Dally, Harting and Aamodt:
Digital Design Using VHDL

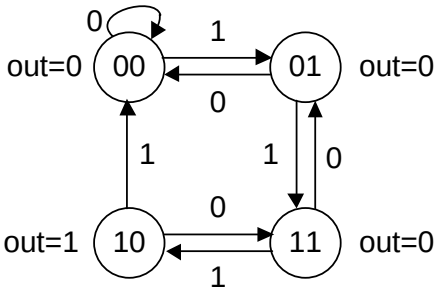
Kapitel 14

14.1 Vi har tillståndstabellen från *Tabell 14.2*, sida 308 som vi upprepar i *Figur 14.1a*.
Låt oss rita tillståndsgraf, *Figur 14.1b*.

	Next state		*out*	
	in		*in*	
State	*0*	*1*	*0*	*1*
00	00	01	0	0
01	00	11	0	0
11	01	10	0	0
10	11	00	0	1

Figur 14.1a

Vi söker en insignal som gör att vi alltid stegar tillbaka till starttillståndet 00 oberoende av nuvarande tillstånd. Vi ser att det inträffar om `in=0`.



Figur 14.1b

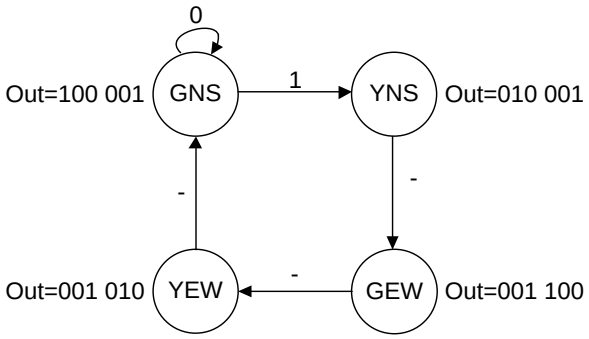
14.2 Vi har tillståndstabellen från *Tabell 14.4*, sida 311 som vi upprepar i *Figur 14.2a*.
Låt oss rita tillståndsgraf, *Figur 14.2b*.

	Next state		
	carew		
State	*0*	*1*	*Out*
GNS	GNS	YNS	100 001
YNS	GEW	GEW	010 001
GEW	YEW	YEW	001 100
YEW	GNS	GNS	001 010

Figur 14.2a

14.2 forts.

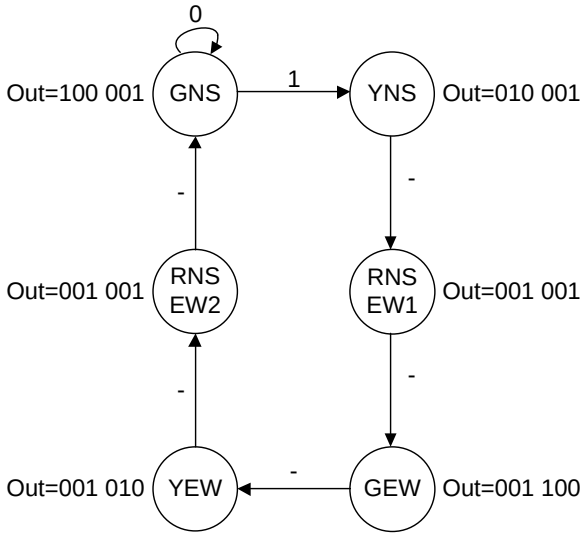
Vi söker en insignal som gör att vi alltid stegar tillbaka till starttillståndet GNS oberoende av nuvarande tillstånd. Vi ser att det inträffar om `carew=0`.



Figur 14.2b

14.3 Vi utgår från tillståndstabell och tillståndsgraf i *Uppgift 14.2*, dvs *Figur 14.2a* respektive *Figur 14.2b*. Våra utsignaler ligger enligt grön-gul-röd i nord-sydlig riktning följt av grön-gul-röd i öst-västlig riktning. Vi vill modifiera sekvensen så att det blir rött i båda riktningarna innan ett trafikljus går till grönt. Vi kan få det genom att lägga in tillstånd sär båda riktningar har rött mellan YNS och GEW samt mellan YEW och GNS, *Figur 14.3a*.

Vi får tillståndstabellen i *Figur 4.3b*.



Figur 14.3a

	Next state		
	carew		
State	*0*	*1*	*Out*
GNS	GNS	YNS	100 001
YNS	RNSEW1	RNSEW1	010 001
RNSEW1	GEW	GEW	001 001
GEW	YEW	YEW	001 100
YEW	RNSEW2	RNSEW2	001 010
RNSEW2	GNS	GNS	001 001

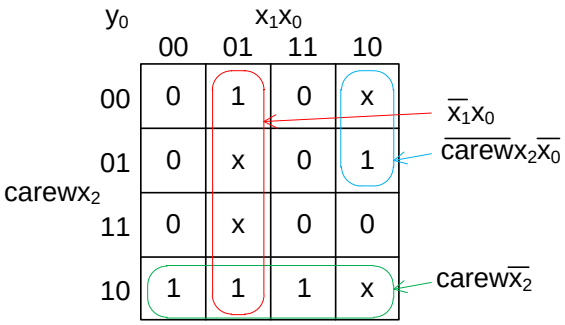
Figur 14.3b

14.4 Vi får ersätta tillståndsnamnen i *Figur 14.3b* med tillståndsvariabler och utsignalen behöver delas upp i sina individuella bitar. Vi har sex tillstånd så vi behöver tre tillståndsvariabler. Vi tilldelar tillstånd slumpmässigt och får *Figur 14.4a*.

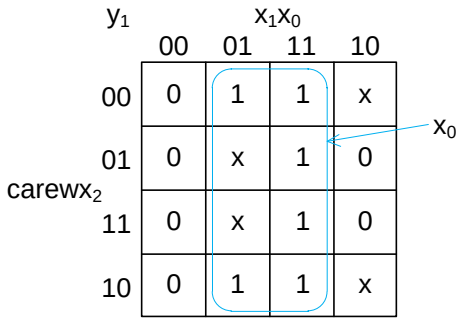
Vi ställer upp Karnaughdiagram för espektive signal.För tillstånden krävs diagram med fyra variabler medan utsignalerna kräver tre variabler. Då vi har ett förlopp uppifrån och ner i tabellen så är det förmodligen lämpligt att göra en tillståndstilldelning där vi rad för rad ändrar en tillståndsvariabel.

	Next state		
	carew		
State	*0*	*1*	*Out*
$x_2x_1x_0$	$y_2y_1y_0$	$y_2y_1y_0$	$z_5z_4z_3z_2z_1z_0$
000	000	001	100 001
001	011	011	010 001
011	111	111	001 001
111	110	110	001 100
110	100	100	001 010
100	000	000	001 001

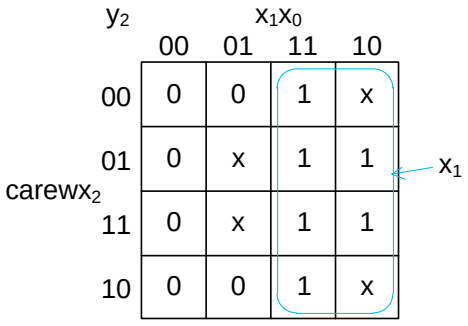
Figur 14.4a



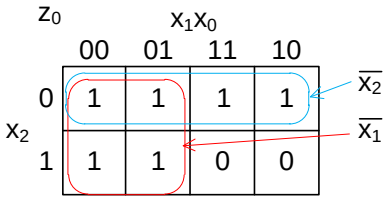
Figur 14.4b



Figur 14.4c

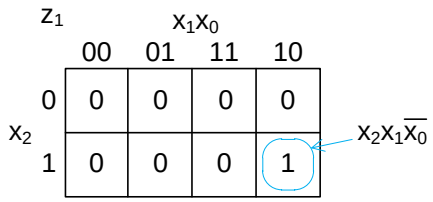


Figur 14.4d

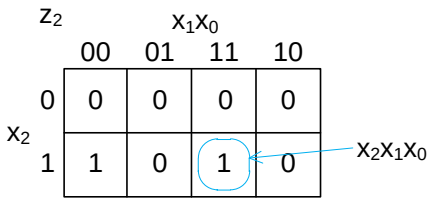


Figur 14.4e

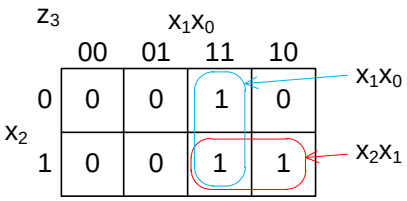
14.4 forts.



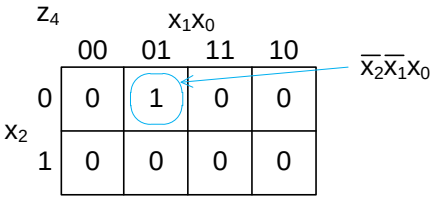
Figur 14.4f



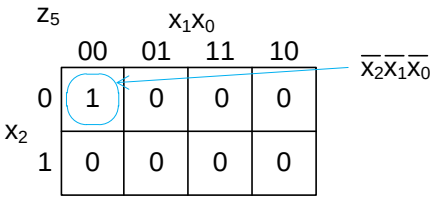
Figur 14.4g



Figur 14.4h



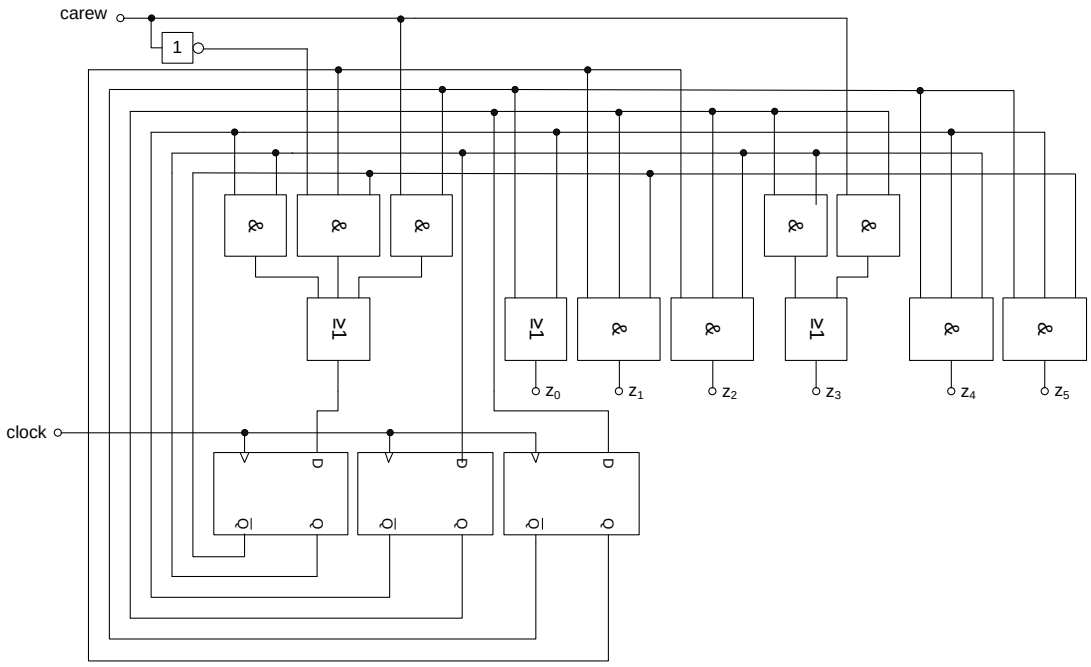
Figur 14.4i



Figur 14.4j

14.4 forts.

Vi får kopplingen i *Figur 14.4k*



Figur 14.4k

14.5 Vi skriver VHDLK-kod för trafikljuset i *Exempel 14.3*

```
-- ex_14_5.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ex14_5 IS
   PORT(Clock:IN STD_LOGIC;
        Resetn:IN STD_LOGIC;
        carew:IN STD_LOGIC;
        z:OUT STD_LOGIC_VECTOR(5 DOWNTO 0));
END ex14_5;

ARCHITECTURE arch_ex14_5 OF ex14_5 IS
   TYPE state_type IS (GNS,YNS,RNSEW1,GEW,YEW,RNSEW2);
   SIGNAL state_signal:state_type;
   SIGNAL next_state_signal:state_type;
BEGIN
```

14.5 forts.

```
state_transition_proc:
   PROCESS(Resetn,Clock)
   BEGIN
      IF (Resetn='0') THEN
         state_signal<=GNS;
      ELSIF rising_edge(Clock) THEN
         state_signal<=next_state_signal;
      END IF;
END PROCESS state_transition_proc;

stateflow_proc:
PROCESS(state_signal,carew)
BEGIN
   CASE state_signal IS
      WHEN GNS =>
         IF carew = '1' THEN
            next_state_signal <= YNS;
         ELSE
            next_state_signal <= GNS;
         END IF;
      WHEN YNS =>
         next_state_signal <= RNSEW1;
      WHEN RNSEW1 =>
         next_state_signal <= GEW;
      WHEN GEW =>
          next_state_signal <= YEW;
      WHEN YEW =>
          next_state_signal <= RNSEW2;
      WHEN RNSEW2 =>
          next_state_signal <= GNS;
    END CASE;
END PROCESS stateflow_proc;

assignment_proc:
PROCESS(state_signal,carew)
BEGIN
   CASE state_signal IS
      WHEN GNS =>
         z <= "100001";
      WHEN YNS =>
         z <= "010001";
      WHEN RNSEW1 =>
         z <= "001001";
```

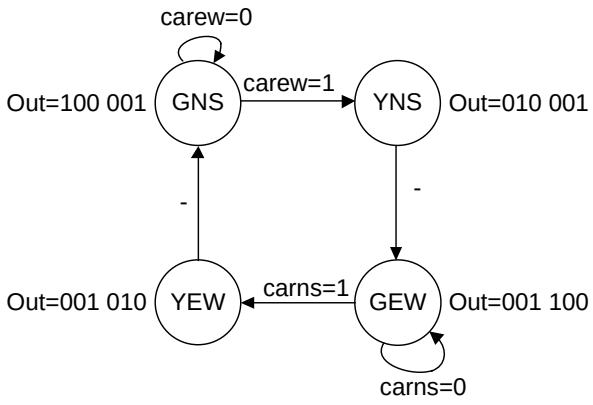
14.5 forts.

```
            WHEN GEW =>
                z <= "001100";
            WHEN YEW =>
                z <= "001010";
            WHEN RNSEW2 =>
                z <= "001001";
        END CASE;
    END PROCESS assignment_proc;
END arch_ex14_5;
```

Och simulerar med en do-fil

```
-- ex_14_5.do
restart -f -nowave
view signals wave
add wave Clock Resetn carew state_signal
add wave next_state_signal z
force Clock 0 0, 1 50ns -repeat 100ns
force carew 0
force Resetn 0
run 225ns
force Resetn 1
force carew 1
run 100ns
force carew 0
run 600ns
force carew 1
run 100ns
force carew 0
run 600ns
```

14.6 Vi gör en liten modifiering av tillståndsgrafen i *Uppgift 14.2*, dvs *Figur 14.2a* respektive *Figur 14.6b* och får *Figur 14.6a*. Vad som inte framgår av uppgiften är vad som skall hända om det finns bilar i båda riktningarna men vi antar att `carew` har prioritet i tillstånd GNS medan `carens` har prioritet i GEW.
Vi uppdaterar tillståndstabellen, *Figur 14.6b*



Figur 14.6a

14.6 forts.

State	Next state carew/carsn 00	01	10	11	Out
GNS	GNS	GNS	YNS	YNS	100 001
YNS	GEW	GEW	GEW	GEW	010 001
GEW	GEW	YEW	GEW	YEW	001 100
YEW	GNS	GNS	GNS	GNS	001 010

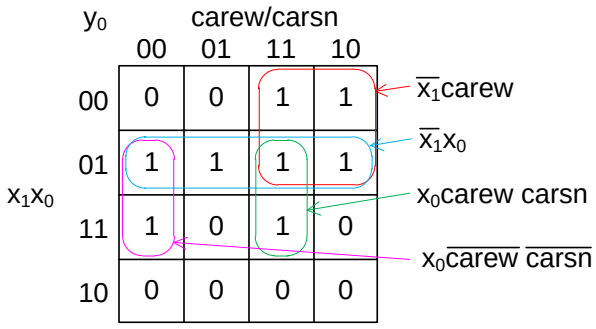
Figur 14.6b

14.7 Vi gör samma typ av tillståndstilldelning som i *Exempel 14.4* och låter bara en variabel ändra tillstånd i varje övergång, *Figur 14.7a*.

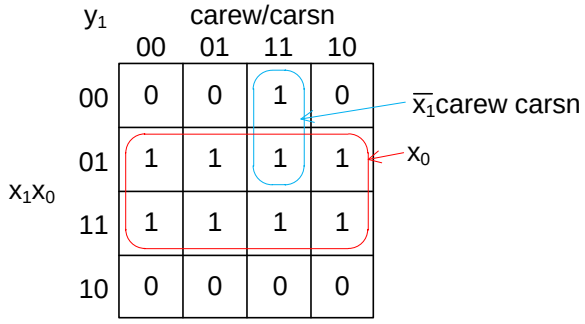
State	Next state carew/carsn 00	01	10	11	Out
x_1x_0	y_1y_0	y_1y_0	y_1y_0	y_1y_0	$z_5z_4z_3z_2z_1z_0$
00	00	00	01	01	100 001
01	11	11	11	11	010 001
11	11	10	11	10	001 100
10	00	00	00	00	001 010

Figur 14.7a

Vi ställer upp Karnaughdiagram, vi har fyra insignaler och behöver fyra-variablers diagram för tillstånden, *Figur 14.7b-c*, och två-variablers diagram för utsignalerna, *Figur 14.7d-i*.

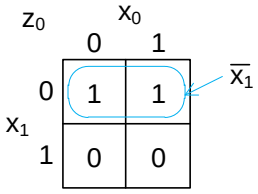


Figur 14.7b

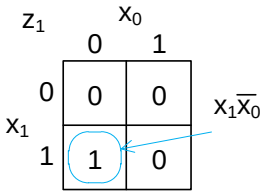


Figur 14.7c

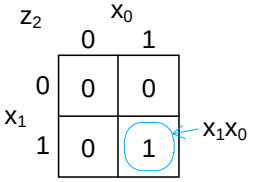
14.7 forts.



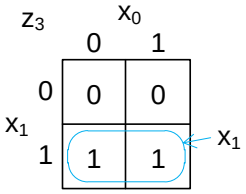
Figur 14.7d



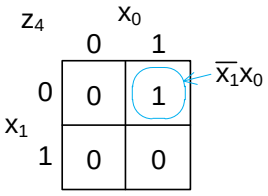
Figur 14.7e



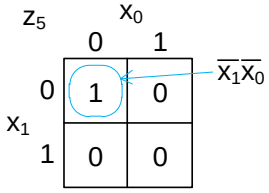
Figur 14.7f



Figur 14.7g

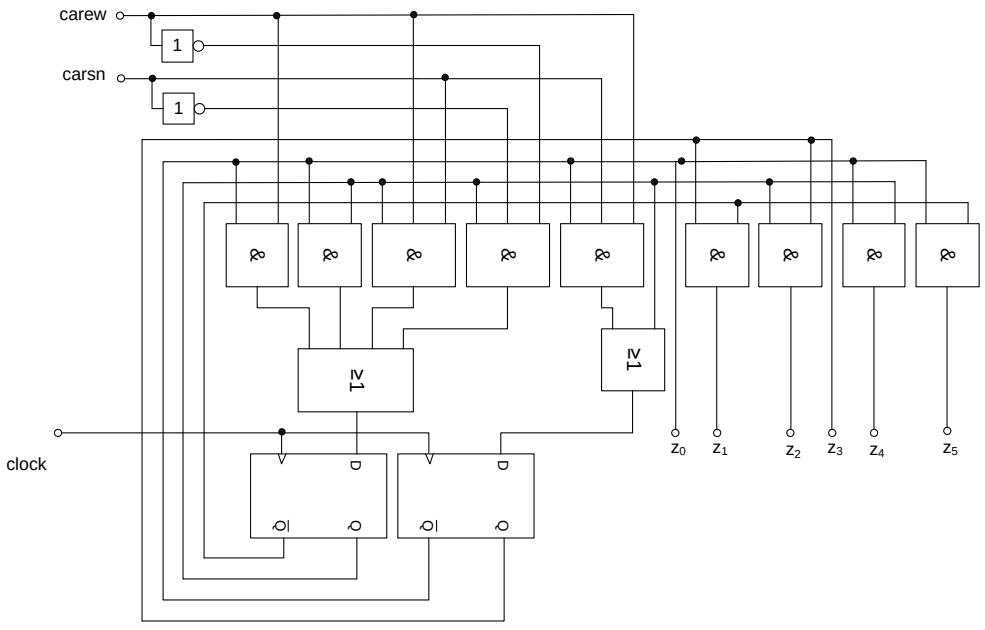


Figur 14.7h



Figur 14.7i

Vi får kopplingen i *Figur 14.7j*



Figur 14.7j

14.8 Vi skriver VHDL-kod

```
-- ex_14_8.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ex_14_8 IS
   PORT(Clock:IN STD_LOGIC;
        Resetn:IN STD_LOGIC;
        carew:IN STD_LOGIC;
        carsn:IN STD_LOGIC;
        z:OUT STD_LOGIC_VECTOR(5 DOWNTO 0));
END ex_14_8;

ARCHITECTURE arch_ex14_8 OF ex_14_8 IS
   TYPE state_type IS (GNS,YNS,GEW,YEW);
   SIGNAL state_signal:state_type;
   SIGNAL next_state_signal:state_type;
   BEGIN
   state_transition_proc:
      PROCESS(Resetn,Clock)
      BEGIN
         IF (Resetn='0') THEN
            state_signal<=GNS;
         ELSIF rising_edge(Clock) THEN
            state_signal<=next_state_signal;
         END IF;
   END PROCESS state_transition_proc;

   stateflow_proc:
   PROCESS(state_signal,carew)
   BEGIN
      CASE state_signal IS
         WHEN GNS =>
            IF carew = '1' THEN
               next_state_signal <= YNS;
            ELSE
               next_state_signal <= GNS;
            END IF;
         WHEN YNS =>
            next_state_signal <= GEW;
         WHEN GEW =>
             IF carsn= '1' THEN
             next_state_signal <= YEW;
```

14.8 forts.

```
               ELSE
                  next_state_signal <= GEW;
                END IF;
             WHEN YEW =>
                 next_state_signal <= GNS;
          END CASE;
       END PROCESS stateflow_proc;

       assignment_proc:
       PROCESS(state_signal,carew)
       BEGIN
          CASE state_signal IS
             WHEN GNS =>
                z <= "100001";
             WHEN YNS =>
                z <= "010001";
             WHEN GEW =>
                 z <= "001100";
             WHEN YEW =>
                 z <= "001010";
          END CASE;
       END PROCESS assignment_proc;
    END arch_ex14_8;
```

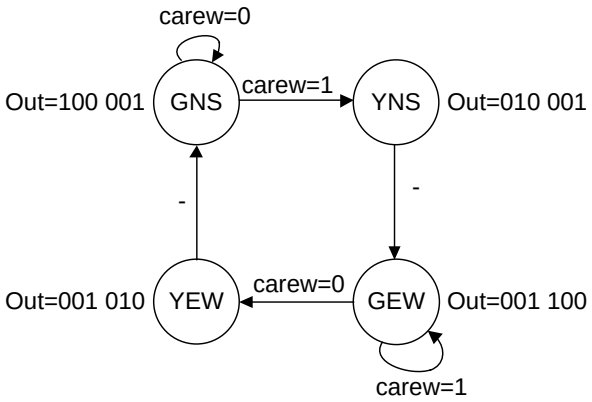
Vi simulerar med en do-fil

```
-- ex_14_8.do
restart -f -nowave
view signals wave
add wave Clock Resetn carew carsn
add wave state_signal next_state_signal z
force Clock 0 0, 1 50ns -repeat 100ns
force carew 0
force carsn 0
force Resetn 0
run 225ns
force Resetn 1
force carew 1
run 100ns
force carew 0
run 600ns
force carew 1
run 100ns
force carew 0
```

14.8 forts.

```
force carsn 1
run 600ns
force carsn 0
run 600ns
```

14.9 Vi gör en liten modifiering av tillståndsgrafen i *Uppgift 14.2*, dvs *Figur 14.2a* respektive *Figur 14.6b* och får *Figur 14.9a*.
Vi uppdaterar tillståndstabellen, *Figur 14.9b*



Figur 14.9a

	Next state		
	carew		
State	*0*	*1*	*Out*
GNS	GNS	YNS	100 001
YNS	GEW	GEW	010 001
GEW	YEW	GEW	001 100
YEW	GNS	GNS	001 010

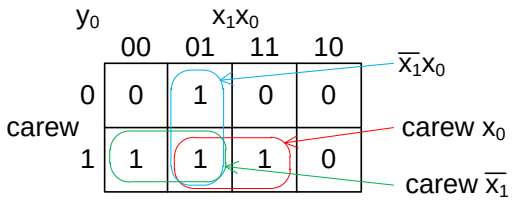
Figur 14.9b

14.10 Vi gör samma typ av tillståndstilldelning som i *Exempel 14.4* och låter bara en variabel ändra tillstånd i varje övergång, *Figur 14.10a*.
Vi ställer upp Karnaughdiagram, vi har fyra insignaler och behöver tre-variablers diagram för tillstånden, *Figur 14.10b-c*, och två variablers diagram för utsignalerna, *Figur 14.7d-i*. Utsignallogiken förändras inte från *Exempel 14.7*.

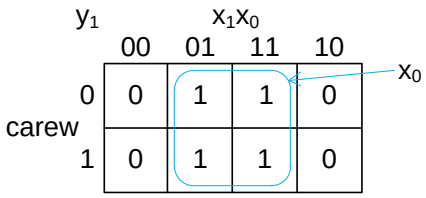
	Next state		
	carew		
State	*0*	*1*	*Out*
x_1x_0	y_1y_0	y_1y_0	$z_5z_4z_3z_2z_1z_0$
00	00	01	100 001
01	11	11	010 001
11	10	11	001 100
10	00	00	001 010

Figur 14.10a

14.10 forts.

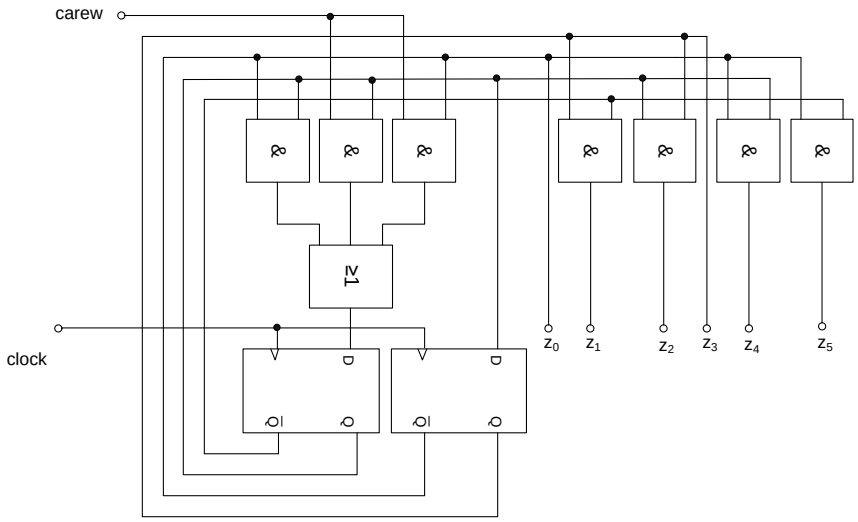


Figur 14.10b



Figur 14.10c

Vi får kopplingen i *Figur 14.10d*



Figur 14.10d

14.11 Vi skriver VHDL-kod

```
-- ex_14_11.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ex_14_11 IS
   PORT(Clock:IN STD_LOGIC;
        Resetn:IN STD_LOGIC;
        carew:IN STD_LOGIC;
        z:OUT STD_LOGIC_VECTOR(5 DOWNTO 0));
END ex_14_11;
```

14.11 forts.

```
ARCHITECTURE arch_ex_14_11 OF ex_14_11 IS
   TYPE state_type IS (GNS,YNS,GEW,YEW);
   SIGNAL state_signal:state_type;
   SIGNAL next_state_signal:state_type;
   BEGIN
   state_transition_proc:
      PROCESS(Resetn,Clock)
      BEGIN
         IF (Resetn='0') THEN
            state_signal<=GNS;
         ELSIF rising_edge(Clock) THEN
            state_signal<=next_state_signal;
         END IF;
   END PROCESS state_transition_proc;

   stateflow_proc:
   PROCESS(state_signal,carew)
   BEGIN
      CASE state_signal IS
         WHEN GNS =>
            IF carew = '1' THEN
               next_state_signal <= YNS;
            ELSE
               next_state_signal <= GNS;
            END IF;
         WHEN YNS =>
            next_state_signal <= GEW;
         WHEN GEW =>
             IF carew= '1' THEN
            next_state_signal <= GEW;
          ELSE
            next_state_signal <= YEW;
            END IF;
         WHEN YEW =>
             next_state_signal <= GNS;
       END CASE;
   END PROCESS stateflow_proc;

   assignment_proc:
   PROCESS(state_signal,carew)
   BEGIN
      CASE state_signal IS
         WHEN GNS =>
            z <= "100001";
```

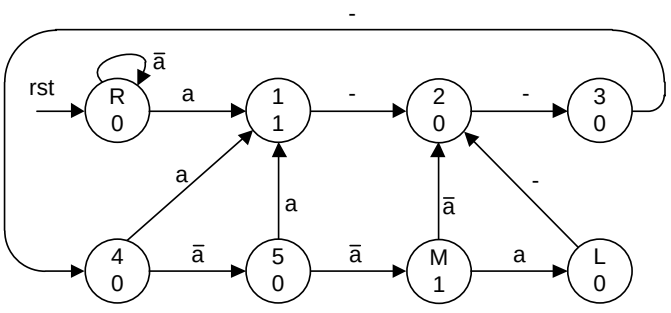
14.11 forts.

```
                WHEN YNS =>
                   z <= "010001";
                WHEN GEW =>
                    z <= "001100";
                WHEN YEW =>
                    z <= "001010";
            END CASE;
        END PROCESS assignment_proc;
    END arch_ex_14_11;
```

och simulerar med en do-fil

```
-- ex_14_11.do
restart -f -nowave
view signals wave
add wave Clock Resetn carew
add wave state_signal next_state_signal z
force Clock 0 0, 1 50ns -repeat 100ns
force carew 0
force Resetn 0
run 225ns
force Resetn 1
force carew 1
run 100ns
force carew 0
run 600ns
force carew 1
run 600ns
force carew 0
run 600ns
```

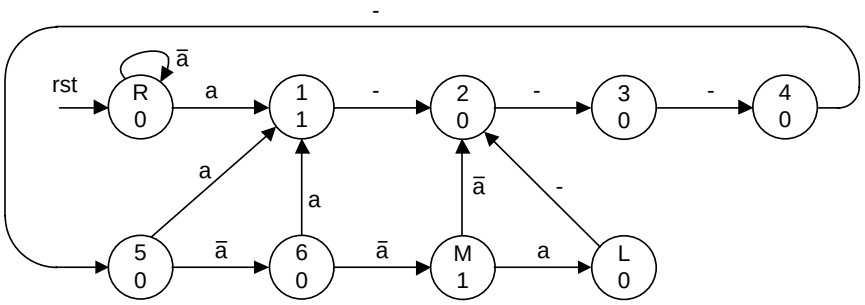
14.12 Vi utgår från tillståndsgrafen i *Figur 14.9*, sidan 312, som vi upprepar i *Figur 14.12a*



Figur 14.12a

14.12 forts.

För att utöka räknecykeln till sex klockpulser så får vi lägga in et extra tillstånd i kedjan R – S3. Vi får också korrigera vägarna då en puls saknas eller är sen. Vi får *Figur 14.12b*



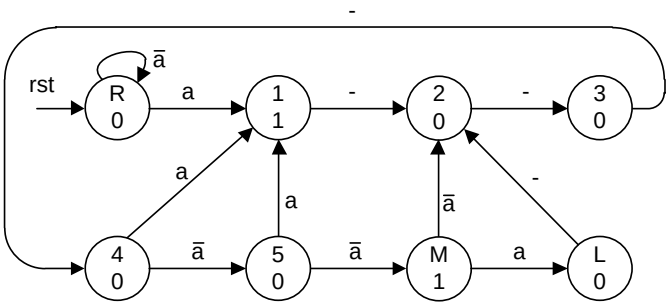
Figur 14.12b

Vi tecknar tillståndstabell, *Figur 14.12c*

	a		
State	*0*	*1*	*z*
R	R	S1	0
S1	S2	S2	1
S2	S3	S3	0
S3	S4	S4	0
S4	S5	S5	0
S5	S5	S1	0
S6	M	S1	0
M	S2	L	1
L	S2	S2	0

14.13 Vi har tillståndsgrafen i *Figur 14.9*, sidan 312, som vi upprepar i *Figur 14.13a*

14.13 forts.



Figur 14.13a

Vi tecknar tillståndstabell, *Figur 14.13b*

	a		
State	*0*	*1*	*z*
R	R	S1	0
S1	S2	S2	1
S2	S3	S3	0
S3	S4	S4	0
S4	S5	S1	0
S5	M	S1	0
M	S2	L	1
L	S2	S2	0

Figur 14.13b

14.14 Vi ska nu göra om tillståndstabellen i *Figur 14.13b* med, *Figur 14.14a*.

	a		
State	*0*	*1*	
$x_2x_1x_0$	$y_2y_1y_0$	$y_2y_1y_0$	*z*
000	000	001	0
001	011	011	1
011	010	010	0
010	110	110	0
110	111	001	0
111	101	001	0
101	011	100	1
100	011	011	0

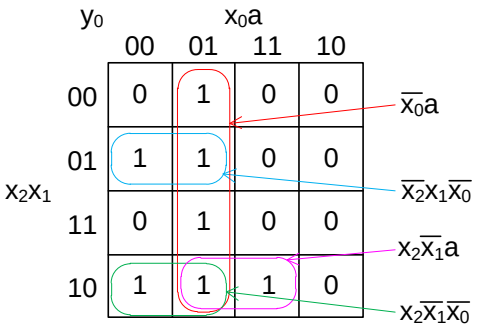
Figur 14.14a

Vi har tyvärr fyra diagonaler kvar där mer än en variabel byter värde. Vi kan nog inte eliminera det utan att dela upp tillstånd.

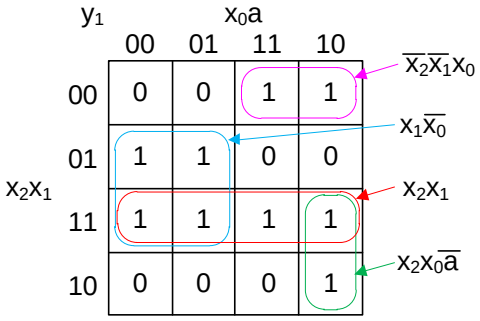
14.15 Vi ska nu använda en annan uppsättning tillståndsvariabler, *Figur 14.15a*.
Vi skall ta fram de logiska uttrycken och använder Karnaughdiagram. För tillstånden så har vi fyra variabler så vi behöver fyra-variablers Karnughdiagram, *Figur 14.15b-d* medan vi behöver ett tre-variablers diagram för utsignalen, *Figur 14.15e*

	a		
State	*0*	*1*	
$x_2x_1x_0$	$y_2y_1y_0$	$y_2y_1y_0$	*z*
000	000	001	0
001	010	010	1
010	011	011	0
011	100	100	0
100	101	001	0
101	110	001	0
110	010	111	1
111	010	010	0

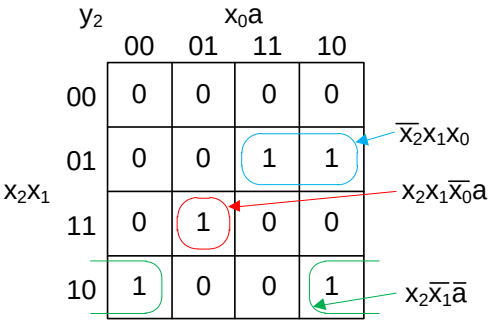
Figur 14.15a



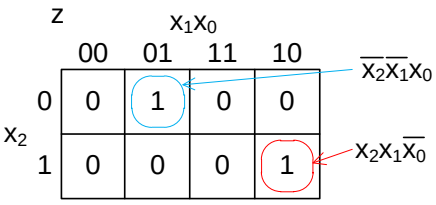
Figur 14.15d



Figur 14.15d



Figur 14.15d



Figur 14.15d

$$y_0 = x_0 \cdot a + \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot a + x_2 \cdot \overline{x_1} \cdot \overline{x_0}$$

$$y_1 = \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_1 \cdot \overline{x_0} + x_2 \cdot x_1 + x_2 \cdot x_0 \cdot \bar{a}$$

$$y_2 = \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot x_1 \cdot \overline{x_0} \cdot a + x_2 \cdot \overline{x_1} \cdot \bar{a}$$

$$z = \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot \overline{x_0}$$

14.16 Vi ska nu göra en one-hot tillståndstilldelning av FSM:n från Exempel 14.12. Vi har åtta tillstånd så vi behöver åtta tillståndsvariabler samt en variabel för utsignalen. Vi får *Figur 14.16a*.
Vi kan inte söka en bra lösning på detta utan att använda optimeringsverktyg.

State	*a*		*z*
	0	*1*	
$x_7x_6x_5x_4x_3x_2x_1x_0$	$y_7y_6y_5y_4y_3y_2y_1y_0$	$y_7y_6y_5y_4y_3y_2y_1y_0$	
00000001	00000001	00000010	0
00000010	00000100	00000100	1
00000100	00001000	00001000	0
00001000	00010000	00010000	0
00010000	00100000	00000010	0
00100000	01000000	00000010	0
01000000	00000100	10000000	1
10000000	00000100	00000100	0

Figur 14.16a

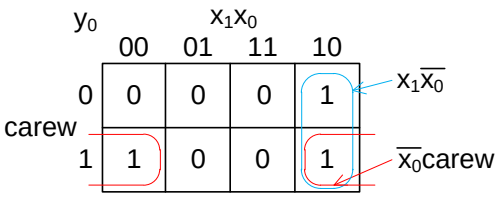
14.17 Vi har tillståndstabellen för trafikljuset från *Tabell 14.4*, sida 311 som vi upprepar i *Figur 14.17a*.
Vi inför tillståndskodning, Figur 14.17b.
För tillstånden har vi tre variabler varför vi behöver tre-variablers Karnaughdiagram, *Figur 14.17c-d*, medan utsignalerna bestäms av bara två variabler så där behöver vi två-variablers Karnaughdigram, *Figur 14.17e-j*.

State	*Next state carew*		*Out*
	0	*1*	
GNS	GNS	YNS	100 001
YNS	GEW	GEW	010 001
GEW	YEW	YEW	001 100
YEW	GNS	GNS	001 010

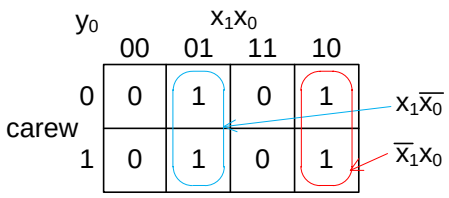
Figur 14.17a

State	*Next state carew*		*Out*
	0	*1*	
x_1x_0	y_1y_0	y_1y_0	$z_5z_4z_3z_2z_1y_0$
00	00	01	100 001
01	10	10	010 001
10	11	11	001 100
11	00	00	001 010

Figur 14.17a

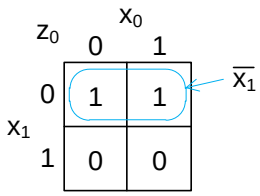


Figur 14.17c

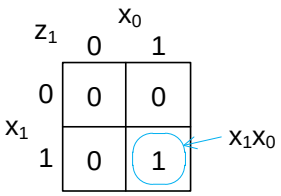


Figur 14.17d

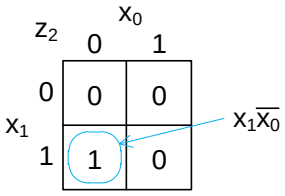
14.17 forts.



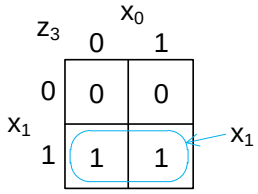
Figur 14.17e



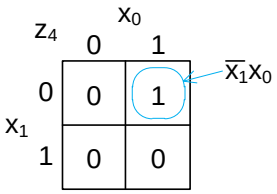
Figur 14.17f



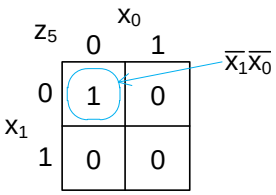
Figur 14.17g



Figur 14.17h

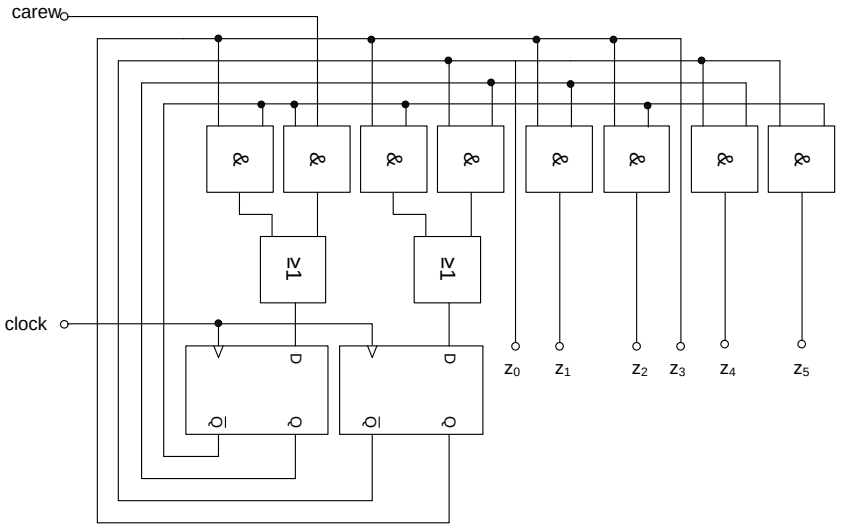


Figur 14.17i



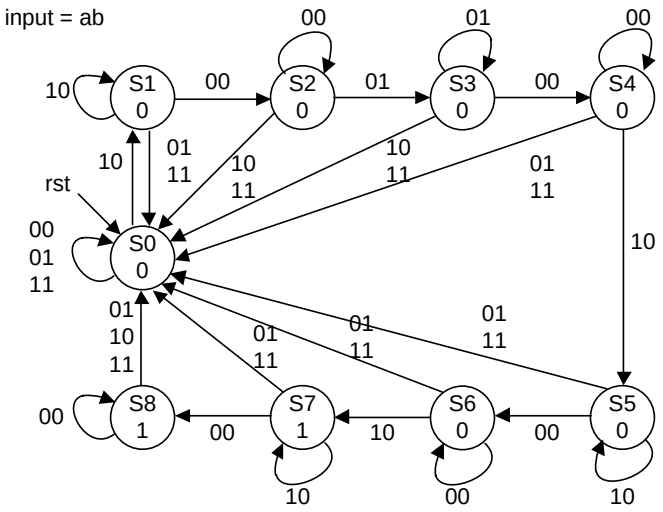
Figur 14.17j

Vi får kopplingen i *Figur 14.17k*



Figur 14.17k

14.18 Om vi tar med alla eventualiteter, även att både a och b är höga så får vi *Figur 14.18a*



Figur 14.18

och vi får tillståndstabellen i *Figur 4.18b*

	Next state				
	ab				
State	*00*	*01*	*10*	*11*	*UL*
S0	S0	S0	S1	S0	0
S1	S2	S0	S1	S0	0
S2	S2	S3	S0	S0	0
S3	S4	S3	S0	S0	0
S4	S4	S0	S5	S0	0
S5	S6	S0	S5	S0	0
S6	S6	S0	S7	S0	0
S7	S8	S0	S7	S0	1
S8	S8	S0	S0	S0	1

Figur 14.18b

14.19 Vi skriver VHDL-kod

```
-- ex_14_19.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

14.19 forts.

```
ENTITY ex_14_19 IS
   PORT(Clock:IN STD_LOGIC;
        Resetn:IN STD_LOGIC;
        a:IN STD_LOGIC;
        b:IN STD_LOGIC;
        z:OUT STD_LOGIC);
END ex_14_19;

ARCHITECTURE arch_ex_14_19 OF ex_14_19 IS
   TYPE state_type IS (S0,S1,S2,S3,S4,S5,S6,S7,S8);
   SIGNAL state_signal:state_type;
   SIGNAL next_state_signal:state_type;
   SIGNAL ab:STD_LOGIC_VECTOR(1 DOWNTO 0);
   BEGIN
      ab <= a & b;

   state_transition_proc:
      PROCESS(Resetn,Clock)
      BEGIN
         IF (Resetn='0') THEN
            state_signal<=S0;
         ELSIF rising_edge(Clock) THEN
            state_signal<=next_state_signal;
         END IF;
   END PROCESS state_transition_proc;

   stateflow_proc:
   PROCESS(state_signal,a,b)
   BEGIN
      CASE state_signal IS
         WHEN S0 =>
            IF ab = "00" THEN
               next_state_signal <= S0;
            ELSIF ab = "01" THEN
               next_state_signal <= S0;
            ELSIF ab = "10" THEN
               next_state_signal <= S1;
            ELSE
               next_state_signal <= S0;
            END IF;
         WHEN S1 =>
         IF ab = "00" THEN
            next_state_signal <= S2;
```

14.19 forts.

```
ELSIF ab = "01" THEN
   next_state_signal <= S0;
ELSIF ab = "10" THEN
   next_state_signal <= S1;
ELSE
   next_state_signal <= S0;
   END IF;
WHEN S2 =>
IF ab = "00" THEN
   next_state_signal <= S2;
ELSIF ab = "01" THEN
   next_state_signal <= S3;
ELSIF ab = "10" THEN
   next_state_signal <= S0;
ELSE
   next_state_signal <= S0;
   END IF;
WHEN S3 =>
IF ab = "00" THEN
   next_state_signal <= S4;
ELSIF ab = "01" THEN
   next_state_signal <= S3;
ELSIF ab = "10" THEN
   next_state_signal <= S0;
ELSE
   next_state_signal <= S0;
   END IF;
WHEN S4 =>
IF ab = "00" THEN
   next_state_signal <= S4;
ELSIF ab = "01" THEN
   next_state_signal <= S0;
ELSIF ab = "10" THEN
   next_state_signal <= S5;
ELSE
   next_state_signal <= S0;
   END IF;
WHEN S5 =>
IF ab = "00" THEN
   next_state_signal <= S6;
ELSIF ab = "01" THEN
   next_state_signal <= S0;
ELSIF ab = "10" THEN
   next_state_signal <= S5;
```

14.19 forts.

```
            ELSE
               next_state_signal <= S0;
               END IF;
            WHEN S6 =>
            IF ab = "00" THEN
               next_state_signal <= S6;
            ELSIF ab = "01" THEN
               next_state_signal <= S0;
            ELSIF ab = "10" THEN
               next_state_signal <= S7;
            ELSE
               next_state_signal <= S0;
               END IF;
            WHEN S7 =>
            IF ab = "00" THEN
               next_state_signal <= S8;
            ELSIF ab = "01" THEN
               next_state_signal <= S0;
            ELSIF ab = "10" THEN
               next_state_signal <= S7;
            ELSE
               next_state_signal <= S0;
               END IF;
            WHEN S8 =>
            IF ab = "00" THEN
               next_state_signal <= S8;
            ELSIF ab = "01" THEN
               next_state_signal <= S0;
            ELSIF ab = "10" THEN
               next_state_signal <= S0;
            ELSE
               next_state_signal <= S0;
               END IF;
         END CASE;
      END PROCESS stateflow_proc;

      assignment_proc:
      PROCESS(state_signal,a,b)
      BEGIN
         CASE state_signal IS
            WHEN S7 =>
               z <= '1';
            WHEN S8 =>
               z <= '1';
```

14.19 forts.

```
            WHEN OTHERS =>
                z <= '0';
        END CASE;
    END PROCESS assignment_proc;
END arch_ex_14_19;
```

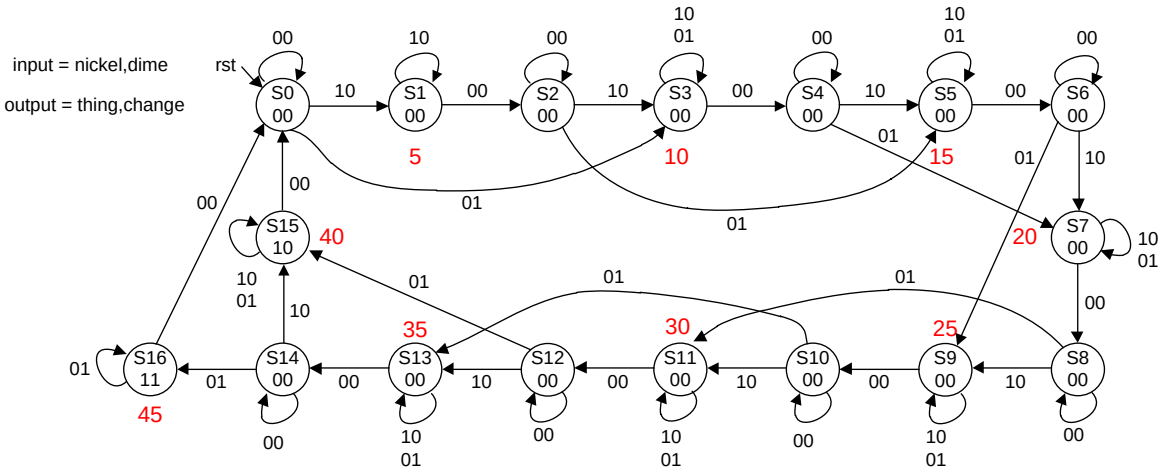
och simulerar med en do-fil

```
-- ex_14_19.do
restart -f -nowave
view signals wave
add wave Clock Resetn a b
add wave state_signal next_state_signal z
force Clock 0 0, 1 50ns -repeat 100ns
force a 0
force b 0
force Resetn 0
run 225ns
force Resetn 1
force a 1
run 100ns
force a 0
run 100ns
force b 1
run 100ns
force b 0
run 100ns
force a 1
run 100ns
force a 0
run 100ns
force a 1
run 100ns
force a 0
run 100ns
force b 1
run 100ns
force b 0
run 100ns
force a 1
run 100ns
force a 0
run 100ns
force b 1
```

14.19 forts.

```
run 100ns
force a 1
run 200ns
```

14.20 Vi ritar en tillståndsgraf, *Figur 14.20a*



Figur 14.20a

14.20 forts.

och ställer upp tillståndstabell, *Figur 4.20b*. Vi utelämnar alternativet att båda insignalerna skulle varahöga dåt inte kan inträffa. I VHDL-koden låter vi i det fallet FSM:n gå tillbaka till starttillståndet.

	Next state					
	Nickel/dime					
State	*00*	*01*	*10*	*11*	*thing*	*change*
S0	S0	S3	S1	-	0	0
S1	S2	S0	S1	-	0	0
S2	S2	S5	S3	-	0	0
S3	S4	S3	S3	-	0	0
S4	S4	S7	S5	-	0	0
S5	S6	S5	S5	-	0	0
S6	S6	S9	S7	-	0	0
S7	S8	S7	S7	-	0	0
S8	S8	S11	S9	-	0	0
S9	S10	S9	S9	-	0	0
S10	S10	S13	S11	-	0	0
S11	S12	S11	S11	-	0	0
S12	S12	S15	S13	-	0	0
S13	S14	S13	S13	-	0	0
S14	S14	S16	S15	-	0	0
S15	S0	S15	S15	-	1	0
S16	S0	S16	S0		0	1

Figur 14.20b

14.21 17 tillstånd så vi behöver 5 tillståndsvariabler

	Next state					
	Nickel/dime					
State	*00*	*01*	*10*	*11*	*thing*	*change*
00000	00000	00011	00001	-	0	0
00001	00010	00000	00001	-	0	0
00010	00010	00101	00011	-	0	0
00011	00100	00011	00011	-	0	0
00100	00100	00111	00101	-	0	0
00101	00110	00101	00101	-	0	0
00110	00110	01001	00111	-	0	0
00111	01000	00111	00111	-	0	0
01000	01000	01011	01001	-	0	0
01001	01010	01001	01001	-	0	0
01010	01010	01101	01011	-	0	0
01011	01100	01011	01011	-	0	0
01100	01100	01111	01101	-	0	0
01101	01110	01101	01101	-	0	0
01110	01110	10000	01111	-	0	0
01111	00000	01111	01111	-	1	0
10000	00000	10000	00000		0	1

Figur 14.20a

Lösningen blir rätt komplicerad med många variabler så vi utelämnar den här

14.22 Vi får VHDL-koden

```
-- ex_14_22.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY ex_14_22 IS
   PORT(Clock:IN STD_LOGIC;
        Resetn:IN STD_LOGIC;
        nickel:IN STD_LOGIC;
        dime:IN STD_LOGIC;
        z:OUT STD_LOGIC;
        change:OUT STD_LOGIC);
END ex_14_22;
```

14.22 forts.

```
ARCHITECTURE arch_ex_14_22 OF ex_14_22 IS
   TYPE state_type IS
    (S0,S1,S2,S3,S4,S5,S6,S7,S8,S9,
             S10,S11,S12,S13,S14,S15,S16);
   SIGNAL state_signal:state_type;
   SIGNAL next_state_signal:state_type;
   SIGNAL nickel_dime_signal:STD_LOGIC_VECTOR(1
    DOWNTO 0);
   SIGNAL value:NATURAL RANGE 0 TO 45:=0;
   BEGIN
      nickel_dime_signal <= nickel & dime;

   state_transition_proc:
      PROCESS(Resetn,Clock)
      BEGIN
         IF (Resetn='0') THEN
            state_signal<=S0;
         ELSIF rising_edge(Clock) THEN
            state_signal<=next_state_signal;
         END IF;
   END PROCESS state_transition_proc;

   stateflow_proc:
   PROCESS(state_signal,nickel,dime)
   BEGIN
      CASE state_signal IS
         WHEN S0 =>
            value <= 0;
            IF nickel_dime_signal = "00" THEN
               next_state_signal <= S0;
            ELSIF nickel_dime_signal = "01" THEN
               next_state_signal <= S3;
            ELSIF nickel_dime_signal = "10" THEN
               next_state_signal <= S1;
            ELSE
               next_state_signal <= S0;
            END IF;
          WHEN S1 =>
            value <= 5;
            IF nickel_dime_signal = "00" THEN
               next_state_signal <= S2;
            ELSIF nickel_dime_signal = "01" THEN
               next_state_signal <= S0;
```

14.22 forts.

```
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S1;
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S2 =>
      value <= 5;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S2;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S5;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S3;
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S3 =>
      value <= 10;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S4;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S3;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S3;
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S4 =>
      value <= 10;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S4;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S7;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S5;
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S5 =>
      value <= 15;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S6;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S5;
```

14.22 forts.

```
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S5;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S6 =>
      value <= 15;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S6;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S9;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S7;
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S7 =>
      value <= 20;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S8;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S7;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S7;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S8 =>
      value <= 20;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S8;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S11;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S9;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S9 =>
      value <= 25;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S10;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S9;
```

14.22 forts.

```
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S9;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S10 =>
      value <= 25;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S10;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S13;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S11;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S11 =>
      value <= 30;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S12;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S11;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S11;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S12 =>
      value <= 30;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S12;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S15;
      ELSIF nickel_dime_signal = "10" THEN
         next_state_signal <= S13;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S13 =>
      value <= 35;
      IF nickel_dime_signal = "00" THEN
         next_state_signal <= S14;
      ELSIF nickel_dime_signal = "01" THEN
         next_state_signal <= S13;
```

14.22 forts.

```
                  ELSIF nickel_dime_signal = "10" THEN
                     next_state_signal <= S13;
                  ELSE
                     next_state_signal <= S0;
                  END IF;
               WHEN S14 =>
                  value <= 35;
                  IF nickel_dime_signal = "00" THEN
                     next_state_signal <= S14;
                  ELSIF nickel_dime_signal = "01" THEN
                     next_state_signal <= S16;
                  ELSIF nickel_dime_signal = "10" THEN
                     next_state_signal <= S15;
                  ELSE
                     next_state_signal <= S0;
                  END IF;
               WHEN S15 =>
                  value <= 40;
                  IF nickel_dime_signal = "00" THEN
                     next_state_signal <= S0;
                  ELSIF nickel_dime_signal = "01" THEN
                     next_state_signal <= S15;
                  ELSIF nickel_dime_signal = "10" THEN
                     next_state_signal <= S15;
                  ELSE
                     next_state_signal <= S0;
                  END IF;
               WHEN S16 =>
                  value <= 45;
                  IF nickel_dime_signal = "00" THEN
                     next_state_signal <= S0;
                  ELSIF nickel_dime_signal = "01" THEN
                     next_state_signal <= S16;
                  ELSIF nickel_dime_signal = "10" THEN
                     next_state_signal <= S0;
                  ELSE
                     next_state_signal <= S0;
                  END IF;
            END CASE;
         END PROCESS stateflow_proc;
```

14.22 forts.

```
            assignment_proc:
            PROCESS(state_signal,nickel,dime)
            BEGIN
               CASE state_signal IS
                  WHEN S15 =>
                     z <= '1';
                     change <= '0';
                  WHEN S16 =>
                     z <= '1';
                     change <= '1';
                  WHEN OTHERS =>
                      z <= '0';
                      change <= '0';
               END CASE;

               END PROCESS assignment_proc;
         END arch_ex_14_22;
```

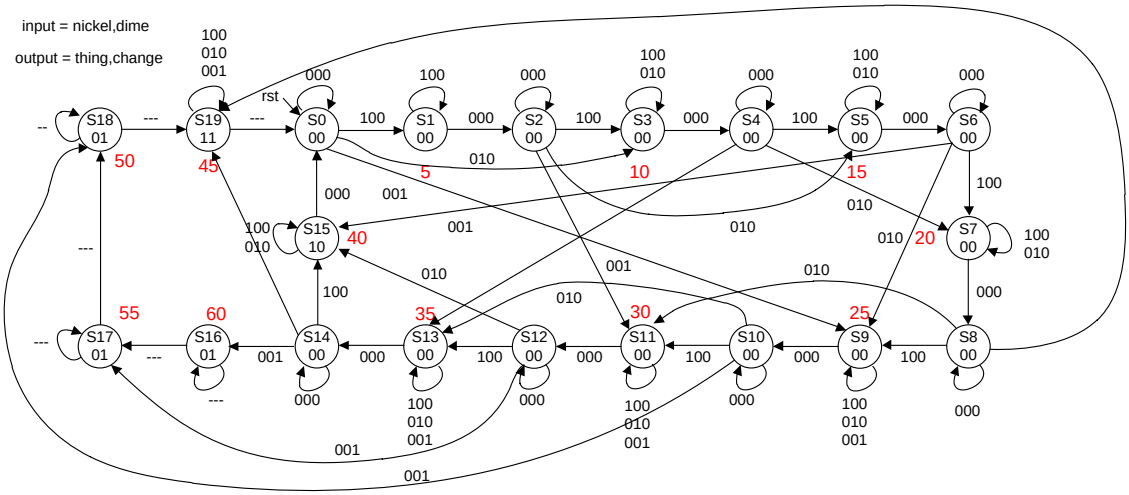
Som vi simulerar med en do-fil

```
-- ex_14_20.do
restart -f -nowave
view signals wave
add wave Clock Resetn nickel dime
add wave state_signal next_state_signal z change
add wave -radix unsigned value
force Clock 0 0, 1 50ns -repeat 100ns
force nickel 0
force dime 0
force Resetn 0
run 225ns
force Resetn 1
#1 dime = 10
force dime 1
run 100ns
force dime 0
run 100ns
#1+1 dime = 20
force dime 1
run 100ns
force dime 0
run 100ns
```

14.22 forts.

```
#1+1+1 dime = 30
force dime 1
run 100ns
force dime 0
run 100ns
#1+1+1+1 dime = 40
force dime 1
run 100ns
force dime 0
run 100ns
#10
force dime 1
run 100ns
force dime 0
run 100ns
#15
force nickel 1
run 100ns
force nickel 0
run 100ns
#25
force dime 1
run 100ns
force dime 0
run 100ns
#35
force dime 1
run 100ns
force dime 0
run 100ns
#45
force dime 1
run 100ns
force dime 0
run 100ns
```

14.23 Vi ritar en tillståndsgraf, *Figur 14.23a*



Figur 14.23a

14.23 forts.

Vi ställer upp tillståndstabell

	Next state									
	Nickel/dime/quarter									
State	*000*	*001*	*010*	*100*	*011*	*101*	*110*	*111*	*thing*	*change*
S0	S0	S9	S3	S1	-	-	-	-	0	0
S1	S2	S1	S1	S1	-	-	-	-	0	0
S2	S2	S11	S5	S3	-	-	-	-	0	0
S3	S4	S3	S3	S3	-	-	-	-	0	0
S4	S4	S13	S7	S5	-	-	-	-	0	0
S5	S6	S5	S5	S5	-	-	-	-	0	0
S6	S6	S15	S9	S7	-	-	-	-	0	0
S7	S8	S7	S7	S7	-	-	-	-	0	0
S8	S8	S19	S11	S9	-	-	-	-	0	0
S9	S10	S9	S9	S9	-	-	-	-	0	0
S10	S10	S18	S13	S11	-	-	-	-	0	0
S11	S12	S11	S11	S11	-	-	-	-	0	0
S12	S12	S17	S15	S13	-	-	-	-	0	0
S13	S14	S13	S13	S13	-	-	-	-	0	0
S14	S14	S16	S19	S15	-	-	-	-	0	0
S15	S0	S15	S15	S15	-	-	-	-	1	0
S16	S17	S17	S17	S17	-	-	-	-	0	1
S17	S18	S18	S18	S18	-	-	-	-	0	1
S18	S19	S19	S19	S19	-	-	-	-	0	1
S19	S0	S0	S0	S0	-	-	-	-	1	1

Figur 14.23b

14.24 Vi skriver VHDL-kod

```
-- ex_14_24.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
```

14.24 forts.

```
ENTITY ex_14_24 IS
   PORT(Clock:IN STD_LOGIC;
        Resetn:IN STD_LOGIC;
        nickel:IN STD_LOGIC;
        dime:IN STD_LOGIC;
        quarter:IN STD_LOGIC;
        z:OUT STD_LOGIC;
        change:OUT STD_LOGIC);
END ex_14_24;

ARCHITECTURE arch_ex_14_24 OF ex_14_24 IS
   TYPE state_type IS (S0,S1,S2,S3,S4,S5,S6,S7,S8,S9,
                10,S11,S12,S13,S14,S15,S16,S17,S18,S19);
   SIGNAL state_signal:state_type;
   SIGNAL next_state_signal:state_type;
   SIGNAL nickel_dime_quarter_signal:
                            STD_LOGIC_VECTOR(2 DOWNTO 0);
   SIGNAL value:NATURAL RANGE 0 TO 60:=0;
   BEGIN
      nickel_dime_quarter_signal <=
                              nickel & dime & quarter;

   state_transition_proc:
      PROCESS(Resetn,Clock)
      BEGIN
         IF (Resetn='0') THEN
            state_signal<=S0;
         ELSIF rising_edge(Clock) THEN
            state_signal<=next_state_signal;
         END IF;
   END PROCESS state_transition_proc;

   stateflow_proc:
   PROCESS(state_signal,nickel,dime)
   BEGIN
      CASE state_signal IS
         WHEN S0 =>
            value <= 0;
            IF nickel_dime_quarter_signal = "000" THEN
               next_state_signal <= S0;
            ELSIF nickel_dime_quarter_signal = "001" THEN
               next_state_signal <= S9;
```

14.24 forts.

```
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S3;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S1;
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S1 =>
      value <= 5;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S2;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S1;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S1;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S1;
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S2 =>
      value <= 5;
       IF nickel_dime_quarter_signal = "000" THEN
          next_state_signal <= S2;
       ELSIF nickel_dime_quarter_signal = "001"
          THEN
             next_state_signal <= S11;
       ELSIF nickel_dime_quarter_signal = "010"
          THEN
             next_state_signal <= S5;
       ELSIF nickel_dime_quarter_signal = "100"
          THEN
             next_state_signal <= S3;
       ELSE
          next_state_signal <= S0;
      END IF;
   WHEN S3 =>
      value <= 10;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S4;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S3;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S3;
```

14.24 forts.

```
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S3;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S4 =>
      value <= 10;
       IF nickel_dime_quarter_signal = "000" THEN
          next_state_signal <= S4;
       ELSIF nickel_dime_quarter_signal = "001"
          THEN
             next_state_signal <= S13;
       ELSIF nickel_dime_quarter_signal = "010"
          THEN
             next_state_signal <= S7;
       ELSIF nickel_dime_quarter_signal = "100"
          THEN
             next_state_signal <= S5;
       ELSE
          next_state_signal <= S0;
       END IF;
    WHEN S5 =>
      value <= 15;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S6;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S5;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S5;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S5;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S6 =>
      value <= 15;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S6;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S15;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S9;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S7;
```

14.24 forts.

```
      ELSE
         next_state_signal <= S0;
      END IF;
    WHEN S7 =>
      value <= 20;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S8;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S7;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S7;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S7;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S8 =>
      value <= 20;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S8;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S19;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S11;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S9;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S9 =>
      value <= 25;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S10;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S9;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S9;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S9;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S10 =>
      value <= 25;
```

14.24 forts.

```
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S10;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S18;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S13;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S11;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S11 =>
      value <= 30;
       IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S12;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S11;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S11;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S11;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S12 =>
      value <= 30;
       IF nickel_dime_quarter_signal = "000" THEN
          next_state_signal <= S12;
       ELSIF nickel_dime_quarter_signal = "001"
          THEN
             next_state_signal <= S17;
       ELSIF nickel_dime_quarter_signal = "010"
          THEN
             next_state_signal <= S15;
       ELSIF nickel_dime_quarter_signal = "100"
          THEN
             next_state_signal <= S13;
       ELSE
          next_state_signal <= S0;
      END IF;
   WHEN S13 =>
      value <= 35;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S14;
```

```
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S13;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S13;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S13;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S14 =>
      value <= 35;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S14;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S16;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S19;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S15;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S15 =>
      value <= 40;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S0;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S15;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S15;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S15;
      ELSE
         next_state_signal <= S0;
      END IF;
   WHEN S16 =>
      value <= 60;
      IF nickel_dime_quarter_signal = "000" THEN
         next_state_signal <= S17;
      ELSIF nickel_dime_quarter_signal = "001" THEN
         next_state_signal <= S17;
      ELSIF nickel_dime_quarter_signal = "010" THEN
         next_state_signal <= S17;
      ELSIF nickel_dime_quarter_signal = "100" THEN
         next_state_signal <= S17;
```

14.24 forts.

```
            ELSE
               next_state_signal <= S0;
            END IF;
         WHEN S17 =>
            value <= 55;
            IF nickel_dime_quarter_signal = "000" THEN
               next_state_signal <= S18;
            ELSIF nickel_dime_quarter_signal = "001" THEN
               next_state_signal <= S18;
            ELSIF nickel_dime_quarter_signal = "010" THEN
               next_state_signal <= S18;
            ELSIF nickel_dime_quarter_signal = "100" THEN
               next_state_signal <= S18;
            ELSE
               next_state_signal <= S0;
            END IF;
         WHEN S18 =>
            value <= 50;
            IF nickel_dime_quarter_signal = "000" THEN
               next_state_signal <= S19;
            ELSIF nickel_dime_quarter_signal = "001" THEN
               next_state_signal <= S19;
            ELSIF nickel_dime_quarter_signal = "010" THEN
               next_state_signal <= S19;
            ELSIF nickel_dime_quarter_signal = "100" THEN
               next_state_signal <= S19;
            ELSE
               next_state_signal <= S0;
            END IF;
         WHEN S19 =>
            value <= 45;
            IF nickel_dime_quarter_signal = "000" THEN
               next_state_signal <= S0;
            ELSIF nickel_dime_quarter_signal = "001" THEN
               next_state_signal <= S0;
            ELSIF nickel_dime_quarter_signal = "010" THEN
               next_state_signal <= S0;
            ELSIF nickel_dime_quarter_signal = "100" THEN
               next_state_signal <= S0;
            ELSE
               next_state_signal <= S0;
            END IF;
      END CASE;
   END PROCESS stateflow_proc;
```

14.24 forts.

```
        assignment_proc:
        PROCESS(state_signal,nickel,dime)
        BEGIN
           CASE state_signal IS
              WHEN S15 =>
                 z <= '1';
                 change <= '0';
              WHEN S16 =>
                 z <= '0';
                 change <= '1';
              WHEN S17 =>
                 z <= '0';
                 change <= '1';
              WHEN S18 =>
                 z <= '0';
                 change <= '1';
              WHEN S19 =>
                 z <= '1';
                 change <= '1';
              WHEN OTHERS =>
                  z <= '0';
                  change <= '0';
           END CASE;
        END PROCESS assignment_proc;
     END arch_ex_14_24;
```

Och vi simulerar med en do-fil som inte är heltäckande

```
-- ex_14_24.do
restart -f -nowave
view signals wave
add wave Clock Resetn nickel dime quarter
add wave state_signal next_state_signal z change
add wave -radix unsigned value
force Clock 0 0, 1 50ns -repeat 100ns
force nickel 0
force dime 0
force quarter 0
force Resetn 0
run 225ns
force Resetn 1
```

14.24 forts.

```
#1 dime = 10
force dime 1
run 100ns
force dime 0
run 100ns
#1+1 dime = 20
force dime 1
run 100ns
force dime 0
run 100ns
#1+1+1 dime = 30
force dime 1
run 100ns
force dime 0
run 100ns
#1+1+1+1 dime = 40
force dime 1
run 100ns
force dime 0
run 100ns
#5
force nickel 1
run 100ns
force nickel 0
run 100ns
#10
force nickel 1
run 100ns
force nickel 0
run 100ns
#15
force nickel 1
run 100ns
force nickel 0
run 100ns
#20
force nickel 1
run 100ns
force nickel 0
run 100ns
```

14.24 forts.

```
#25
force nickel 1
run 100ns
force nickel 0
run 100ns
#30
force nickel 1
run 100ns
force nickel 0
run 100ns
#35
force nickel 1
run 100ns
force nickel 0
run 100ns
#40
force nickel 1
run 100ns
force nickel 0
run 100ns
force quarter 1
run 100ns
force quarter 0
run 100ns
force quarter 1
run 100ns
force quarter 0
run 400ns
```

14.25 Lösningen utelämnas nu den blir så komplicerad att vi ändå inte kan gå igenom den

14.26 Lösningen utelämnas nu den blir så komplicerad att vi ändå inte kan gå igenom den

14.27 Lösningen utelämnas nu den blir så komplicerad att vi ändå inte kan gå igenom den

14.28 Lösningen utelämnas nu den blir så komplicerad att vi ändå inte kan gå igenom den

14.29 Lösningen utelämnas nu den blir så komplicerad att vi ändå inte kan gå igenom den

14.30 Lösningen utelämnas nu den blir så komplicerad att vi ändå inte kan gå igenom den

14.31 Lösningen utelämnas nu den blir så komplicerad att vi ändå inte kan gå igenom den