CS303 - Logic Design – 2023 Fall Term

Lab 3 Report

Berke Ayyıldızlı 31018

3 files have been made for this lab assignment. A 4 bit carry-look-ahead adder, CLA\_4bit.v, A 16 bit carry-look-ahead adder that uses 4 of the 4 bit CLA’s, CLA\_16bit\_top.v, and a test bench, CLA\_16bit\_tb.v .

First, let’s take a look at the 4-bit CLA,

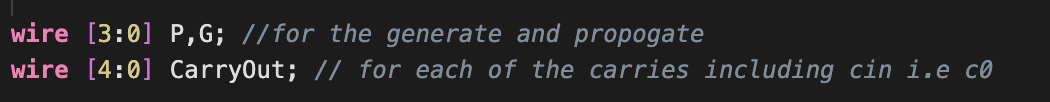
metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

This is the file that is given to us, the input and output values are as follows.

Then, I declared 2 values, a generate signal as G, and a propagate signal as P.

Also a CarryOut signal is also necessary, for both holding the values of the carries from each adder circuit.



After that, I assigned each P and G value, with their bitwise values, also implementing the mode value with xor’ing it with an input.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

This approach is taken from here:

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

The firs carryout is our inputted carry value.

After the declaration, I assigned each carry out, according to the chart in the lecture slides.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

The last carryout is our main carry input.

Also we assign the sums(RES) according to again the chart in the lecture slides.

metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

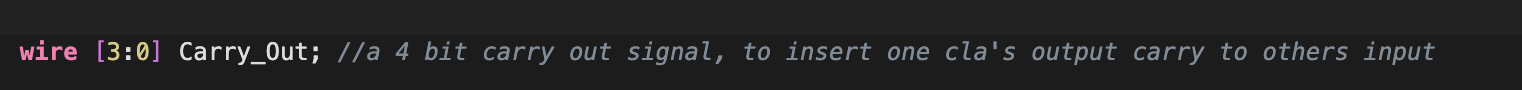
Let’s continue with the 16 bit CLA.

First we start with including our 4 bit CLA, and with the module declaration.

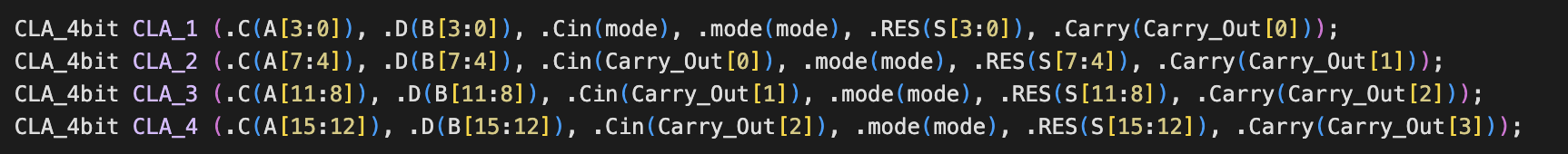
metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

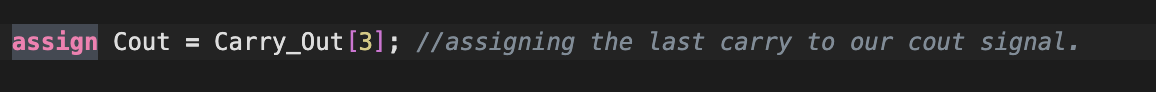
Then, a 4-bit Carry\_Out input is declared. This is keeping track of each 4-bit CLA’s carry and assigning it to one another’s cin value.



After that, I called the 4-bit CLA module for 4 times, with each times, changing the carry and sum values.



This approach is again taken by the lecture slides.

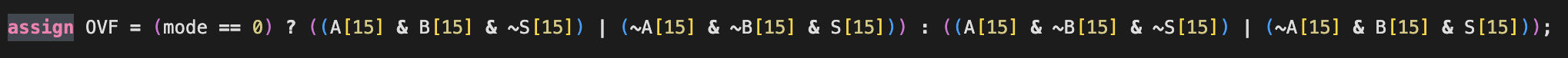


Last but not least, the last bit of the carry out signal, is our main Cout.

For the overflow detection part, I tried to use the second approach that is given by the lecture slides. Since I couldn’t came up with a solution with the xor approach.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu



Since I needed 2 separate values for the mode (adder and subtracter) and I did not want to use if or else operations, I used ternary operator here. But since I had 2 midterms and 1 project phase submission, i am having errors on the simulation part. I know I could solve the issues by debugging or by calculating values differently than this approach, sadly this is the best solution I can come up with now.

After the end of the 16 bit file, I am continuing with the test bench file.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

After the declaration part, I tried to give 8 different examples for each wanted output, but again, no matter the numbers I tried, (for example case 3) I couldn’t properly found a way to demonstrate the overflow mechanic. This can be happening due to wrong approach on the overflow declaration or wrong binary inputs.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Now, lets continue with the simulation results.

ekran görüntüsü, multimedya yazılımı, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

As you can see, the results of case1, 2 ,4, 5, 7 is correct. But for some reason some cases do not show the overflows and some cases do not show the Cout signal.

Again, I did tried to do my best in the given time, and I know that the remaining part can be fixable. I just couldn’t figure out how to fix the overflow mechanic and which numbers to give for the test bench.

Thank you for your time.