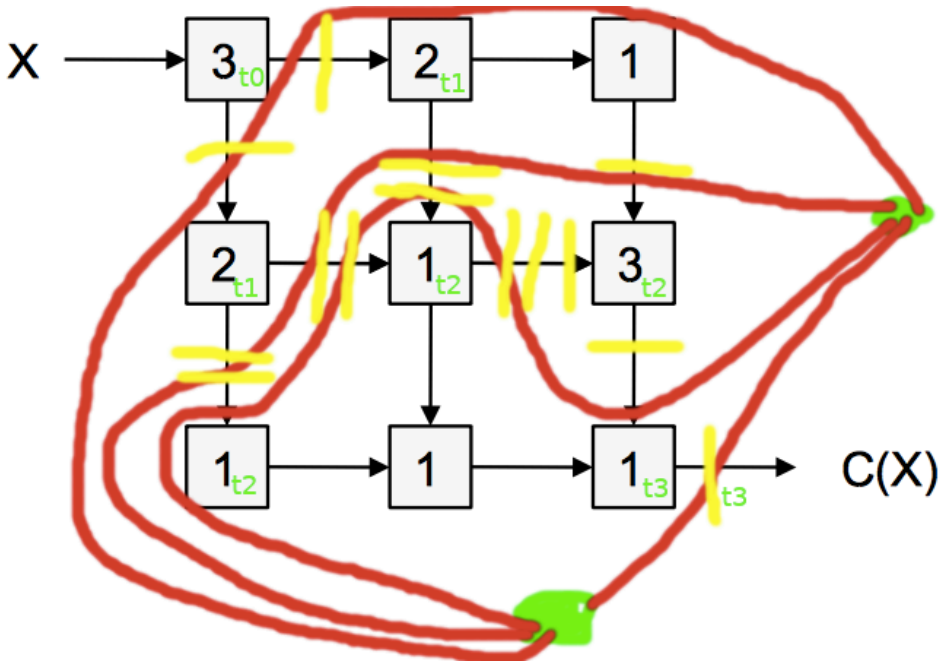
Why isn't the solution below for the pipeline stages correct?

question posted 6 years ago by **Acsor**

I'm pretty faithful about my understanding of combinational pipelining: we slice a whole combinational phase into smaller ones with clock t_{CLK}, where $t_{CLK} = max\{t_{PD,i}\ \forall i\} + t_{SETUP,REG} + t_{PD,REG}$, that is the prop. delay of the slowest component plus delay and setup time of a *single* register. The rate of processed outputs will be then $\frac{1}{t_{CLK}}$ if t_{CLK} is expressed in seconds; the latency $k \cdot t_{CLK}$, with k the number of pipeline stages.

Now, my subdivision below is made under this logic: at each t_{CLK} a signal traverses a pipeline region, as the green t_is show in the figure. Every component is "padded" with registers so that the amount of time a signal travels from one region to the next is exactly t_{CLK}. As the notes advise the pipeline lines (in red) traverse output wires, starting with the first ones. Most importantly, when grabbing new components we group them to achieve the greatest possible added latency.

This reasoning all makes sense, especially if translated into theory, yet my answers are invariably marked as incorrect... I demand help.

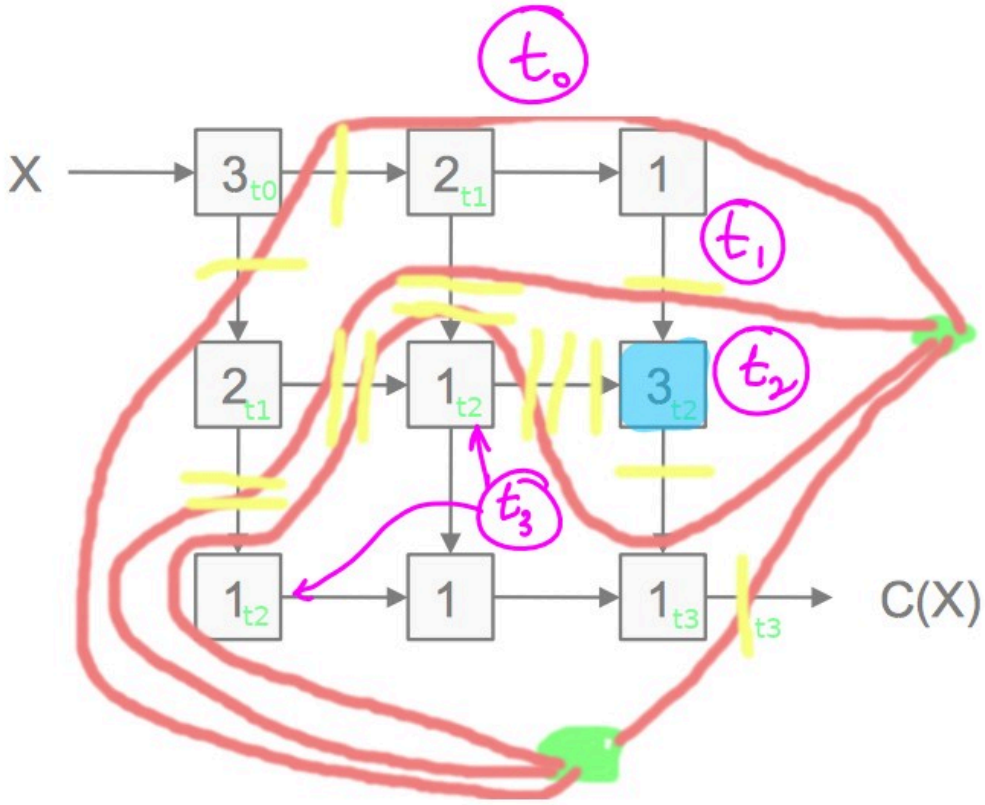


This post is visible to everyone.

rhodesd

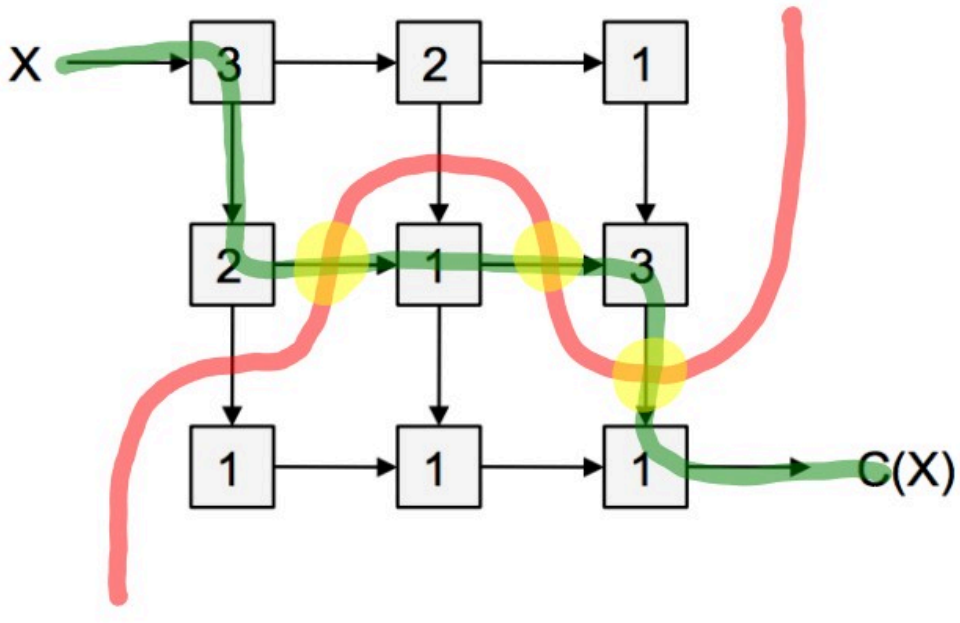
6 years ago - marked as answer 6 years ago by **Acsor**

Hi Acsor, there is a problem with the inputs of the blue component at middle right. Depending on the path it could be in either stage 2 or stage 4. The bits arriving at the top inputs are two cycles ahead of the inputs at the left input, so the output differs from combinational equivalent.



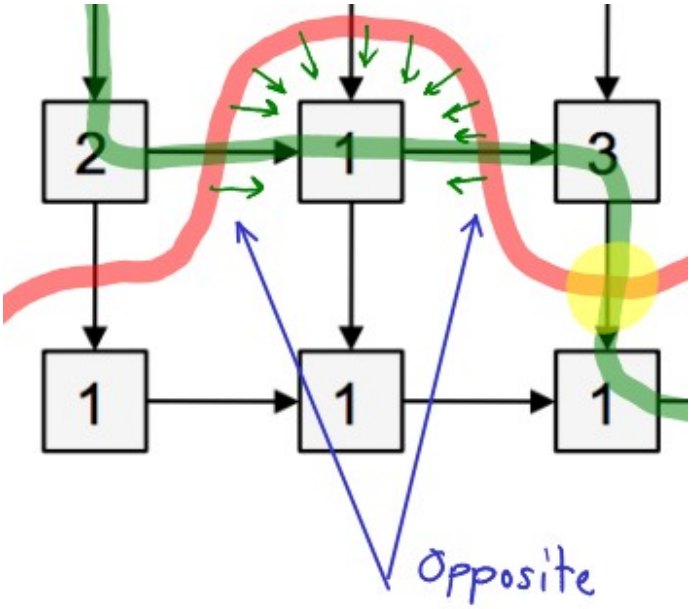
The culprit is the 3rd contour. Below, a green path was chosen that is intersected multiple times by the same contour. The contour is making a declaration: "Here begins stage 3 on all crossed paths", when it cross the same path more than once,

it's saying that different parts of the same path have the registers in the same stage, which is a contradiction.



Rule 2 of the K-stage pipeline says:

*... (b) intersect lines so that every signal crosses the contour in the **same direction.** Each additional contour increases the number of pipeline stages by one.*

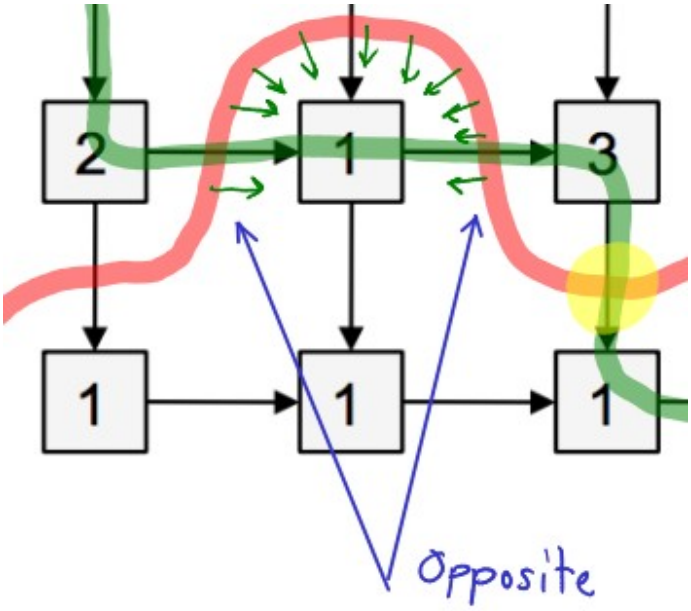


Do you agree?

I believe my two main mistakes were:

1. Following my personal brand of the "padding rule" (registers need only to be added according to the *exact* number of pipeline contours).
2. Not knowing about the second rule of the K-stage pipeline.

For those who would ask something like "How to figure out the "direction" of a contour line?": consider rhodesd's picture



the contour line divides the space into two regions; to my own understanding the only signals involved must be going *into* the demarcated region, not *outside* it, which is what also happens in my wronged attempt.

Everything else was pretty clear, thanks for the help so far!

posted 6 years ago by **Acsor**

Add a comment

Preview

Submit