EDA321 Digitalteknik-syntes D2 och E3

GU **DIT795**

Tentamen *(EDA321-0205)*

Fredag den 13 januari 2012, fm i M-salarna

Examinator
Arne Linde, tel. 772 1683

Tillåtna hjälpmedel

Inga hjälpmedel tillåtna. Detta innefattar även kalkylatorer och alla tabellverk.

Oläsliga eller svårtydda lösningar ger poängavdrag.

Allmänt:

Ett lösningsblad får endast innehålla redovisningsdelar som hör ihop med en uppgift (dock flera deluppgifter).

För full poäng på de uppgifter som omfattar konstruktioner krävs förutom korrekt funktion även en optimal (minimal) eller nära optimal lösning.
Fungerande men onödigt komplicerade lösningar ger varierande poängavdrag beroende på hur mycket lösningen avviker från den optimala.
För samtliga uppgifter gäller, att ofullständiga lösningar eller lösningar innehållande felaktigheter ger poängavdrag även om resultatet är korrekt.

Betygsättning
För godkänt betyg fordras minst 20 p av totalt 50 p.
20p ≤ **betyg 3** < 30p ≤ **betyg 4** < 40p ≤ **betyg 5**

För godkänt slutbetyg på hela kursen fordras godkänt betyg på tentamen och dessutom fordras godkänd laborationskurs.

Lösningar

Anslås senast måndag 16 januari, kl 16.00 på kursens hemsida.

Granskning
Av rättning kan göras
tisdag 14/2 kl 11.45 - 12.30 rum E-4128 och onsdag 16/2 kl 11.45-12.00 rum E-4128.

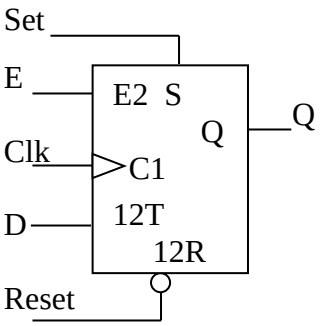
1. *Småfrågor (10p)*

a) Vilka är skillnaderna mellan en FPGA och en CPLD för konstruktören.
(antal minneselement, typ av logik, snabbhet, timing och struktur skall finnas med för full poäng.) (3p)

b) Koda D-vippan enligt figur 1 i VHDL.
Den har: Asynkron Set (aktiv hög),
Synkron Reset (aktiv låg), Enable och
Data ingångar. Enable påverkar D och Reset funktionerna.
Använd följande entitet:

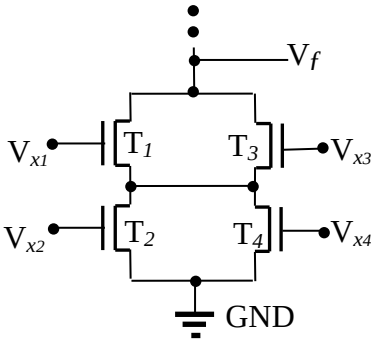
```
ENTITY D_Vippa IS
  PORT ( Clk,Reset,Set, D, E :IN std_logic;
      Q :OUT std_logic);
END D_Vippa;
```

(4p)



Figur 1: JK-vippa till uppgift 1:b

c) Figur 2 visar halva transistornätet för en CMOS krets.
Rita den andra halvan som innehåller PMOS transistorerna.
(3p)



Figur 2: Transistornät till uppgift 1:e

2. *Minimering (10p)*

a) Bestäm samtliga primimplikatorer till funktionen: (3p)

$$f(x_3, x_2, x_1, x_0) = \sum m\,(2, 4, 6, 7, 13, 15) + d\,(0, 8, 9, 12)$$

b) Ta fram en lösning på minimal disjunktiv form till funktionen från uppgift 2a. (2p)

c) Ta fram alla minimala lösningar på minimal disjunktiv form till funktionen från uppgift 2a. (3p)

d) Ta fram en hasard fri lösning på minimal disjunktiv form till funktionen från uppgift 2a. (2p)

3. *VHDL (10p)*

Betrakta VHDL- koden.

a) Rita upp en tillståndsgraf utgående från VHDL-koden (1p)

b) Vilken typ av tillstånds-maskin beskrivs? Motivera! (2p)

c) Fyll i pulsdiagrammet som finns som bilaga och bifoga den. (3p)

Systemet behöver modifieras. Antalet klockintervall som A och B minst skall vara lika behöver bestämmas av en Signalvektor.

Så entiteten behöver ändras:

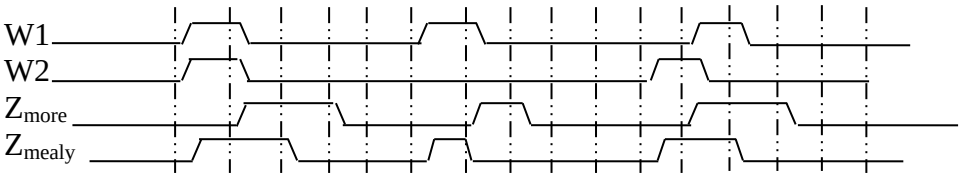
```
ENTITY Upg_3d IS
 PORT (A, B :IN std_logic;
  Comp : IN std_logic_vector(3 DOWNTO 0);
      Clk,Reset :IN std_logic;
      AA, BB, CE :OUT std_logic);
END Upg_3d;
```

d) Redogöra för hur koden skall modifieras. För att uppfylla kraven ovan. (4p)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY Upg_3a IS
  PORT (A, B :IN std_logic;
        Clk,Reset :IN std_logic;
        AA, BB, CE :OUT std_logic);
END Upg_3a;

ARCHITECTURE Beha OF Upg_3a IS
TYPE State_Type IS (St,S0,S1,S2);
Signal Y, Next_Y : State_type;

BEGIN
  P1: PROCESS(Clk,Reset)
  BEGIN
    IF Reset='1' THEN
      Y<=St;
    ELSIF Clk'event AND Clk='1' THEN
      Y<= Next_Y;
    END IF;
  END PROCESS P1;

  P2: PROCESS(Y, A, B)
  BEGIN
    Next_Y<=St; AA<='0'; BB<='0'; CE<='0';
    CASE Y IS
      WHEN St =>
       IF A=B THEN Next_Y<=S0; END IF;
      WHEN S0 => CE<='1';
        IF A=B THEN Next_Y<=S1; END IF;
      WHEN S1 => CE<='1';
        IF A=B THEN Next_Y<=S2; END IF;
      WHEN S2 =>
        IF A=B THEN Next_Y<=S2;
        ELSIF A='1' THEN AA<='1';
        ELSE BB<='1'; END IF;
    END CASE;
  END PROCESS P2;

END Beha;
```

4. *Synkrona sekvensnät (12p)*

Ett synkront sekvensnät har två ingångar (w1,w2) och en utgång z.
Sekvensnätets funktion är att genera en enable signal till en räknare som räknar pulserna på w1 och w2.
Pulserna kommer med **långa mellanrum** men när pulsen kommer så kan man få pulser samtidigt eller nästan samtidigt på w1 och w2.



Streckad linje markerar aktiv klockflank

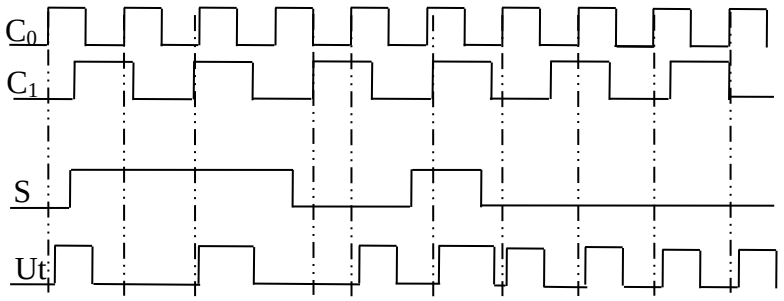
a) Rita upp en tillståndsgraf för nätet i form av ett Moore-nät. (3p)

b) Rita upp en tillståndsgraf för nätet i form av ett Mealy-nät. (3p)

c) Ta fram de Booleska ekvationerna för implementering av nätet (Q^+ och Z) från del uppgift a), Moore-nät.
Använd One-Hot kodning där begynnelsetillståndet kodas som ’0000’. (3p)

d) Ta fram VHDL koden för att implementera deluppgift b), Mealy-nät. (3p)

5. *Asynkrona sekvensnät (8p)*
En klockväljare skall konstrueras. Insignalerna är två klockor (C_1, C_0) som har olika frekvens och en styrsignal S. Funktionen är sådan att när S=0 så släpper väljaren igenom C_0 och när S=1 så släpper den igenom C_1. När S byter värde så ska ditt asynkrona nät släppa igenom hela den påbörjade klockperioden och därefter börja på nästa positiva flank på den andra klockan. Insignalrestriktionen gäller, d v s två insignaler kan ej byta värde samtidigt.



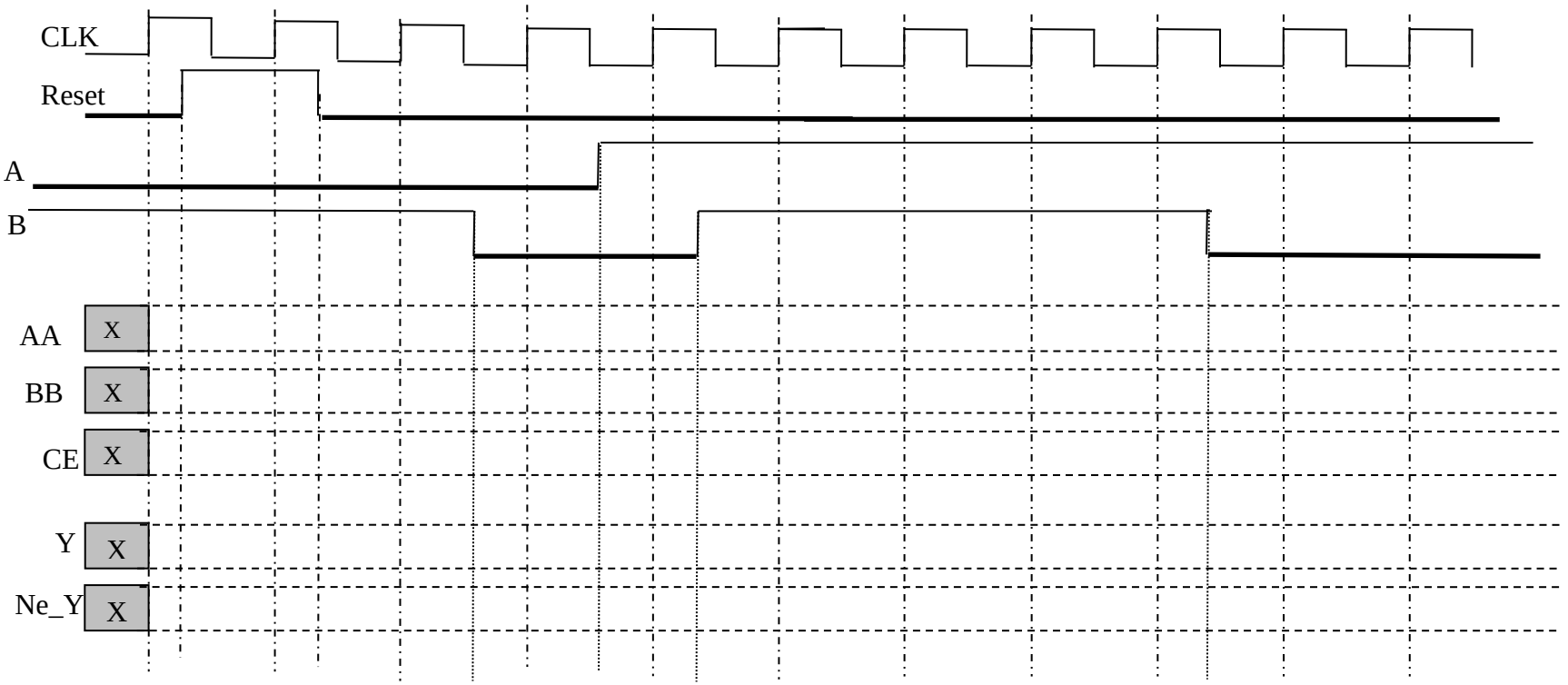
a) Rita upp en tillståndsgraf för det asynkrona sekvensnätet. (4p)

b) Hur många tillståndsvariabler behöver du? (2p)
(Motivering krävs)

c) Beskriv hur du skulle gå till väga för att göra tillståndstilldelningen. (2p)

Kod	Poäng	Sida

Bilaga A: till uppgift 3.

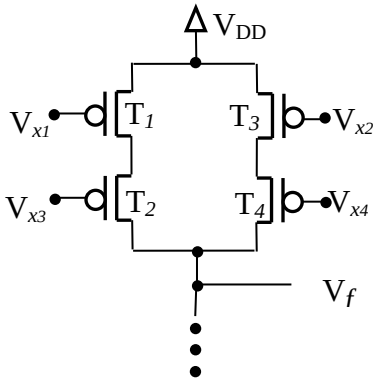


Markera tydligt vid vilken flank som signalen (AA,BB,CE,Y och Ne_Y) slår om samt vilket värde signalen får.

Riv här!

Lösningar
1. a)
FPGA:er är stora och innehåller många minneselement men mindre med logik.
CPLD:er innehåller få minneselement men förhållandevis mer logik.
FPGA:er består av LUT:ar (Lock upp table) och minneselement (3-7 ingångar).
CPLD:er består av flera PAL:kretsar och en kopplingsmatris.
PAL:en är en stor Och array som ger produkttermer av i storleksordningen 32 variabler sedan summeras produkttermerna upp och kopplas till ett minneselement.
CPLD:er är snabba och deterministiska, har men god tjänsla för kretsen så vet man timingen innan man har utfört syntes.
FPGA:er är också snabba men strukturen gör att timingen starkt beror på konstruktion och vilken tur man har i syntesen.
b)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY D_Vippa IS
   PORT ( Clk,Reset,Set, D, E :IN std_logic;
       Q :OUT std_logic);
END T_Vippa;

ARCHITECTURE Beha OF T_Vippa IS
Signal Int_Q : std_logic;

BEGIN
  DPros: PROCESS(Clk,Reset)
  BEGIN
    IF Set='1' THEN
      Q<='1';
    ELSIF Clk'event AND Clk='1' THEN
      IF E='1' THEN
        IF Reset='1' then Q<='0';
        ELSE Q<= D;
        END IF; -- Reset
      END IF; -- E
    END IF; -- Set
  END PROCESS DPros;
END Beha;
```



c) Se figur

2 a. f (x3, x2, x1, x0) = ∑ m (2, 4, 6, 7, 13, 15) + d (0, 8, 9, 12)

0 0000 V
2 0010 V
4 0100 V
8 1000 V
6 0110 v
9 1001 v
12 1100 v
7 0111 v
13 1101 v
15 1111 v

0,2 00-0 v
0,4 0-00 v
0,8 -000 v
2,6 0-10 v
4,6 01-0 v
4,12 -100 v
8,9 100- v
8,12 1-00 v
6,7 011- →a
9,13 1-01 v
12,13 110- v

0,2,4,6 0--0 →e
0,4,8,12 --00 →f
8,9,12,13 1-0- →g

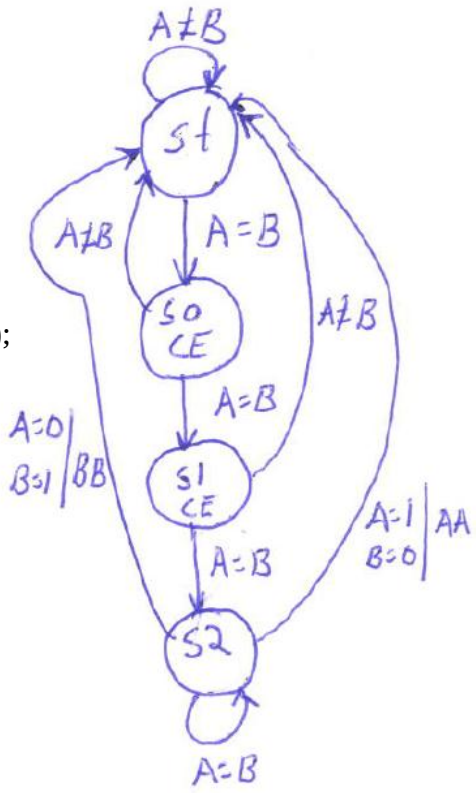
	2	4	6	7	13	15
a 6,7			*x*	x		
b 7,15				x		x
c 13,15					x	x
d 0,2,4,6	**x**	x	x			
e 0,4,8,12		*x*				x
f 8,9,12,13					x	

7,15 -111 →c
13,15 11-1 →d

2. b+c) f = d + ((b + (c eller f)) eller (c + (a eller b))) →[minimalt]→ d + b + f

2. d) f= a + d + b + c

3.
a) Se figure:
b) Mealy eftersom den beter sig som Mealy i tillstånd S2

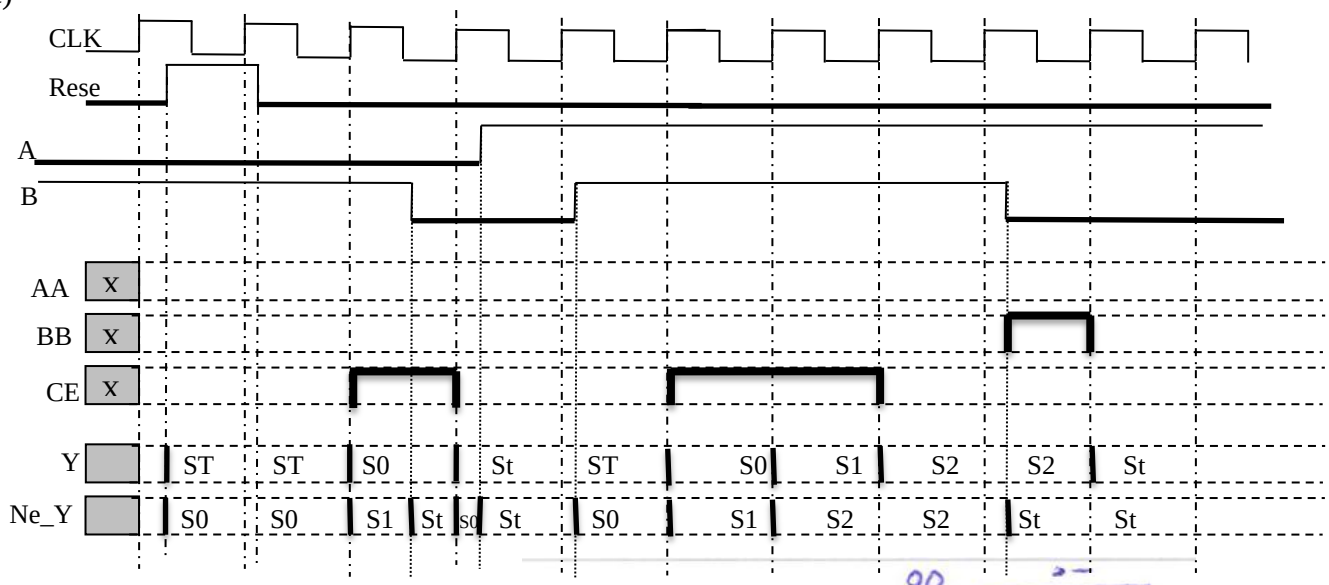


d)

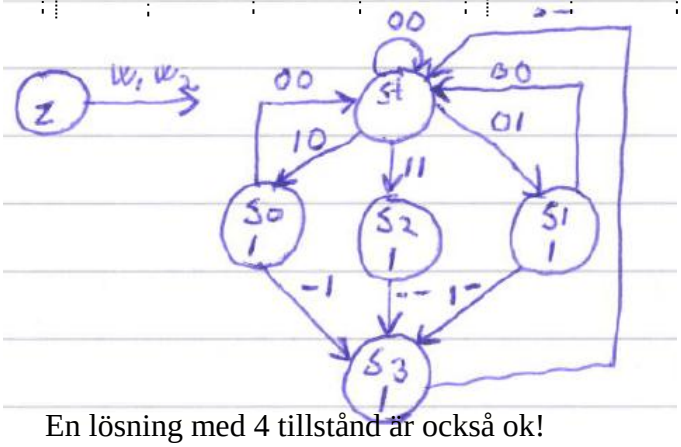
```
ARCHITECTURE Beha OF Upg_3d IS
TYPE State_Type IS (St,S0,S2);
Signal Y, Next_Y : State_type;
SIGNAL Cnt, N_CNT : std_logic_vector(3 DOWNTO 0);

BEGIN
  P1: PROCESS(Clk,Reset)
  BEGIN
    IF Reset='1' THEN
      Y<=St;
      Cnt<="0000";
    ELSIF Clk'event AND Clk='1' THEN
      Y<= Next_Y;
      Cnt<=N_Cnt;
    END IF;
  END PROCESS P1;
```

```
 P2: PROCESS(Y,Cnt, A, B)
 BEGIN
  Next_Y<=St; AA<='0'; BB<='0'; CE<='0';
  N_Cnt<=Cnt;
  CASE Y IS
   WHEN St => N_Cnt<="0000";
    IF A=B THEN Next_Y<=S0; END IF;
   WHEN S0 => CE<='1'; N_Cnt<= Cnt+1;
    IF A=B THEN Next_Y<=S0;
      IF Cnt=Comp THEN Next_Y<=S2; END IF;
    END IF;
   WHEN S2 =>
    IF A=B THEN Next_Y<=S2;
    ELSIF A='1' THEN AA<='1';
    ELSE BB<='1'; END IF;
  END CASE;
 END PROCESS P2;

END Beha;
```

c)



4. a)



En lösning med 4 tillstånd är också ok!
(Strikt enligt spec så kan S0 och S1 slås samman)

c) $Z = Q_3 + Q_2 + Q_1 + Q_0$

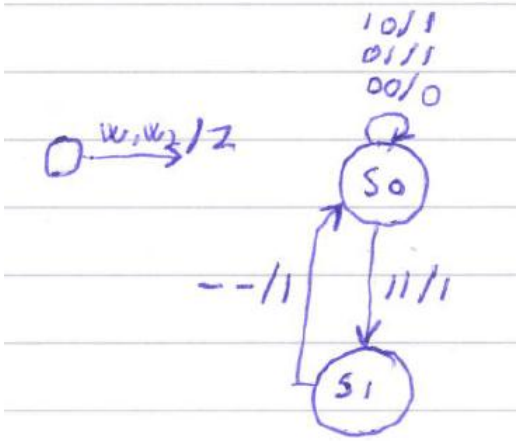
$Q_0 = Z' W_1W_2'$

$Q_1 = Z' W_1'W_2$

$Q_2 = Z' W_1W_2$

$Q3 = Q_0W_2 + Q_1W_1 + Q_2$

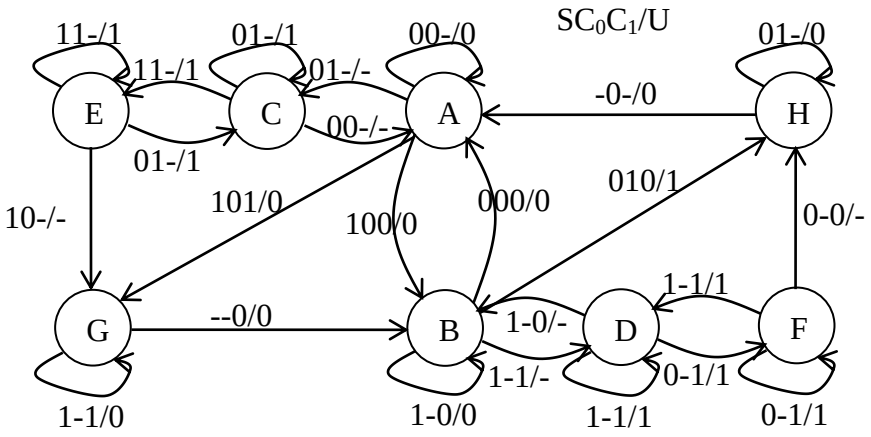
b)



d)

```
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY Upg_4d IS
  PORT (W1, W2 :IN std_logic;
        Clk,Reset :IN std_logic;
        Z :OUT std_logic);
END Upg_4d;

ARCHITECTURE Beha OF Upg_4d IS
TYPE State_Type IS (S0,S1);
Signal Y, Next_Y : State_type;

BEGIN
  P1: PROCESS(Clk,Reset)
  BEGIN
    IF Reset='1' THEN
      Y<=S0;
    ELSIF Clk'event AND Clk='1' THEN
      Y<= Next_Y;
    END IF;
  END PROCESS P1;

  P2: PROCESS(Y, W1, W2)
  BEGIN
    Next_Y<=S0; Z<='0';
    CASE Y IS
      WHEN S0 => Z<= W1 OR W2;
       IF (W1 AND W2)='1' THEN Next_Y<=S1; END IF;
      WHEN S1 => Z<='1';
    END CASE;
  END PROCESS P2;

END Beha;
```

5. a)



b) 8-tilltånd ger minst 3 variabler. För att implementera grafen ovan behövs dock minst en extra tillståndsvariabel.

c) Tillstånden läggs ut i en Boolesk-hyperkub. Alla tillståndsövergångar via diagonaler elimineras genom användning av transient tillstånd.