MITx 6.004.1x

Computation Structures 1: Digital Circuits

Help

Course Progress Dates Discussion Course Notes

Course / 8. Design Tradeoffs / Lecture Videos (37:03)

‹ Previous ✓ Next ›

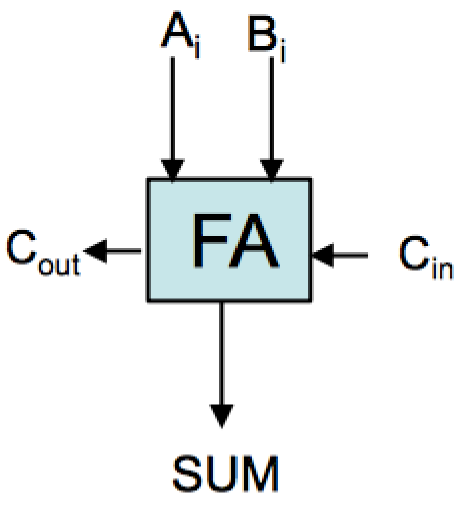
LE8.2

Bookmark this page

Calculator

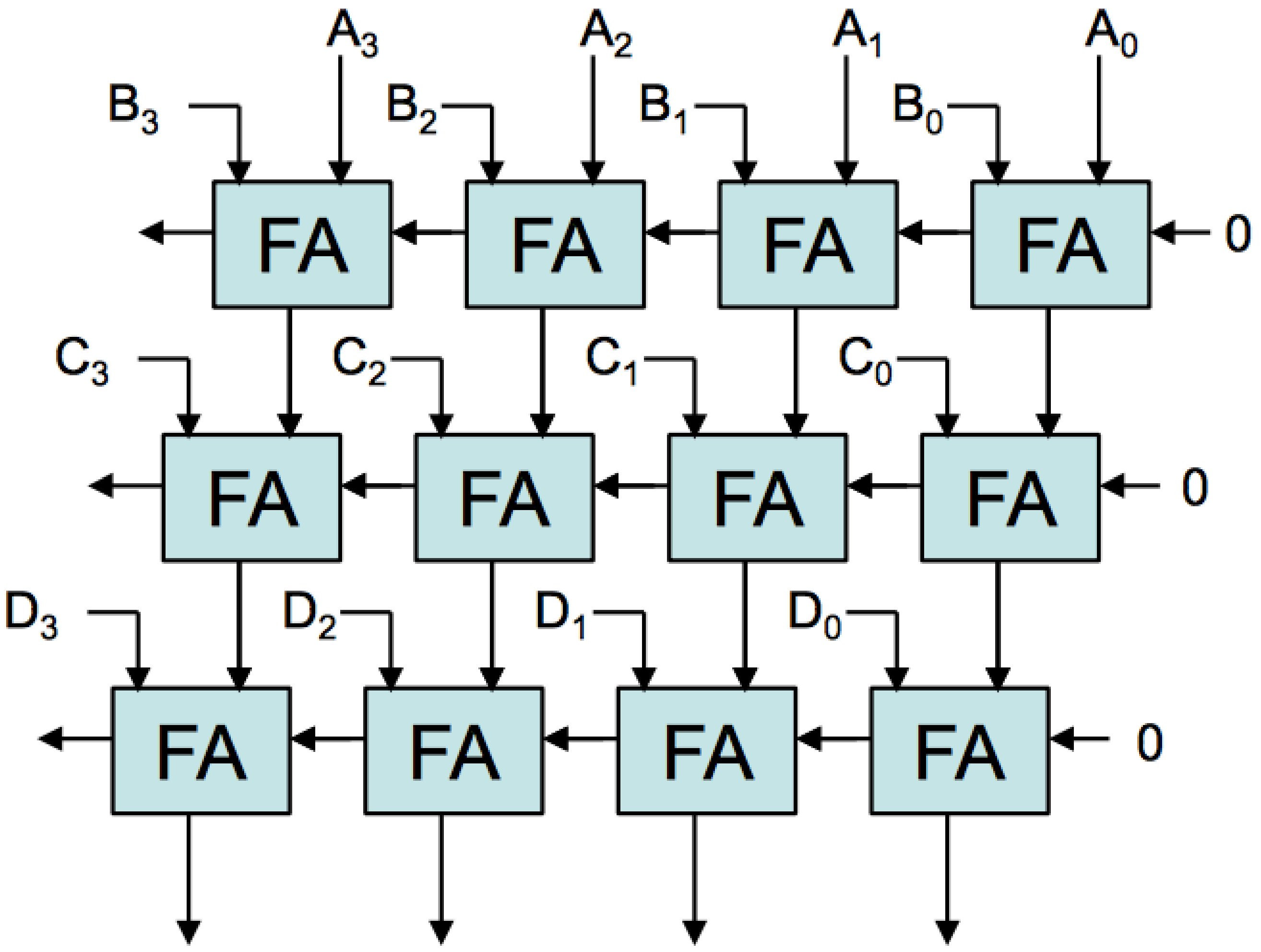
LE8.2.1 Pipelining and Design Tradeoffs

7 points possible (ungraded)



You've been hired by Addplex, a startup specializing in adders capable of summing an entire column of binary numbers as a single operation. Until you came on board, they had no engineers, but had a number of customers for combinational devices capable of adding N k-bit numbers together for various values of N and k. You need to quickly design products to satisfy this demand. You remember the full adder device you designed in lab 2, shown here. Fortunately, Addplex has a large inventory of these on hand.

You begin by tackling the problem of designing combinational circuit that adds four 4-bit binary numbers, producing a 4-bit binary result. Your approach is to combine three copies of the 4-bit adder you built in lab 2 and combine them as shown below:



You connect the low-order carry inputs to 0, and simply ignore the high-order carry outputs. Note that the four 4-bit inputs are designated A[3:0], B[3:0], C[3:0], and D[3:0].

You learn that the propagation delay of each full adder module is 1 nanosecond. What is the propagation delay of the above adder?

Answer: 6 ns

Explanation

The longest path that any bit must take in order to influence the output is the path for A_0 as it influences the output bit of the adder with D_3. In order to get there, it must be added to B_0, added to C_0, added to D_0, carried to D_1, carried to D_2, carried to D_3 and then finally output, going through a total of 6 adders. Because it takes a nanosecond to go through a single adder, that means that it takes 6 nanoseconds for that bit to make it through the system.

You consider generalizing this approach to add N k-bit binary numbers for arbitrary values of N and k. You are particularly interested in the asymptotic cost and performance characteristics of the adder as N and k become large, and recall the $\Theta(\ldots)$ notation that abstracts out additive and multiplicative constants.

Using $\Theta(\ldots)$ notation, give asymptotic **latency** of an adder constructed as above capable of adding N k-bit quantities. Give expressions in terms of N and k.

$\Theta(\ldots)$ Answer: N+k

Calculator

Explanation
The longest path of any similarly construced adder is going to be the patch of A_0, which must be added to N-1 other words, and then carried k-1 times to the highest order bit of the last adder. The overall latency will therefore be $\Theta\left(N+k\right)$.

Using $\Theta\left(\ldots\right)$ notation, give asymptotic **hardware cost** of an adder constructed as above capable of adding N k-bit quantities. Give expressions in terms of N and k.

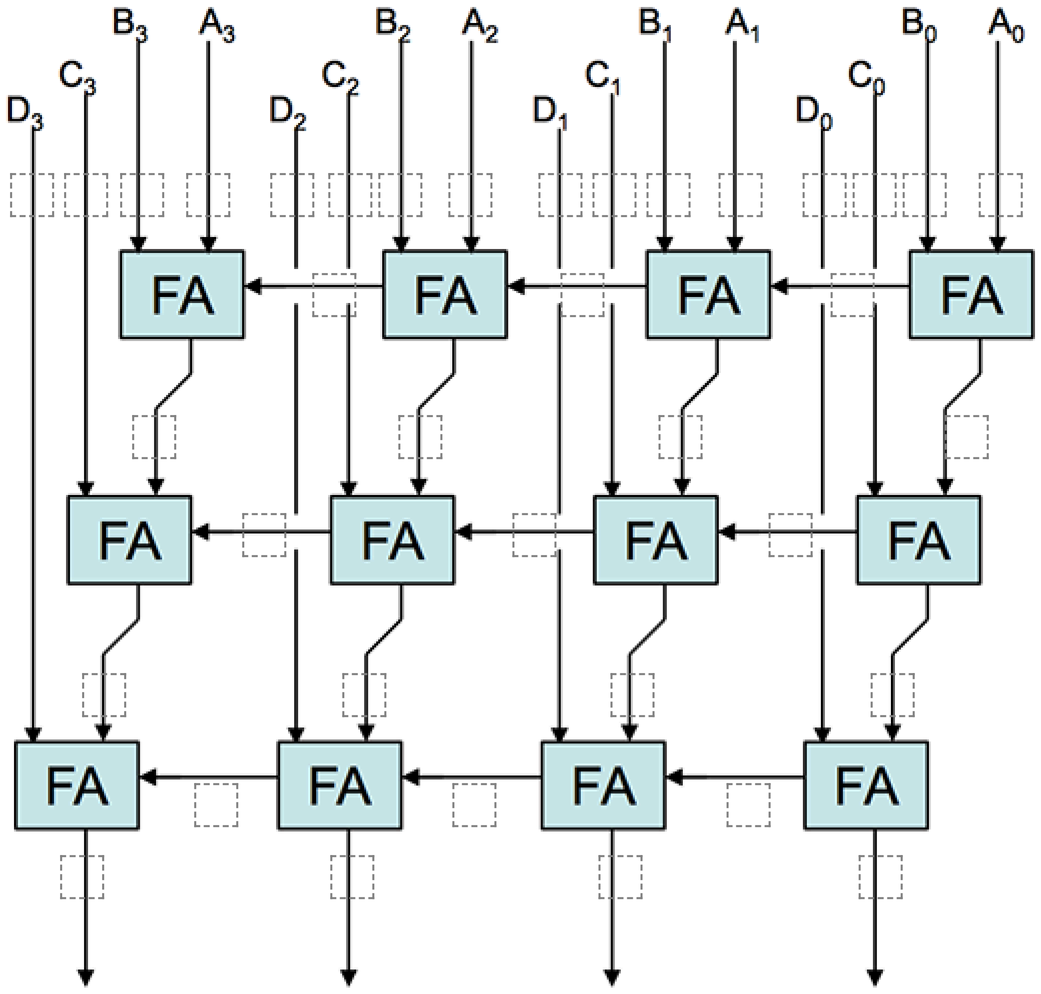
$\Theta\left(\ldots\right)$ Answer: N*k

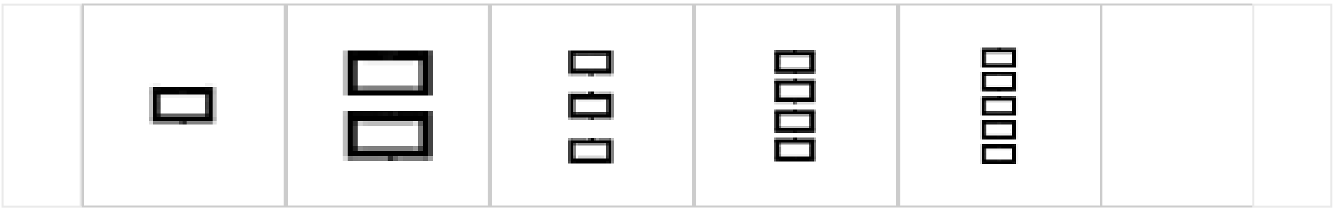
Explanation
The construction of a similarly constructed adder will always be able to form a grid with N-1 rows of k adders, one row to add each new number, and k adders in each row to compute the addition of every bit in those numbers. Therefore the total hardware cost will be $\Theta\left(N*k\right)$.

Next, you turn to the problem of pipelining your adders for maximum throughput. You start by pipelining your 4×4 combinational adder, using ideal (zero-delay) registers inserted at strategic positions:

The diagram below has a single place available for registers on each wire. Indicate how many registers should be added on each wire by dragging the appropriate number of registers to each wire to produce a maximum-throughput pipelined implementation of the 4 by 4 adder.

You may consider printing the image and drawing contours to show the pipeline stage boundaries, and then counting how many pipeline stage boundries are on each wire to know how many registers to put on each wire. Use the minimum number of registers necessary to maximize throughput. Remember to put registers on all outputs. If a wire does not need any registers, do not drag any registers to that wire.





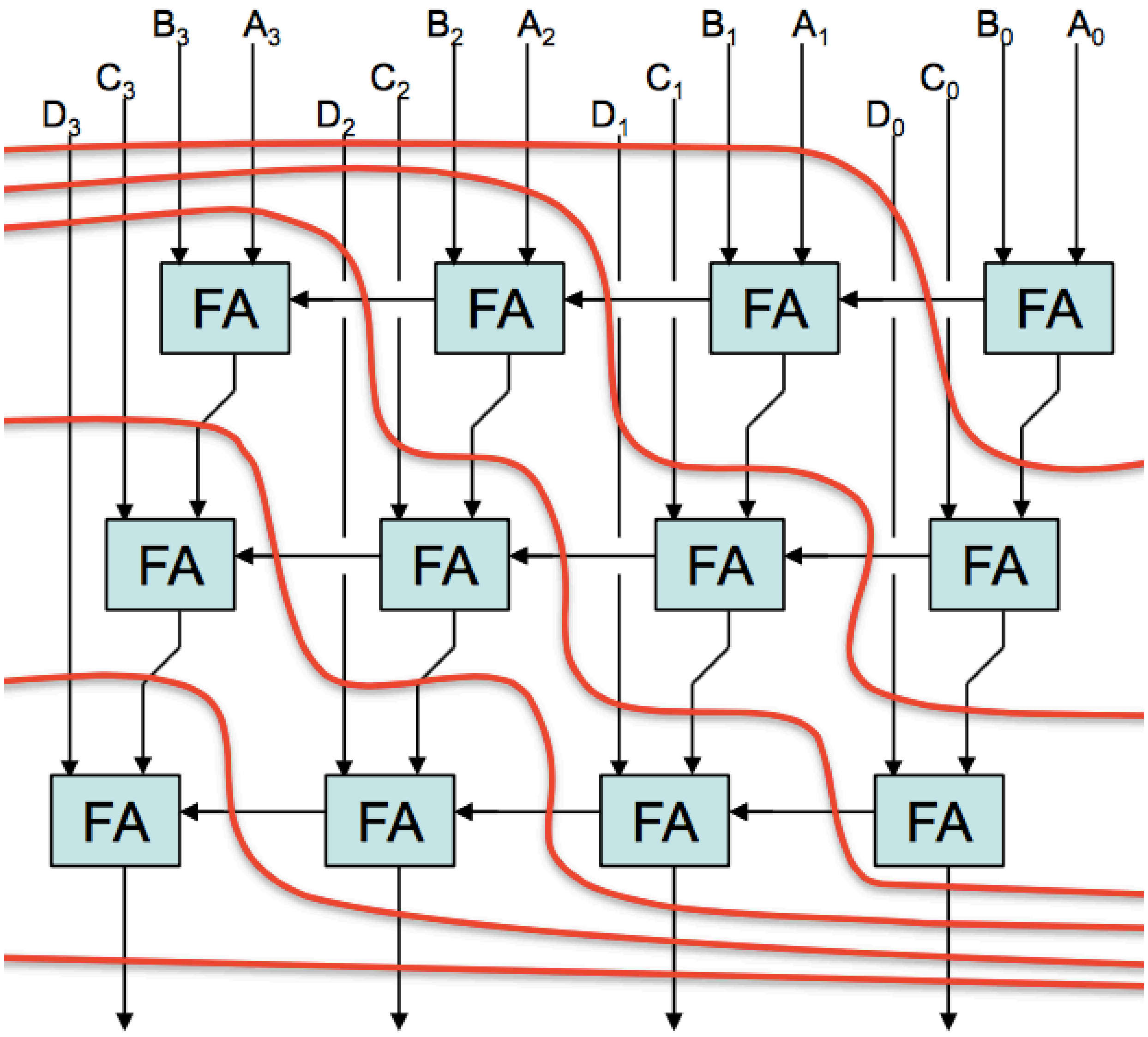
Explanation
There are two constraints on adding registers to inputs to this system.
First, just like in the unoptimized version of the circuit, every input has to propogate through the entire circuit.
Second, even though we are going to have multiple inputs in the system at the same time, every input can c
see other inputs that were added at the same time

Calculator

see other inputs that were added at the same time.

With that in mind, we know that we need to add a register so that the sections of the circuit which take the longest time can have a queue of some length build up behind it. This circuit only has one type of component that takes any time at all: the single bit adder. So first, we add a register before each input to an adder, including the carry bit coming in from other adders. Note that we do NOT need to add any registers to store a value which never changes like the zeros that we use as dummy inputs to our lowest order adders.

After we add those registers, we have to deal with our second constraint: making sure that each bit is only combined with other inputs that were given to the circuit at the same time. We can visualize this constraint much easier if we draw contour lines across the entire circuit as shown below.



As you can see, each contour line provides a break on at least one input to an adder, often more than one, while also making sure that it captures bits coming in from the entire circuit, allowing us to put in our next set of bits, even though the original set of bits haven't been completely processed. We can then count up the total number of contour lines crossing each wire in order to determine how many registers to put on each wire.

Finally, consider generalizing the pipelined adder to add N k-bit numbers.

Using $\Theta(\ldots)$ notation, give asymptotic latency, throughput, and hardware cost of a pipelined adder capable of adding N k-bit quantities. Give expressions in terms of N and k.

Latency $\Theta(\ldots)$: Answer: N+k

Explanation

Our latency does not actually change at all by pipelining our circuit. The overall output is still determined by the longest path, and that longest path is still determined by how A_0 finally affects the D_3 bit, which is $\Theta(N+k)$.

Throughput $\Theta(\ldots)$: Answer: 1

Explanation

The asymptotic throughput of our generalized adder is determined by the stage in the throughput that has the longest propogation delay. In this case, each stage has the same amount of propogation delay - one nanosecond - and that propogation delay is not affected by how many stages there are, nor is it affected by how many inputs there are. The overall asymptotic delay is therefore constant, $\Theta(1)$.

Hardware Cost $\Theta(\ldots)$: Answer: N*k*k + N*N*k

Calculator

Explanation
The hardware cost for the **unpiplined** adder is $\Theta\left(N*k\right)$ because we need $\Theta\left(N*k\right)$ adders. See the solution to the unpiplined hardware cost for more explanation on the unpiplined adder.
For every adder in the circuit, we need to add registers for every stage in the pipeline, making sure that each input is only being added to other inputs that were given at the same time. That gives an overall asymptotic hardware cost of $\Theta\left(N*k*\left(N+k\right)\right)$. Note that if k is assumed to be much larger than N, then this reduces to $\Theta\left(N*k*k\right)$.

Submit

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: 8. Design Tradeoffs / LE8.2

Add a Post

All Posts

[Staff] How registers get counted in \Theta notation?

question posted 7 years ago by **mmdrebi**

Hello,

Please give us clear explanation of how registers get counted in \Theta notation for the last question about **Hardware Cost**

This post is visible to everyone.

silvinahw (Staff)
7 years ago - marked as answer 7 years ago by **silvinahw** (Staff)

See this post.

Add a comment

Previous Next

Calculator

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap
Cookie Policy
Your Privacy Choices

Connect

Idea Hub
Contact Us
Help Center
Security
Media Kit

GET IT ON Google play

Download on the App Store

Calculator