# EE1410: Digitale Systemen

## BSc. EE, 1e jaar, 2012-2013, 4e college

Arjan van Genderen, Stephan Wong, Computer Engineering

21-2-2013

# Mededelingen

- Volgende week tijdens EPO-2: 2 middagen training  Digitale Systemen met VHDL en breadboard oefeningen: combinatorische schakelingen en sequentiele schakelingen
- Het afronden van deze oefeningen is verplicht binnen EPO-2 (resultaten worden afgetekend)
- Degenen die niet aan EPO-2 meedoen kunnen ook aanschuiven op ma. t/m do. middag (1e verd. Drebbelweg)
- Handleiding binnenkort op Blackboard bij EPO-2
- Bereid je goed voor !

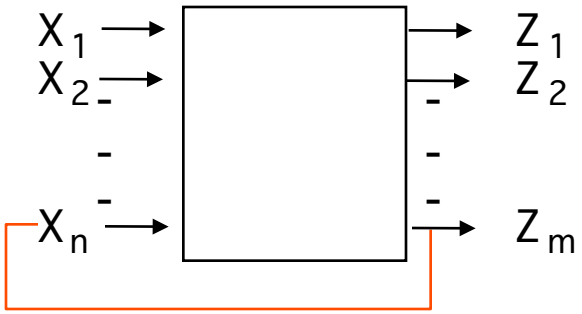
# Hoorcollege 4

- Sequentiële netwerken
  - Terugkoppeling
  - Latch
  - Flipflop
- Finite State Machines
  - State Machine Concept
  - Finite State Diagram
  - Moore and Mealy Machines
- Case Studies: Snoepautomaat en Verkeerslichten

Corresponderende stof in boek "Digital Logic":
7 – 7.4, 8 – 8.3, 8.5.1-2

# Sequentiële Netwerken

- **Circuits met terugkoppeling (Z = f(X, Z))**
- **⇒ outputs = f(inputs èn verleden)**
- **Gelimiteerd aantal stabiele oplossingen Z (toestanden of states genoemd)**
- **⇒ circuit heeft/is een geheugen**

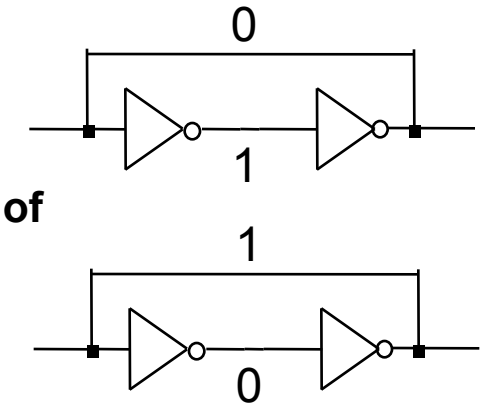



***Synchrone*** **systemen: toestandsovergang bepaald door één centraal *clock* signaal (toestandsveranderingen lopen synchroon met clock)**

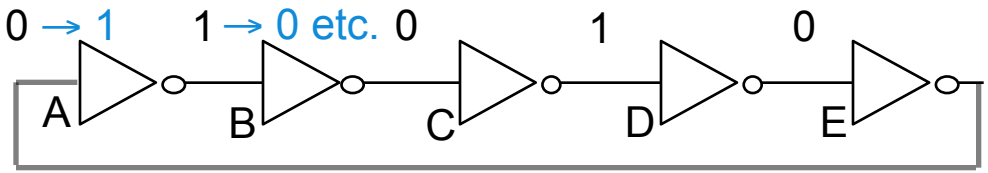
***Asynchrone*** **systemen: toestandsovergang afhankelijk van meerdere signalen (lastig voorspelbaar gedrag, niet behandeld in EE1410)**

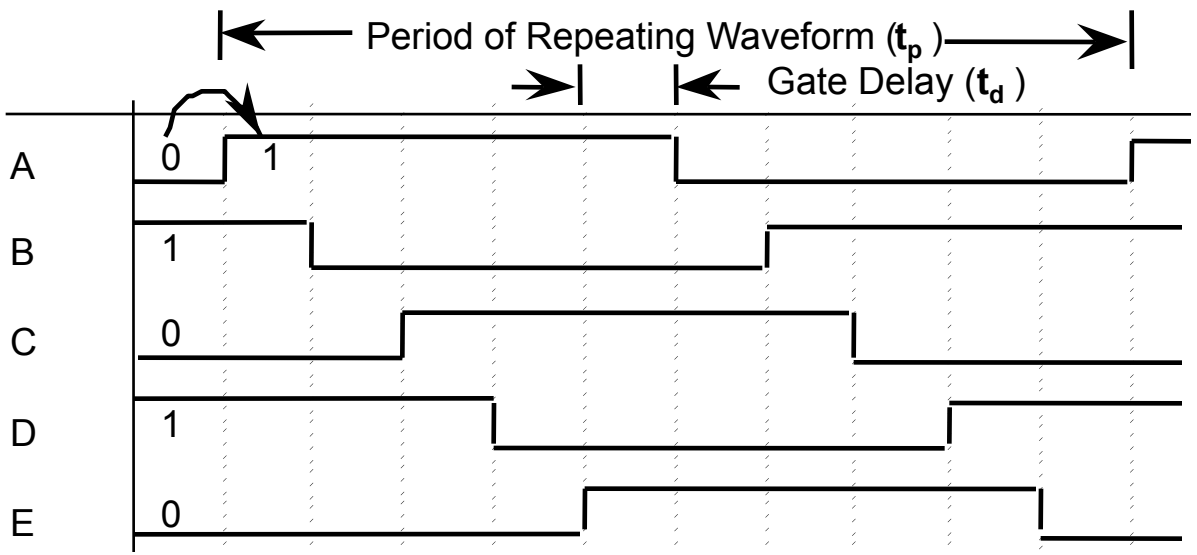
# Inverter-ketens

- **Statische geheugencel (even # inv.):** →




- **Ring Oscillator (*on*even # inv.):**



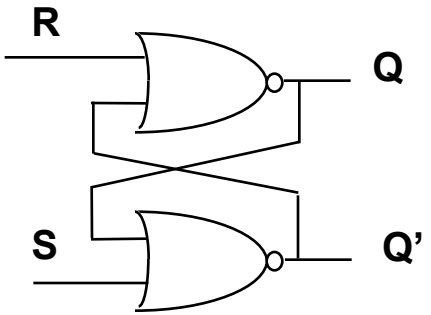




**$t_p = 2n\ t_d$**

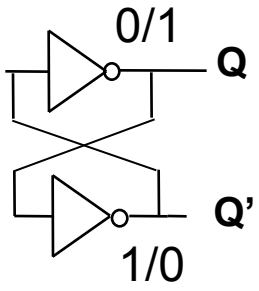
**n = # inv.**

# R-S Latch

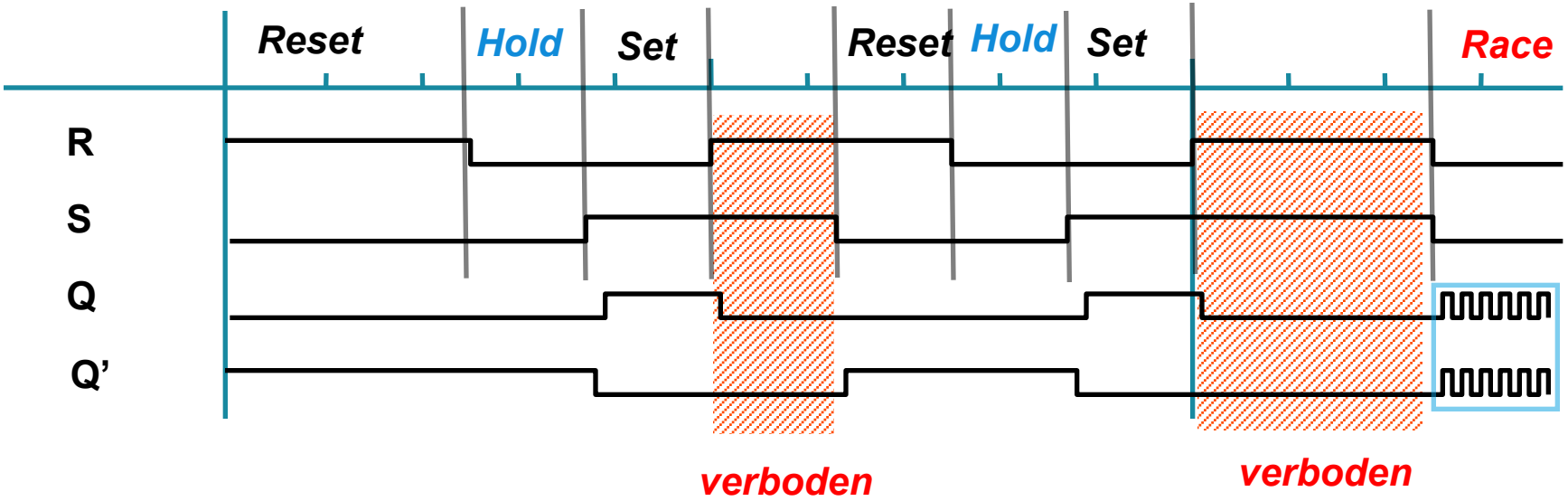
**R-S Latch: (2 gated inverters)**




**R = S = 0**

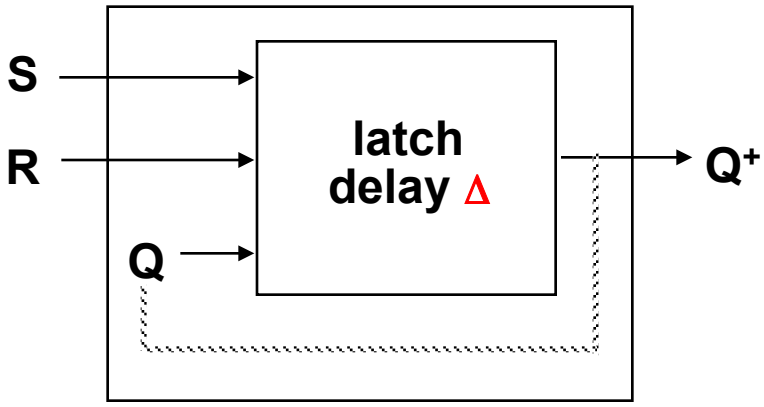



| S | R | Q | Q' | |
|---|---|---|---|---|
| 0 | 0 | 0/1 | 1/0 | hold |
| 0 | 1 | 0 | 1 | reset |
| 1 | 0 | 1 | 0 | set |
| 1 | 1 | 0 | 0 | verboden |



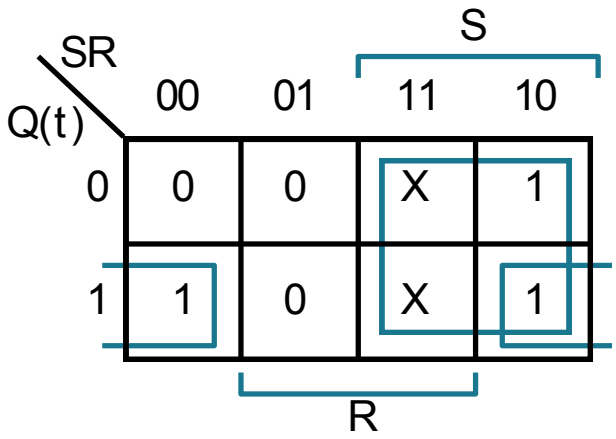
# R-S Latch

**nieuwe state = f(S, R, huidige state):**




| S(t) | R(t) | Q(t) | Q (t + Δ) | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | HOLD |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | RESET |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | SET |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | x | VERBODEN |
| 1 | 1 | 1 | x | |

Q (t + Δ)




**Karakteristieke vergelijking: $Q^+ = S + R' Q$**
**(of $Q^+ = R' S + R' Q = (R + (S + Q)')'$ wanneer 0 voor don't care, zie vorige slide)**

# Synchrone netwerken

**Transitie naar nieuwe state(s) *synchroon* (m.b.v. clock):**

**De clock geeft, nadat het circuit to rust is gekomen, het teken voor de volgende state transitie**

**Voordelen: goed gedefinieerde states en transities, storingsongevoelig(er)**

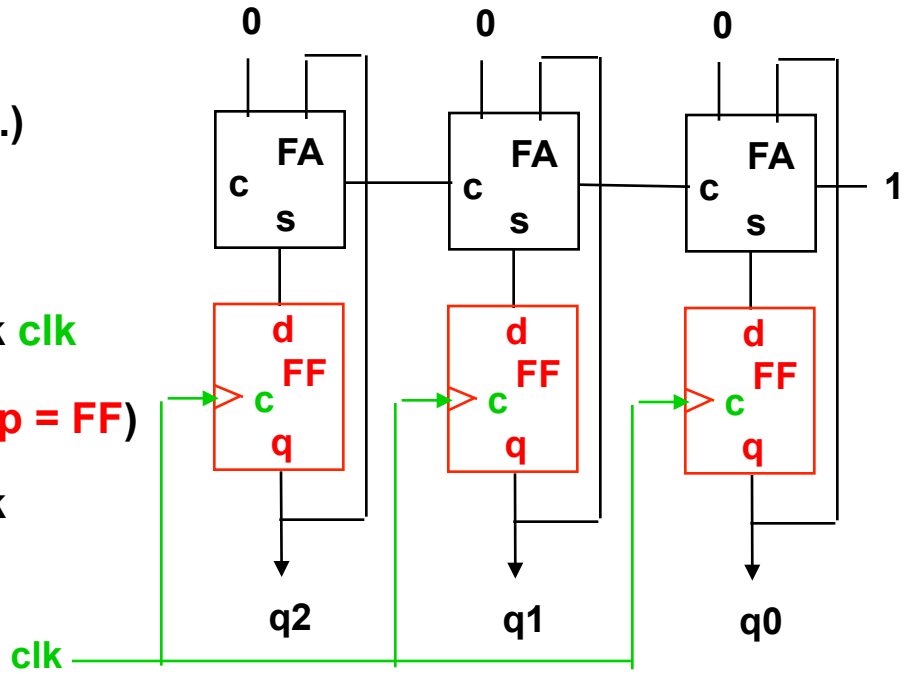
**Voorbeeld:**

**Up-counter ($q_2q_1q_0$ = 0, 1, ..., 7, 0, 1, ...)**

**Nodig:**

**geheugenelement dat d.m.v. de clock clk**
**de momentane ingang d bemonstert**
**en op de uitgang q aanbiedt (Flip Flop = FF)**

**bemonsteringsduur zo klein mogelijk**
**⇒ bemonstering wordt getriggered door clk *flank* (*edge*-triggered)**

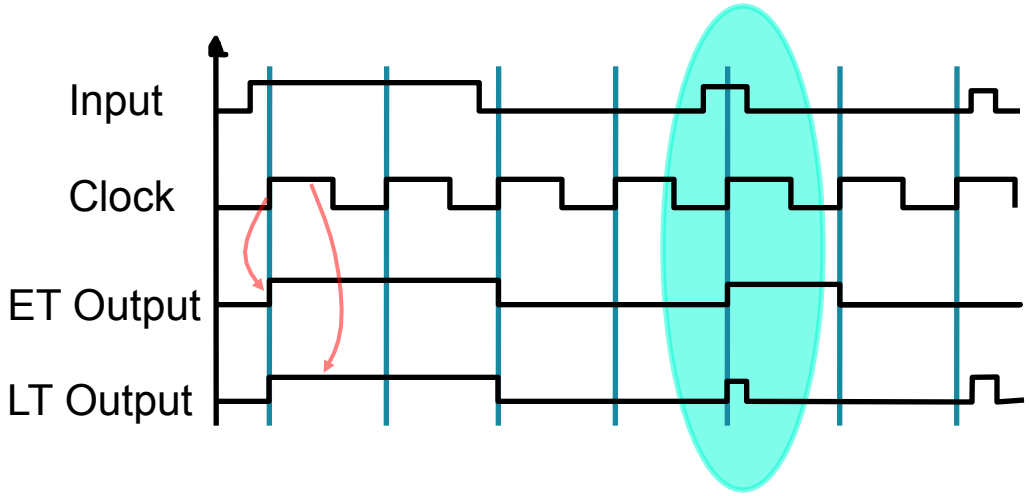


# Level triggered vs. edge triggered

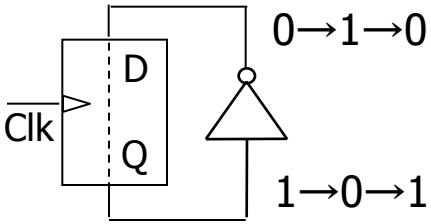
**Clock*:*** **periodieke gebeurtenis (event) triggert state verandering in geheugencel**

**Pos/neg *level* triggered circuit (of *gated latch*) bemonstert gedurende *gehele* hoge/lage *nivo***

**Pos/Neg *edge* triggered circuit (of *flipflop*) bemonstert op pos/neg *flank***

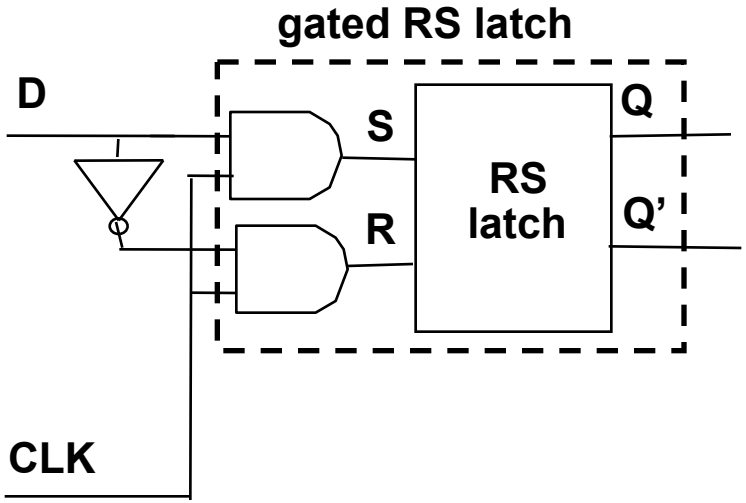



**Level triggered circuit heeft ook probleem wanneer uitgang terugkoppelt naar ingang: oscillaties kunnen optreden:**




**Edge triggered circuit wel aantrekkelijk om state mee vast te leggen**

# Level-triggered D flip-flop (Gated Latch)



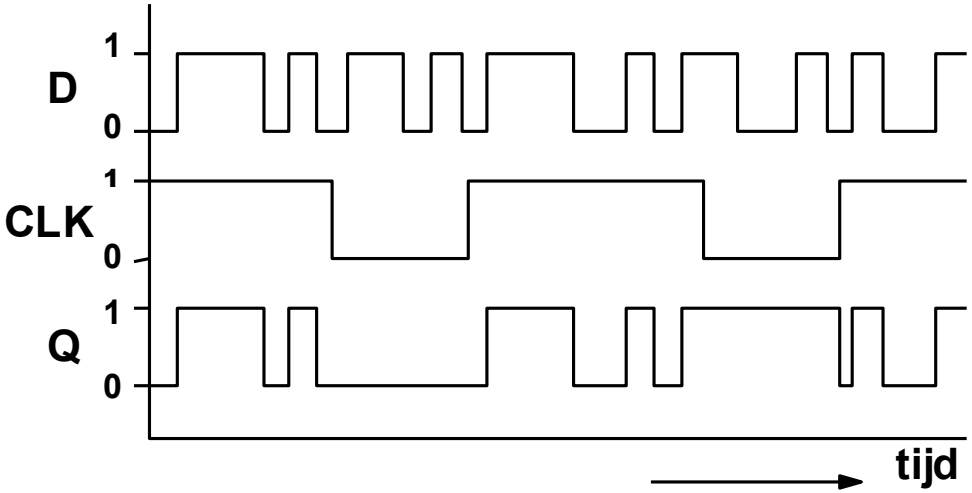

| S | R | Q | Q' | |
|---|---|---|---|---|
| 0 | 0 | 0/1 | 1/0 | hold |
| 0 | 1 | 0 | 1 | transport |
| 1 | 0 | 1 | 0 | transport |
| 1 | 1 | 0 | 0 | |

CLK = 1: → S = D, R = D'

→ Q(t + Δ) = D (transport)

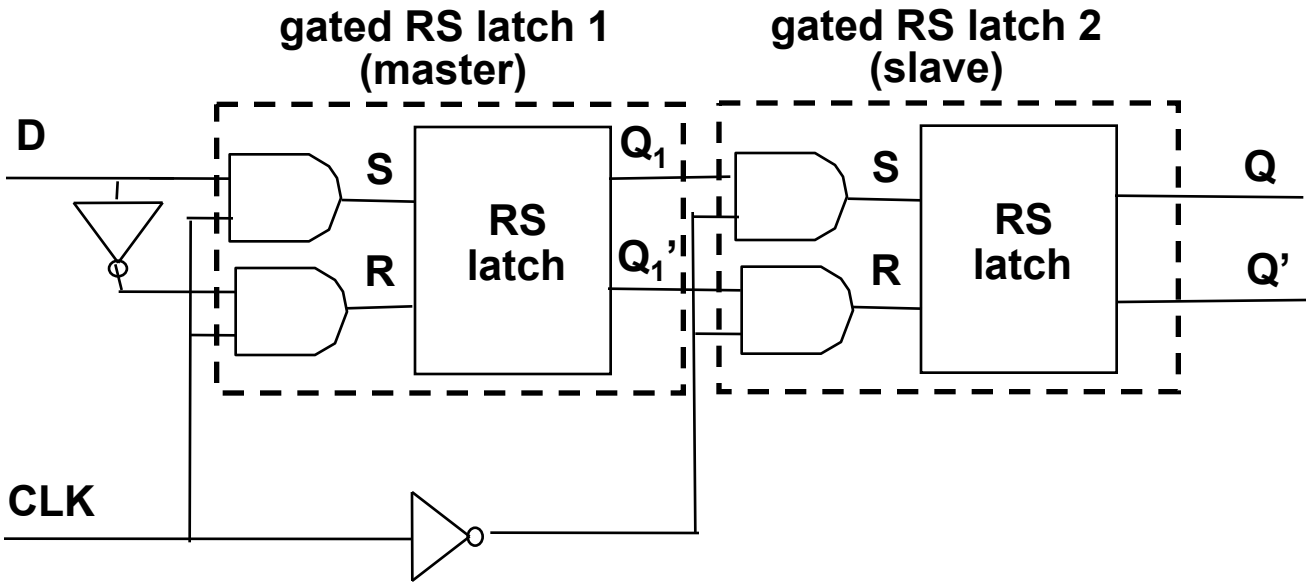
CLK = 0: → S = 0, R = 0

→ Q(t + Δ) = Q(t) (hold)



# Edge-triggered D flipflop: principe

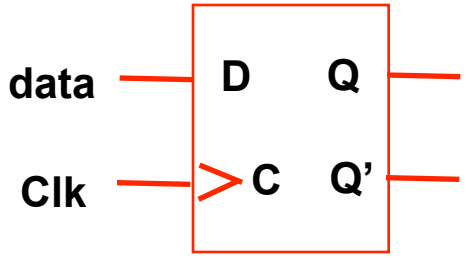
**Principe edge-triggered D flip flop: combinatie van gated RS latches:**




| CLK | latch 1 ($Q_1$) | latch 2 (Q) |
|---|---|---|
| 1 | transport D | hold Q |

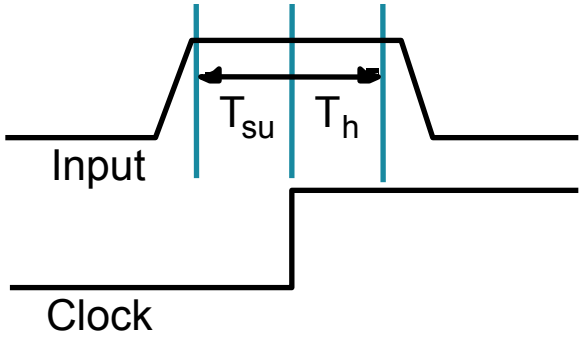
# Edge-triggered D flipflop: realisatie

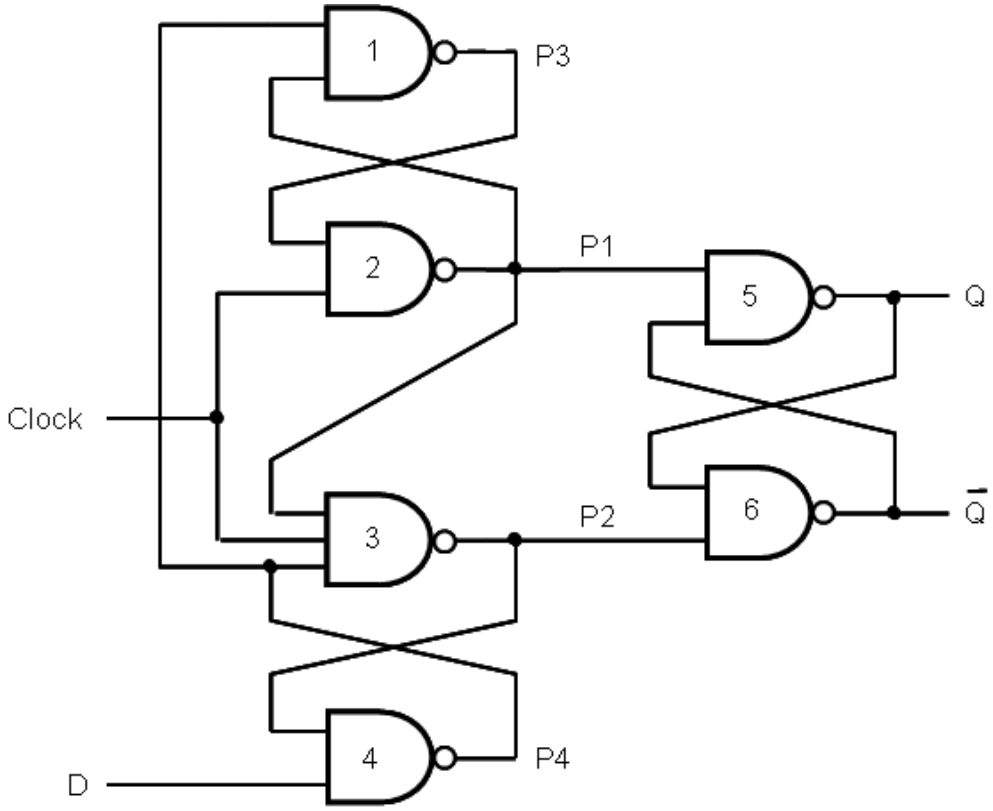
**D flipflop (D-FF):**




**Karakteristieke vergelijking: $Q^+ = D$**

**Input moet stabiel zijn rond clk flank (setup time $T_{su}$, hold time $T_h$) om succesvol te bemonsteren**




**positive-edge triggered D-flip-flop**

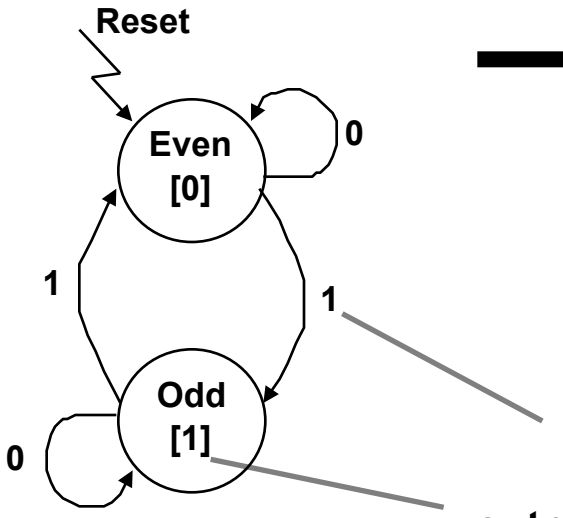
# Case study: Parity checker

**Maak uitgang 1 bij oneven aantal 1-en in invoerstroom:**

| | | |
|---|---|---|
| **reset:** | **1 0 ...** | |
| **input:** | **X 0 1 0 0 1 0 1 0 0 1** | **(waarde juist voor CLK edge)** |
| **output:** | **0 0 1 1 1 0 0 1 1 1 0** | **(waarde na CLK edge)** |

**Codeer "oneven aantal 1-en gedetecteerd" als: 1 (state)**
**Codeer "even aantal 1-en gedetecteerd" als: 0 (state)**

**Toestandsdiagram**




**Toestandstabel:**

| Present State | Input | Next State | Output |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |

**input juist voor CLK edge**

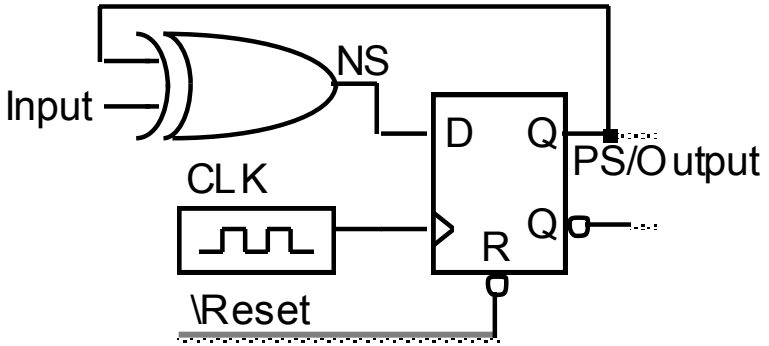
**output na CLK edge**

**NS (= D) = PS ⊕ Input**
**Output = PS**

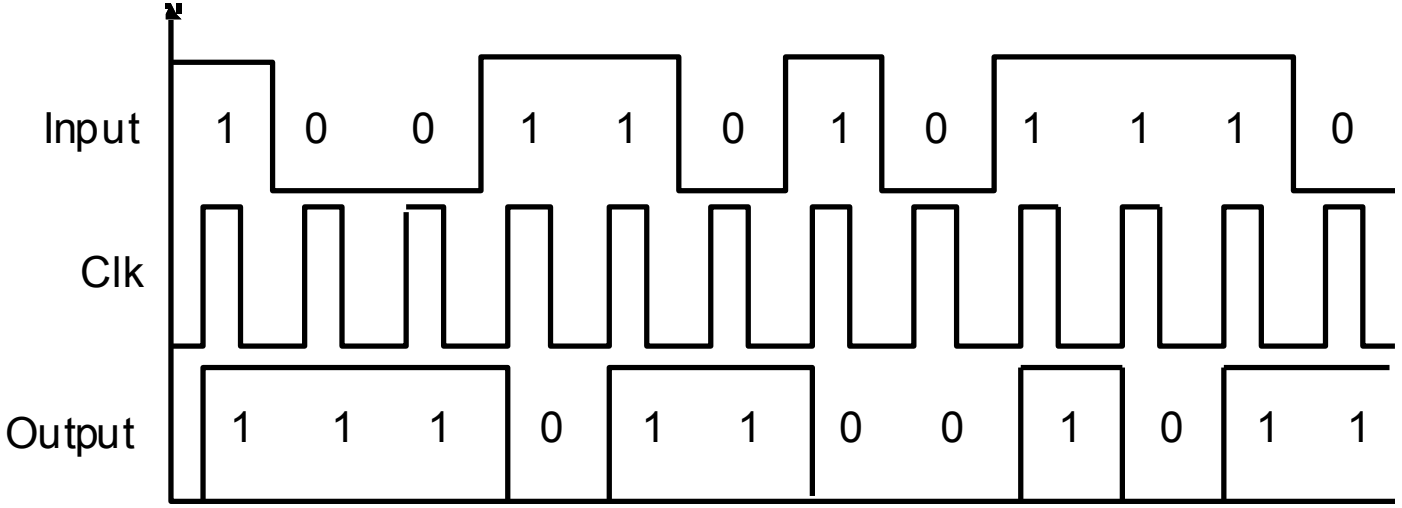
# Case study: Parity checker

**Opslag state (PS) mbv D-FF:**

- **D = NS = PS ⊕ Input;**
- **Output = PS**




**Tijdsgedrag voor Input = 1 0 0 1 1 0 1 0 1 1 1 0:**



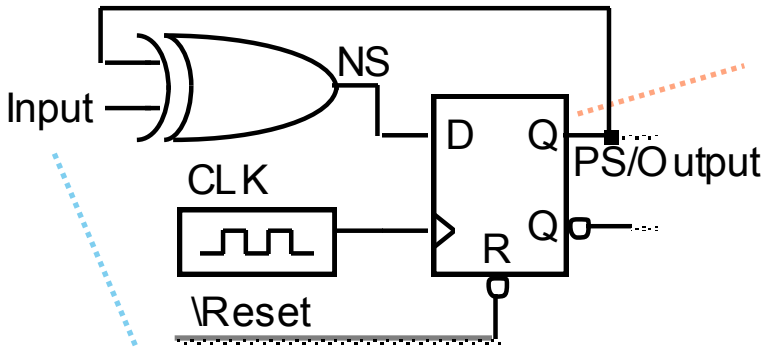
# Finite State Machines (FSM)

# FSM: Definitie

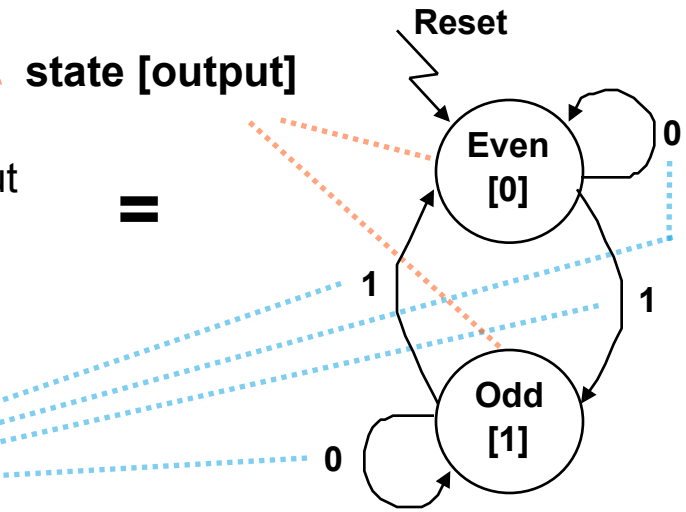
**FSM = seq. circuit met *eindig* # states (s).**
**Executie: $s^{k+1} = f(s^k, \text{input})$ waarbij:**

- **$s^k$ = state als gevolg van klokflank *k***
- **$s^k$ en input worden bemonsterd tijdens klokflank *k+1***

**Vb. FSM: circuit-representatie**




**Vb. FSM: toest.diagr.-representatie (FSD, Finite State Diagram)**



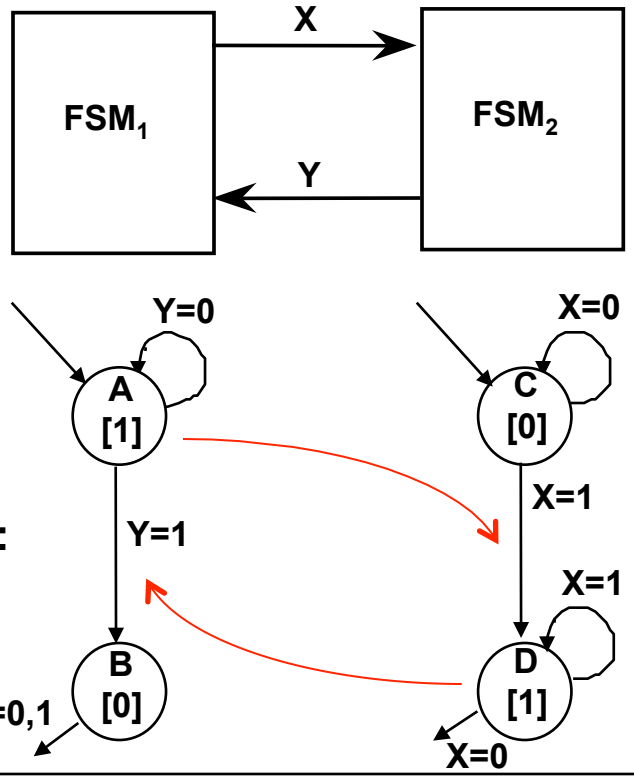

**vereiste input tijdens CLK edge voor deze transitie**

**FSM verandert state in het tempo van de klok. *Kloksnelheid* begrensd door *propagatietijden combinatoriek* in combinatie met *setup/hold tijden* van FF's en hun *interne propagatietijden*.**

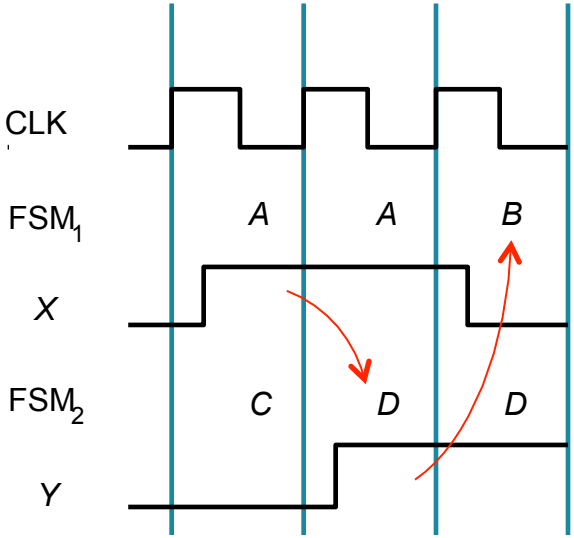
# FSM: Voorbeeld van interactie/timing

**Beschouw $FSM_1$ en $FSM_2$**

**$FSM_1$ wil zijn huidige state A via X doorseinen aan $FSM_2$, die via Y terugmeldt dat hij X gezien heeft (state C ⇒ D), waarop $FSM_1$ naar B gaat (en verder).**
**Initieel geldt: uitgang X = 0 van $FSM_1$, state van $FSM_2$ = C met Y = 0.**
**Dan gaat state van $FSM_1$ naar A.**




**Tijdsdiagram:**

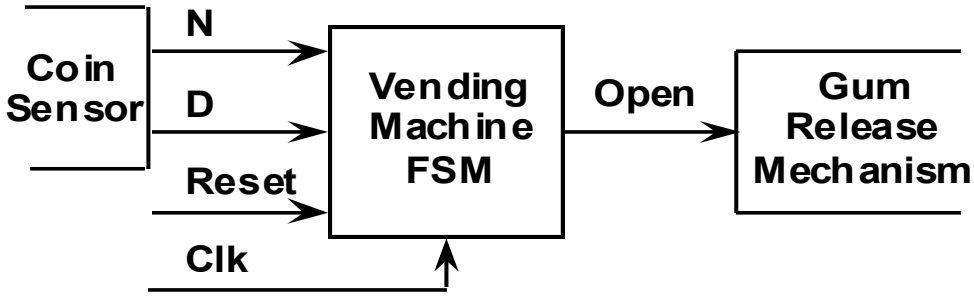


# FSM voorbeeld: Snoepautomaat

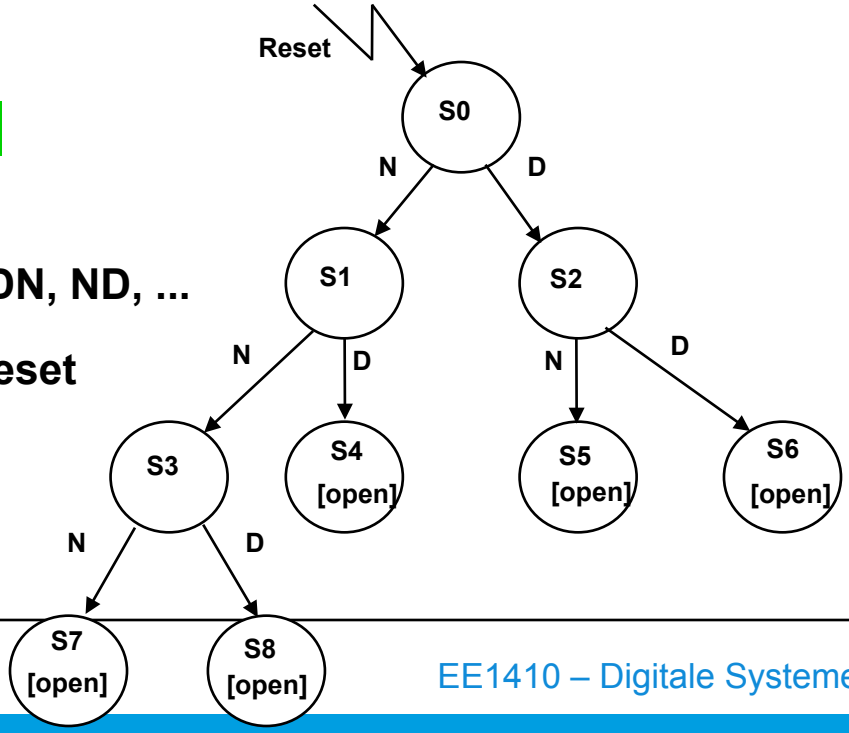
**Probleemdefinitie:**

**retourneer kauwgom nadat minimaal 15 cent is ingevoerd met dubbeltjes (dimes) en stuivers (nickels) (geen wisselgeld)**

**Stap 1: Blokschema:**




**Stap 2: Toestandsdiagram:**



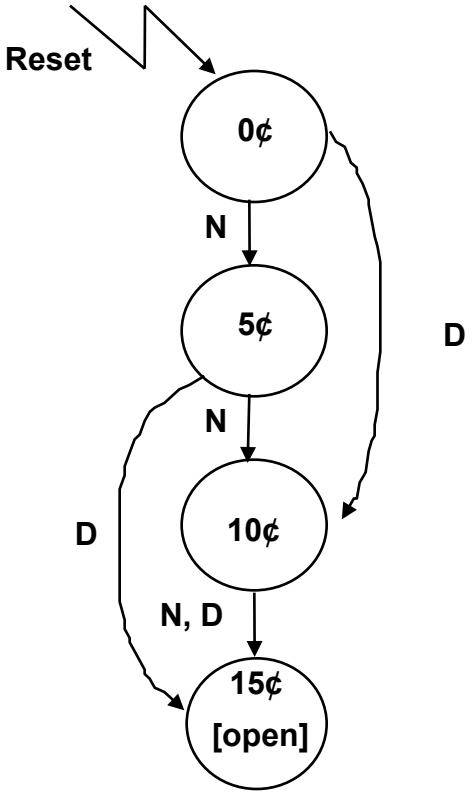

**States:** 0, N, NN, NNN, D, DN, ND, ...

**Inputs:** N(ickel), D(ime), Reset

**Output:** open

# FSM voorbeeld: Snoepautomaat

**Step 3: Vereenvoudiging:**




**Step 4: Toestandcodering:**

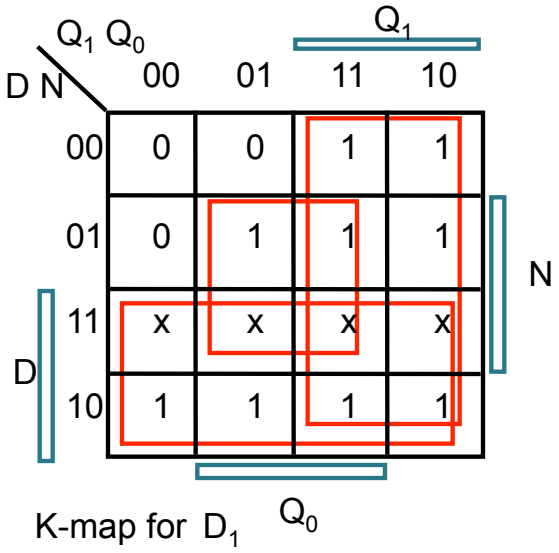
| Present State | $Q_1$ | $Q_0$ | Inputs D | N | Next State | $D_1$ | $D_0$ | Output Open |
|---|---|---|---|---|---|---|---|---|
| 0¢ | 0 | 0 | 0 | 0 | 0¢ | 0 | 0 | 0 |
| | | | 0 | 1 | 5¢ | 0 | 1 | 0 |
| | | | 1 | 0 | 10¢ | 1 | 0 | 0 |
| | | | 1 | 1 | X | X | X | X |
| 5¢ | 0 | 1 | 0 | 0 | 5¢ | 0 | 1 | 0 |
| | | | 0 | 1 | 10¢ | 1 | 0 | 0 |
| | | | 1 | 0 | 15¢ | 1 | 1 | 0 |
| | | | 1 | 1 | X | X | X | X |
| 10¢ | 1 | 0 | 0 | 0 | 10¢ | 1 | 0 | 0 |
| | | | 0 | 1 | 15¢ | 1 | 1 | 0 |
| | | | 1 | 0 | 15¢ | 1 | 1 | 0 |
| | | | 1 | 1 | X | X | X | X |
| 15¢ | 1 | 1 | 0 | 0 | 15¢ | 1 | 1 | 1 |
| | | | 0 | 1 | | 1 | 1 | 1 |
| | | | 1 | 0 | | 1 | 1 | 1 |
| | | | 1 | 1 | | X | X | X |

**Step 5: Toestandstabel:**

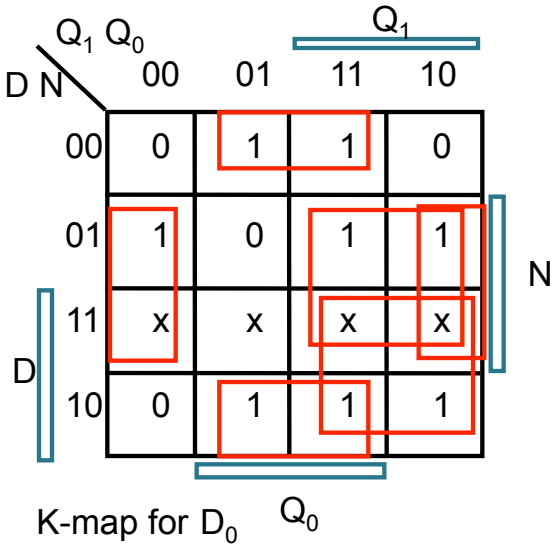
**Hergebruik states waar mogelijk (S2 = S3, S4 = S5 = ... = S8)**

# FSM voorbeeld: Snoepautomaat

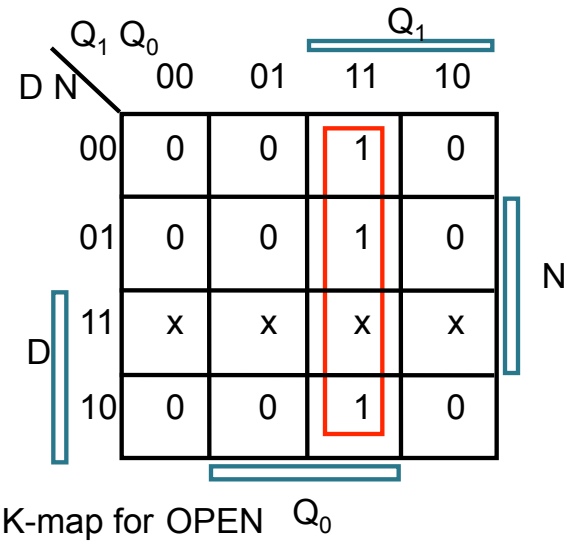
**Step 6: Implementatie:**




K-map for $D_1$




K-map for $D_0$

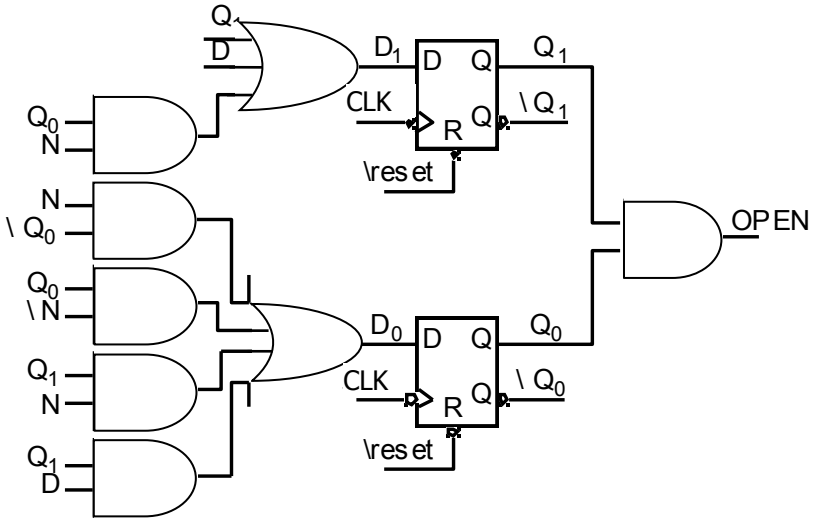



K-map for OPEN

$D_1 = Q_1 + D + Q_0\,N$

$D_0 = N\,Q_0' + Q_0\,N' + Q_1\,N + Q_1\,D$

$OPEN = Q_1\,Q_0$

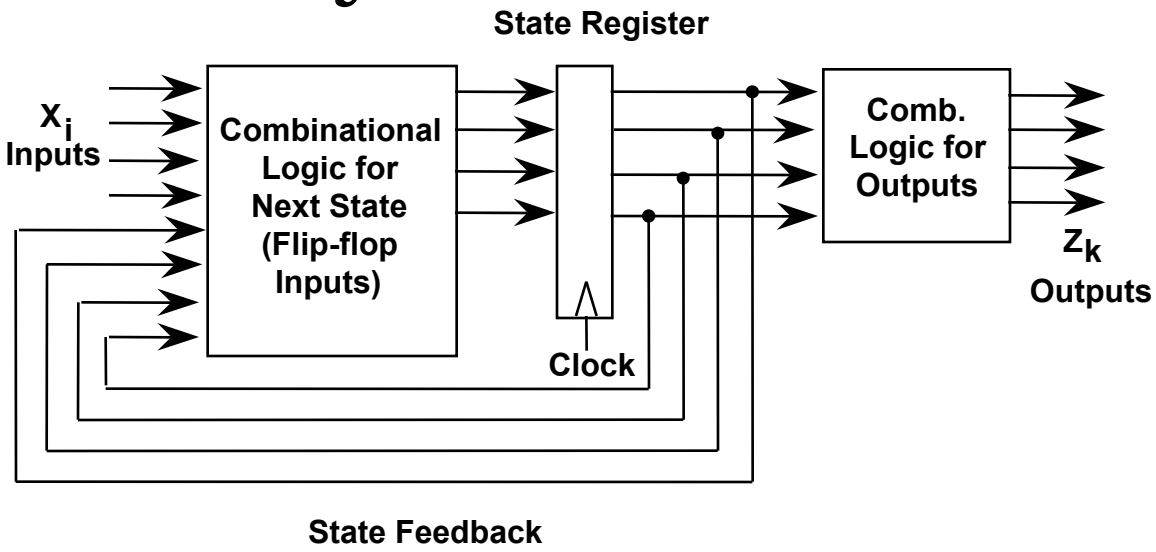


# Moore vs. Mealy machines

**Moore Machine**

**output = f(state)**

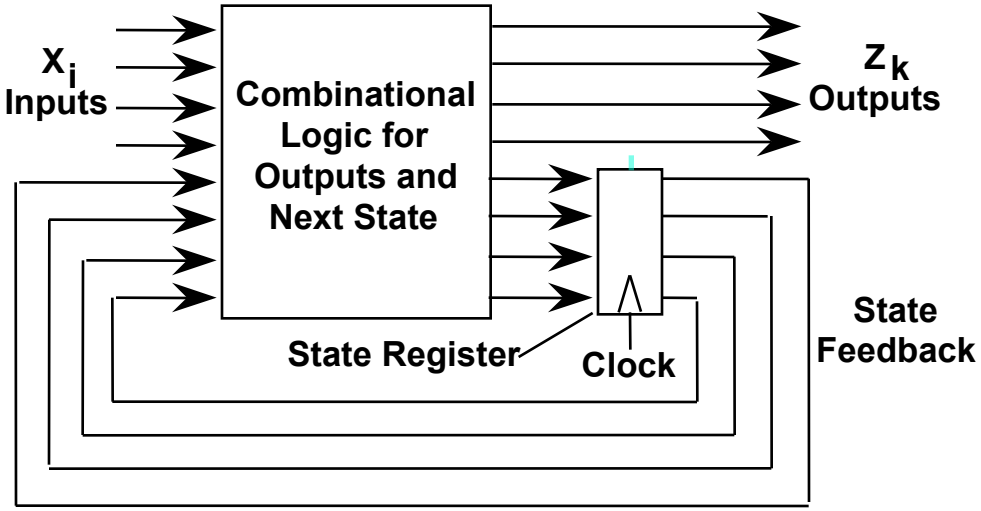
**output verandert synchronoon met toestandsverandering**




**Mealy Machine**

**output = f(state,input)**

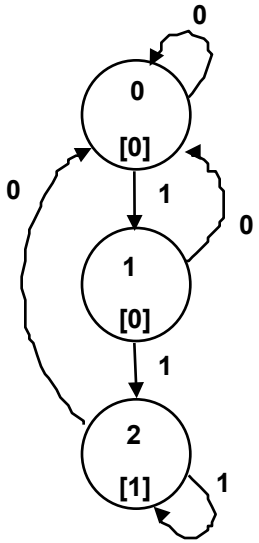
**output verandert direct met input-verandering (asynchrone output)**



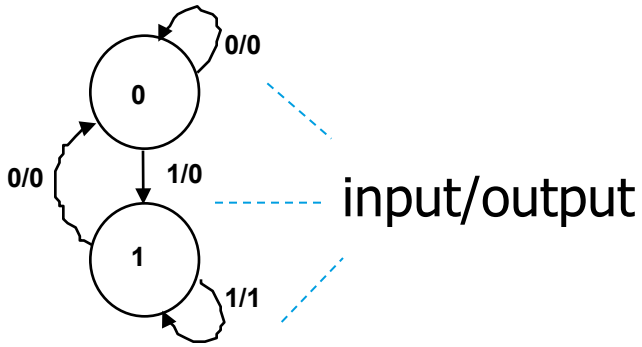
# Moore vs. Mealy machines

**Voorbeeld: FSM die 2 opeenvolgende 1-en in invoerstroom detecteert**

**Moore machine:**



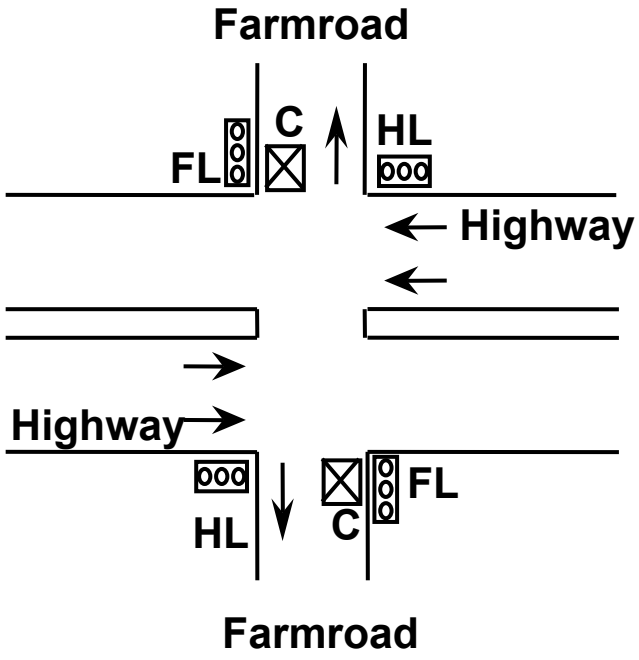

**Mealy machine:**




**Zelfde gedrag maar verschillend # states**

**Nadeel Mealy: eerder kans op problemen (instabiliteit bij terugkoppeling van outputs naar inputs)**

# Case study: Verkeerslichten




**Specificatie:**

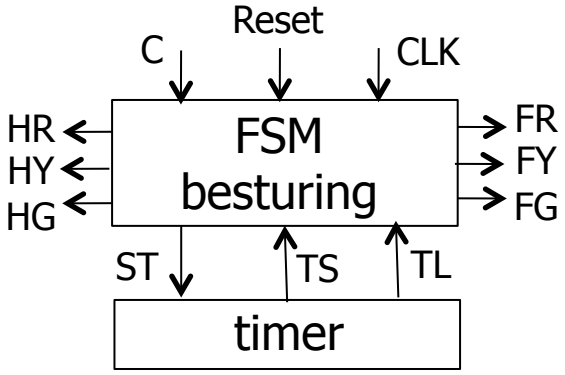
**Basistoestand (reset): HL, FL**

**Indien wagen op C:**

- **lange wachttijd**
- **HL, FL**
- **korte wachttijd**
- **HL, FL**
- **lange wachttijd**
  **tenzij geen wagens meer op C**
- **HL, FL**
- **korte wachttijd**
- **terug naar basis toestand**

**Er is een timer beschikbaar welke start met signaal ST:**
**geeft signaal TS na korte tijd, signaal TL na lange tijd**

**Digitale besturing:**
**States:** S0 (HL, FL), S1 (HL, FL), S2 (HL, FL), S3 (HL, FL)
**Inputs:** Reset, C (meetlus), TS (korte timer gaat af),
TL (lange timer gaat af)
**Outputs:** HG, HY, HR, FG, FY, FR,
ST (set timer, korte puls om timer te (her)starten)



# Case study: Verkeerslichten

**S0** (highway, farmroad)

**overgangsconditie: wagen op C en lange wachttijd verstreken, dus C • TL**
**➔ ST wordt gezet tijdens overgang naar S1**

**S1** (highway, farmroad)

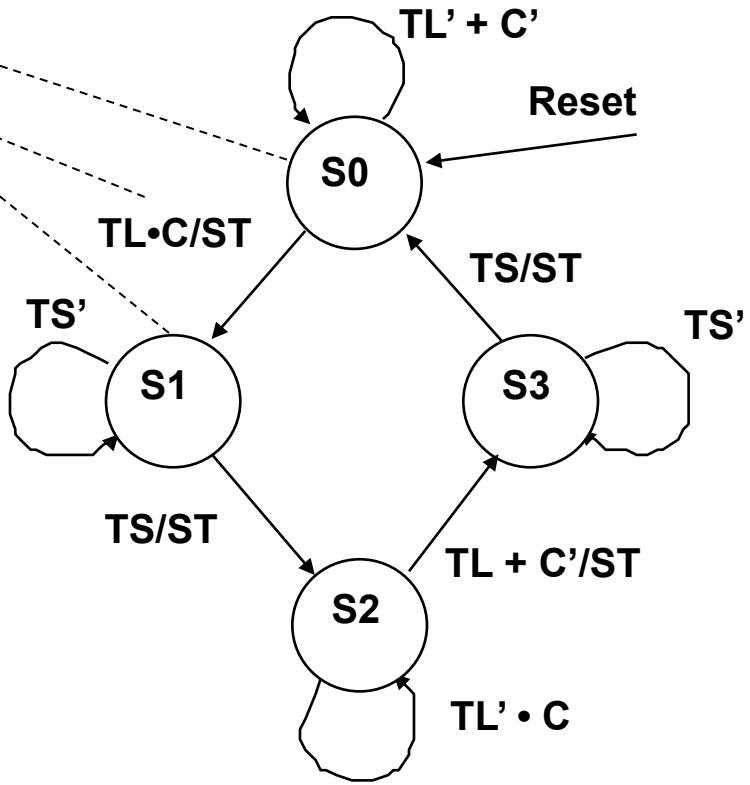
**overgangsconditie: korte tijd verstreken, dus TS**
**➔ ST wordt gezet tijdens overgang naar S2**

**S2** (highway, farmroad)

**overgangsconditie: geen wagen meer op C of lange wachttijd verstreken, dus TL + C'**
**➔ ST wordt gezet tijdens overgang naar S3**

**S3** (highway, farmroad)

**overgangsconditie: korte tijd verstreken, dus TS**
**➔ ST wordt gezet tijdens overgang naar S0**




**Mealy of Moore ?**

# Samenvatting

**Belangrijkste elementen**:

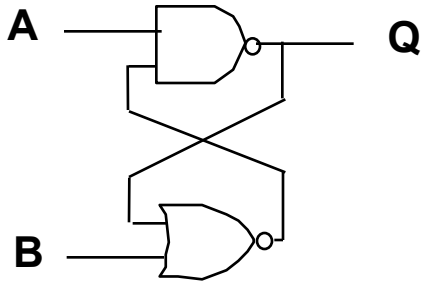
- wat zijn synchrone/asynchrone seq. systemen?
- wat zijn de principes achter latches en flipflops?
- wat is verschil tussen level-triggered en edge-triggered circuits?
- waar wordt maximale clocksnelheid door bepaald?
- wat zijn finite state machines?
- wat zijn de drie belangrijkste FSM representaties?
- wat is de ontwerpmethode van probleem naar FSM implementatie?
- wat zijn Moore en Mealy machines?

**Volgende keer: VHDL voor seq. systemen**

# Huiswerkopgaven h4

**Vraag 1:**

**Gegeven het volgende circuit:**




| A | B | Q |
|---|---|---|
| 0 | 0 | ? |
| 0 | 1 | ? |
| 1 | 0 | ? |
| 1 | 1 | ? |

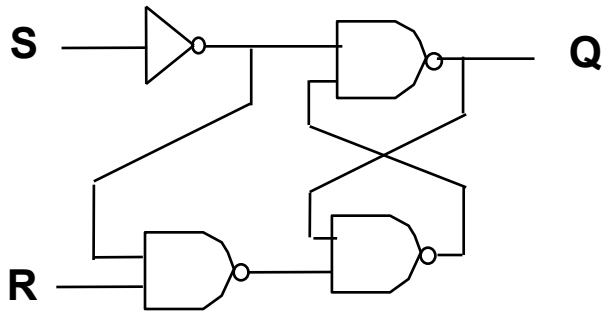
**Het circuit is:**

**a.** een latch; AB = 00 is de verboden ingangscombinatie
**b.** een latch; AB = 01 is de verboden ingangscombinatie
**c.** een latch; AB = 10 is de verboden ingangscombinatie
**d.** geen bruikbaar geheugenelement; de Reset-combinatie ontbreekt

# Huiswerkopgaven h4

**Vraag 2:**

**Bij SR latches is SR = 11 verboden ivm racecondities en niet-complementaire uitgangswaarden. Bij de volgende implementatie:**

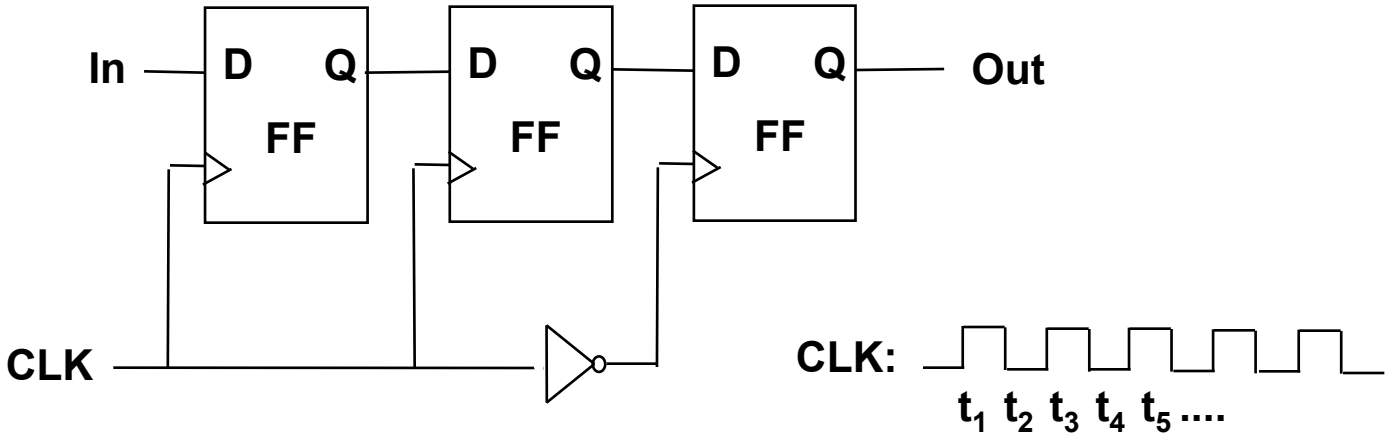



a. **zijn all problemen rond SR = 11 opgelost. Voor SR = 11 is gekozen voor een extra setopdracht**
b. **zijn all problemen rond SR = 11 opgelost. Voor SR = 11 is gekozen voor een extra resetopdracht**
c. **zijn all problemen rond SR = 11 opgelost. Voor SR = 11 is gekozen voor een extra onthoud (hold) opdracht**
d. **zijn niet alle problemen rond SR = 11 opgelost.**

# Huiswerkopgaven h4

**Vraag 3:**

**Gegeven het volgende circuit met drie positive edge-triggered D flipflops:**



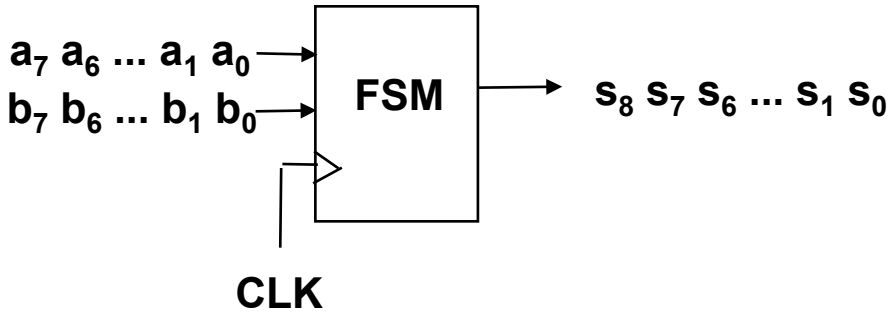

**Indien voor $t_1$ op In het constante signaal X wordt gezet, waneer wordt X op zijn vroegst op Out waargenomen? Op of direct na:**

**a. $t_2$**
**b. $t_3$**
**c. $t_4$**
**d. $t_5$ of later**

# Huiswerkopgaven h4

Casus: sequentiële opteller

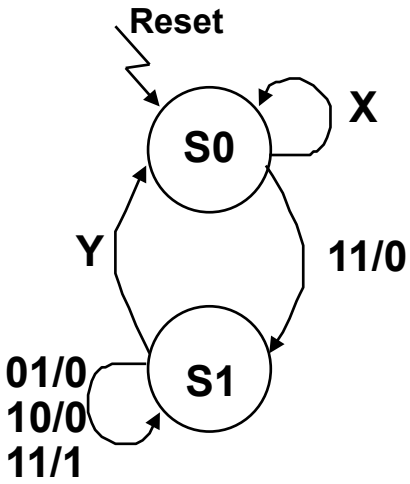
Men wil een circuit ontwerpen dat twee 8-bits getallen A = $a_7$ $a_6$ ... $a_1$ $a_0$ en B = $b_7$ $b_6$ ... $b_1$ $b_0$ wil optellen tot het getal S = $s_8$ $s_7$ $s_6$ ... $s_1$ $s_0$.
Om componenten te sparen wil men de optelling op een *sequentiële* manier realiseren: op tijdstip (klokflank) $t_i$ (i = 0,1,2,...,7) ontvangt het circuit de input bits $a_i$ en $b_i$ en berekent bit $s_i$. Tijdens de berekening speelt de carry $c_{i-1}$ die mogelijk is ontstaan tijdens de vorige berekening op tijdstip i-1 een belangrijke rol (tijdens de eerste berekening op i = 0 is de "vorige" carry natuurlijk 0). De carry speelt uiteraard ook een rol tijdens de berekening van $s_8$. De sequentiële opteller kan worden gebouwd m.b.v. een FSM met 2 states zoals onderstaand circuit:



# Huiswerkopgaven h4

**Vraag 4 (zie de casus "sequentiële opteller"):**

**De FSM kan worden gerepresenteerd door de volgende FSD (Mealy vorm) waarin X en Y nog niet zijn ingevuld:**




Welke uitspraak is correct?

a. S0 representeert $s_i = 0$, S1 representeert $s_i = 1$

b. S0 representeert $c_i = 0$, S1 representeert $c_i = 1$

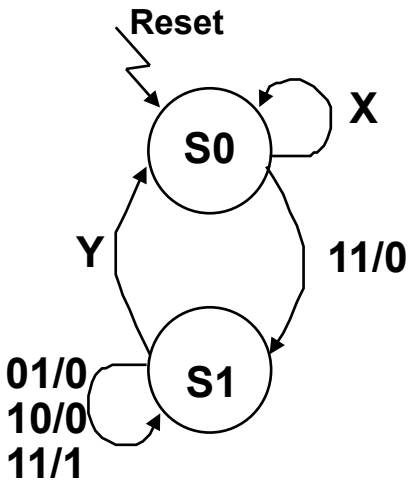
c. S0 representeert $a_i = 1$, S1 representeert $b_i = 1$

d. Geen van bovenstaande antwoorden.

# Huiswerkopgaven h4

**Vraag 5 (zie de casus "sequentiële opteller"):**

**In de FSD staan twee labels X en Y. Wat moet er voor X en Y worden ingevuld?**




a. X = (01/1, 10/1) Y = 11/0

b. X = (01/1, 10/1, 00/0) Y = 00/1

c. X = 00/0 Y = 00/1

d. Geen van bovenstaande antwoorden.

# Huiswerkopgaven h4

**Vraag 6 (zie de casus "sequentiële opteller"):**

**Stel dat we nu twee 64-bit getallen A en B sequentieel willen optellen. Welke van onderstaande uitspraken is het meest correct?**

a. We moeten de vorige FSM aanpassen

b. We kunnen de vorige FSM gebruiken en de berekening kost evenveel tijd (= # klokcycli)

c. We kunnen de vorige FSM gebruiken maar de berekening kost wel meer tijd (= # klokcycli)

d. Geen van bovenstaande antwoorden.

# Huiswerkopgaven h4

**Vraag 7 (zie de casus "sequentiële opteller"):**

**Stel dat de opteller met D flip-flops en 4-input NAND gates moet worden gerealiseerd.**
**Hoeveel flip-flops en NAND gates zijn benodigd (een invertor kost ook een NAND gate)?**

a. 1 D FF en 11 gates

b. 1 D FF en 9 gates

c. 2 D FF en 5 gates

d. Geen van bovenstaande antwoorden.

# Huiswerkopgaven h4

**Vraag 8 (zie de casus "sequentiële opteller"):**

**Stel dat de opteller m.b.v. een Moore machine wordt gerealiseerd. Welke van de volgende uitspraken is het meest correct?**

a. De FSD van de Moore machine heeft 2 states

b. De FSD van de Moore machine heeft 3 states

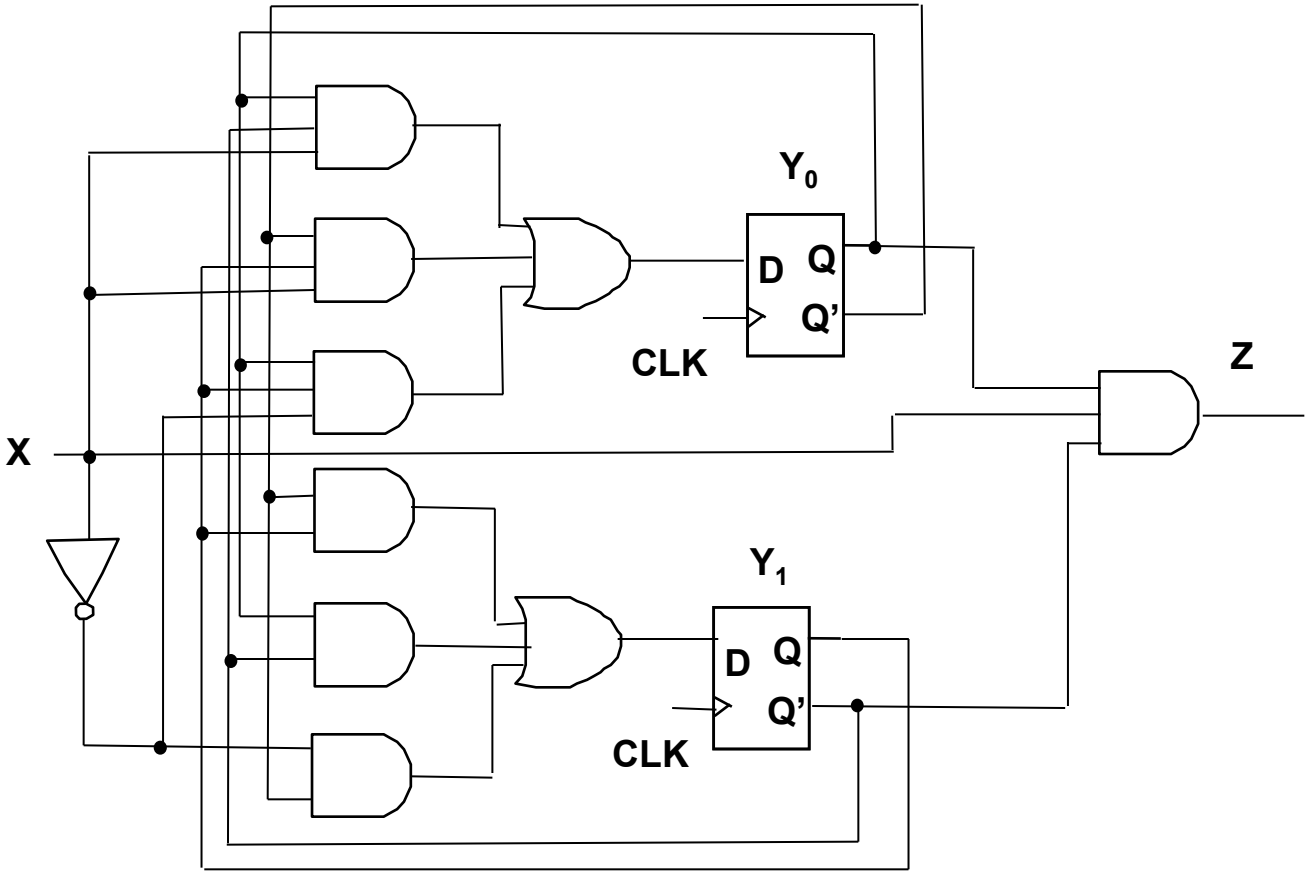
c. De FSD van de Moore machine heeft 4 states

d. Geen van bovenstaande antwoorden.

# Huiswerkopgaven h4

Casus: 4-bits detector

Onderstaand circuit detecteert een bepaalde reeks van 4 bits:



# Huiswerkopgaven h4

Vraag 9 (zie de casus "4-bit detector"):

Welke type FSM is hier gerealiseerd?

a. Moore machine

b. Mealy machine

c. Synchronous Mealy machine

d. Geen van bovenstaande antwoorden.

# Huiswerkopgaven h4

Vraag 10 (zie de casus "4-bit detector"):

Welke 4-bits reeks wordt door het circuit gedetecteerd?

a. 0011

b. 1010

c. 0101

d. Geen van bovenstaande antwoorden.