Verslag

Titel: Project 2: Adders

Dit verslag werd opgesteld door:

- Naam: Van pellicom Gilles
  Studentennummer: s0211449
  Email adres: gilles.vanpelicom@student.uantwerpen.be
- Naam: Caluwé Jonas
  Studentennummer: s0210051
  Email adres: jonas.caluwe@student.uantwerpen.be

Aantal man-uren besteed: 4 uur

Moeilijkheidsgraad: 6 / 10 (1 is heel makkelijk, 10 is heel moeilijk)

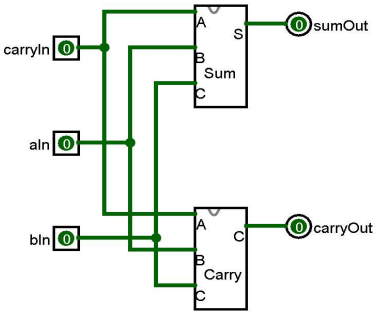
Inhoud van de oplossing

De oplossing bestaat uit de volgende bestanden (geef alle bestanden op):

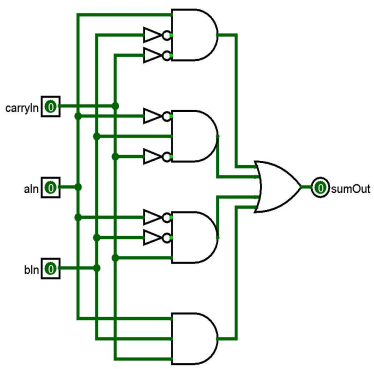
- Adders.circ

Verslag

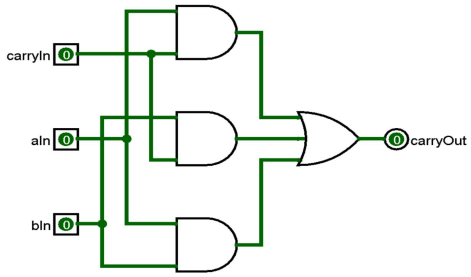
1. **The 1 bit full adder**



De one bit full adder werkt als een soort van decoder. Hij krijgt drie inputs: een A (é é n bit), een B (é é n bit) en een Carry (é é n bit). Op basis van deze inputs genereert hij dan een result (het optellen van A, B en de Carry) en een CarryOut mocht dit resultaat twee bits bevatten. Omdat ze elk hun eigen functie hebben, besloten we om ze in twee aparte circuits te decoderen. De Sum ziet er als volgt uit.

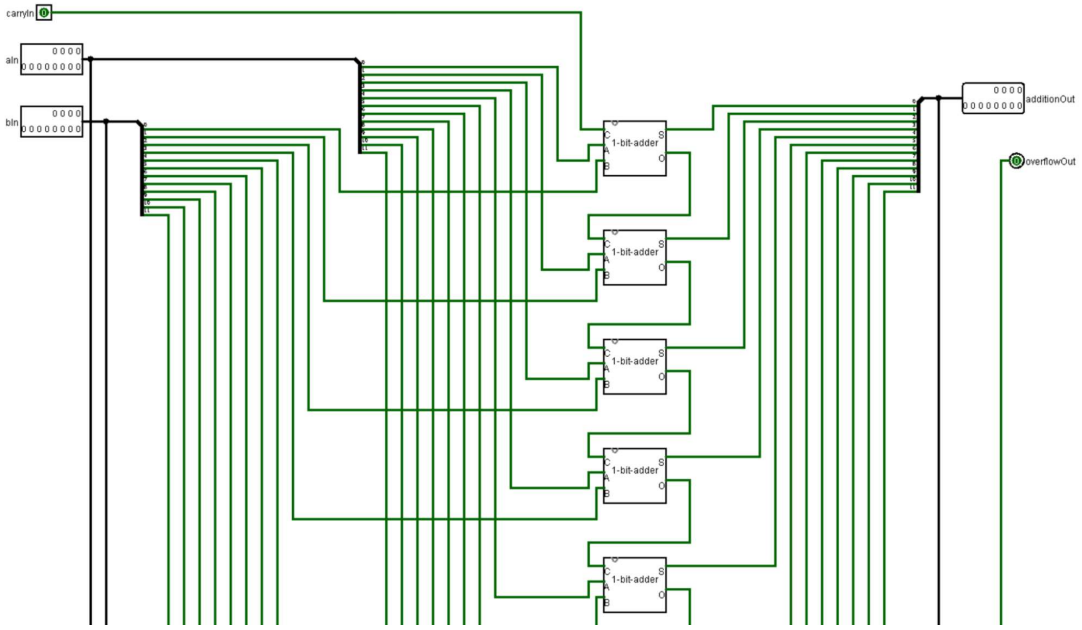


Zoals u kan zien decodeert de sum een A (é é n bit), een B (é é n bit) en een Carry (é é n bit) naar een SumOut (é é n bit). Om deze “ decoder” op te stellen hebben we een truth table opgesteld en die expressies geï mplementeerd als een PLA (Programmable Logic Array). Het voordeel aan een pla is dat deze snel is, namelijk een latency van 2eps met eps de vertraging die é é n gate teweeg brengt. Het nadeel is dat een PLA per definitie meer gates nodig heeft dan custom gemaakt circuits. In dit geval hebben we relatief weinig gates, omdat we niet veel sommen hebben die é é n als resultaat hebben (alleen als é é n van de drie of alle drie de inputs aan staan). Daarom maken we dan ook gretig gebruik maken van de snelheid van een PLA.

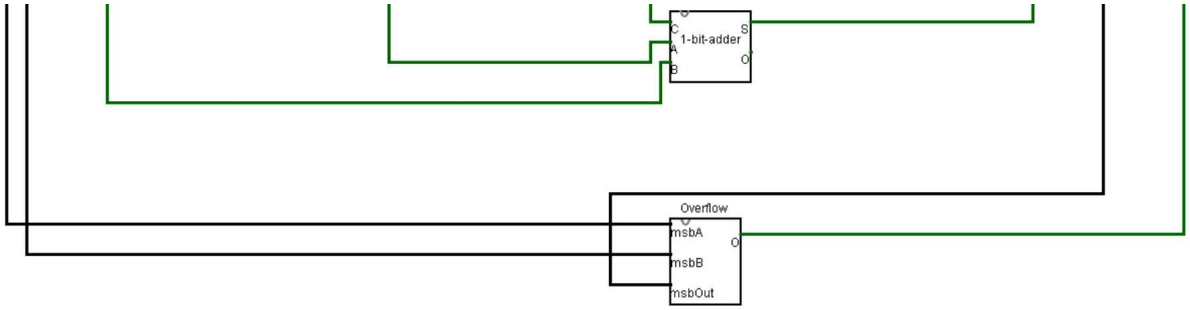


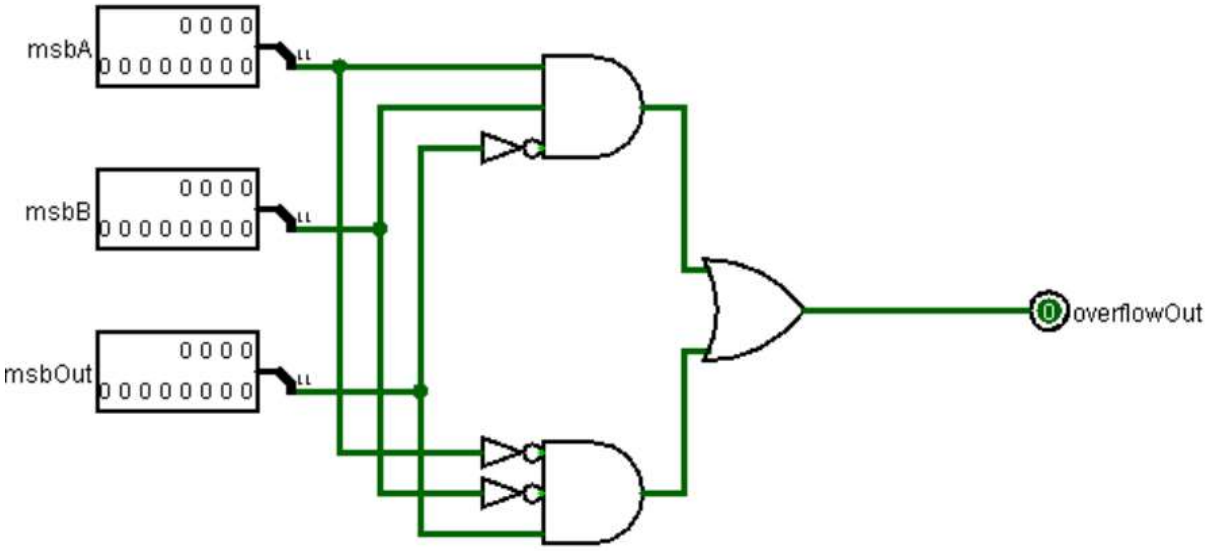
Ook bij het carry gedeelte zijn we gegaan voor de PLA implementatie aangezien het aantal gates hier nog minder is dan bij de som en deze had al relatief weinig gates. Dus ook hier gaan we voor de snelheid aangezien dit belangrijk is om de ripple carry en de carry lookahead (zie verder) zo efficië nt mogelijk te maken.

2. The 12 bit two’ s complement ripple carry adder



De ripple carry adder heeft é é n groot voordeel, hij is niet moeilijk om te maken. Hiermee bedoel ik dat hij geen complexe abstractie niveaus heeft of parallelle berekeningen moet doen. Op dat vlak is hij eenvoudig. Echter heeft hij é é n groot nadeel, zijn latency. Vanaf dat er een input wordt gegeven, begint elk circuit met rekenen, echter klopt de berekening op tijdstip 2eps (de circuits zijn PLA’ s) maar voor é é n circuit en dat is circuit é é n. Al de anderen produceren wel een uitkomst, maar die is gewoonweg fout. De manier waarop de volledige adder is gebouwd is dat eerst circuit 1 zijn berekening doet en een CarryOut genereert. Deze CarryOut is de CarryIn voor het volgende circuit, etc. De circuits staan in serie en dus kunnen we er pas vanuit gaan dat de output (AdditionOut) klopt wanneer n*2eps gepasseerd is (n is in dit geval 12), tenzij de overflow bit aanstaat. De overflow bit is een manier om zeggen dat de output die dit geval is gegenereerd niet klopt. Het is te zeggen hij klopt wel, maar zijn betekenis niet (bijvoorbeeld de som van twee negatieve getallen die een positief getal oplevert). Om deze overflow te berekenen gebruiken we een extra circuit.

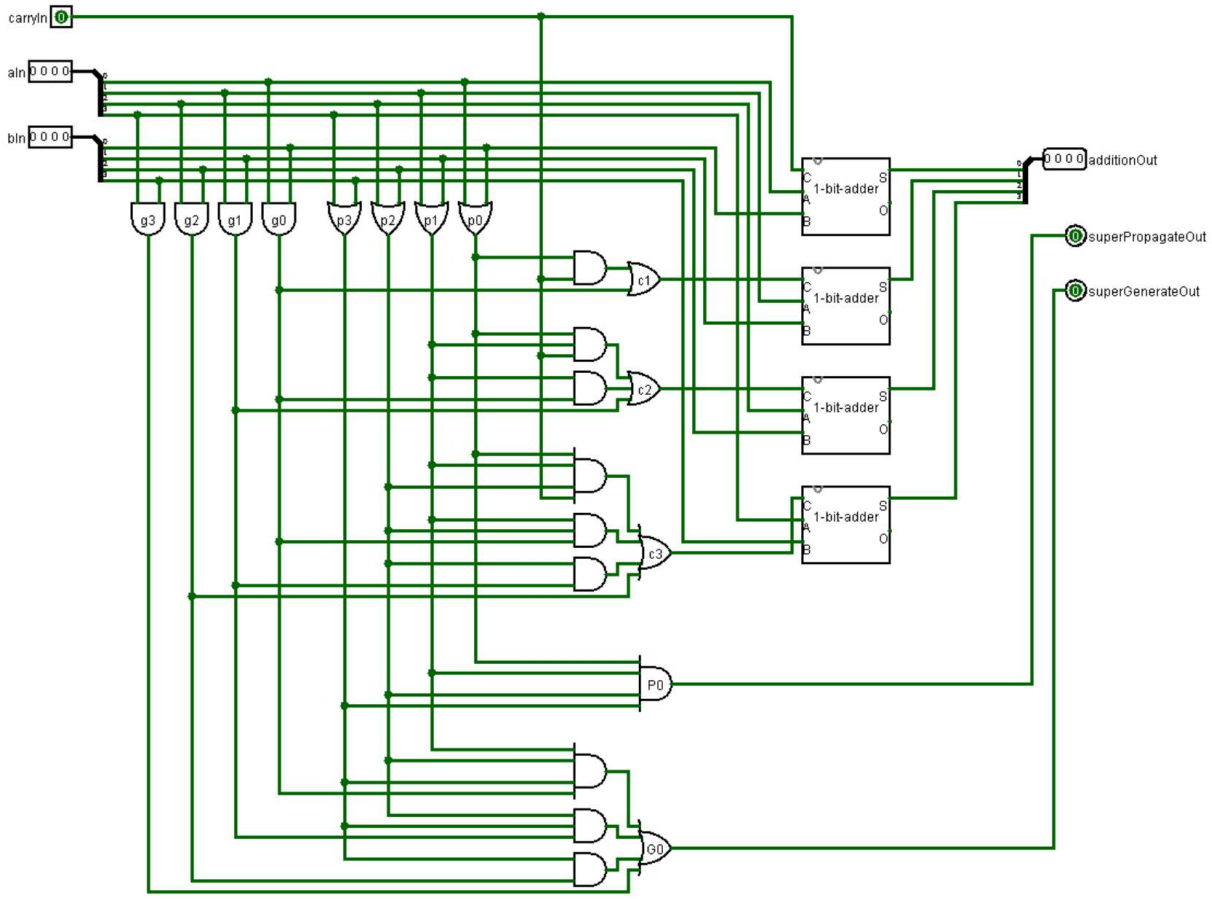




De overflow baseert zich op drie inputs: de MSB (most significant bit) van A, de MSB van B en de MSB van de volledige som . Deze overflow detectie implementeert alleen de regels die gelden voor de som, niet voor substraction, bit shiften, etc. (cfr. Verslag Project 3: ALU). Wanneer een positief getal bij een positief getal wordt opgeteld en een negatief getal de uitkomst is of wanneer een negatief getal wordt opgeteld bij een negatief getal en het resultaat positief is, dan spreken we van overflow. Voor gemakkelijkheids redenen aanvaarden we hier niet de MSB' s maar de volledige dataline. Dit om de circuits zo eenvoudig mogelijk te houden.

3. The 12 bit two' s complement carry lookahead adder

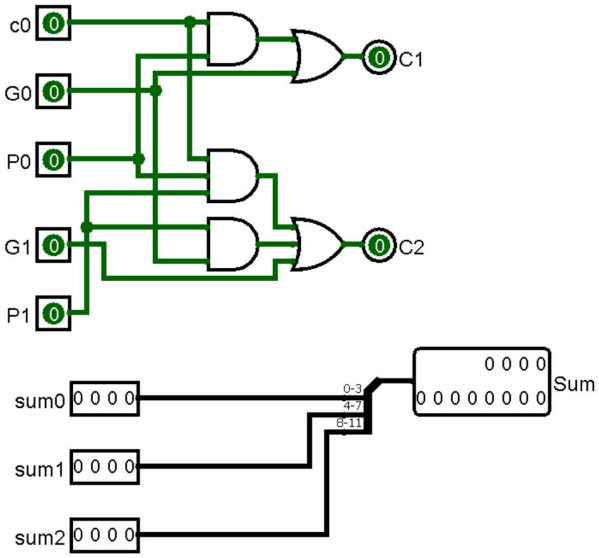
1. The 4 bit carry lookahead adder (abstractieniveau 1)



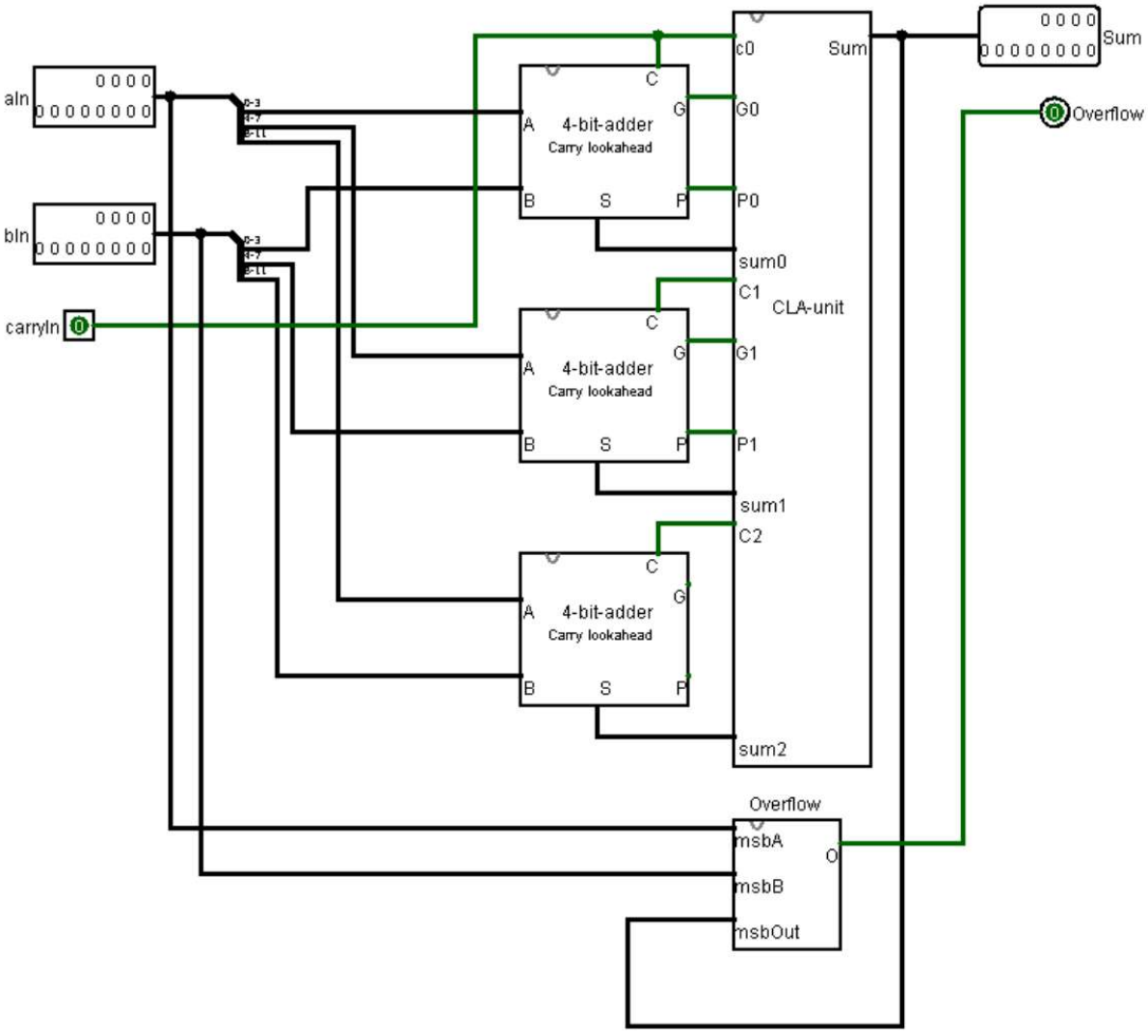
De 12 bit CLA (Carry Lookahead Adder) is een stuk complexer dan de ripple carry adder. We hebben namelijk twee abstractieniveaus. Het idee van een CLA is dat de circuits niet meer moeten wachten op een output (CarryOut) van het vorige circuit, alles gebeurt namelijk in parallel. Alle CarryIn' s worden hier tegelijk berekent in 3eps (behalve Carry 0, deze is namelijk een input) waardoor elke 1 bit adder circuit een output klaar heeft na 5eps. Deze berekeningen gebeuren aan de hand van propagation en generation (wordt hier een nieuwe bit gegenereerd of wordt de vorige doorgelaten). In vergelijking met de latency van een vier bit ripple carry adder, is 5eps een kleine 40 procent sneller. Ook ziet u hier twee andere outputs: Super Generate en Super Propagate dit is hoe het circuit aan het tweede abstractieniveau zegt of hij een extra carry zal genereren of hij de vorige zal doorlaten.

2. **The 12 bit carry lookahead adder (abstractieniveau 2)**

Het parallel berekenen van de carries had al een redelijk groot voordeel bij de 4 bit CLA (40 procent sneller). Wanneer we echter meer bits gaan toevoegen, wordt een niveau 1 CLA zeer complex en heeft hij exponentieel meer gates nodig om de carries te berekenen. Dit is waar een tweede abstractieniveau in het spel komt. We laten namelijk elk circuit vier bits in parallel berekenen, maar het propageren en genereren van de Carry zal jammer genoeg wel in serie gebeuren over de drie vier bit adders (per vier bit circuit gebeurt dit parallel). Daarom hebben we de CLA Unit nodig.



De CLA Unit berekend op basis van de Super Generations en Super Propagations of er een carry in nodig is voor het volgende circuit. En zo zal de uiteindelijke latency neerkomen op 15eps overflow exclusief. Dit is een aanzienlijk verschil over een 12 ripple carry adder (24 eps overflow exclusief), namelijk een verschil van 40 procent in snelheid. De overflow wordt in beide gevallen echter niet parallel berekent want deze moet wachten op de MSB van het resultaat, dus moeten we er in beide gevallen een constante van 2 eps bij rekenen. Dit komt neer op een snelheidsverschil van 35 procent, wat nog steeds een prachtig resultaat is. Hieronder de volledige 12 bit CLA.



4. Conclusion

In het algemeen was dit geen moeilijke taak, maar zat het hem meer in het begrijpen van de verschillende adders en het begrijpen van overflow bij two’ s complement. We hebben dan ook het meeste van onze tijd gespendeerd aan dit verslag en aan het labelen, rewiring van onze circuits.