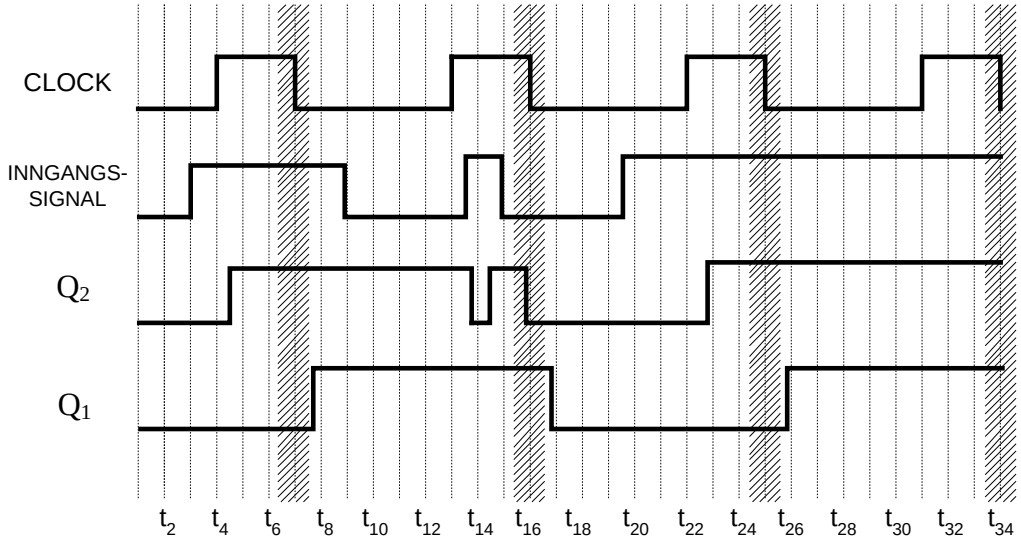
Løsningsforslag øving 3
Digitalteknikk og Datamaskiner – TFE4105 – H12

Oppgave 1: Flanke- og nivåstyrte vipper

a) Vi ser fra figuren at pulstog Q_2 skifter fra høy til lav, og tilbake til høy, *mens* klokken er høy. Dette pulstoget tilhører derfor en D-latch.

Pulstog Q_1 skifter nivå kun i tilknytning til den fallende klokke-flanken, og hører derfor til en D-vippe (flankestyrt). Plasseringen av invertereren gjør at slave-låsen leder når CLOCK er lav. Følgelig vil utgangen Q endre seg i det CLOCK skifter fra høy til lav, og vi får en vippe som er styrt av den fallende (eller negative) flanken.

b) $T_{oppsett}$-T_{holde} er et tidsintervall som omslutter den aktive flanken til klokken. Hvis man endrer inngangssignalet i dette tidsrommet har man ikke kontroll på hvordan det vil innvirke på utgangsverdien til kretsen.



Figur 3.1: De skraverte området er tidsintervallet $T_{oppsett}$ - T_{holde}.

c)

Q	$Q_{(next)}$	D
0	0	0
0	1	1
1	0	0
1	1	1

Tabell 3.1: Eksitasjonstabell for D-vippe.

Vi ser at nestetilstanden til D-vippen er lik inngangsignalet til vippen.

Q	$Q_{(next)}$	T
0	0	0
0	1	1
1	0	1
1	1	0

Tabell 3.2: Eksitasjonstabell for T-vippe.

Hvis inngangssignalet T er 1, vil vippen skifte tilstand ved neste aktive klokkeflanke (toggle-vippe).

Q	$Q_{(next)}$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Tabell 3.3: Eksitasjonstabell for SR-vippe.

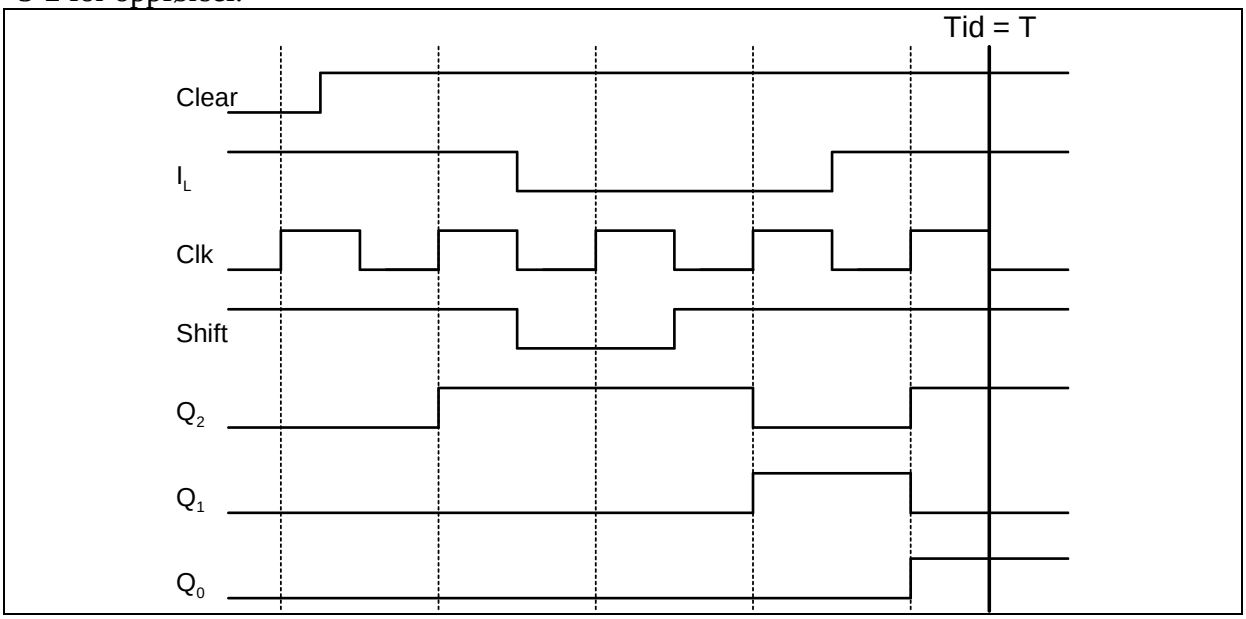
Dersom vippen er i tilstanden 0, er det S som bestemmer om vippen skal settes til 1 eller forbli 0. S kalles derfor set-inngangen. Dersom vippen er i tilstand 1, er det R som bestemmer om vippen skal resettes (settes til 0). R kalles derfor reset-inngangen til vippen. Merk at det ikke finnes noen rad i tabellen der både S og R er 1, siden dette er ulovlig for SR-vippen.

Q	$Q_{(next)}$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabell 3.4: Eksitasjonstabell for JK-vippe.

Dersom vippen er i tilstanden 0, er det J som bestemmer om vippen skal settes til 1 eller forbli 0. J kalles derfor set-inngangen. Dersom vippen er i tilstand 1, er det K som bestemmer om vippen skal resettes (settes til 0). K kalles derfor reset-inngangen til vippen. Merk at dersom både J og K er 1, så vil vippen skifte tilstand ved neste aktive klokkeflanke (toggle).

d) Ved tiden T er Q2=1, Q1=0 og Q0=1. Skiftregisteret svarer til figur 7.4 i Gajski. Se figur 3-2 for oppførsel.



Figur 3-2: Skiftregister med inngangsverdier

e) For å kunne adressere 16 registre kreves fire adresselinjer. Siden vi har tre porter som skal adressere hvert sitt register samtidig, trenger registerfila 4x3=12 adresselinjer totalt. I tillegg kommer WE, RE, og datalinjer. Merk at registerbredden ikke har innvirkning på antall adresselinjer. Se figur 7.13 i læreboka for et eksempel med fire registre.

Oppgave 2: Analyse av sekvensielle kretser

a) Kretsen er en Moore-type tilstandsmaskin siden utgangene kun er funksjoner av nåtilstandene til vippene, og ikke av inngangen INN.

En Mealy-type tilstandsmaskin har asynkrone utgangssignaler. Dette forvansker design av eventuelle delkretser som har disse som innganger. Til gjengjeld kan en ofte klare seg med færre tilstander i en Mealy-tilstandsmaskin. Dersom man i en gitt tilstand skal ha høyt signal ut dersom en inngang er høy og lavt signal ut ellers, så må man i en Moore-maskin eksplisitt innføre en ny tilstand som man forflytter seg til når inngangen er høy. Dette er ikke nødvendig i en Mealy-maskin. Her kan vi la kretsens utgangssignal være styrt av at vi er i den opprinnelige tilstanden OG har høyt inngangssignal.

b) For å komme frem til tilstandsdiagrammet må vi gå gjennom flere trinn. Først må vi sette opp likningene for J- og K-inngangene. Verdiene for disse kan man føre direkte i egne kolonner i nestetilstandsdiagrammet, for så å benytte verdiene i hver rad til å finne tilhørende nestetilstand (se forelesningsnotater fra Digleik-eksemplet). Som et alternativ til dette skal vi nå isteden bruke likningene for J- og K-inngangene til å finne nestetilstands-likningene for JK-vippene. Disse bruker vi så til å sette opp nestetilstands-tabellen, og til slutt tilstandsdiagrammet.

Analyse av kombinatorikken for J- og K-inngangene gir følgende likninger:

J1 = Q0
K1 = Q0’ + INN’
J0 = Q1’ INN
K0 = Q1 INN’

Tilsvarende er likningen for utgangene:

$$UT = Q1' \; Q0$$

Nå kan vi sette opp nestetilstandslikningene for vippene. Her bruker vi uttrykket $Q_{next} = J\overline{Q} + \overline{K}Q$ fra tabell 6.1 i Gajski.

Q1(Neste) = J1 Q1' + K1' Q1 =
Q0 Q1' + (Q0' + INN')' Q1 =
Q0 Q1' + Q0 INN Q1 =
Q1' Q0 + Q1 Q0 INN

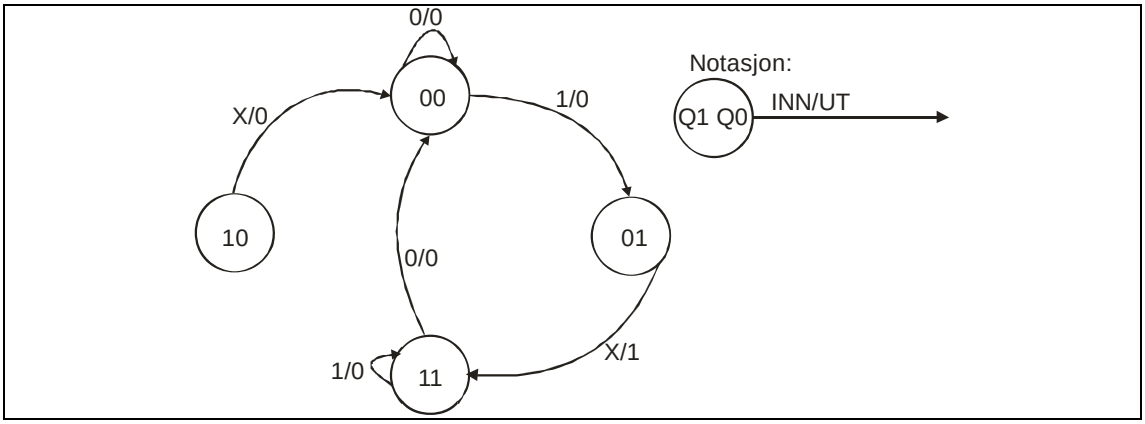
Q0(Neste) = J0 Q0' + K0' Q0 =
Q1' INN Q0' + (Q1 INN')' Q0 =
Q1' INN Q0' + (Q1' + INN) Q0 =
Q1' INN Q0' + Q1' Q0 + INN Q0 =
Q1' Q0' INN + Q1' Q0 + Q0 INN

Etter at alle likningene nå er satt opp, kan vi lage nestetilstandstabellen. Da vi har 2 vipper og en inngangsvariabel, vil vi totalt ha 8 rader. Nestetilstandene vil avhenge av både nåtilstander og inngangsverdier (dette ser vi utfra nestetilstandslikningene). Tabellen må også inneholde en kolonne for utgangsvariabelen. Merk at vi har skrevet likningene på en slik form (sortert sum av produkt) at det er så enkelt som mulig å finne hvilke rader som skal ha 0 og 1 for hver likning. Se tabell 3.5 for den endelige tabellen.

Nå			Neste		
Q1	Q0	INN	Q1	Q0	UT
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	1	0

Tabell 3.5: Nestetilstand- og utgangstabell for design med JK-vipper

For å komme fra nestetilstandstabellen til tilstandsdiagrammet, tar vi utgangspunkt i hver nåtilstand og noterer hvilken nestetilstand den vil skifte til for forskjellige inngangsverdier (i vårt tilfelle for INN=0 og INN=1). Ved å bruke notasjonen i fig 3.3, tegner vi hver nåtilstand og merker dem med tilstandsnavn og tilstandskoding. Pilene indikerer overganger til nestetilstander. I de tilfellene der det er overgang til samme nestetilstand for begge inngangsverdier, så skriver vi X på overgangspilen. Merk at utgangsverdien er knyttet til den tilstanden pilen kommer fra. Når tilstandsmaskinen er i tilstand Q1Q0 = 00 vil det altså stå 0 på utgangen, mens det står 1 på utgangen når tilstandsmaskinen er i tilstand Q1Q0 = 01. Siden dette er en Moore-maskin kunne vi også godt ha skrevet utgangsverdien inne i den enkelte tilstanden.



Figur 3.3: Tilstandsdiagram for JK-design, oppgave 2

Kommentar: Vi kan observere at kodingen av tilstander er utført i henhold til minimum bitendring. Dette gjør at kombinatorikken er relativt enkel. Dersom tilstanden som nå er kodet med Q1Q0 = 11 isteden hadde vært kodet med Q1Q0 = 10, så ville det gitt betydelig mer kombinatorikk. Se også oppgave 4 i denne øvingen.

c) Ved å studere tilstandsdiagrammet, ser vi at tilstandsmaskinen gir ut en puls på en klokkeperiodes varighet når den påtrykkes en puls av vilkårlig lengde. Dette er altså en positiv flanke detektor (positive edge detector, PED). Se i labheftet for lab 3 for en alternativ implementering av dette.

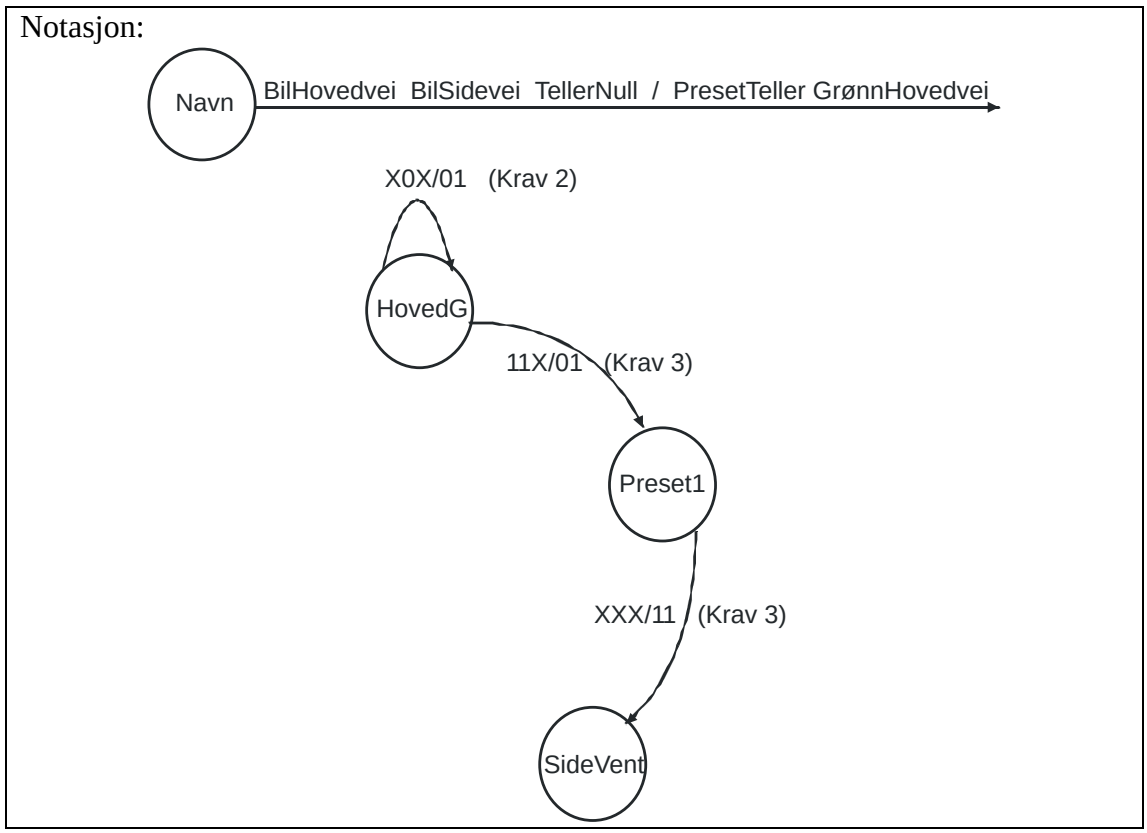
Oppgave 3: Design av tilstandsdiagram fra skriftlig spesifikasjon

Nedenfor beskrives den type vurderinger som du som designer må gå gjennom når du skal utvikle et tilstandsdiagram. Merk at det finnes mange forskjellige og like gode løsninger på dette. Blant annet er det her valgt å bruke en Moore-type tilstandsmaskin. Dette er gjort for å sikre at oppsett og holdetider overholdes for den synkrone telleren. Med en Mealy-type tilstandsmaskin kunne vi risikert at BilSidevei, og dermed PresetTeller, gikk høyt rett før en positiv klokkeflanke.

- Start med en initialtilstand GrønnH der det ikke er biler på sidevei og hovedveien derfor skal ha grønt lys kontinuerlig.
- I og med at vi har tre inngangssignaler, BilHovedvei, BilSidevei og TellerNull, har vi åtte mulig overganger ut fra GrønnH. Verdien på TellerNull kan vi imidlertid ignorere ettersom vi i denne tilstanden ikke har startet noen tidtakning.
- Så lenge BilSidevei er lav, kan vi dessuten også ignorere BilHovedvei, siden hovedveien skal ha grønt lys uavhengig av om det er biler der eller ikke. Dette gir en overgangspil tilbake til seg selv.
- En høy verdi på BilSidevei indikerer at vi må bevege oss bort fra tilstand HovedG. Dersom det også er bil på hovedveien på dette tidspunkt (BilHovedvei=1), skal sideveien vente en tid T før den får grønt. For å få til dette må PresetTeller aktiviseres. Siden vi lager en Moore-type tilstandsmaskin, krever dette overgang til en egen tilstand, Preset1, der PresetTeller settes høy. Fra Preset1 skal vi på neste positive klokkeflanke, uavhengig av inngangssignalene, flytte oss til en ny tilstand, SideVent, der vi venter på at telleren skal telle ned til null.

Se figur 3.4 for en beskrivelse av tilstandsdiagrammet så langt. (Krav #) angir hvilket kravnummer i spesifikasjonen den aktuelle overgangen er basert på. Merk at det er benyttet en

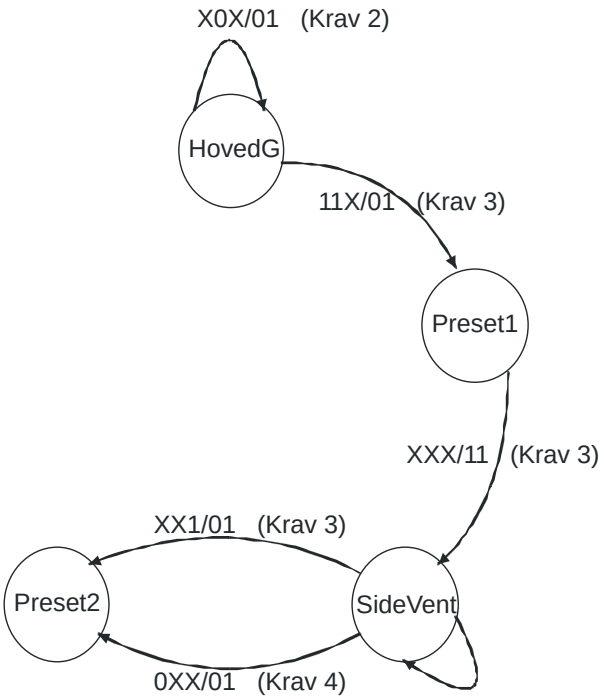
Mealy diagramnotasjon selv om maskinen til slutt viser seg å være av Moore type. Det samme er (som kjent) stort sett gjort i fagets forelesninger.



Figur 3.4 Del av tilstandsdiagram

- I tilstand SideVent, venter vi på at signalet TellerNull skal gå høyt, eller at det ikke lenger er biler på hovedveien (BilHovedvei=0).
- Dette gir oss en overgangspil tilbake til tilstanden selv dersom det fortsatt er bil på hovedveien og telleren ikke har nådd null enda.
- Det gir i tillegg to overgangspiler til en ny tilstand der vi skal gjøre oss klar til å gi sideveien grønt lys. En overgang er basert på at telleren når null (TellerNull=1) og den andre på at det ikke lenger er bil på hovedveien (BilHovedvei=0).
- Fordi sideveien bare skal ha grønt lys en tid T, har vi imidlertid i begge tilfellene behov for å resette telleren på nytt. Vi går derfor ikke direkte til en tilstand SideG, men går først til en tilstand Preset2.

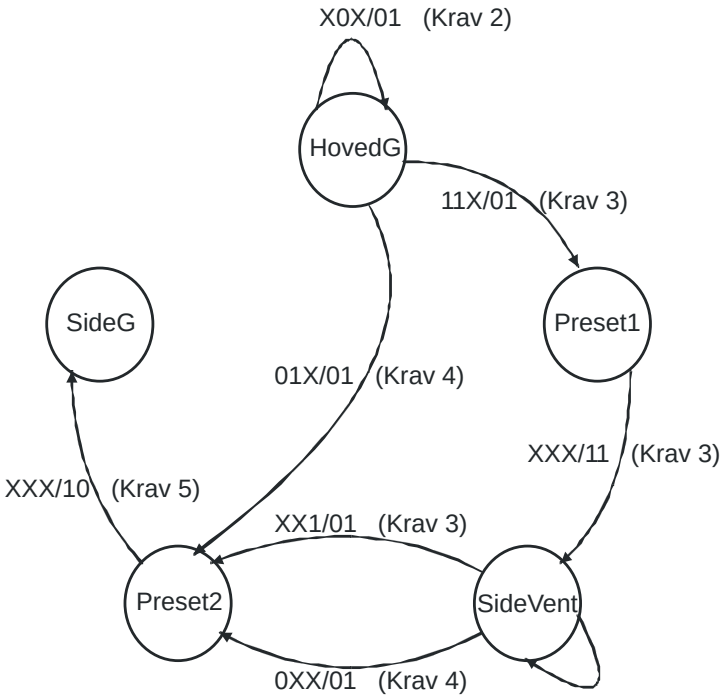
Se figur 3.5 for en beskrivelse av tilstandsdiagrammet så langt.



Figur 3.5 Del av tilstandsdiagram

- La oss nå gå tilbake til tilstand HovedG. Her er ikke alle mulige kombinasjoner av 0 og 1 på inngangene dekket enda. Dersom det kommer bil på sideveien (BilSidevei=1) uten at det er bil på hovedveien (BilHovedvei=0), så skal sideveien ikke innom ventеperioden. Det svarer til at vi får en direkte overgang fra tilstand HovedG til Preset2.
- Fra Preset2 skal vi på neste positive klokkeflanke, uavhengig av inngangssignalene, flytte oss til en ny tilstand, SideG, der vi gir sideveien grønt lys (GrønnHovedvei=0) og venter på at telleren skal telle ned til null eller at det skal bli tomt for biler på sideveien.

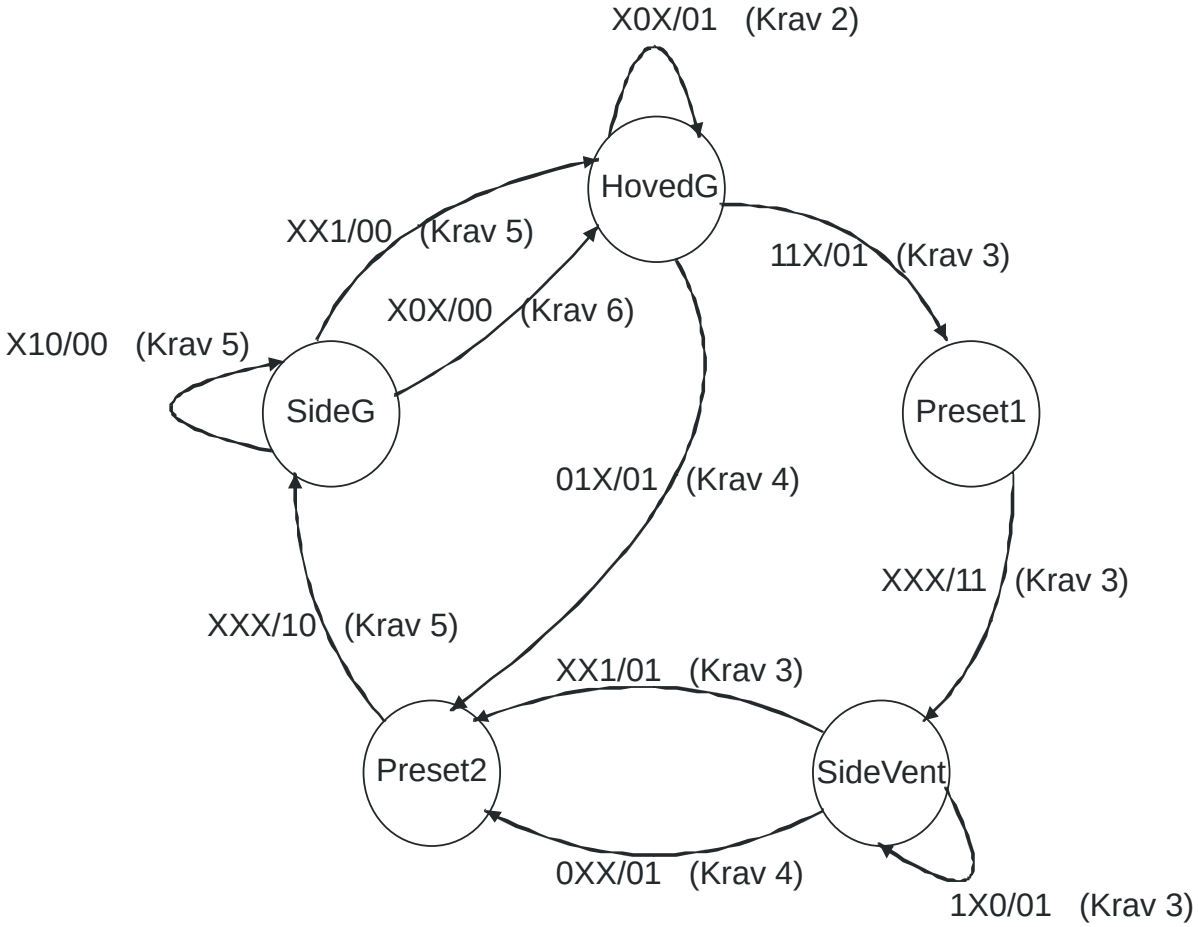
Se figur 3.6 for en beskrivelse av tilstandsdiagrammet så langt.



Figur 3.6 Del av tilstandsdiagram

- I tilstand SideG venter vi på at signalet TellerNull skal gå høyt, eller at det ikke lenger er biler på sideveien (BilSidevei=0).
- Dette gir oss en overgangspil tilbake til tilstanden selv dersom det fortsatt er bil på sideveien og telleren ikke har nådd null enda.
- Det gir i tillegg to overgangspiler tilbake til initialtilstanden HovedG, der hovedveien har grønt lys. En overgang er basert på at telleren når null (TellerNull=1) og den andre på at det ikke lenger er bil på sideveien (BilSidevei=0).

Se figur 3.7 for en komplett beskrivelse tilstandsdiagrammet.



Figur 3.7 Komplett tilstandsdiagram

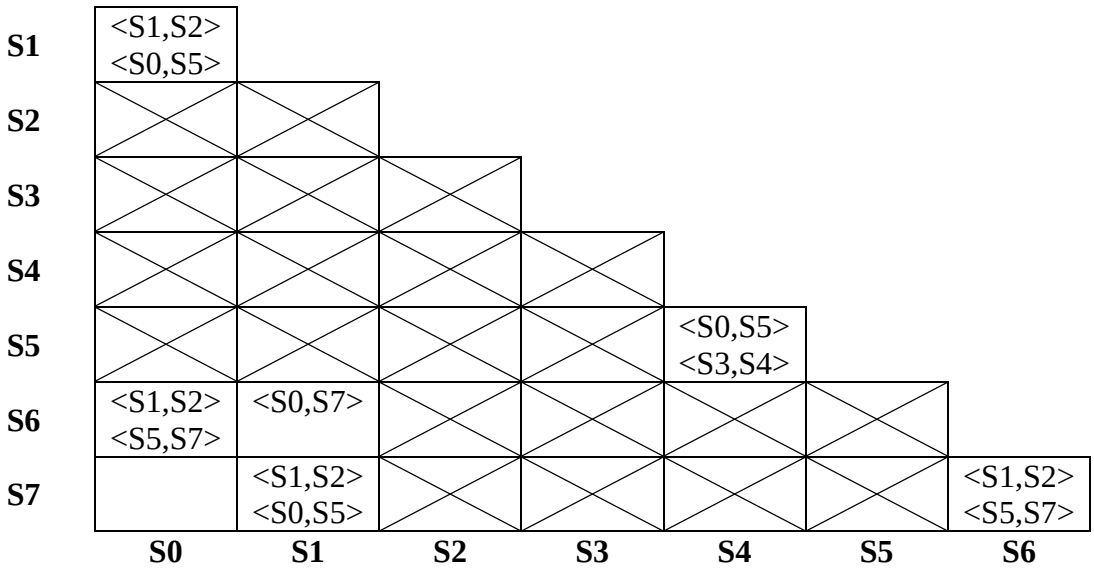
Oppgave 4: Design av tilstandsmaskin fra tilstandsdiagram

a) Nestetilstands- og utgangstabellen for tilstandsdiagrammet er gitt i tabell 3.6. Dette er en direkte oversettelse fra grafisk til tabellmessig framstilling, og inneholder ikke noen ny informasjon. For hver tilstand er utgangsverdiene gitt av det som er angitt inne i tilsvarende tilstandsboble. Nestetilstanden finner vi ved å følge overgangspilene.

Nåtilstand	I	Nestetilstand	Ut1	Ut0
S0	0	S1	0	0
S0	1	S5	0	0
S1	0	S2	0	0
S1	1	S0	0	0
S2	0	S3	1	1
S2	1	S6	1	1
S3	0	S4	0	1
S3	1	S2	0	1
S4	0	S5	1	0
S4	1	S3	1	0
S5	0	S0	1	0
S5	1	S4	1	0
S6	0	S2	0	0
S6	1	S7	0	0
S7	0	S1	0	0
S7	1	S5	0	0

Tabell 3.6: Nestetilstands- og utgangstabell for gitt tilstandsmaskin

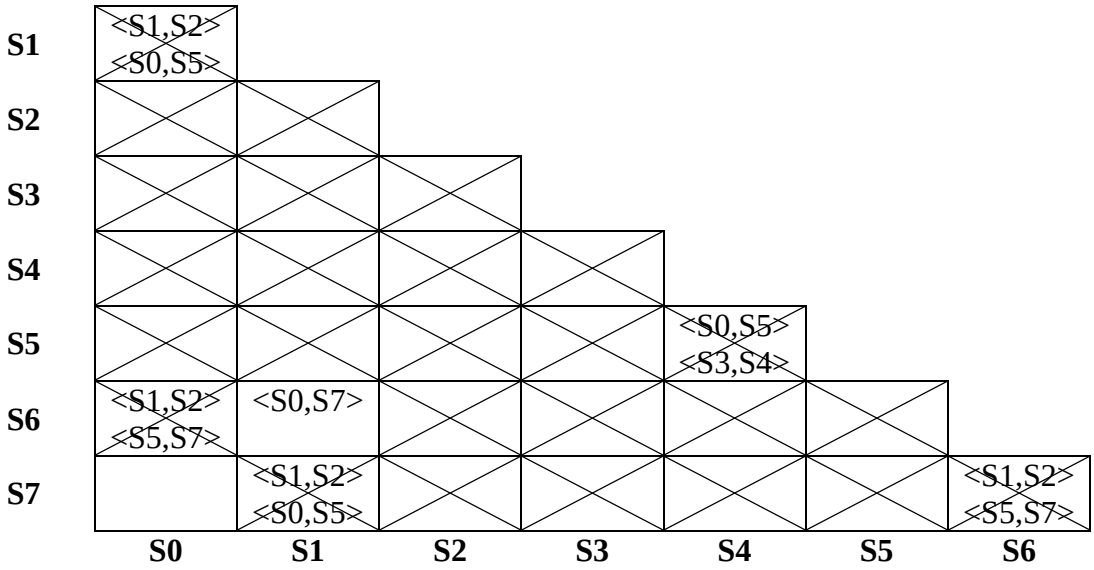
b) Vi bruker her implikasjonstabell, som angitt i Gajski, kapittel 6.10, side 248 - 250. Vi starter med å sette opp en "tom" tabell. Deretter fyller vi ut rutene i henhold til reglene nederst på side 248 i Gajski. Vi setter kryss i ruter der tilstandene har forskjellige utgangsverdier. Dette er for eksempel tilfelle for S0 og S2, siden vi har utgangsverdiene 00 for S0 og 11 for S2. I de tilfellene der tilstandene har samme utgangsverdier, ser vi på nestetilstandene for I=0 og I=1 og setter inn kravene til ekvivalens. For at for eksempel S0 og S1 skal være ekvivalente, må S1 og S2 være ekvivalente (nestetilstander når I=0) og S0 og S5 være ekvivalente (nestetilstander når I=1). S0 og S7 får ingen krav, siden disse allerede har samme nestetilstander for både I=0 og I=1.



Mulige kandidater for ekvivalente tilstander er parene <S0, S1>, <S0, S6>, <S0, S7>, <S1, S6>, <S1, S7>, <S4, S5> og <S6, S7>. Vi ser at:

- S0 og S7 er ekvivalente tilstander, da de har samme utgangsverdier og samme nestetilstander (for alle sett av inngangsverdier).

Etter å ha satt ekvivalens mellom S0 og S7, går vi gjennom tabellen på nytt. Vi setter nå X i alle ruter der minst et av kravene til ekvivalens ikke er oppfylt. Vi får for eksempel kryss i ruten mellom S0 og S1 fordi det allerede er kryss i ruten mellom S1 og S2 (og mellom S0 og S5).



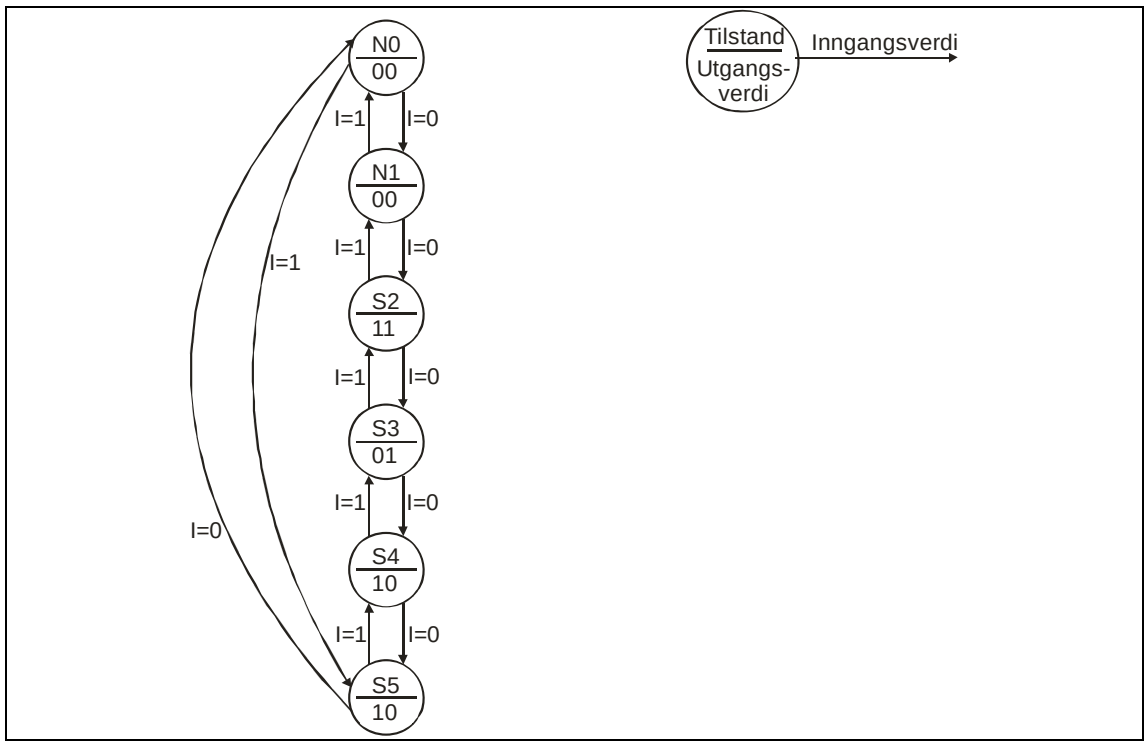
Vi setter ikke kryss i ruten mellom S1 og S6, siden kravet til ekvivalent nestetilstand her er oppfylt (S0 og S7 er ekvivalente). Dette gir oss følgelig en ny ekvivalent tilstand.

Vi definerer nå to nye tilstander:
N0 = <S0, S7>
N1 = <S1, S6>
Dette gir en forenklede nestetilstands- og utgangstabell som vist i tabell 3.6. Vi har fjernet den ene av hvert ekvivalent par, og omdøpt den andre til nytt navn. I resten av tabellen endrer vi alle henvisninger til S0 og S7 til N0, og alle henvisninger til S1 og S6 til N1.

Nåtilstand	I	Nestetilstand	Ut1	Ut0
N0	0	N1	0	0
N0	1	S5	0	0
N1	0	S2	0	0
N1	1	N0	0	0
S2	0	S3	1	1
S2	1	N1	1	1
S3	0	S4	0	1
S3	1	S2	0	1
S4	0	S5	1	0
S4	1	S3	1	0
S5	0	N0	1	0
S5	1	S4	1	0

Tabell 3.6: Nestetilstands- og utgangstabell for forenklet tilstandsmaskin

Dette gir forenklet tilstandsdiagram som vist i figur 3.5.



Figur 3.8: Tilstandsdiagram for forenklet tilstandsmaskin

c) Vi skal nå kode tilstandene med minimum-bit-change metoden. I dette tilfellet har vi en enkel struktur, der hver tilstand bare har overganger til to nabotilstander. Målet vårt blir da å kode tilstandene slik at to nabotilstander bare har ett bitt forskjell. Dette kan vi for eksempel oppnå ved å bruke følgende tilstandskoding:

N0 : 000 N1 : 001 S2 : 011 S3 : 111 S4 : 101 S5 : 100

Tabell 3.7 gir nestetilstands- og utgangstabell med de valgte tilstandkodene. De ubrukte kodene er også lagt inn. Siden det er et mål å få en så enkel logisk krets som mulig, er det plassert X for nestetilstand og Nåutganger for disse tilstandskodene. Radene i tabellen er stokket om i forhold til tabell 3.6, slik at de kommer i binær rekkefølge.

Nåtilstand	**Inngang**	**Nestetilstand**	**Nåutganger**	
$Q_2Q_1Q_0$	**I**	**$Q_2Q_1Q_0$**	**Ut1**	**Ut0**
000	0	001	0	0
000	1	100	0	0
001	0	011	0	0
001	1	000	0	0
010	X	XXX	X	X
011	0	111	1	1
011	1	001	1	1
100	0	000	1	0
100	1	101	1	0
101	0	100	1	0
101	1	111	1	0
110	X	XXX	X	X
111	0	101	0	1
111	1	011	0	1

Tabell 3.7: Nestetilstands- og utgangstabell for forenklet tilstandsmaskin

Med hot-one koding ville det vært behov for 6 vipper siden det trengs like mange vipper som tilstander.

d) Nedenfor følger Karnaugdiagram og forenklede likninger for de tre nestetilstandene og de to utgangene. Vi kunne fått færre transistorer ved bruk av NAND eller NOR porter. Dette er ikke gjort her.

Q_2Q_1 \ Q_0I	00	01	11	10
00	0	1	0	0
01	X	X	0	1
11	X	X	0	1
10	0	1	1	1

$D_2 = Q_2(\text{Neste}) = Q_0' \, I + Q_1 \, I' + Q_2 \, Q_1' \, Q_0$

Q_2Q_1 \ Q_0I	00	01	11	10
00	0	0	0	1
01	X	X	0	1
11	X	X	1	0
10	0	0	1	0

$D_1 = Q_1(\text{Neste}) = Q_2 \, Q_0 \, I + Q_2' \, Q_0 \, I' = Q_0 \, (Q_2 \odot I)$

Q_2Q_1 \ Q_0I	00	01	11	10
00	1	0	0	1
01	X	X	1	1
11	X	X	1	1
10	0	1	1	0

$D_0 = Q_0(\text{Neste}) = Q_1 + Q_2' \, I' + Q_2 \, I = Q_1 + (Q_2 \odot I)$

Q_2Q_1 \ Q_0I	00	01	11	10
00	0	0	0	0
01	X	X	1	1
11	X	X	0	0
10	1	1	1	1

$Ut_1 = Q_2' \, Q_1 + Q_2 \, Q_1' = Q_2 \oplus Q_1$

Q_2Q_1 \ Q_0I	00	01	11	10
00	0	0	0	0
01	X	X	1	1
11	X	X	1	1
10	0	0	0	0

$Ut_0 = Q_1$

e) Vi har to ubrukte tilstander $Q_2Q_1Q_0 = 010$ og $Q_2Q_1Q_0 = 110$. Vi setter hver av dem inn i likningene for nestetilstander for å finne nestetilstandene for disse:

$Q_2Q_1Q_0 = 010$:
$Q_2(\text{Neste}) = Q_0' I + Q_1 I' + Q_2 Q_1' Q_0 = 1 \bullet I + 1 \bullet I' + 0 = (I + I') = 1$
$Q_1(\text{Neste}) = Q_2 Q_0 I + Q_2' Q_0 I' = 0 \bullet 0 \bullet I + 1 \bullet 0 \bullet I' = 0$
$Q_0(\text{Neste}) = Q_1 + Q_2' I' + Q_2 I = 1 + 1 \bullet I' + 0 \bullet I = 1$

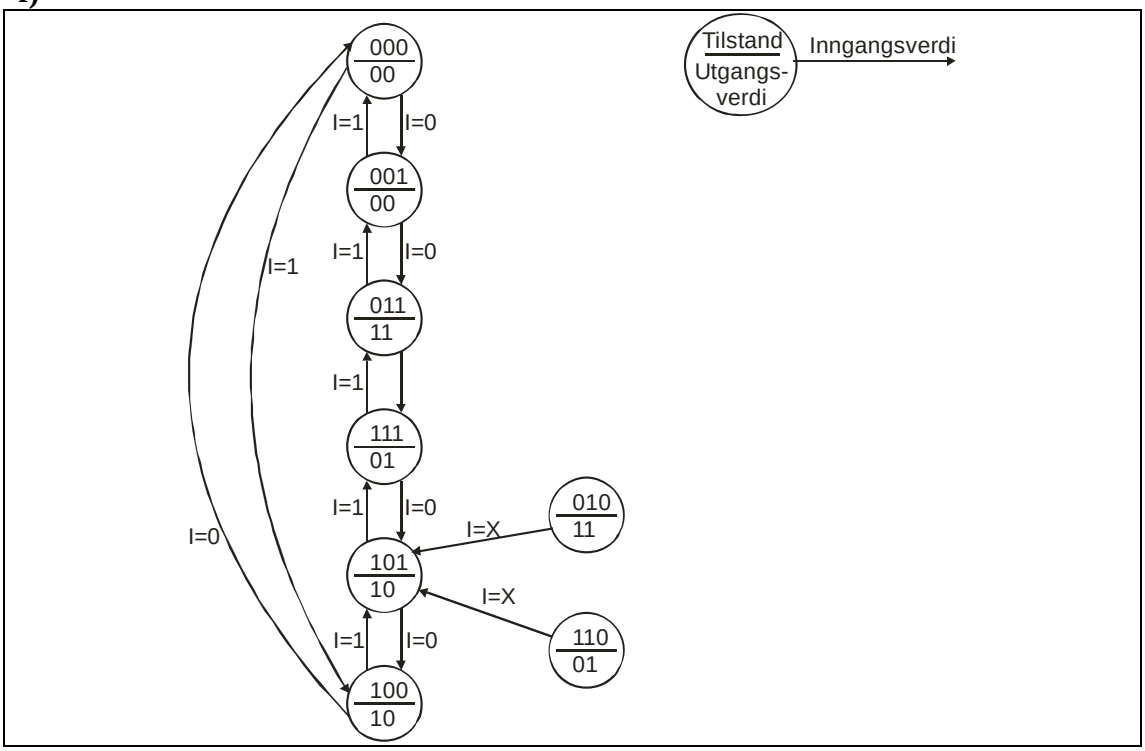
Dette gir altså den lovlige tilstanden $Q_2Q_1Q_0 = 101$

$Q_2Q_1Q_0 = 110$:
$Q_2(\text{Neste}) = Q_0' I + Q_1 I' + Q_2 Q_1' Q_0 = 1 \bullet I + 1 \bullet I' + 0 = (I + I') = 1$
$Q_1(\text{Neste}) = Q_2 Q_0 I + Q_2' Q_0 I' = 1 \bullet 0 \bullet I + 0 \bullet 0 \bullet I' = 0$
$Q_0(\text{Neste}) = Q_1 + Q_2' I' + Q_2 I = 1 + 0 \bullet I' + 1 \bullet I = 1$

Dette gir også den lovlige tilstanden $Q_2Q_1Q_0 = 101$

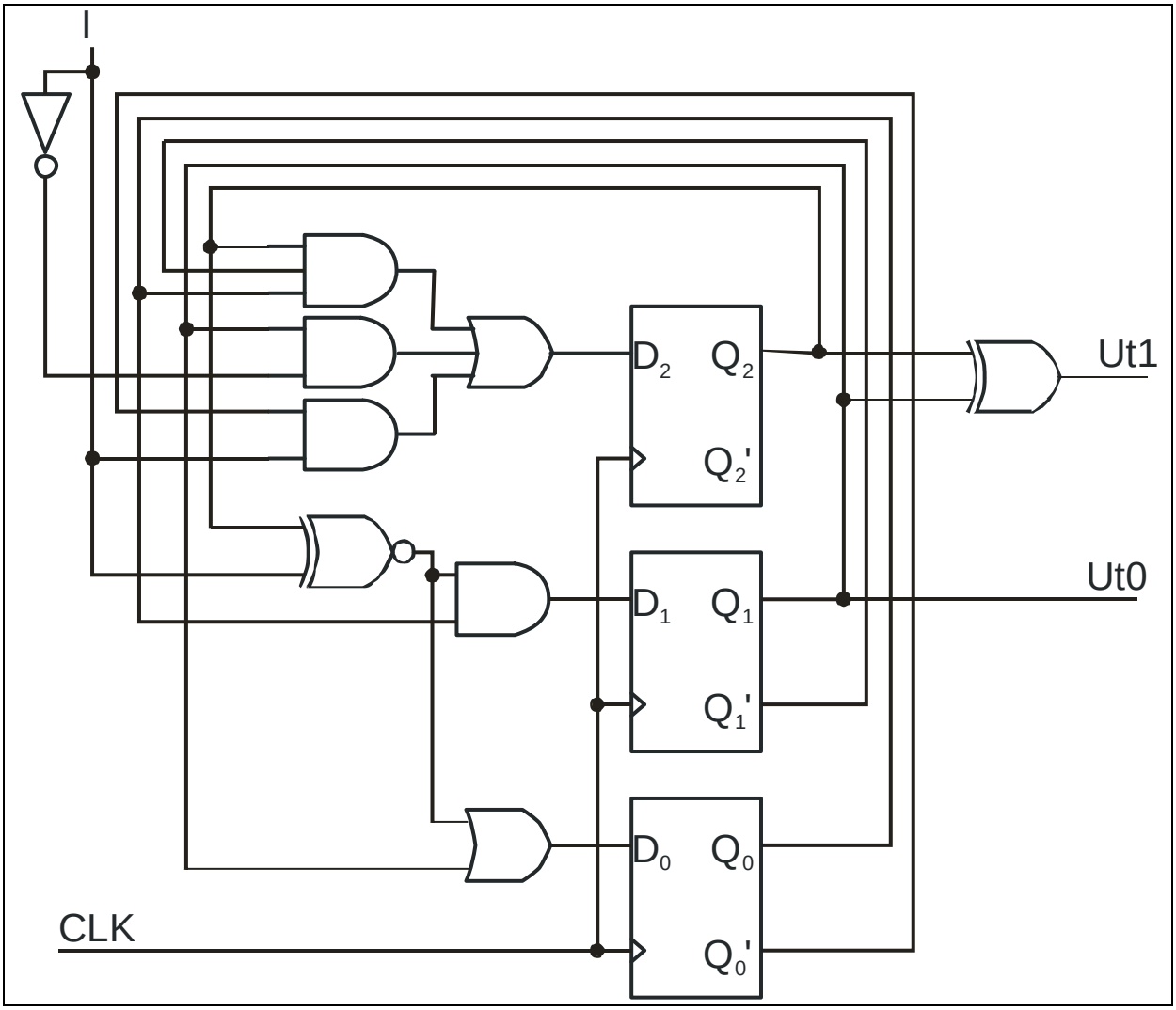
Her er det altså ikke fare for vranglås.

f)



Figur 3.9: Tilstandsdiagram med tilstandskoding og ulovlige tilstander.

g)



Figur 3.10: Logisk kretsskjema for tilstandsmaskinen

h) Klokkehastigheten bestemmes her av forsinkelse gjennom vippe + forsinkelse gjennom kombinatorikk fra vippeutgang til vippeinngang + oppsettid. Kritisk sti i kombinatorikk går gjennom en tre-inngangs AND-port og en tre-inngangs OR-port, alternativt gjennom en to-inngangs XNOR-port og en to-inngangs AND-port eller OR-port. Alle disse stiene har totalt 5,6ns forsinkelse. Det gir:

$$t_{(Q_i \rightarrow D_j)}\Big|_{max} = 2{,}8ns + 2{,}8ns = 5{,}6ns$$

Maksimal klokkefrekvens blir da

$$f_{clk}\Big|_{max} = \frac{1}{t_{pd(max)} + t_{(Q_i \rightarrow D_j)}\Big|_{max} + t_{setup}} = \frac{1}{4ns + 5{,}6ns + 2{,}8ns} = 80{,}6MHz$$