ELE10 Opdrachten

Inhoudsopgave

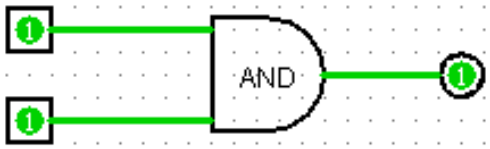
Introductie

De onderstaande opdrachten zijn geschreven met als doel gevoel te ontwikkelen voor digitale logica. Werk deze opdrachten door en zoek zo nodig de theorie op in Hambley. Bij elke opdracht wordt gebruik gemaakt van Logisim. Bekijk de Logisim handleiding in de netwerkmap voor meer uitleg.

Week 1

Opdracht 1

De basis van de digitale logica wordt gelegd door de logische poorten AND, OR, NOT. Alle andere poorten kunnen hier uit voort vloeien. Een basis poort heeft altijd maximaal 1 uitgang maar minstens 1 ingang.



Om te zien hoe een bepaalde poort zich gedraagt bij verschillende ingangssignalen kunnen we Logisim gebruiken. Vervolgens kunnen we de resultaten in zogenaamde waarheidstabellen neerzetten. Gebruik het simulatie programma om de onderstaande waarheidstabellen in te vullen. Eén waarde is gegeven.

AND Poort		
IN1	IN2	UIT
0	0	
0	1	
1	0	
1	1	**1**

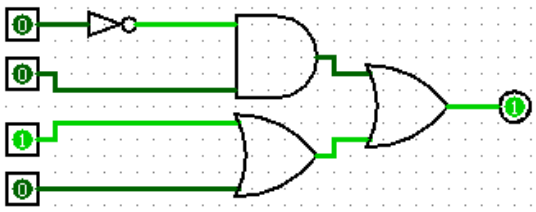
OR Poort		
IN1	IN2	UIT
0	0	
0	1	
1	0	
1	1	

NOT Poort	
IN1	UIT
0	
1	

Een waarheidstabel laat het verband zien tussen de ingangen en de uitgangen bij een specifieke poort of schakeling. Om alle mogelijkheden na te gaan, worden de ingangen binair nagelopen van 0 (00) tot en met 3 (11)

Opdracht 2

Met behulp van de poorten die we nu kennen, kunnen we andere 'poorten' bouwen. Om dit te bereiken moeten we verschillende poorten combineren zoals weergeven in figuur 1. Maak zelf een willekeurige combinatie van poorten en vind de daarbij behorende waarheidstabel. (4 ingangen, 1 uitgang)



Figuur 1 - Schakeling met 4 ingangen en 1 uitgang

We noemen zo'n gecombineerde 'poort' ook wel een (digitale) schakeling. Jouw schakeling moet beschikken over 4 ingangen. De waarheidstabel van 'Figuur 1' zou de 16 mogelijke combinaties van de 4 ingangen beschrijven en de daarbij behorende output.

Vind het decimale equivalent van elke rij in je waarheidstabel. Bijvoorbeeld:

IN1	IN2	IN3	IN4	Uit	Dec
1	0	0	1	1	19
1	0	1	0	1	21

Opdracht 3

Hieronder zijn 3 waarheidstabellen weergeven. Maak schakeling 1, 2 en 3 in logisim.

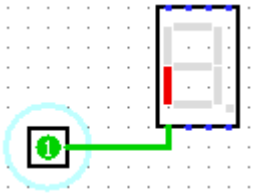
Schakeling 1		
IN1	IN2	UIT
0	0	**0**
0	1	**1**
1	0	**0**
1	1	**1**

Schakeling 2		
IN1	IN2	UIT
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Schakeling 3		
IN1	IN2	UIT
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**0**

Opdracht 4

In Logisim is ook het component '7-Segment Display' te vinden onder de categorie 'Input/Output'. Zo'n 7-segment display heeft voor elk segment een aparte ingang. Maak een schakeling met de volgende eisen:



- De schakeling heeft 1 ingang en zoveel uitgangen als je nodig hebt.
  - Een 0 op de ingang laat een 0 zien
  - Een 1 op de ingang laat een 9 zien.
- We negeren de 'dot' van het display

Voor meer informatie over hoe je een display aansluit, kijk naar opdracht 1 van week 3.

Tip: Maak ook gebruik van het 'constant' element onder 'wiring'.

Week 2

Opdracht 1

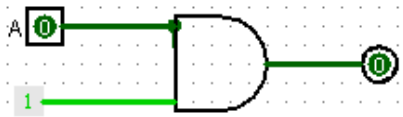
Naast waarheidstabellen is er ook een tweede manier om digitale schakelingen te beschrijven. Door de eigenschappen van poorten te koppelen aan aritmetische operaties ontstaat er een nieuw soort algebra; Booleaanse Algebra. Deze algebraïsche vorm is handig omdat het toe staat equivalente maar kleinere vormen te vinden.

Hieronder zijn een aantal basisregels gegeven van deze algebra. Bevestig deze regels met behulp van Logism simulaties.

Zie Paragraaf 7.3 in Hambley voor meer theorie.

AND

$A1 = A$
$A0 = 0$
$A(BC) = (AB)C = ABC$



Figuur 2 - Schakeling A1=A

NOT

$A\bar{A} = 0$
$\bar{\bar{A}} = A$

OR

$(A + B) + C = A + (B + C) = A + B + C$
$A(B + C) = AB + AC$

$A + 0 = A$
$A + 1 = 1$
$A + \bar{A} = 1$
$A + A = A$

Opdracht 2

Naast de bekende 3 poorten bestaan ook de NAND en de NOR. De eerste N staat voor NOT bij beide poorten. Een NAND is in feite een AND poort gekoppeld aan een NOT poort.

De wetten van 'De Morgan' (7.3 Hambley) impliceren dat enkel NANDs of NORs gebruikt kunnen worden om ALLE andere poorten te bouwen. Bouw de AND, OR en NOT poorten met behulp van enkel NANDs, vervolgens met behulp van enkel NORs..

Simuleer dit in Logisim en controleer de werking met de tabellen van opdracht 1 uit week 1.

Opdracht 3

Bouw de onderstaande schakeling in Logisim en genereer de waarheidstabel.

$$E = D + BA + BC + B\bar{A} + ABD$$

Optimaliseer de expressie met algebra en verifieer dat deze schakeling identiek werkt als de eerste met behulp van Logisim. Schrijf goed je tussenstappen op.

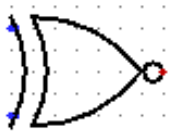
Opdracht 4

Verifieer de onderstaande regels van De Morgan met Logisim. (Vergelijk de waarheidstabellen)

$$ABC = \overline{\bar{A} + \bar{B} + \bar{C}}$$
$$(A + B + C) = \overline{\bar{A}\bar{B}\bar{C}}$$

Opdracht 5

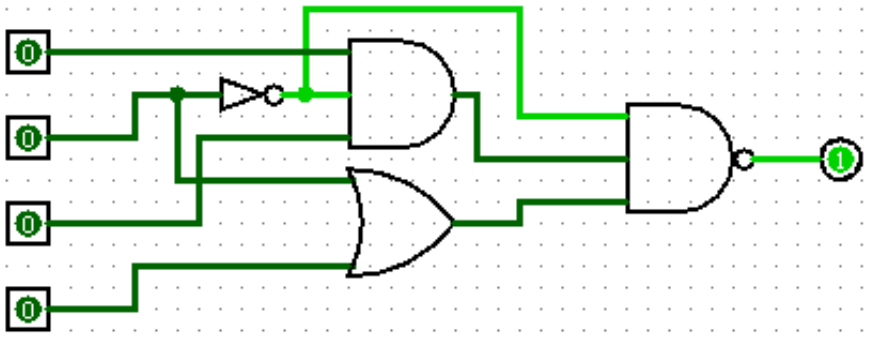
Logisim beschikt over twee poorten welke we tot nu toe nog niet hebben gebruikt. Test de XOR en XNOR poorten en beschrijf ze in waarheidstabellen.



Figuur 3 - Een XNOR / Equivalence poort

Opdracht 6

Beschrijf de nevenstaande schakeling met behulp van algebra. Minimaliseer dit en test de werking. Laat je tussenstappen zien.

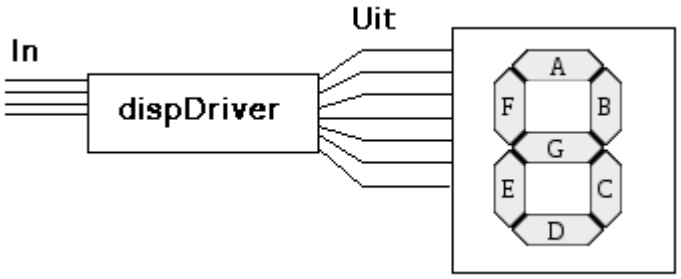


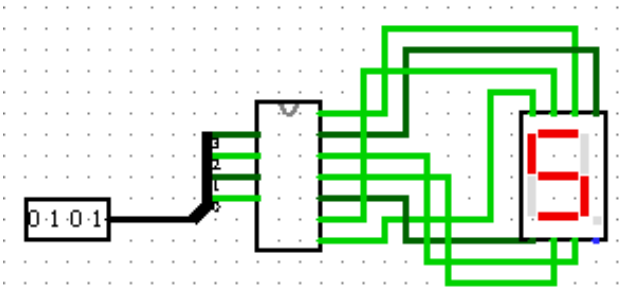
Figuur 4 - Opdracht 6

Week 3

Opdracht 1 –Display driver

Rekenmachines maken gebruik van zogenaamde 7-segment displays. Veelal wordt er BCD code gebruikt omdat dit het rekenen makkelijker maakt (7.2 Hambley).





Ontwerp een display driver waarbij we als ingang 4 bits hebben, en als uitgang de 7 bits voor elk segment van een enkele display. Staat op de ingang 0101 dan zien we op de display '5'.

BCD gaat maar tot 9 dus geef een error 'E' als de decimale waarde van de ingang boven de 9 uitkomt.

Maak voor deze opdracht gebruik van SOP.

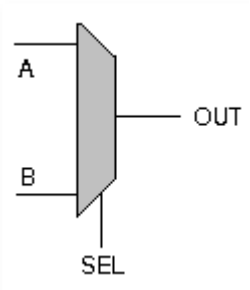
Let op, het component wat hierboven de 5 uitstuurt is wat je zelf moet maken. Wanneer jouw schakeling af is, kan je het opslaan als component en gebruiken in volgende opdrachten.

Opdracht 2

In grote digitale schakelingen is er vaak de wens, om te kunnen 'kiezen' of schakelen tussen twee digitale signalen. Deze wens heeft uiteindelijk geleid tot de multiplexer schakeling.

De 1-Bits mux (multiplexer) heeft de volgende eigenschap: Er zijn 2 ingangen en 1 uitgang. Er is ook een SELECT bit. Deze bit bepaald welk ingangssignaal op de uitgang komt.

SEL	OUT
0	A
1	B

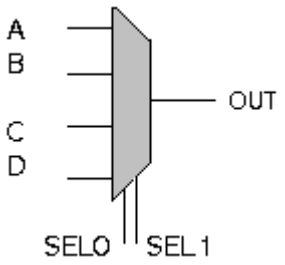


Figuur 5 - Een Multiplexer

Met een OR, NOT en twee AND poorten is deze schakeling te realiseren. Bedenk de oplossing en test het in Logisim.

Opdracht 3

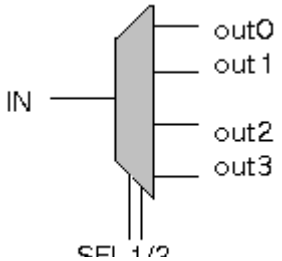
De 2-bits mux kan selecteren tussen 4 ingangen, maar heeft nog steeds 1 uitgang. Er zijn wel 2 SELECT bits nodig om dat te realiseren. Maak gebruik van 1-Bits multiplexers en extra logische poorten om de 2-bits multiplexer te realiseren.



Figuur 6 - Een 2 bits multiplexer

Opdracht 4

De demultiplexer (demux) is de tegenhanger van de mux. Het nevenstaande figuur laat zien hoe het werkt. Er is 1 ingangssignaal, en meerdere uitgangssignalen. Op deze manier is het bijvoorbeeld mogelijk om te selecteren tussen verschillende geheugenplekken om 'IN' in op te slaan. Aan elke uitgang is dan een geheugenplek gekoppeld. De demux wordt ook wel een 'Data verdeler' of 'decoder' genoemd.



Figuur 7 - Een 2 bits demux

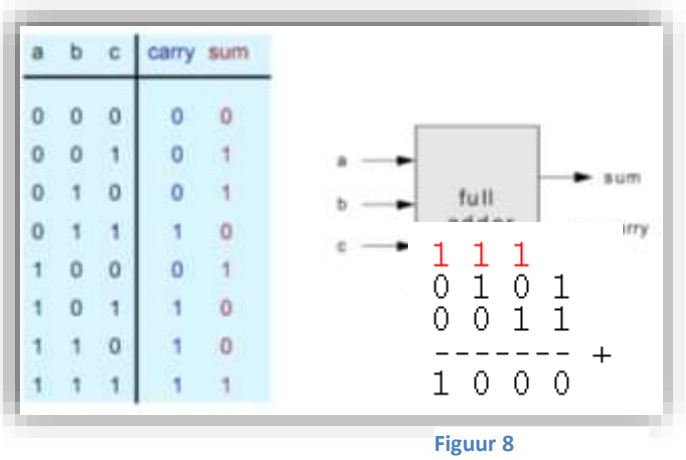
DEMUX – Ingang = A					
SEL0	SEL1	OUT3	OUT2	OUT1	OUT0
0	0	0	0	0	A
0	1	0	0	A	0
1	0	0	A	0	0
1	1	A	0	0	0

Maak en simuleer eerst een 1-bit demux en vervolgens een 2-bit demux.

Opdracht 5

a	b	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Een rekenmachine moet ook in staat zijn om twee 4-bits getallen bij elkaar op te tellen. In de processor van een computer gebeurt dit met zogenaamde 'Adders'. Nu zijn er twee soorten Adders. Als eerste is er de half-Adder. Deze is slechts in staat om twee 1-bit getallen bij elkaar op te tellen met als resultaat een 2-bit getal. De full adder is in staat om drie 1-bit getallen bij elkaar op te tellen. Hieronder zijn de tabellen en figuren weergeven. Gebruik SOP om de expressies van deze schakelingen te bepalen. Simuleer het vervolgens in Logisim.



Figuur 8

Opdracht 6

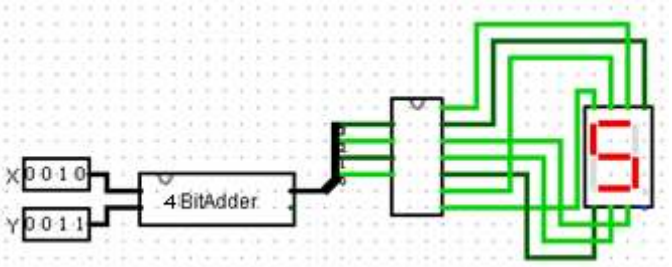
Ook de full-Adder alleen is niet genoeg om twee 4-bits getallen bij elkaar op te tellen. Hoe doen we dit dan? Hiervoor moeten we kijken naar een simpele binaire berekening.

De bovenstaande optelling werkt identiek aan optellen in het decimale stelsel. Bij decimale getallen, wanneer de som van twee getallen boven de 9 komt, ontstaat er een 'rest' getal. Deze rest gebruiken we vervolgens bij de volgende linker berekening. Ontstaat daar ook een rest? Dan schuift deze door naar de volgende berekening. Etc.

In het binaire stelsel is het rest getal (de Carry) 1 of 0. Is het een 1 dan heeft de volgende berekening een extra bit om mee op te tellen. Maximaal moeten dus 3 bits bij elkaar worden opgeteld. Met andere woorden, voor elke kolom van Figuur 8 is een full adder nodig!

Maak een Adder waarbij twee 4-bits getallen bij elkaar kunnen worden opgeteld. Noem de bits van getal 1 [X_0 t/m X_3] en de bits van getal 2 [Y_0 t/m Y_3]. De uitgang bestaat uit [Z_0 t/m Z_3] plus de Carry.

Koppel uiteindelijk je Adder aan de ingang van je display driver en test je kleine rekenmachine.

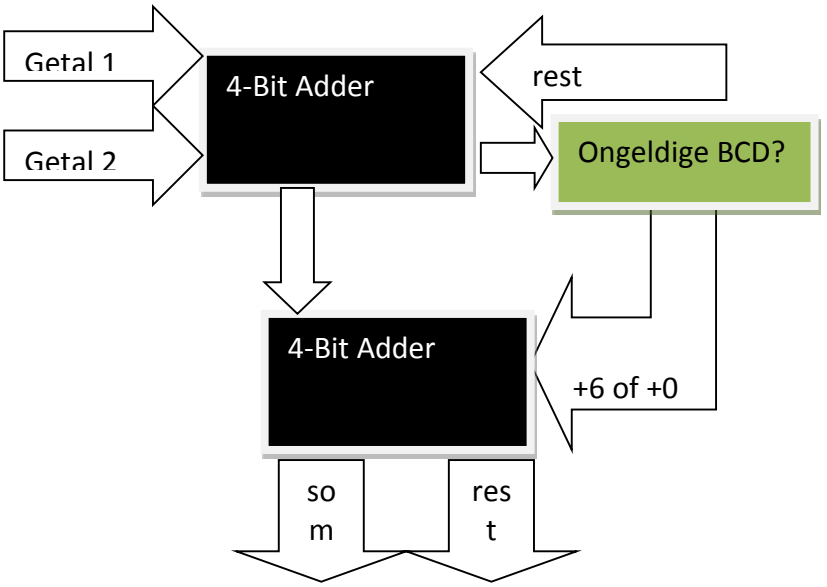


Wanneer de som boven de 9 komt zie je dat er een ‘E’ te voorschijn komt. Hoe gaan we dit oplossen met meerdere displays?

Opdracht 7 – BCD Adder

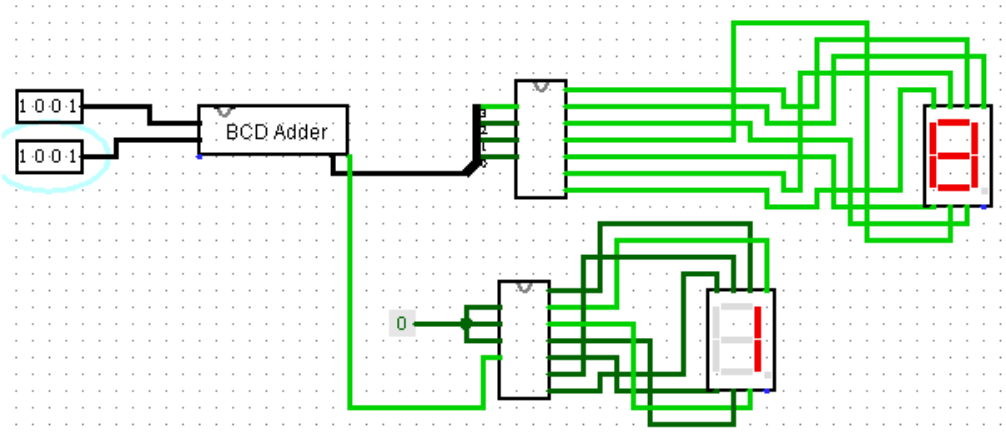
BCD getallen representeren de decimale cijfers 0-9. Een 4-bits BCD code wordt gebruikt voor deze getallen. Omdat de 4-bits code tot 16 mogelijkheden toestaat, worden de eerste 10 mogelijkheden als ‘juiste’ BCD code beschouwd. De laatste 6 mogelijkheden zijn onjuist en komt niet voor (onze ‘E’).

BCD nummers kunnen worden opgeteld met behulp van de BCD Adder. Een BCD optelling is gelijk aan normaal binair optellen met een uitzondering wanneer de som van twee BCD getallen boven de 9 komt, of een Carry oplevert. In dit geval wordt er een 6 bij de som opgeteld om het onjuiste BCD getal om te zetten naar een correct BCD getal. De rest die ontstaat door het optellen van 6 aan het onjuiste BCD getal, wordt doorgegeven aan het volgende BCD getal.



Een 4-bits BCD Adder ziet er dan uit als het bovenstaande schema. Twee getallen worden opgeteld. Is de som meer dan 9? Dan wordt de som met 6 opgehoogd. Er volgt dan uiteindelijke een nieuwe som en al dan niet een rest bit.

Maak eerst de waarheidstabel van het groene blok. Deze heeft 5 bits als ingang (de 4 bits van de som, 1 bit voor de Carry) en 3 bits voor de uitgang (maximaal binair 6).



Maak vervolgens de volledige BCD Adder en sluit hem aan op een display zoals hier weergeven. Normaliter gaat de Carry naar een volgende BCD Adder, en niet direct naar een display, dit is puur voor illustratie.

Week 4

Opdracht 1 – Optimalisatie

Tot nu toe hebben we ons weinig bezig gehouden met optimalisatie. Gezien onze basiselementen niet geoptimaliseerd zijn krijgen we problemen met performance en grootte bij de simulatie. Ook, indien we het in het echt zouden bouwen, wil je zo min mogelijk logische poorten (IC's) gebruiken.

Deze opdracht bestaat uit het optimaliseren van de schakelingen. We gaan gebruik maken van karnaugh. Gebruik karnaugh om de kleinste expressies te vinden en gebruik dit om in logisim de schakelingen kleiner te maken.

Dus optimaliseer indien mogelijk:

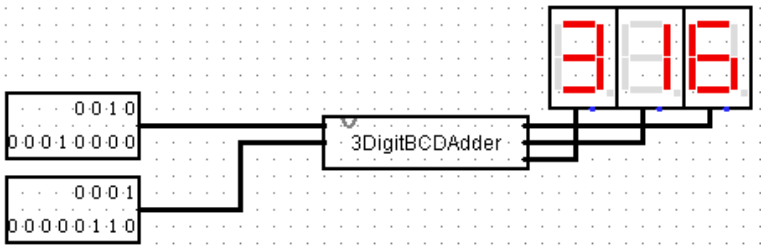
- De 7-Segment driver
- De Half en Full-Adder
- De BCD Controle (>9 ?)

Opdracht 2

Onze BCD Adder kan tot nu toe slecht getallen onder de 10 bij elkaar optellen. Wat als je 11+12 wilt uitrekenen? Het mooie van BCD getallen is dat elk cijfer een aparte binaire beschrijving heeft. De optelling zou dus het volgende zijn:

0001 0001 (11)
0001 0010 (12)
---------- +
0010 0011 (23)

Deze berekening maakt gebruik van twee BCD Adders. In feite is dit het zelfde als wat we met de full-Adder hebben gedaan. Voor elke kolom een aparte Adder.



Figuur 9 - 3-Digit BCD Adder

Bouw een 3-cijferige BCD Adder. Gebruik voor het overzicht en gemak niet onze eigen display driver, maar de al bestaande 'Hex Digit Display'. Deze heeft de driver al ingebouwd.

Week 5 – Sequentiële logica

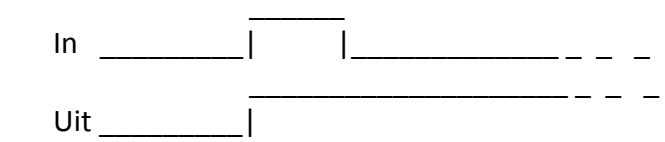
Opdracht 1

Bouw een Set/Reset flipflop met behulp van twee NOR gates in Logisim.

Opdracht 2

Een flipflop is in essentie een 1-bit geheugenelement. Afgezien van de SR flipflop hierboven, maken alle flipflops gebruik van een klok signaal. In Logisim is zo'n klok signaal te vinden onder 'Wiring'. De ingangs-waarde van de flip flop wordt onthouden bij elke op- of neergaande klok-flank.

Bouw een 'puls' detector met een D-flipflop (DFF). De duur van de puls is minstens gelijk aan de klok-periode. Wanneer een puls is opgetreden aan de ingang, moet de uitgang van de DFF hoog blijven. Dit is alleen te bereiken met terugkoppeling (feedback).



(Er kan een verschuiving optreden afhankelijk van de positie van het kloksignaal.)

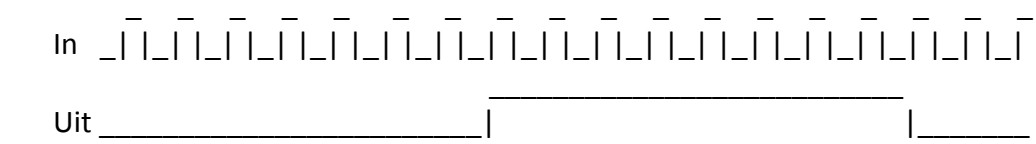
Opdracht 3

Bij de vorige opdracht konden we de DFF enkel '1' laten onthouden bij de eerstvolgende op- of neergaande klok-flank. Pas de schakeling van opdracht 2 aan zodat we ook een '0' kunnen opslaan. Deze nieuwe schakeling heet een 'register'. Een register bindt nog een voorwaarde aan het onthouden van de nieuwe waarde; Naast de klokflank moet ook de ingang 'Load' gelijk zijn aan 1. Dus pas wanneer load 1 is en de gewenste klok-flank optreedt , DAN wordt de ingang opgeslagen.

Je kan voor de oplossing gebruik maken van een 1-bits multiplexer.

Opdracht 4

Soms is er de wens om bepaalde logica bijvoorbeeld twee keer zo langzaam te laten lopen dan het klok-signaal toe laat. Een oplossing is om een klok-deler te maken. Met behulp van een JK-FlipFlop is zo'n klok-deling te bewerkstelligen. Maak een 8-voudige klokdeler.

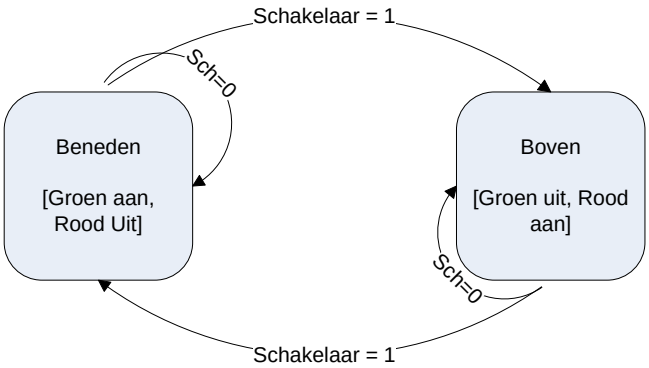


Week 6/7

Er zijn verschillende manieren om een sequentieel systeem te beschrijven en te ontwerpen. Eén van die manieren is met behulp van een Finite-State machine (Eindig-aantal-toestanden-systeem). Een FSM komt in twee vormen, 'Mealy' en 'Moore'. Gezien er een eindig aantal toestanden zijn kan de toestand van zo'n systeem worden opgeslagen in flipflops. Heeft het systeem 10 toestanden, dan zijn er vier flopflops nodig om elke unieke toestand te kunnen opslaan. Een systeem met maar 1 flipflop kan maar twee toestanden onderscheiden.

Opdracht 1

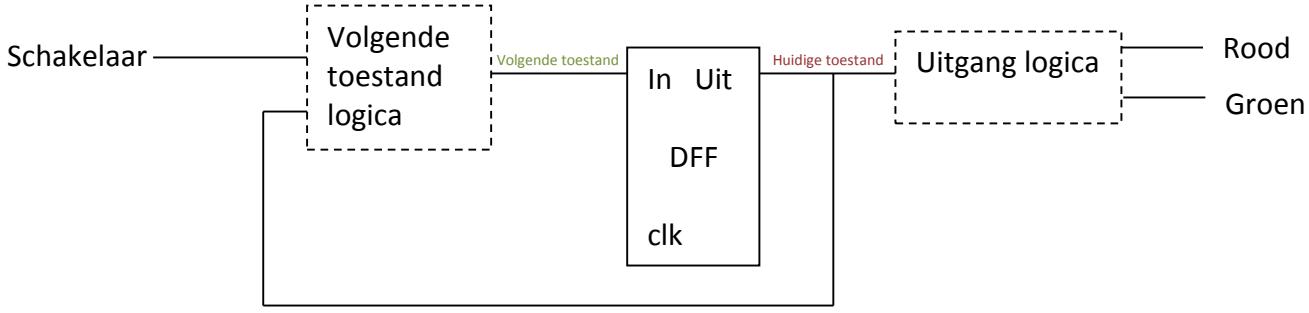
We maken een schakeling voor een lift. Er zijn twee verdiepingen waar de lift heen kan. Op de begane grond brandt er alleen een groen lampje, op de 1e verdieping brandt alleen een rood lampje. De lift heeft een schakelaar waarmee je kiest tussen 'naar boven' of 'naar beneden'.



Dit systeem bestaat uit twee toestanden (boven en beneden) en is een Moore systeem. Maak het STT (State Transition Table) en bepaal uiteindelijk de missende logica in de onderstaande schakeling. Simuleer je resultaat en bevestig de werking van je lift schakeling.

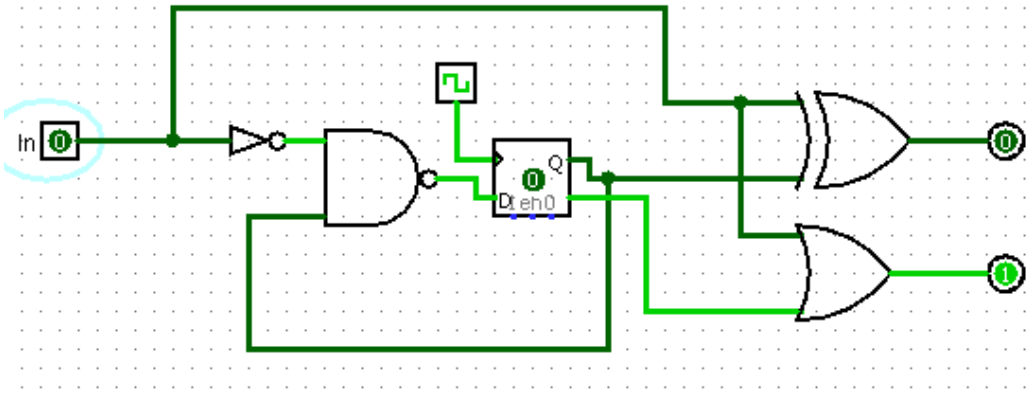
Huidige Toestand	Schakelaar (In)	Volgende Toestand	Groen (Out)	Rood (Out)

Bepaal de logische expressies voor "Volgende toestand" en voor "Rood" en "Groen"



Opdracht 2

Teken het STD van de onderstaande schakeling.
Tip: Is het mealy of moore? Simuleer de schakeling en bekijk het gedrag.



Opdracht 3

We maken een mealy machine van een cola automaat. Gegeven is de onderstaande STT, bepaal hiermee de schakeling. De automaat geeft geen geld terug en accepteert enkel euro's en 50 cent munten. De uitgang is gekoppeld aan een mechanisme wat een blikje loslaat wanneer de uitgang even 1 is geweest.

Gezien het een mealy systeem betreft, zal het asynchrone gedrag van de input problemen veroorzaken. Wanneer de input synchroon loopt met de klok gebeurt dit niet. Bedenk hiervoor een oplossing in de vorm van flip-flops.

Huidige Toestand	Euro	50 Cent	Volgende Toestand	Uitgang
0 Euro	0	0	0 Euro	0
	0	1	50 Cent	0
	1	0	1 Euro	0
	1	1	X	X
50 Cent	0	0	50 Cent	0
	0	1	1 Euro	0
	1	0	0 Euro	1
	1	1	X	X
1 Euro	0	0	1 euro	0
	0	1	0 Euro	1
	1	0	0 Euro	1
	1	1	X	X

De X'en staan voor 'Don't Cares'. Het is immers vrijwel onmogelijk om tegelijkertijd een euro en 50 cent in een machine te gooien.

Tip: Het systeem heeft twee flipflops nodig. (Waarom?). Dit betekent dat je een logische expressie moet bepalen voor de ingang van elke flipflop.

De uitgang van de schakeling heeft een logische expressie op basis van de ingang van de schakeling en huidige-toestand (uitgang van de flipflops).

Opdracht 4

Maak een schakeling welke de functie heeft om een correct wachtwoord te accepteren. De schakeling heeft als input een keypad (0-9) en als output een 1bits logisch signaal. Wanneer het wachtwoord correct wordt ingevoerd dient de uitgang 1 te worden.

Het wachtwoord moet gelijk zijn aan ‘3914’ .

Tip: Niet de volle 4-bits van het keypad zijn nodig.
Tip: Gezien de tabel groot wordt, is het verstandig het door Logisim te laten omzetten.