Design van synchrone sequentiele schakelingen – Quick and Dirty

[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 “Quick and Dirty” 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 $^{2}\log(\text{aantal toestanden})$. 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

Zoals gezegd, dit document behandelt de eerstgenoemde 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 JK-flipflops en combinatoriek.

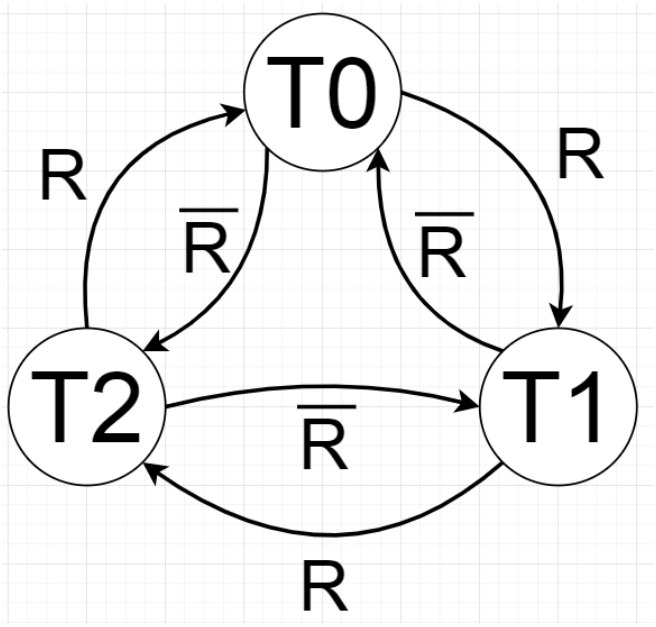
- De uitgangen Q van de JK-flipflops representeren de huidige toestand.
- De clock-ingangen van alle JK-flipflops zijn op een clock-lijn aangesloten.
- Op de J-ingang (een soort “set ingang”) van een JK-flipflops wordt een 1 gezet als de gewenste uitgang Q bij de volgende opgaande flank van de clock ingang 1 moet worden.
- Op de K-ingang (een soort “reset ingang”) van een JK-flipflops wordt een 1 gezet als de gewenste uitgang Q bij de volgende opgaande flank van de clock ingang 0 moet worden.
- Er kunnen ook (externe-) input variabelen zijn. Bijvoorbeeld gekoppeld aan een button.
- Die J en K-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

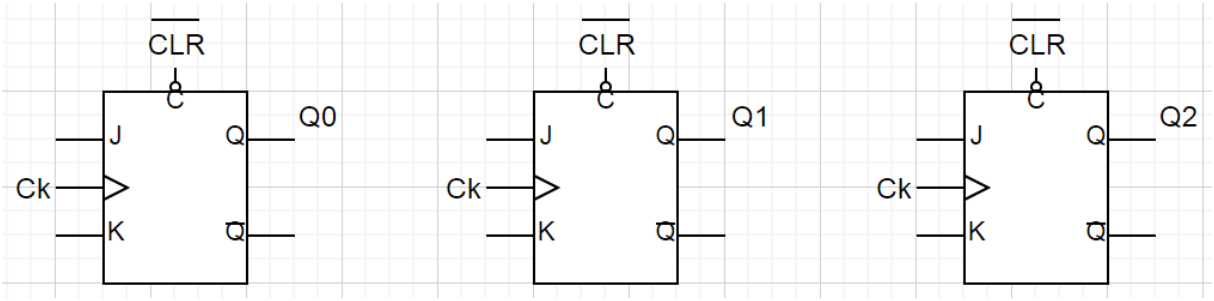
We maken het toestandsdiagram door de drie toestanden als circels op papier te zetten en de toestandsovergangen ertussen met pijlen weer te geven. Naast de pijlen zetten we de condities van de toestandsovergangen. Die condities hangen af van de input variabelen. In dit geval hebben we maar 1 input variabele: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



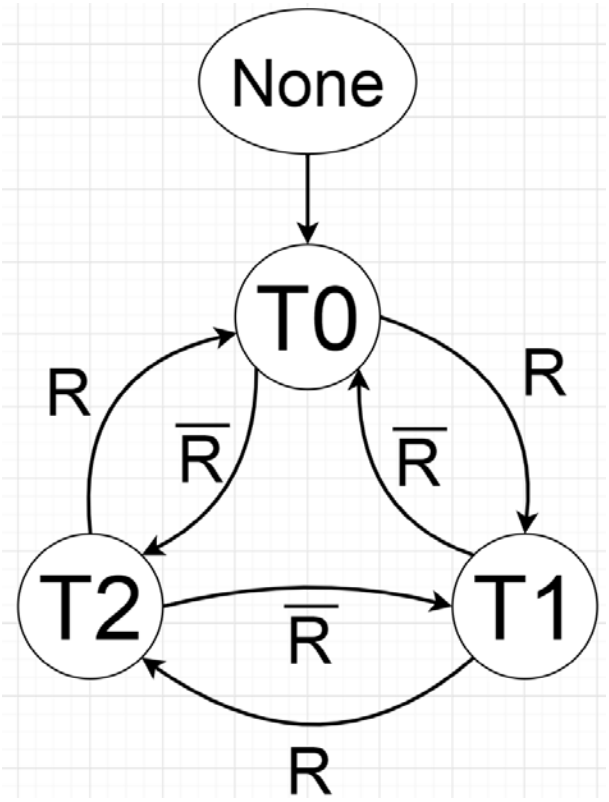
Methode 1: Quick en Dirty

Voor de Quick en Dirty methode gebruiken we per toestand 1 D-flipflop: in toestand T0 is alleen Q0 hoog, en zijn de overige flipflop-uitgangen laag. Hetzelfde geldt voor de overige toestanden.

Bij de quick en dirty methode is het belangrijk dat het systeem begint in een bekende toestand. Een handige manier daarvoor is om een D-flipflop met Clear input te gebruiken. Zo kunnen aan het begin alle flipflops eenvoudig gereset worden (met bijvoorbeeld een CLR knopje) in een toestand waarbij alle uitgangen (Q0, Q1 en Q2) nul zijn. Die toestand noem ik de “None” toestand.



Het toestandsdiagram, aangevuld met deze “None” toestand wordt dus:



Ik heb gekozen voor een transitie vanuit de None toestand naar toestand T0. Dat betekent dat vanuit de bekende None opstart-toestand bij de eerstvolgende opgaande flank van Ck de bekende valide toestand T0 wordt bereikt.

Vinden van de "Quick en dirty"-toestandstransities

Elk van de toestanden T0, T1 en T2 wordt gerepresenteerd door respectievelijk de flipflop toestanden Q0, Q1 en Q2.

Voor elk van hen moeten 2 vragen beantwoord worden:

1. Onder welke condities moet de betreffende flipflop bij de volgende klokflank 1 worden? Anders geformuleerd: onder welke condities moet zijn J-ingang 1 worden?

2. Onder welke condities moet de betreffende flipflop bij de volgende klokflank 0 worden? Anders geformuleerd: onder welke condities moet zijn K-ingang 1 worden?

In het toestandsdiagram zien we dat **T0 bereikt** wordt en dus Q0 1 moet worden door **alle pijlen naar T0** langs te lopen. Daaruit volgt: Q0 moet 1 worden / J0 moet 1 zijn als:

1. T0, T1 en T2 geen van alle actief is - de None toestand: $\underline{Q0} \cdot \underline{Q1} \cdot \underline{Q2}$ (underscore betekent "niet", · betekent AND)
2. T2 actief is en R hoog
3. T1 actief is en R laag

Hiermee is de eerste van bovenstaande vragen beantwoord voor flipflop 0. In formulevorm levert dit samen:

- **J0 = ($\underline{Q0}$ · $\underline{Q1}$ · $\underline{Q2}$) + (Q2 · R) + (Q1 · $\underline{R}$)**

Voor het beantwoorden van de tweede vraag zien we dat **T0 wordt verlaten** door te kijken naar de condities langs de **pijlen die uit T0 vandaan** lopen. Q0 moet in die gevallen gaan veranderen naar 0, dus K0 moet dan 1 worden. K0 moet dus 1 worden als:

1. (T0 · R)
2. (T0 · $\underline{R}$)

Dus K0 = (T0 · R) **+** (T0 · $\underline{R}$), hetgeen vereenvoudigd kan worden tot:

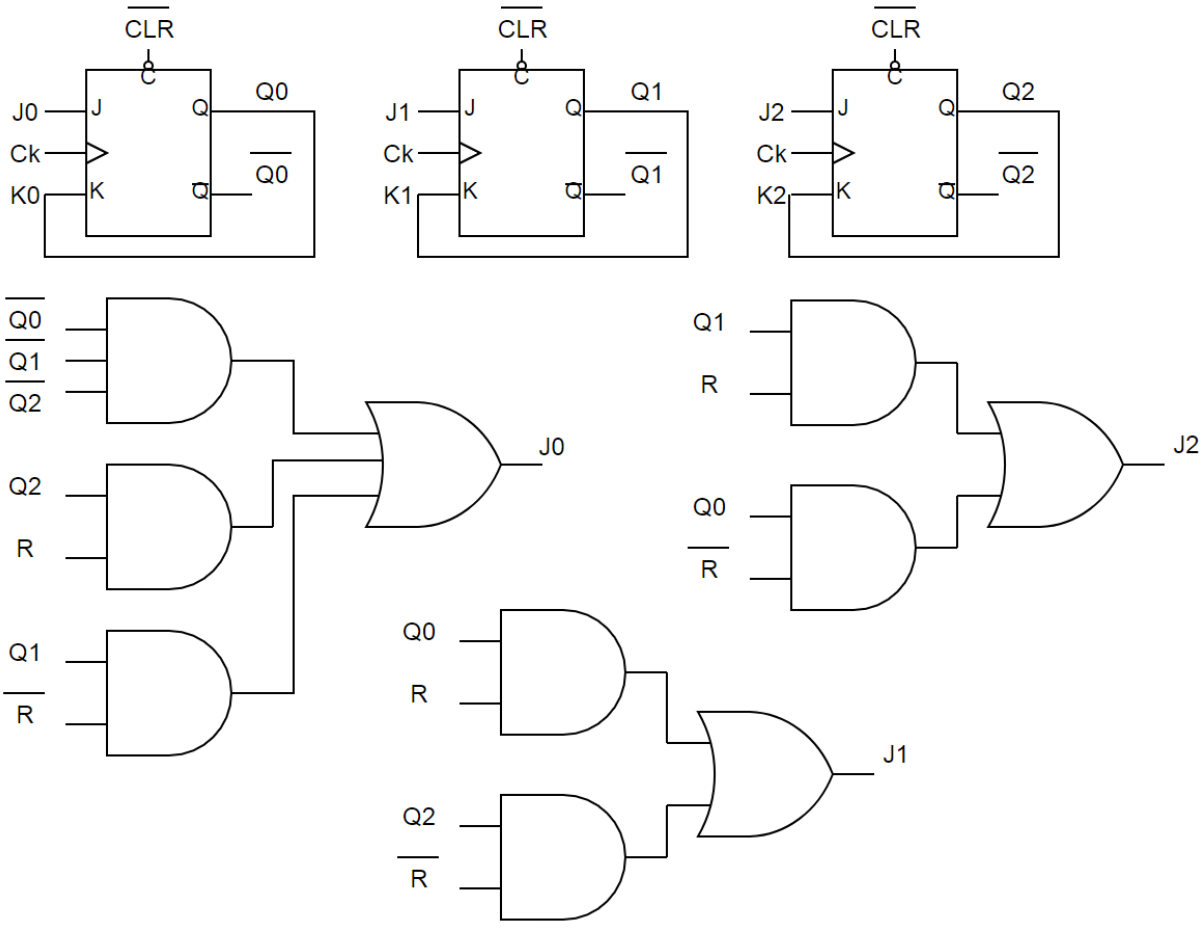
- **K0=Q0**

Hetzelfde kun je doen voor de toestanden T1 en T2. Je vindt dan:

- **J1 = (Q0 · R) + (Q2 · $\underline{R}$)**
- **K1=Q1**
- **J2 = (Q1 · R) + (Q0 · $\underline{R}$)**
- **K2=Q2**

Resulterende schema

We hebben hierboven dus de combinatoriek gevonden die zorgt dat de toestandsmachine kan werken. We kunnen het rechttoe-rechtaan met logische poorten implementeren:



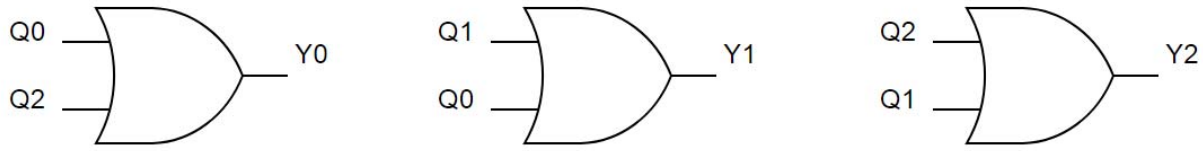
NB: Knooppunten met dezelfde naam worden verondersteld met elkaar verbonden te zijn.
Zo is de triple-OR uitgang J0 verbonden met de J0 ingang van JK-flipflop 0.
Met bijvoorbeeld Logisim (een gratis simulator van logische circuits) is een soortgelijke notatiewijze mogelijk (de verbindingslabels heten daar “tunnels”).

Combinatoriek voor de outputs

We hadden bepaald dat in T0 moet gelden dat led-aansturingen Y0 en Y1 hoog moeten zijn, dat in T1 zowel Y1 als Y2 hoog moeten zijn en in T2 zowel Y2 als Y0.
Anders gezegd: Y0 is hoog als Q0 of Q2 hoog is, Y1 is hoog als Q1 of Q0 hoog is en Y2 is hoog als Q2 of Q1 hoog is.
Deze functionaliteit is eenvoudig toe te voegen met de volgende formules:

- **Y0 =Q0 + Q2**
- **Y1 =Q1 + Q0**
- **Y2 =Q2 + Q1**

Ook hiervoor kunnen eenvoudig logische poorten worden toegevoegd:



Nabeschouwing

- De getoonde methode is eenvoudig en rechttoe-rechtaan.
- Het levert een systeem op met 3 flipflops / toestandsvariabelen, terwijl de 3 toestanden van het looplicht systeem ook met 2 flipflops / toestandsvariabelen gerealiseerd had kunnen worden.
- Het ontwerp is later zonder veel werk uitbreidbaar met extra toestanden. Dat komt, doordat de vergelijkingen per toestand onafhankelijk worden opgeschreven.
- Het systeem is niet zo robuust: Bij de initiele reset wordt het systeem in een valide toestand gedwongen. Van daar uit worden de nieuwe valide toestanden bepaald. Stel nu dat door een of andere spanningspiek of zo het systeem later in een toestand komt waarbij bijvoorbeeld Q0 en Q1 gelijktijdig hoog worden (een invalide toestand), dan kan het systeem zich daar niet meer van herstellen.

Bij de andere methode (“Compact and Clean”) 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.