Design van synchrone sequentiele - schakelingen – Compact and Clean

[2020, Marius Versteegen, v0.01]

Over dit document

Bij dit vak behandelen we twee methoden waarmee je synchrone sequentiele schakelingen kunt designen: “Quick and Dirty” en “Compact and Clean”.

In dit document wordt een voorbeeld gegeven van het design van een synchrone sequentiele schakeling volgens de “Compact and Clean” methode. Onder de blauwe headers wordt nog wat theorie herhaald. Vervolgens wordt de theorie toegepast op een voorbeeld met groene headers.

“Quick and dirty” versus “Compact en clean”

Het verschil tussen deze methodes ligt in het kiezen van het aantal flipflops en het type flipflops dat gebruikt wordt.

Quick and Dirty

Bij de “Quick and dirty” methode gebruik je per toestand een JK-flipflop, en bepaal je per toestand apart de condities waaronder die geset of gereset moeten worden.

Voordelen van die methode zijn:

- Eenvoudig, overzichtelijk ontwerp.
- Het ontwerp is later zonder veel werk uitbreidbaar met extra toestanden.
- Soms minder combinatoriek nodig dan bij “Compact en clean”.

Compact en Clean

Bij de “Compact and Clean” methode gebruik je een aantal D-flipflops dat niet hoger is dan $^{2}\log$(aantal toestanden). Dus bij 3 of 4 toestanden gebruik je 2 D-flipflops, bij 5 tot en met 8 toestanden 3 D-flipflops, etcetera.

Voordelen van die methode zijn:

- Een minimaal aantal flipflops
- Meestal ook een minimale hoeveelheid componenten.
- Robuuster

Dit document behandelt de laatstgenoemde methode.

Toestandsdiagram

Het design van een synchrone sequentiele schakeling begint met het tekenen van een toestandsdiagram. De toestanden worden gerepresenteerd door circels. De toestandsovervangen door pijlen. De toestandsovergangen kunnen afhankelijk zijn van ingangsvariabelen.

Combinatoriek en registers

We veronderstellen in dit document dat we synchrone sequentiele schakelingen opbouwen uit geclockte D-flipflops en combinatoriek.

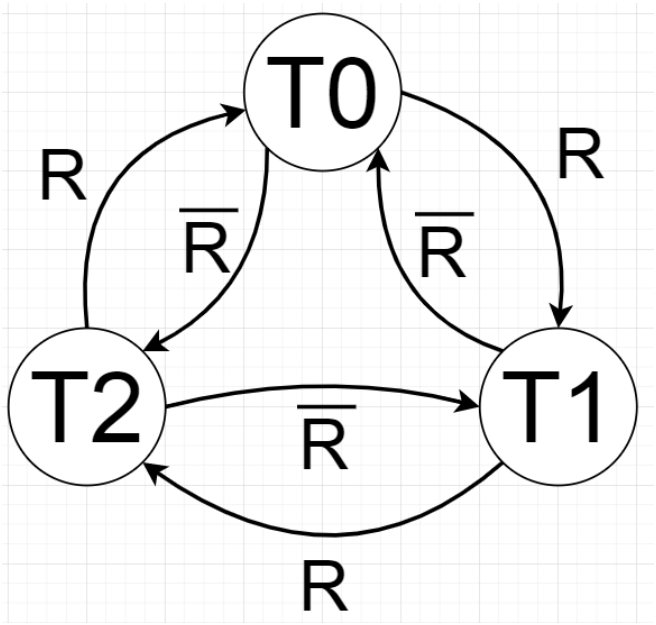
- De uitgangen Q van de D-flipflops representeren de huidige toestand.
- De clock-ingangen van alle D-flipflops zijn op een clock-lijn aangesloten.
- Op de D-ingang van elke D-flipflop staat de waarde klaar van de “volgende gewenste Q”. Die waarde word bij de volgende opgaande flank van de clock-ingang overgenomen door uitgang Q. De waarde op de D-ingang noem ik daarom “Q_{next}”.
- De waarde op de uitgang van flipflop 0 noemen we dus Q0, terwijl we de waarde op zijn (D-) ingang $Q0_{next}$ noemen.
- Er kunnen ook (externe-) input variabelen zijn. Bijvoorbeeld gekoppeld aan een button.
- Die (D-) ingangen worden bepaald door combinatoriek die afhangt van de huidige toestand (de uitgangen Q) en de input variabelen.

Voorbeeld:
Een circulair looplicht van 3 LEDs

- Drie leds zijn opgesteld in een 3-hoekje/”circeltje”. Er zijn drie toestanden: L0, L1 en L2. In toestand T0 branden alleen Leds 0 en 1, in toestand T1 branden alleen Leds 1 en 2 en in toestand T2 branden alleen Leds 2 en 0. Er branden dus steeds twee leds, terwijl de derde led uit is.
- De drie leds zijn aangesloten op binaire uitgangswaarden Y0, Y1 en Y2. Als Y0==1 brandt Led0, als Y1==1 brandt Led1 en als Y2==1 brandt Led2. Er mag maar 1 led tegelijk aan zijn.
- Er is een klok-signaal. Op de opgaande flank gaat de huidig brandende led uit en de volgende aan.
- Er is een drukknopje die de binaire ingangswaarde R levert.
  Als R==1 dan is de volgende led die aangaat “rechtsom” de volgende.
  Zo niet, dan is de volgende led die aangaat “linksom” de volgende.

Toestandsdiagram

We maken het toestandsdiagram door de drie toestanden als circels op papier te zetten en de toestandsovergangen ertussen met pijlen weer te geven. Naast de pijlen zetten we de condities van de toestandsovergangen. Die condities hangen af van de input variabelen. In dit geval hebben we maar 1 input variabele: de waarde R die het “rechtsom-drukknopje” produceert.

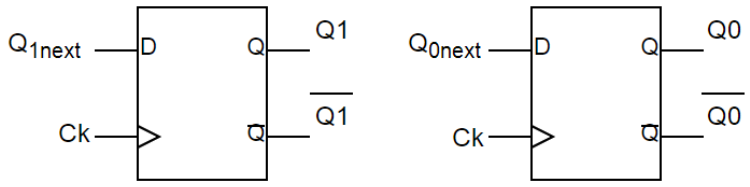


Compact and Clean

Aantal toestanden

Uit het toestandsdiagram blijkt dat we 3 toestanden willen kunnen modelleren. We kunnen T0 modelleren met het binaire getal 0x00, T1 met 0x01 en T2 met bijvoorbeeld 0x10. Kortom: we hebben aan twee bits (twee D-flipflops) voldoende om de 3 toestanden te representeren.

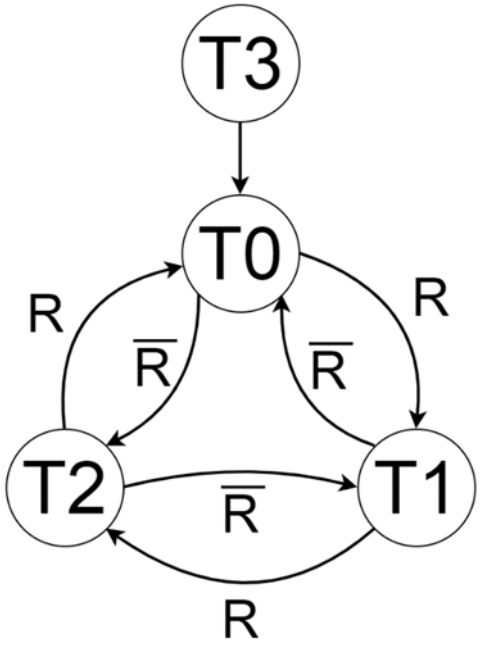
We gebruiken twee flipflops. Met uitgang Q0 van flipflop 0 representeren we het laagste bit en met uitgang Q1 van flipflop 1 representeren we het hoogste bit (daarom teken is flipflop 0 rechts getekend):



NB: bij gebruik van 2 bits is er ook nog een vierde toestand: 0x11.
Bij het ontwerp moeten we rekening houden met die toestand, en besluiten wat we willen doen, mocht het systeem (bijvoorbeeld tijdens het opstarten) in die toestand terecht komen.

Om die reden voegen we die vierde toestand, T3 ook toe aan ons toestandsdiagram. Ik kies er voor dat mocht het systeem in T3 geraken, dat het dan bij de volgende opgaande flank van de klok te allen tijde naar de valide toestand T0 gaat.

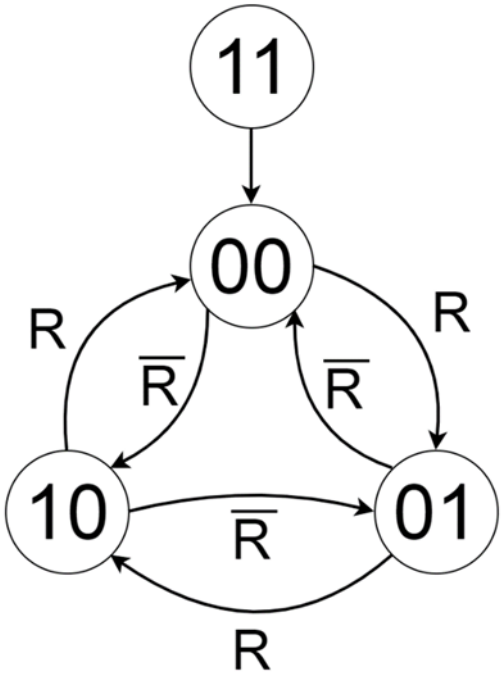
Uitgebreid Toestandsdiagram



Vertaald Toestandsdiagram

Zoals eerder is aangegeven correspondeert toestand T0 met Q_1Q_0=00, toestand T1 met Q_1Q_0=01, toestand T2 met Q_1Q_0=10, en toestand T3 met Q_1Q_0=11

Om dat te verduidelijken zijn in onderstaande toestandsdiagram de toestandsnamen vervangen door de bijbehorende waarden voor het tuple flipflop outputs Q_1Q_0.



Gecombineerde Karnaugh diagram voor de toestandstransities

Gegeven de huidige toestand Q_1Q_0 en de ingangswaarde R willen we bepalen wat de volgende gewenste toestand $Q_{1next}Q_{0next}$ is.

Daartoe maken we een Karnaugh diagram met Q_1, Q_0 en R als inputs, en de twee-bits waarde $Q_{1next}Q_{0next}$ als output.

Om de visualisatie logisch te houden, zetten we de "huidige toestandsbits" Q_1 en Q_0 aan de linkerzijde, en de overige inputs aan de bovenzijde.

$Q_{1next}Q_{0next}$	**R**	
Q_1 |		
| |		
Q0 |		

Elke cel van het toestandsdiagram representeert een mogelijke set input waarden. We moeten nu besluiten welke nieuwe gewenste output waarden $Q_{1next}Q_{0next}$ we willen in elk van die gevallen.

Laten we (willekeurig) met de linksbovenste cel beginnen. De cellen in de bovenste rij horen bij "huidige toestand" Q_1Q_0=10. Verder hoort bij de linksbovenste cel de ingangswaarde R (de rechtsom knop is ingedrukt)

In het toestandsdiagram zien we linksonder de toestand 10. We zien een uitgaande pijl met R die naar de volgende toestand $Q_{1next}Q_{0next}$=00 wijst. Om aan te geven dat dat de gewenste volgende toestand in die situatie is, vullen we 00 in voor die cel in het karnaugh diagram:

$Q_{1next}Q_{0next}$	**R**	
Q_1 |	**00**	
| |		
Q0 |		

We zien dat er vanuit toestand 10 ook een uitgaande pijl met R is. Die wijst naar de volgende toestand $Q_{1next}Q_{0next}$=01. In de meest rechtsbovenste cel noteren we dus 01:

$Q_{1next}Q_{0next}$	**R**	
Q_1 |	00	**01**
| |		
Q0 |		

Door voor alle cellen op deze manier de gewenste toestanden in te vullen, vinden we:

$Q_{1next}Q_{0next}$	R	
Q_1 |	00	01
| |	00	00
Q_0 |	10	00
	01	10

Opsplitsen in een Karnaugh-diagram per bit

Het gecombineerde Karnaugh-diagram was een handige en overzichtelijke manier om de gewenste nieuwe toestanden in te vullen. Voor het bepalen van de bijbehorende formules is het weer handiger om de diagram op te splitsen per uitgang:

$Q_{1next}Q_{0next}$	R	
Q_1 |	00	01
| |	00	00
Q_0 |	10	00
	01	10

Q_{1next}	R	
Q_1 |	0	0
| |	0	0
Q_0 |	1	0
	0	1

Q_{0next}	R	
Q_1 |	0	1
| |	0	0
Q_0 |	0	0
	1	0

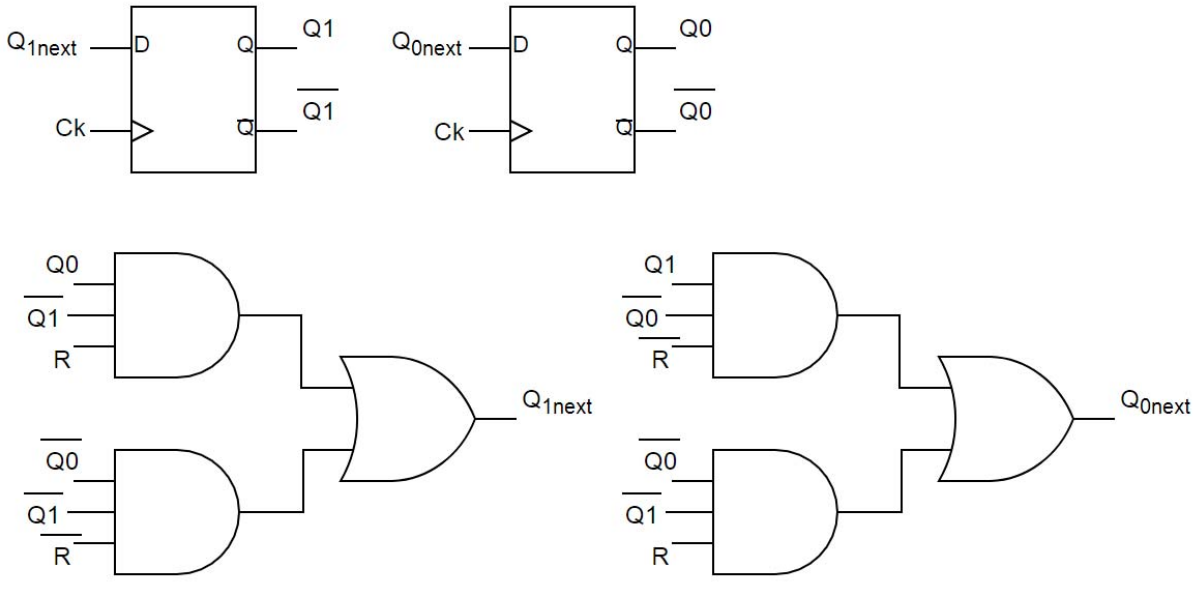
->

Uit de Karnaugh-diagrammen voor Q_{1next} en Q_{0next} volgen deze formules (underscore betekent “niet”, · betekent AND) :

- $Q_{1next}=(Q_0 \cdot \underline{Q_1} \cdot R)+(\underline{Q_0} \cdot \underline{Q_1} \cdot \underline{R})$
- $Q_{0next}=(Q_1 \cdot \underline{Q_0} \cdot \underline{R})+(\underline{Q_0} \cdot \underline{Q_1} \cdot R)$

Resulterende schema

Zetten we deze formules om in logische poorten, dan resulteert het volgende schema voor onze toestandsmachine:



NB: Knooppunten met dezelfde naam worden verondersteld met elkaar verbonden te zijn.
Zo is de OR uitgang Q_{1next} verbonden met de D ingang van D-flipflop 1.
Met bijvoorbeeld Logisim (een gratis simulator van logische circuits) is een soortgelijke notatiewijze mogelijk (de verbindingslabels heten daar “tunnels”).

Combinatoriek voor de outputs

We hadden bepaald dat in T0 moet gelden dat led-aansturingen Y_0 en Y_1 hoog moeten zijn, dat in T1 zowel Y_1 als Y_2 hoog moeten zijn en in T2 zowel Y_2 als Y_0. Voor de potentiele initiele toestand T3 maakt het niet uit. Daarvoor kunnen we dus don’t cares invullen.

Laten we eerst het Karnaugh diagram van Y0 invullen:

Y_0	
Q_1 |	
| |	
Q_0 |	

In de bovenste regel geldt: Q_1Q_0=10, oftewel toestand T2. Daarvoor geldt dat Y_0 hoog moet zijn, dus vullen we daar een een in:

Y_0	
Q_1 |	**1**
| |	
Q_0 |	

Op dezelfde manier bekijken we voor elke andere cel/toestand wat de bijbehorende led-aansturing moet worden. Tesamen levert dat de volgende drie Karnaugh-Diagrammen op:

Y_0	
Q_1 |	1
| |	D
Q_0 |	0
	1

Y_1	
Q_1 |	0
| |	D
Q_0 |	1
	1

Y_2	
Q_1 |	1
| |	D
Q_0 |	1
	0

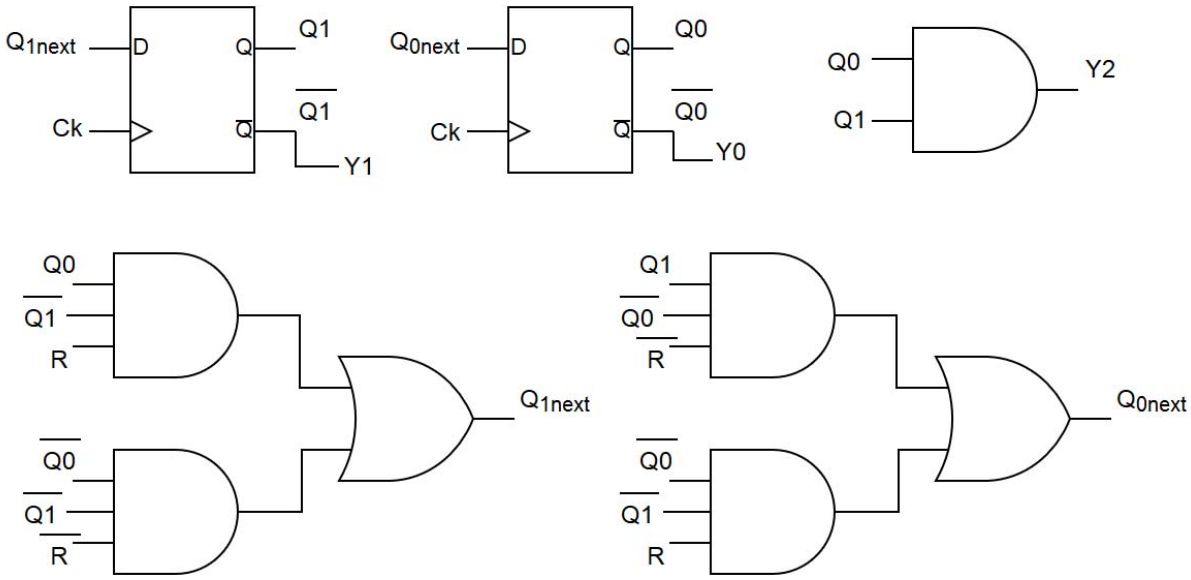
In het geval van Y0 kunnen we de bovenste en onderste 1 samennemen in 1 groep (de “karnaugh wereld is rond”). We kunnen daar voor de don’t care (d) het beste een 0 kiezen, zodat er geen extra groep (enen) nodig is. Voor geval Y1 kunnen we de bestande enen ook makkelijk bij mekaar in een groep stoppen. Ook daar kiezen we dus voor de don’t care een 0.
In geval Y2 grenzen de enen niet aan elkaar. Als we d=0 zouden kiezen, zou dat twee groepen ter

grootte van een enkele 1 opleveren. Als we daarentegen d=1 kiezen, kunnen we twee overlappende groepen ter grootte van 2 enen maken. Grotere groepen leiden tot kleinere formules, dus dat is beter.

De vergelijkingen voor de Led-aanstuursignalen worden dus:

- $Y_0 = \underline{Q_0}$
- $Y_1 = \underline{Q_1}$
- $Y_2 = Q_0 \cdot Q_1$

Het totale schema met deze poorten in ogenschouw wordt dus:



Nabeschouwing

- De getoonde methode vereist het opstellen van karnaugh diagrammen.
- Het levert een systeem op met een minimaal aantal flipflops.
- Later uitbreiden met een extra toestand kan, maar daarvoor moet het hele ontwerp opnieuw.
- Het systeem is robuust: Als nu initieel of door een of andere spanningspiek het systeem in een onbedoelde toestand (T3) komt, dan kan het systeem zich daarvan bij de eerstvolgende opgaande klokflank herstellen.

Bij de andere methode (“Quick and Dirty”) zijn deze voor- en nadelen omgekeerd.

Simuleer!

Simuleer je ontwerp (bijvoorbeeld met Logisim) voordat je het gaat bouwen. Je kunt daarmee design-fouten eenvoudig ontdekken en debuggen, en bespaart daarmee jezelf dikwijls veel tijd.