4.4 Schuifregisters

4.4.1 Toepassingsgebied

Schuifregisters zijn een voorname toepassing van sequentiële logica om binaire data te bewaren en over te dragen. De data die zich op de registeringangen bevindt, wordt ingelezen en op het ritme van een kloksignaal doorgeschoven naar de uitgang. Aan dit proces dankt het schuifregister zijn naam.

In zijn meest eenvoudige vorm bestaat een shuifregister uit een aaneenschakeling van afzonderlijke **D-flipflops** waarbij we één flipflop voorzien voor elke databit. De keten is zo opgebouwd dat de uitgang van elke vorige flipflop doorverbonden is met de ingang van de volgende.

De data kan **serieel** (links of rechts inschuivend) of **parallel** (alle bits tegelijkertijd) aangeboden worden. Het aantal bits dat een schuifregister moet kunnen opslaan bepaalt het aantal individuele flipflops en meestal zijn dat er **8** (1 byte).

Schuifregisters worden, naast gewone **data-opslag**, ook gebruikt voor specifieke bit-operaties bij het uitvoeren van **rekenkundige bewerkingen** in computers. In de **datacommunicatie** zet men deze registers ook in om seriële data om te vormen naar parallelle data-overdracht.

Elke flipflop van het schuifregister wordt door een *gemeenschappelijk* kloksignaal getriggerd en daarom zijn schuifregisters **synchrone** digitale bouwstenen. De meeste schuifregister-IC's zijn uitgerust met een clear-pin (reset) zodat we ze op onze wenken kunnen *setten* of *resetten*.

4.4.2 Soorten

We kunnen schuifregisters opdelen volgens de manier waarop de data doorheen het schuifregister beweegt. We onderscheiden alzo vier verschillende modi:

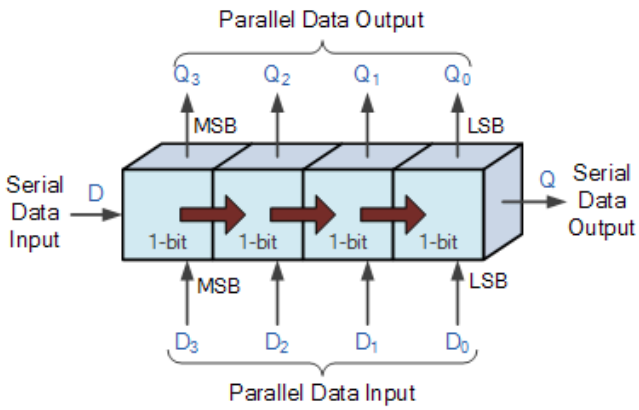
1. **Serial-in to Parallel-out** (SIPO):
   - Het register wordt bit per bit *serieel* ingeladen.
   - De data in het geheugen is beschikbaar aan de uitgangen in *parallelle* vorm.

2. **Serial-in to Serial-out** (SISO):
   - De data wordt bit per bit *serieel* IN en UIT het register geladen op het commando van een klok.
   - Het opschuiven van de data kan zowel naar links als naar rechts.

3. **Parallel-in to Serial-out** (PISO):
   - De *parallelle* data wordt in één beweging in het register ingeladen.
   - De data verlaat bit per bit het register op een *seriële* manier op het commando van een klok.

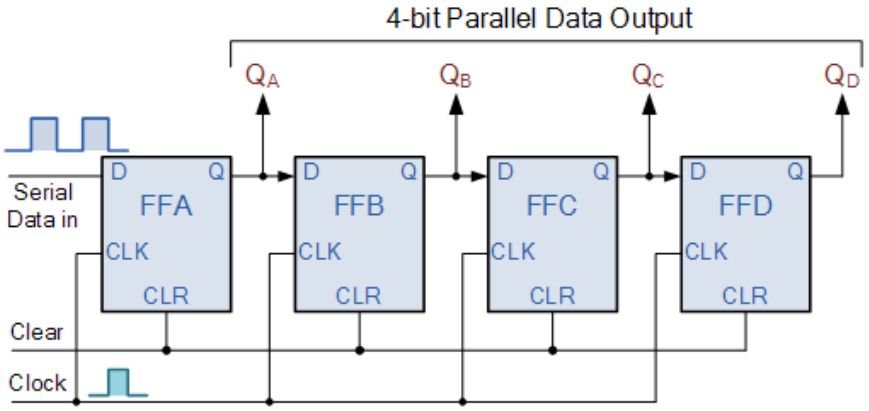
4. **Parallel-in to Parallel-out** (PIPO):
   - De *parallelle* data wordt in één beweging in het register ingeladen.
   - De data wordt op een *parallelle* manier naar de uitgangen overgedragen op commando van dezelfde klokpuls als het inladen.

Het het verplaatsen van data van links naar rechts doorheen het schuifregister kunnen we voorstellen zoals in figuur 4.34:



△ Fig.4.34 Het schuiven van data doorheen een register. Bron: https://www.electronics-tutorials.ws

4.4.3 SIPO-schuifregister



△ Fig.4.35 Het SIPO-schuifregister.
Bron: https://www.electronics-tutorials.ws

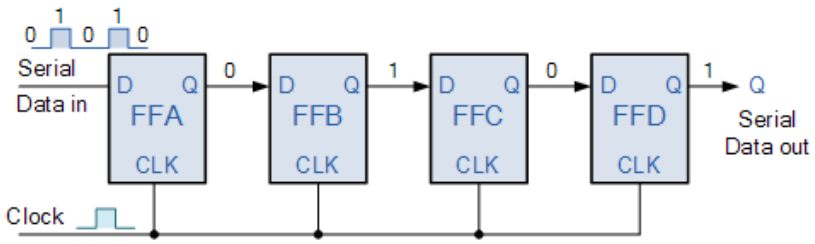
We veronderstellen dat alle flipflops (FFA t.e.m. FFD) zijn gereset zodat alle uitgangen (Q_A t.e.m. Q_D) logisch nul zijn. Wanneer we op de DATA-ingang van flipflop FFA een logische '1' aanbieden dan zal bij de eerstvolgende klokpuls FFA worden geset (Q_A wordt hoog) terwijl alle andere FF-uitgangen laag blijven. Vervolgens veronderstellen we dat de DATA-ingang van FFA terug naar logische '0' gaat. Bij de tweede klokpuls zal Q_A terug omklappen naar logische '0' terwijl de uitgang Q_B hoog wordt omdat de DATA-ingang van FFB (verbonden met Q_A) logisch '1' was tijdens de transitie van het tweede kloksignaal. De logische '1' van FFA is dus één plaats *opgeschoven* naar rechts en bevindt zich nu in FFB. Bij de derde klokpuls zal deze logische '1' weer een plaats opschuiven naar rechts en zal deze '1' verschijnen op uitgang Q_C. Na de vierde klokovergang zal Q_D hoog zijn en na de vijfde klok-overgang verdwijnt de logische '1' uit het register: alle uitgangen zijn nu logisch '0'.

Algemeen kunnen we stellen dat bij elke klokovergang de serieel aangeboden data één plaats opschuift naar rechts terwijl de data van het **SIPO-schuifregister** (Q_A, Q_B, Q_C en Q_D) direct parallel beschikbaar is. Dit proces wordt mooi gevisualiseerd in de waar-heidstabel en het signaal-tijddiagram van figuur 4.36.

Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0

△ Fig.4.36 Het opschuiven van een logische '1' in een SIPO-schuifregister.
Bron: https://www.electronics-tutorials.ws

4.4.4 SISO-schuifregister

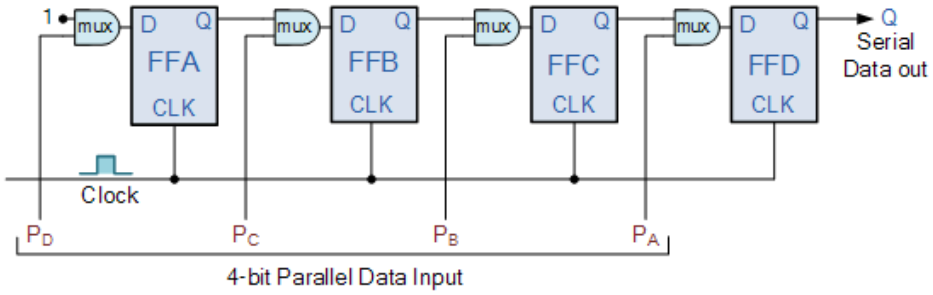


△ Fig.4.37 Het SISO-schuifregister.
Bron: https://www.electronics-tutorials.ws

Het **SISO-schuifregister** is vrij gelijkaardig aan het SIPO-register maar nu is de data van het register niet onmiddellijk beschikbaar. De data beweegt op het ritme van de klok doorheen het register en omdat er maar één uitgang is verlaat de data (bit per bit) het schuifregister op een seriële manier.

Het SISO-schuifregister beschikt over slechts 3 aansluitingen: een seriële ingang (SI), een seriële uitgang (SO) en het kloksignaal (Clk). Dit type van schuifregister is het eenvoudigste van de 4 mogelijke configuraties en je zult je wellicht afvragen wat het nut kan zijn van een register waarvan de ingangsdata identiek is aan de uitgangsdata. Dit schuifregister kan ingezet worden als *tijdelijk* opslagmedium van data of als een *vertraginslijn* waarbij de data gecontroleerd vertraagd kan worden.

4.4.5 PISO-schuifregister



△ Fig.4.38 Het PISO-schuifregister.
Bron: https://www.electronics-tutorials.ws

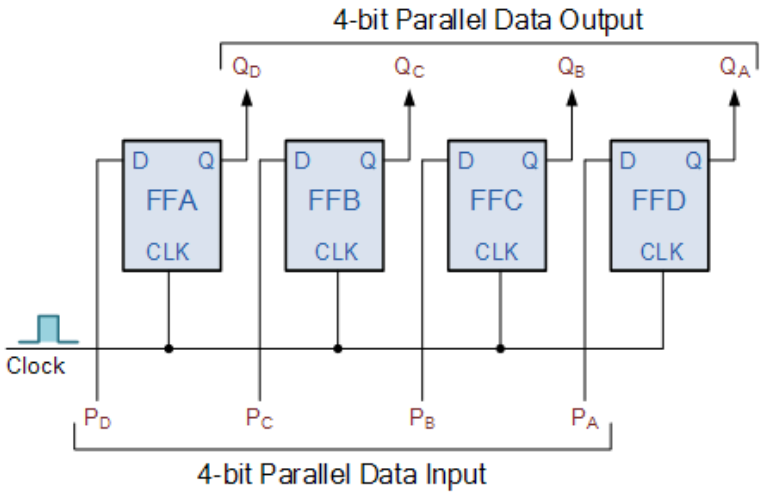
Bij een **Parallel-in to Serial-out** schuifregister wordt de data op een *parallelle* manier ingeladen. Op alle data-ingangen van de D-flipflops worden de databits (P_A t.e.m. P_D) op hetzelfde moment ingegeven. De data die zich nu in de flip-flops bevindt, wordt dan via pin Q (Serial Data Out) bit per bit - op het ritme van het kloksignaal - sequentieel uitgelezen.

Merk op dat er bij een PISO-schuifregister *geen* kloksignaal nodig is om de reeds aanwezige parallelle data in te laden. In ons voorbeeld zijn de 4 klokpulsen nodig om de 4-bits data terug uit het register te halen.

Dit type schuifregister kunnen we gebruiken om 8-bit woorden (1 byte) om te zetten in een serieel formaat zodat het mogelijk wordt om verschillende parallel datalijnen te converteren naar één seriële datastream. Dit pricipe wordt *'multiplexing'*[22] genoemd en is een veel gebruikte techniek in de datacommunicatie.

[22] In de digitale techniek is multiplexing (MUX) een proces waarbij meerdere datastromen worden gecombineerd tot één signaal. Het voordeel hiervan is dat een communicatiemedium efficiënter gebruikt kan worden.

4.4.6 PIPO-schuifregister



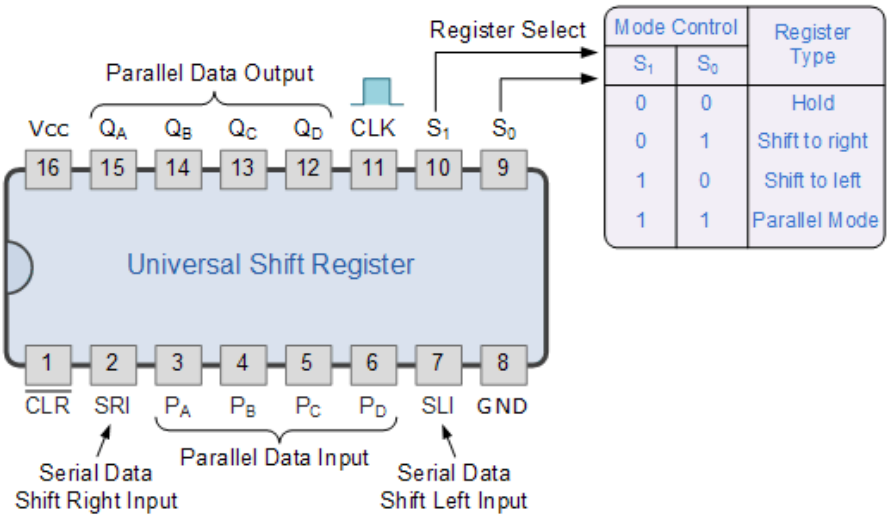
△ Fig.4.39 Het PIPO-schuifregister.
Bron: https://www.electronics-tutorials.ws

Het **Parallel-in to Parallel-out** schuifregister gedraagt zich als een tijdelijk opslagmedium waarbij we de data in de tijd kunnen vertragen. Het gedrag van het PIPO-register lijkt dus op het eerder besproken SISO-register. De ingangsdata wordt *parallel* aangeboden op de ingangspinnen (P_A t.e.m. P_D) en bij de klokpuls direct *parallel* overgedragen naar de respectievelijke uitgangspinnen (Q_A t.e.m. Q_D). Dus 1 enkele klokpuls laadt en ontlaadt het register in één beweging.

Het PIPO-schuifregister bezit slechts 3 aansluitingen: de parallele ingang (PI), de parallelle uitgang (PO) en het kloksignaal (Clk). Er zijn geen extra onderlinge verbindingen tussen de verschillende D-flipflops nodig omdat er geen data van links naar rechts dient geschoven te worden.

4.4.7 Universeel schuifregister

Vandaag kunnen we beschikken over supersnelle bi-directionele **universele schuifregisters** zoals het veel toegepast IC 74LS194. Dit IC bevat een 4-bits multifunctioneel schuifregister waarmee we de eerder besproken modi kunnen realiseren: SISO, SIPO, PISO en PIPO. In de seriële modus is zowel links schuiven als rechts schuiven mogelijk. Universele shuifregisters hebben bijkomende ingangen (S_1 en S_0) nodig om de gewenste mode in te stellen, en om de flipflops te kunnen setten (pre-loading) en resetten.



△ Fig.4.40 Het universele schuifregister 74LS194.
Bron: https://www.electronics-tutorials.ws

Universele schuifregisters zijn zeer nuttige digitale bouwstenen. Ze zijn onmisbaar om *'temporary memory storage'* mogelijk te maken, datastreams gecontroleerd te vertragen, seriële data-stromen om te zetten naar het parallelle formaat en vice versa. Universele schuifregisters worden veelvuldig toegepast bij reken-kundige operaties (vermenigvuldigen en delen) in microcontrollers en computers.

4.4.8 74LS194 datasheet

MOTOROLA

4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

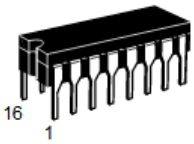
SN54/74LS194A

4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

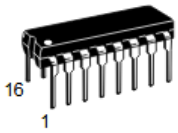
LOW POWER SCHOTTKY

The SN54/74LS194A is a High Speed 4-Bit Bidirectional Universal Shift Register. As a high speed multifunctional sequential building block, it is useful in a wide variety of applications. It may be used in serial-serial, shift left, shift right, serial-parallel, parallel-serial, and parallel-parallel data register transfers. The LS194A is similar in operation to the LS195A Universal Shift Register, with added features of shift left without external connections and hold (do nothing) modes of operation. It utilizes the Schottky diode clamped process to achieve high speeds and is fully compatible with all Motorola TTL families.

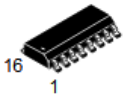
- Typical Shift Frequency of 36 MHz
- Asynchronous Master Reset
- Hold (Do Nothing) Mode
- Fully Synchronous Serial or Parallel Data Transfers
- Input Clamp Diodes Limit High Speed Termination Effects



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

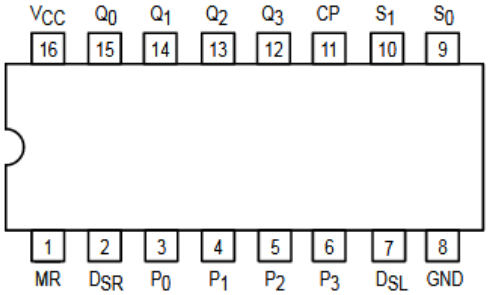


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

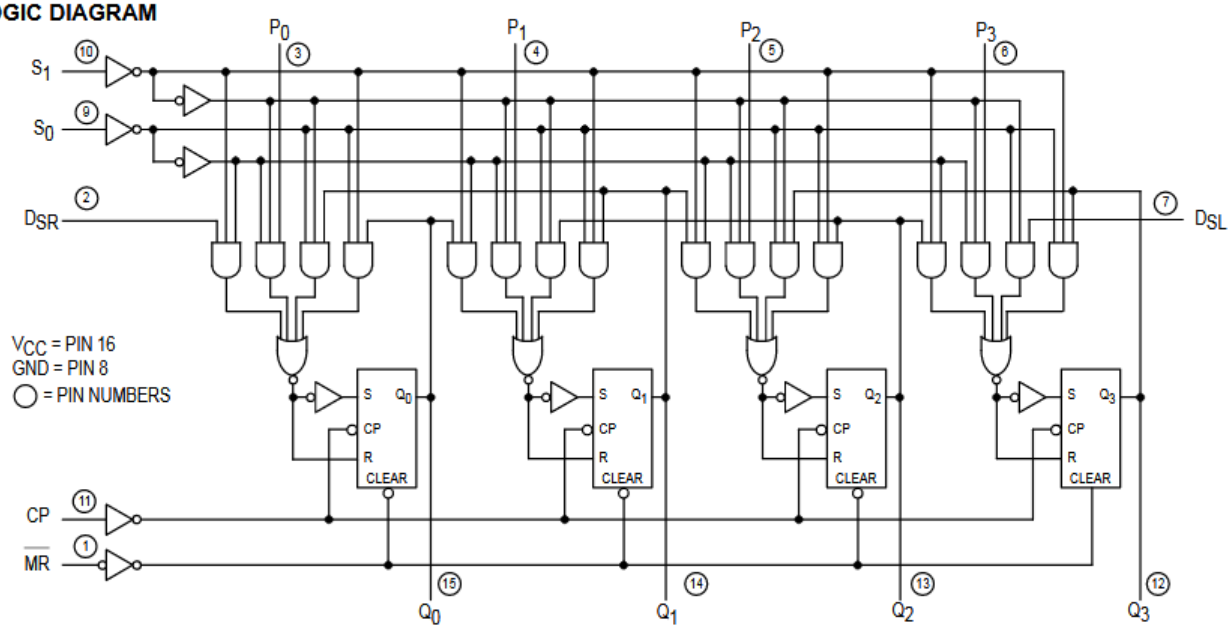
SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXD	SOIC

CONNECTION DIAGRAM DIP (TOP VIEW)



PIN NAMES		LOADING (Note a)	
		HIGH	LOW
S_0, S_1	Mode Control Inputs	0.5 U.L.	0.25 U.L.
P_0–P_3	Parallel Data Inputs	0.5 U.L.	0.25 U.L.
D_{SR}	Serial (Shift Right) Data Input	0.5 U.L.	0.25 U.L.
D_{SL}	Serial (Shift Left) Data Input	0.5 U.L.	0.25 U.L.
$\overline{CP}$	Clock (Active HIGH Going Edge) Input	0.5 U.L.	0.25 U.L.
MR	Master Reset (Active LOW) Input	0.5 U.L.	0.25 U.L.
Q_0–Q_3	Parallel Outputs (Note b)	10 U.L.	5 (2.5) U.L.

.OGIC DIAGRAM



FUNCTIONAL DESCRIPTION

The Logic Diagram and Truth Table indicate the functional characteristics of the LS194A 4-Bit Bidirectional Shift Register. The LS194A is similar in operation to the Motorola LS195A Universal Shift Register when used in serial or parallel data register transfers. Some of the common features of the two devices are described below:

All data and mode control inputs are edge-triggered, responding only to the LOW to HIGH transition of the Clock (CP). The only timing restriction, therefore, is that the mode control and selected data inputs must be stable one set-up time prior to the positive transition of the clock pulse.

The register is fully synchronous, with all operations taking place in less than 15 ns (typical) making the device especially useful for implementing very high speed CPUs, or the memory buffer registers.

The four parallel data inputs (P_0, P_1, P_2, P_3) are D-type inputs. When both S_0 and S_1 are HIGH, the data appearing on P_0, P_1, P_2, and P_3 inputs is transferred to the Q_0, Q_1, Q_2, and Q_3 outputs respectively following the next LOW to HIGH transition of the clock.

The asynchronous Master Reset ($\overline{MR}$), when LOW, overrides all other input conditions and forces the Q outputs LOW.

Special logic features of the LS194A design which increase the range of application are described below:

Two mode control inputs (S_0, S_1) determine the synchronous operation of the device. As shown in the Mode Selection Table, data can be entered and shifted from left to right (shift right, $Q_0 \rightarrow Q_1$, etc.) or right to left (shift left, $Q_3 \rightarrow Q_2$, etc.), or parallel data can be entered loading all four bits of the register simultaneously. When both S_0 and S_1,are LOW, the existing data is retained in a "do nothing" mode without restricting the HIGH to LOW clock transition.

D-type serial data inputs (D_{SR}, D_{SL}) are provided on both the first and last stages to allow multistage shift right or shift left data transfers without interfering with parallel load operation.

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS						OUTPUTS			
	MR	S_1	S_0	D_{SR}	D_{SL}	P_n	Q_0	Q_1	Q_2	Q_3
Reset	L	X	X	X	X	X	L	L	L	L
Hold	H	l	l	X	X	X	q_0	q_1	q_2	q_3
Shift Left	H	h	l	X	l	X	q_1	q_2	q_3	L
	H	h	l	X	h	X	q_1	q_2	q_3	H
Shift Right	H	l	h	l	X	X	L	q_0	q_1	q_2
	H	l	h	h	X	X	H	q_0	q_1	q_2
Parallel Load	H	h	h	X	X	P_n	P_0	P_1	P_2	P_3

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
l = LOW voltage level one set-up time prior to the LOW to HIGH clock transition
h = HIGH voltage level one set-up time prior to the LOW to HIGH clock transition
p_n (q_n) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54, 74			−0.4	mA
I_{OL}	Output Current — Low	54 74			4.0 8.0	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

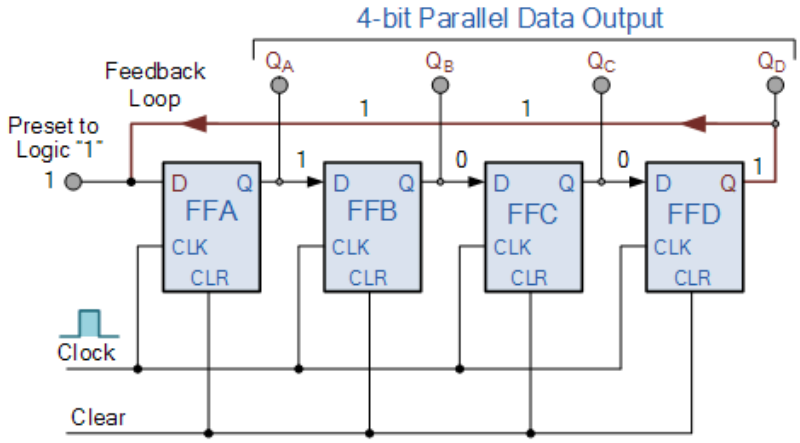
Symbol	Parameter		Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.5		V		
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Short Circuit Current (Note 1)		−20		−100	mA	V_{CC} = MAX	
I_{CC}	Power Supply Current				23	mA	V_{CC} = MAX	

AC CHARACTERISTICS (T_A = 25°C)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
f_{MAX}	Maximum Clock Frequency	25	36		MHz	V_{CC} = 5.0 V C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay, Clock to Output		14 17	22 26	ns	
t_{PHL}	Propagation Delay, MR to Output		19	30	ns	

4.4.9 Ring-teller

Bij het eerder besproken SISO-schuifregister zagen we dat de *sequentie* (volgorde van de opeenvolgende databits) van de input-data ongewijzigd op de uitgang van de laatste flipflop van de keten terug te vinden was. Het SISO-register gedroeg zich als een soort *vertragingscircuit* waarbij het originele datasignaal een vast aantal klokpulsen later terug te vinden was op de seriële uitgang van het register. Maar wat zou nu het effect zijn wanneer we de uitgang van dit schuifregister (Q_D) zouden verbinden met de registeringang (D_A)? Op die manier maken we een *'gesloten lus'*[23] waarbij de databits continu in rondjes draaien op het commando van de klok. Door de registeruitgang *terug te voeren*[24] naar de registeringang vormen we een standaard schuifregister om naar een zogenaamde **ringteller**[25].

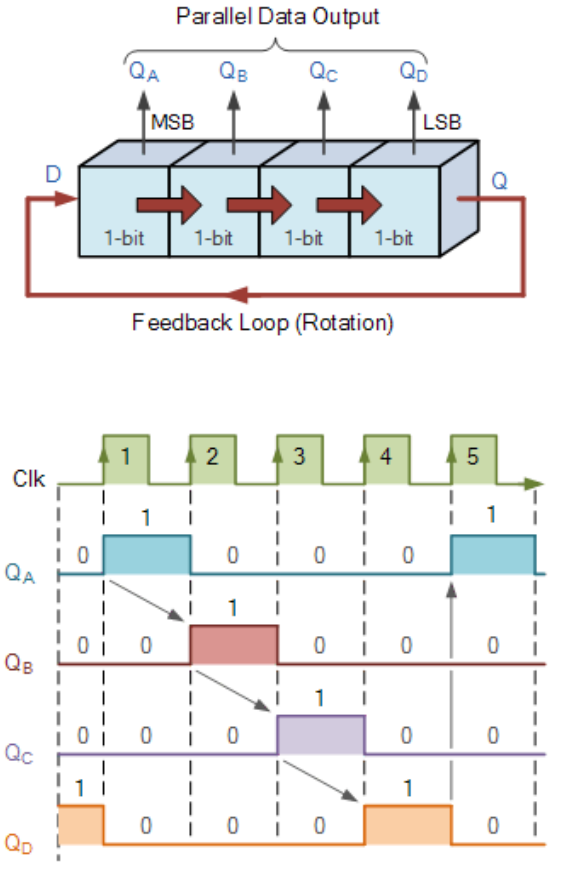


△ Fig.4.41 Principeschema van een 4-bit ringteller.
Bron: https://www.electronics-tutorials.ws

Figuur 4.41 toont een SIPO-schuifregister, geschakeld als een synchrone **4-bit ringteller**. Bij aanvang zetten we de eerste D-flipflop (FFA) op '1' terwijl we alle andere flipflops resetten. Dit noemen we de 'preset' van het register. Om dit praktisch voor elkaar te krijgen zullen we eerst alle flipflops '0' maken (resetten) door het toepassen van een CLR-commando. Dan plaatsen we een preset-puls op de ingang van de eerste flipflop (FFA) waarvan de uitgang (Q_A) '1' wordt bij de eerstvolgende klokpuls. Zodoende hebben we in het schuifregister één enkele '1' ingeladen.

[23] 'gesloten lus' of in het Engels de meer gangbare term: 'closed loop'
[24] elektronici en ingenieurs spreken van 'feedback'-systemen
[25] in vakliteratuur ook wel terug te vinden onder de benaming:'Ring Counter'

Bij de daaropvolgende klokpulsen zal deze enkele databit doorheen het schuifregister beginnen ronddraaien. Elke 4e klokcyclus heeft de databit juist éénmaal de ring doorlopen en dit patroon zal zich eindeloos blijven herhalen zolang het kloksignaal aanwezig is. Dit effect van het circuleren van één enkele databit wordt in het tijddiagram van figuur 4.42 voorgesteld:



△ Fig.4.42 Rotatiebeweging van een 4-bit ringteller.
Bron: https://www.electronics-tutorials.ws

De 4-bit ringteller bezit 4 duidelijk te onderscheiden toestanden en wordt daarom ook wel een *'modulo-4'* teller genoemd. De signaal-frequentie van elke flipflop-uitgang is daarbij juist gelijk aan 1/4 van de aangeboden klokfrequentie.

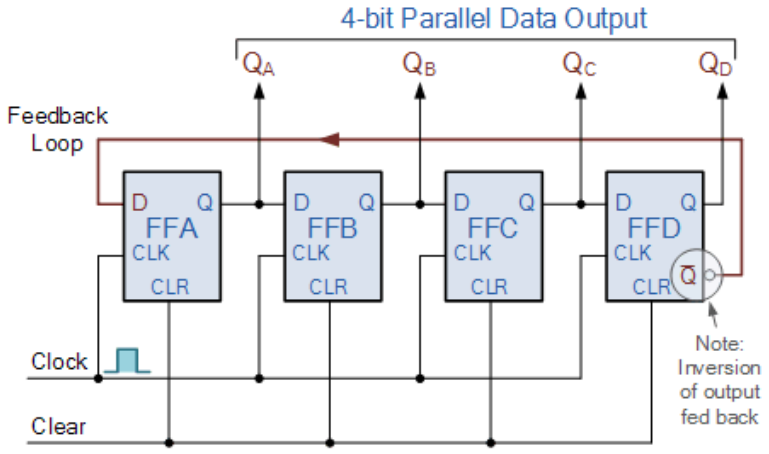
Definitie

> De **modulus** van een teller is een getal dat het aantal verschillende *teltoestanden* aangeeft of het aantal patronen die we kunnen onderscheiden vooraleer de teller zichzelf begint te herhalen.

Een 'modulo-n' ringteller vereist 'n' flipflops om 'n' verschillende uitgangstoestanden te bekomen. Zoals we weten zijn er met 4 binaire uitgangsvariabelen (Q_A t.e.m. Q_D) in principe 16 (2^4) verschillende uitgangstoestanden mogelijk. Ringtellers zijn daarom niet erg efficiënt wat betreft het gebruik van hun mogelijke uitgangstoestanden.

4.4.10 Johnson-teller

De **Johnsonteller** of *'gekruiste ringteler'* maakt, zoals een standaard ringteller, ook gebruik van een feedback-lus. Het verschil zit hem in het feit dat we de *inverse* uitgang van de laatste flipflop verbinden met de D-ingang van de eerste flipflop. Figuur 4.43 maakt veel duidelijk:



△ Fig.4.43 Principeschema van een 4-bit Johnsonteller. Bron: https://www.electronics-tutorials.ws

Het voornaamste voordeel van dit type ringteller is dat we maar de helft zoveel flipflops nodig hebben in vergelijking met een standaard ringteller om een zelfde aantal toestanden te bekomen. De inversie van Q_D zorgt er voor dat dit type ringteller op een andere manier telt. Een standaard ringteller telt als volgt: '0001' = $(1)_{10}$, '0010' = $(2)_{10}$, '0100' = $(4)_{10}$, '1000' = $(8)_{10}$ en deze sequentie wordt dan eindeloos herhaald. Een Johnsonteller daarintegen telt eerst *op* en daarna telt hij *af*. Een 4-bit Johnsonteller levert op die manier **8** verschillende toestanden af, terwijl de 4-bit standaard ringteller slechts 4 verschillende toestanden bezat. De toestandentabel van figuur 4.44 toont dit aan.

Clock Pulse No	FFA	FFB	FFC	FFD
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

△ Fig.4.44 Waarheidstabel van een 4-bit Johnsonteller.
Bron: https://www.electronics-tutorials.ws

Naast het *tellen* of laten *roteren* van databits in een continue loop kunnen ringtellers ook gebruikt worden om bepaalde bitpatronen te herkennen in een gegeven dataset. We doen dit door eenvoudige basispoorten (AND en OR) met de flipflopuitgangen te verbinden. Een andere toepassing van ringtellers is het verlagen van het kloksignaal met een vaste verhouding afhankelijk van de feedback-connectie. Ook het genereren van de typische bitpatronen (vol-stap en half-stap bedrijf) om *stappenmotoren* aan te sturen behoort tot de mogelijkheden.

Laboproef ь

Realiseer m.b.v. het universele schuifregister **74LS194**:

- een *standaard ringteller* met 4 uitgangen
- een *Johnsonteller* met 4 uitgangen

Doe eerst een computersimulatie met afzonderlijke D-flipflops vooraleer je de schakelingen in praktijk brengt op een breadboard.

Maak gebruik van LED's om de respectievelijke toestanden van de uitgangen aan te geven.

Stel de klokfrequentie in op ongeveer 1 hertz.