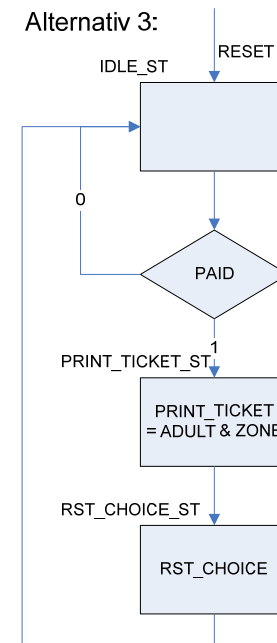
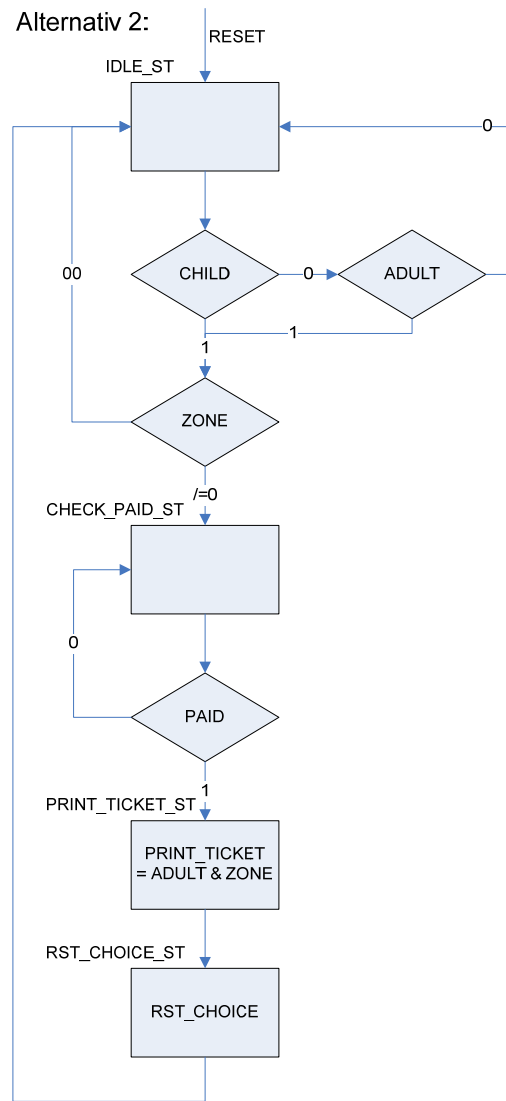
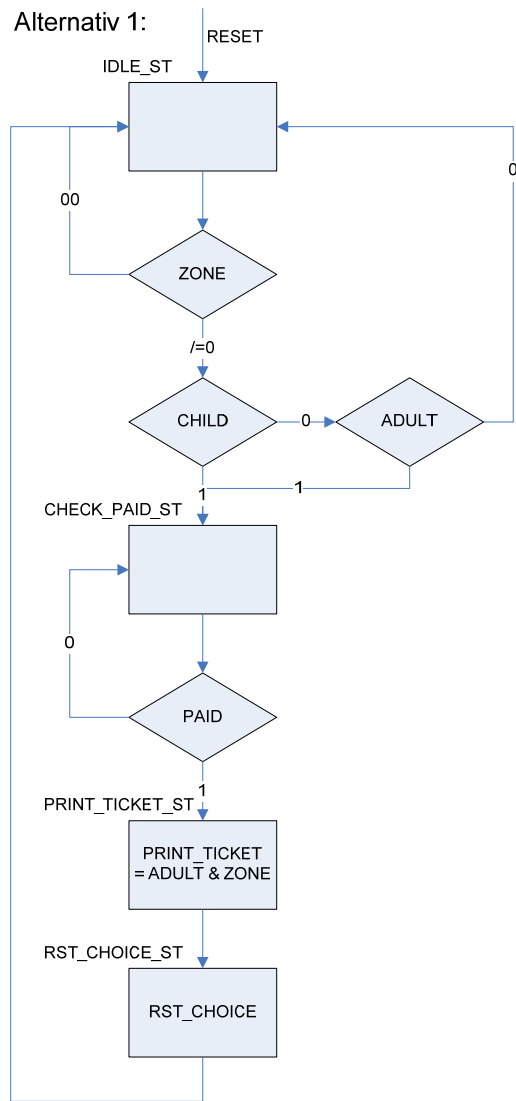


INF3430/INF4430 Fasit eksamen 2006 Oppgave 1-14

Oppgave	A	B	C	D	E
1		O	O		
2	O	O		O	
3		O		O	
4	O			O	
5		O	O		
6		O			
7	O	O			
8		O	O	O	
9	O	O			
10			O	O	
11	O	O	O	O	
12	O		O		
13	O		O	O	
14		O		O	

```
1  --Oppgave 15a:
2  -----
3
4
5  ZONE_REG:
6  process(RESET,CLK)
7  begin
8      if RESET = '1' then
9          ZONE <= (others => '0');
10     elsif rising_edge(CLK) then
11         if RST_CHOICE = '1' then
12             ZONE <= (others => '0');
13         elsif 2_ZONE_PB = '0' and 1_ZONE_PB = '1' then
14             ZONE <= "01";
15         elsif 2_ZONE_PB = '1' and 1_ZONE_PB = '0' then
16             ZONE <= "10";
17         elsif 2_ZONE_PB = '1' and 1_ZONE_PB = '1' then
18             ZONE <= (others => '0');
19         end if;
20     end if;
21 end process;
22
```

Oppgave 15b).



```

39  --Oppgave 15c:
40  =====
41
42  --Alternativ 1 og 2
43  =====
44  architecture RTL_TICKET_MASTER of TICKET_MASTER is
45
46  type TICKET_MASTER_STATE is ( IDLE_ST,CHECK_PAID_ST,PRINT_TICKET_ST,RST_CHOICE_ST
47  )
48  signal CURRENT_ST,NEXT_ST : TICKET_MASTER_STATE;
49
50  begin
51
52  --Next state and output logic
53  NEXT_STATE_COMB:
54  process (CHILD,ADULT,ZONE,PAID,CURRENT_ST)
55  begin
56      RST_CHOICE    <= '0';
57      PRINT_TICKET <= (others => '0');
58      NEXT_ST       <= IDLE_ST;
59
60      case CURRENT_ST is
61
62          when IDLE_ST =>
63              --Alt 1:
64              =====
65              --if ZONE /= "00" then
66              --  if (CHILD = '1' or ADULT = '1') then
67              --    NEXT_ST <= CHECK_PAID_ST; --vi kan benytte if uten else fordi
68              --  end if;                    --vi benytter defaultverdier i starten
69              --end if;                      --av processen
70
71              --Alt 2:
72              =====
73              if CHILD = '1' then
74                  if ZONE /= "00" then
75                      NEXT_ST <= CHECK_PAID_ST;
76                  end if;
77              elsif ADULT = '1' then
78                  if ZONE /= "00" then
79                      NEXT_ST <= CHECK_PAID_ST;
80                  end if;
81              end if;
82
83          when CHECK_PAID_ST =>
84              if PAID = '1' then
85                  NEXT_ST <= PRINT_TICKET_ST;
86              else
87                  NEXT_ST <= CHECK_PAID_ST;
88              end if;
89
90          when PRINT_TICKET_ST =>
91              NEXT_ST <= RST_CHOICE_ST;
92              PRINT_TICKET <= ADULT & ZONE;  --
93
94          when RST_CHOICE_ST =>
95              NEXT_ST <= IDLE_ST;
96              RST_CHOICE <= '1';
97
98      end case;
99
100 end process NEXT_STATE_COMB;
101
102 CURRENT_STATE_REG:
103 process (RESET,CLK)
104 begin
105     if RESET = '1' then
106         CURRENT_ST <= IDLE_ST;
107     elsif rising_edge (CLK) then
108         CURRENT_ST <= NEXT_ST;

```

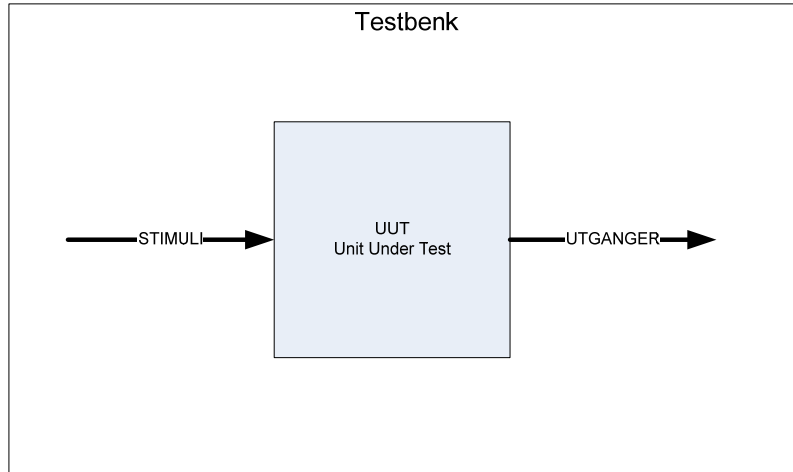
```

109     end if;
110 end process CURRENT_STATE_REG;
111
112 end architecture RTL_TICKET_MASTER;
113
114
115 --Alternativ 3:
116 -----
117 architecture RTL_TICKET_MASTER of TICKET_MASTER is
118
119 type TICKET_MASTER_STATE is (IDLE_ST,PRINT_TICKET_ST,RST_CHOICE_ST)
120 signal CURRENT_ST,NEXT_ST : TICKET_MASTER_STATE;
121
122 begin
123
124 --Next state and output logic
125 NEXT_STATE_COMB:
126 process (PAID,CURRENT_ST)
127 begin
128
129     RST_CHOICE    <= '0';
130     PRINT_TICKET  <= (others => '0');
131
132     case CURRENT_ST is
133
134         when IDLE_ST =>
135             if PAID = '1' then           --Antar at CHILD/ADULT og ZONE er aktivert
136                 NEXT_ST <= PRINT_TICKET_ST; --før PAID kan gå aktivt
137             else                           --Mefører at disse ikke er nødvendige på
138                 NEXT_ST <= IDLE_ST;       --sensitivitetslisten
139             end if;
140
141         when PRINT_TICKET_ST =>
142             NEXT_ST <= RST_CHOICE_ST;
143             PRINT_TICKET <= ADULT & ZONE; --Legg merke til enkel
144                                           --sammenheng i sannhetstabell 2
145
146         when RST_CHOICE_ST =>
147             NEXT_ST <= IDLE_ST;
148             RST_CHOICE <= '1';
149
150     end case;
151 end process NEXT_STATE_COMB;
152
153 CURRENT_STATE_REG:
154 process (RESET,CLK)
155 begin
156     if RESET = '1' then
157         CURRENT_ST <= IDLE_ST;
158     elsif rising_edge (CLK) then
159         CURRENT_ST <= NEXT_ST;
160     end if;
161 end process CURRENT_STATE_REG;
162
163 end architecture RTL_TICKET_MASTER;

```

Oppgave 15d).

For å simulere kretsen i oppgave 15c) må man påtrykke inngangene stimuli og sjekke utgangene.



I VHDL gjør man dette ved å lage en testbenk. I sin enkleste er den bygd opp på følgende måte

1. En (vanligvis) tom entitet for selve testbenken. Dvs. testbenken har vanligvis ikke noe interface mot verden utenfor men er "selfcontained".
2. Komponentdeklarasjon for UUT (Unit Under Test)
3. Deklarsjon av input stimuli signaler
4. Definisjon av klokke
5. Instantiering av UUT
6. Stimuli process der man lager en sekvens av input signaler
7. Sjekker output i "Waveform-viewer"

I mer avanserte testbenker kan man istedenfor stimuliprosessen påtrykke inputstimuli ved å benytte simuleringsmodeller av omkringliggende kretser og instantiere disse i testbenken. Selve testbenken kan bli vesentlig enklere på denne måten.

Et annet alternativ er å hente input stimuli fra fil.

En mer avansert måte å sjekke korrekt funksjon er å lage en fasit over forventede verdier på utgangene og sammenligne disse med utgangene av UUT. Fasiten kan man lagre i en egen fil eller inni testbenken.

På denne måten kan testbenken være selvtestende og man kan slippe å studere timingdiagrammer. Man kan bare rapportere om resultatet er OK eller ikke.