Eindtentamen Digitale Systemen 18/06/2007, 9.00 – 12.00 uur

De tentamen is **open boek** en bestaat uit 18 multiple choice (MC) vragen en 2 open vragen. De MC-vragen dienen beantwoord te worden op het uitgereikte **MC-formulier**. Enkele aanwijzingen bij het invullen van de MC-formulieren:

- slechts 1 antwoord is het correcte antwoord (NB: a,b,c,d staan door elkaar op het antwoordvel)
- vul de gekozen vakjes *helemaal* in (liefst met ballpoint, of met potlood)
- vul het formulier pas aan het einde in om fouten te voorkomen
- *geen* veranderingen aanbrengen: haal dan een nieuw formulier
- het onbeantwoord laten van een vraag werkt altijd in uw nadeel
- vergeet niet uw *studienummer* in te vullen (cijfers *en* vakjes!)

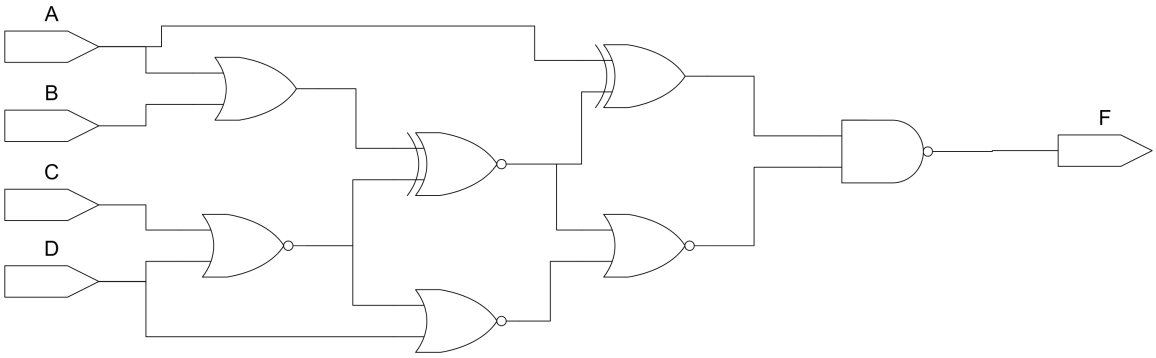
U mag het boek "Contemporary Logic Design", het VHDL boek, en eventuele prints van het **college** slides bij u hebben. Verder dus niets! Wij benadrukken dat u tijdens toetsen het tentamen dus **GEEN** gebruik mag maken van oude examens en toetsen. Gebruikt u deze toch dan zijn de tentamen fraude regels van toepassing.

Als een 5 is behaald, wordt nader onderzocht of er op grond van toetsresultaten aanleiding is om het cijfer om te zetten in een 6.

Succes!

A. MC-vragen (gewicht: 18 × 3.5 % = 63 %)

Vraag 1



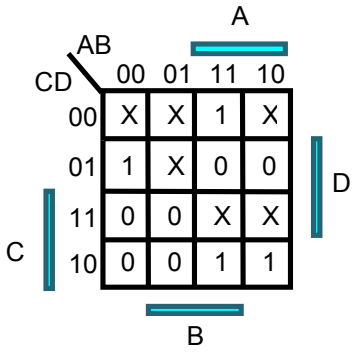
Voor bijgaand circuit gelden de volgende vertragingstijden:
NAND = 3ns, NOR = OR = 5ns, XOR = XNOR = 7ns.

Oorspronkelijk geldt A = B = C = D = 0. Dan wordt A = 1.
Welke van de volgende uitspraken is correct?

a. F wordt 0 na 10 ns en verandert niet meer
b. Er treedt geen hazard op mbt. F
c. Er treedt een 0-1-0 hazard op mbt. F
d. Er treedt een 1-0-1 hazard op mbt. F

Vraag 2

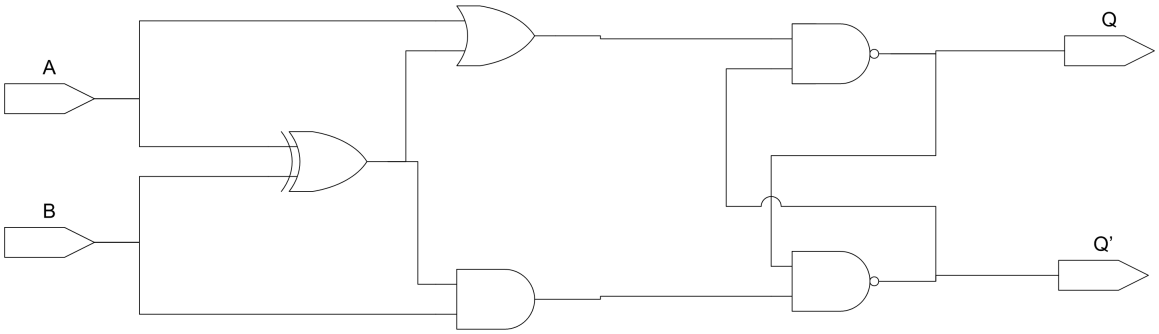
Welke expressie representeert een hazard-free netwerk?
(we nemen aan dat slechts één ingang van waarde verandert)



a. F = A'C' + AC + C'D'
b. F = AD' + A'C'+ AC
c. F = A'C' + AD' + C'D'
d. geen van bovenstaande antwoorden.

Vraag 3

Gegeven bijgaand circuit:



Welke van de volgende uitspraken is correct?

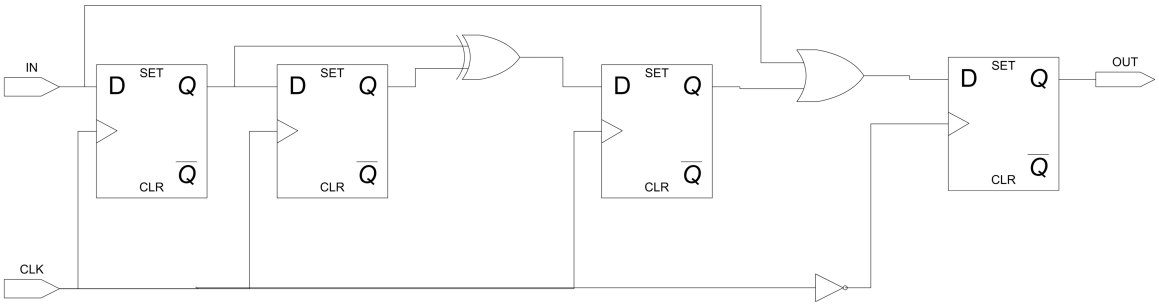
a. Dit circuit is geen bruikbaar geheugenelement: de Set-combinatie ontbreekt.
b. Dit circuit is geen bruikbaar geheugenelement: de Reset-combinatie ontbreekt
c. Dit circuit is een latch; AB = 00 is de verboden ingangscombinatie
d. Dit circuit is een latch zonder verboden ingangscombinatie

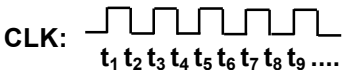
Vraag 4

Gegeven de berekening **C** = **A** – **B** in 2's complement notatie. Er geldt **A** = 011010 en **B** = 101100.
Wat is de waarde van **C**?

a. -18
b. 46
c. 14
d. **C** kan niet in een 6-bit 2's complement representatie worden berekend wegens overflow.

Vraag 5 Gegeven het volgende circuit met positive edge-triggered D flipflops:



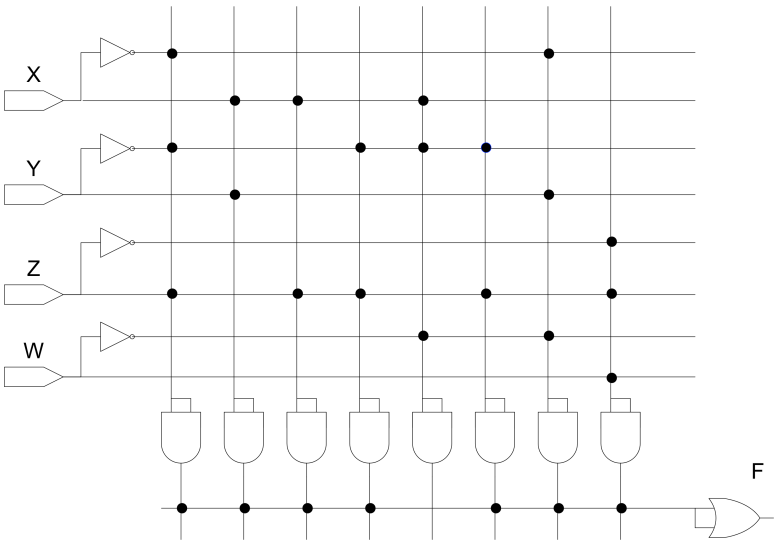


Voor bijgaand circuit gelden de volgende vertragingstijden: INV = 1ns, OR = 2ns, XOR = 3ns en CLK periode is 16ns.

Indien voor tijdstip t_1 alle Qs **1** zijn en op 'In' het constante signaal **0** wordt gezet, wanneer wordt dan voor het eerst **0** op 'Out' waargenomen? Op of direct na:

a. t_2
b. t_3
c. t_4
d. t_5 of later.

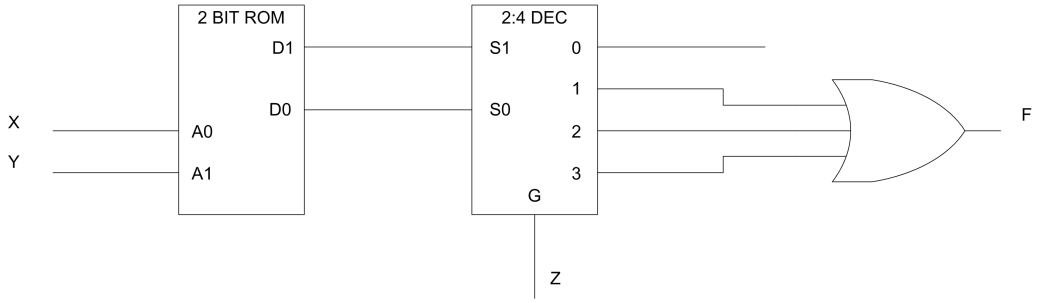
Vraag 6 Welke van de volgende vereenvoudigingen is correct?



a. F = ((Y + Z')(X' + Z))'
b. F = X(Y + Z) + Y'Z + YW'
c. F = X'W + Y'(X + W)
d. geen van bovenstaande antwoorden.

Vraag 7

Gegeven bijgaand circuit:



De ROM heeft de volgende inhoud:

A1	A0	D1	D0
0	0	1	1
0	1	0	0
1	0	0	1
1	1	1	0

Welke van de volgende vereenvoudigingen is correct?

a. F = z(x' + y')
b. F = z (x ⊕ y)'
c. F = (z' + xy')'
d. geen van bovenstaande antwoorden.

Vraag 8

Gegeven de volgende Delta I assembly code:

```
clr   c
ld    10100101b
and   01011010b
xor   01010101b
add   11000111b
xor   10000011b
add   00101100b
```

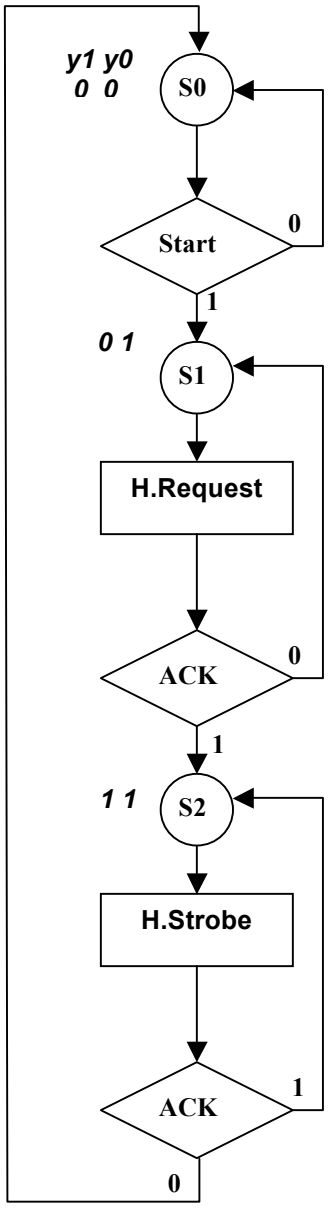
Wat gebeurt met A, Z en C na de executie van de laatste instructie?

a. A = CC_H , C = 1, Z = 0
b. A = CC_H , C = 0, Z = 1
c. A = CB_H , C = 0, Z = 0
d. A = CB_H , C = 1, Z = 0

Vraag 9

Gegeven de volgende ASM.
Stel dat we de toestand Si gecodeerd hebben met 2 variabelen y_0, y_1 volgens:

y_1	y_0	state
0	0	S0
0	1	S1
1	1	S2



Stel dat de ASM met D flip-flops en NAND gates moet worden gerealiseerd.

Hoeveel NAND gates zijn ***minimaal*** benodigd voor een netwerk?

a. 6 NAND
b. 7 NAND
c. 8 NAND
d. geen van de bovenstaande antwoorden geeft de minimale oplossing weer.

Vraag 10

Gegeven het volgende VHDL process:

```
PROCESS (s1, s2, s3)
BEGIN
      out <= s1 OR s2;
      out <= NOT(s3);
END PROCESS;
```

Alle signalen zijn van het type **std_logic**. Neem aan dat **s1**, **s2** en **s3** resp. de waarden 'W', '0' en 'L' hebben. Wat is de waarde van het **out** signaal?

a. 'W'
b. '-'
c. '1'
d. geen van bovenstaande

Vraag 11

Gegeven het volgende VHDL process:

```
PROCESS (s1, s2, s3, s4)
VARIABLE temp : std_logic;
BEGIN
      out <= s1 AND s4 AFTER 4ns;
      temp := (s1 AND s2) AND (s3 AND s4);
      out <= NOT(s3) AFTER 10ns;
      out <= temp AFTER 12ns;
END PROCESS;
```

Alle signalen zijn van het type **std_logic**. Neem aan dat alle signalen de waarde '1' hebben. Hoe lang duurt het om dit process uit te voeren (uitgedrukt in simulatietijd)?

a. geen tijd (0 ns)
b. een delta delay (Δ ns)
c. de som van alle delay statements (26 ns)
d. de maximale delay (12 ns)

Vraag 12

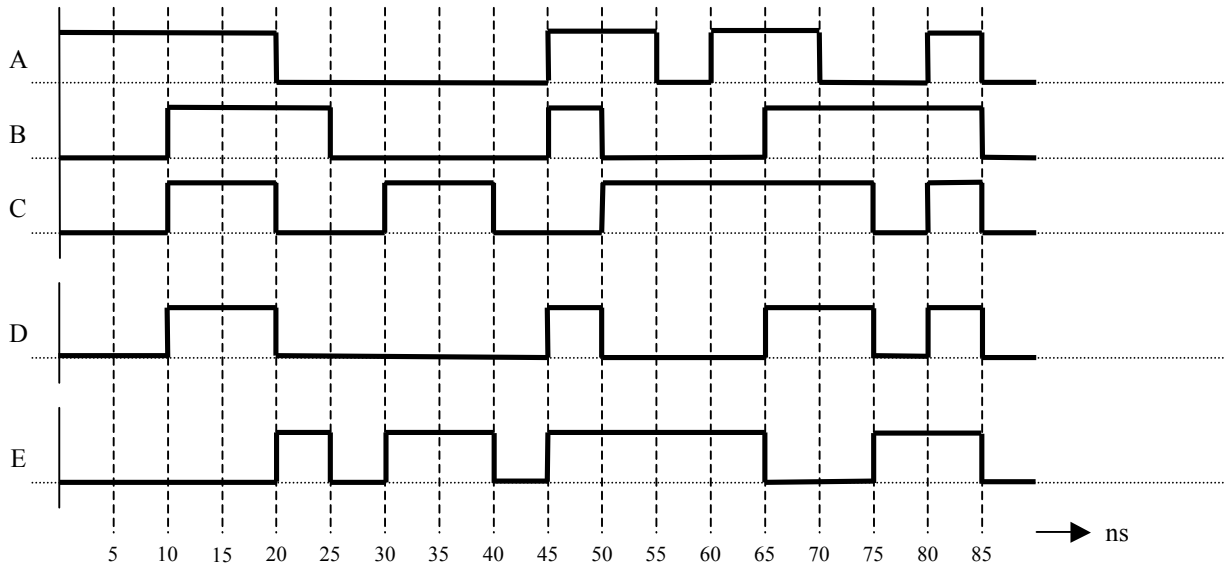
Gevraagd wordt hoe een concurrent statement kan voorkomen in VHDL.

a. Een concurrent statement kan voorkomen in een functie.
b. Een concurrent statement komt alleen voor in een architecture body.
c. Een concurrent statement kan voorkomen als variable assignment.
d. Geen van de drie genoemde mogelijkheden is goed.

Vraag 13

In VHDL worden naast signalen ook variabelen gebruikt. Welke van de volgende 4 beweringen is waar?

a. De waarde van een variabele mag niet direct aan een signaal worden toegekend.
b. Toekenning van een nieuwe waarde aan een variabele gebeurt pas een delta delay later.
c. In een process body mogen zowel signalen als variabelen worden gebruikt.
d. Alle bovenstaande beweringen zijn waar.



Hierboven zijn de ingangssignalen A, B, en C gegeven. Aan de hand van deze signalen worden de uitgangssignalen D en E (in simulatie) gegenereerd door de volgende processen:

```
L1:   PROCESS (sensitivity-list)  -- de sensitivity-list ontbreekt
      BEGIN
            D <= (A AND B);
      END PROCESS;

L2:   PROCESS (sensitivity-list) -- de sensitivity-list ontbreekt
      BEGIN
            E <= (B XOR C);
      END PROCESS;
```

Vraag 14 Welke sensitivity list moet worden ingevuld bij het genereren van signaal D?

a. (A, B)
b. (A, C)
c. (B, C)
d. Geen van bovenstaande.

Vraag 15 Welke sensitivity list moet worden ingevuld bij het genereren van signaal E?

a. (A, B)
b. (A, C)
c. (B, C)
d. Geen van bovenstaande.

Vraag 16

In een VHDL beschrijving worden 3 std_logic_vectoren a, b, en c gebruikt. Ze zijn als volgt gedeclareerd:

```
signal a, b, c:  std_logic_vector (3 downto 0);
```

Deze signalen worden gebruikt in het beschrijven van een binair rekenwerk. In dat rekenwerk gebruiken we de signalen als positieve binaire getallen in een 4 bits representatie. We willen zonder meer kunnen schrijven:

```
c <= a + b;
```

Gevraagd wordt de juiste keus te maken voor de zichtbaarheid van bibliotheek en packages die deze schrijfwijze mogelijk maakt.

Mogelijkheid 1:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
```

Mogelijkheid 2:

```
library IEEE;
use IEEE.std_logic_1164.all;
```

Mogelijkheid 3:

```
library IEEE;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

Mogelijkheid 4:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
```

Welke mogelijkheid kiest U?

a. Mogelijkheid 1.
b. Mogelijkheid 2.
c. Mogelijkheid 3.
d. Mogelijkheid 4.

Vraag 17

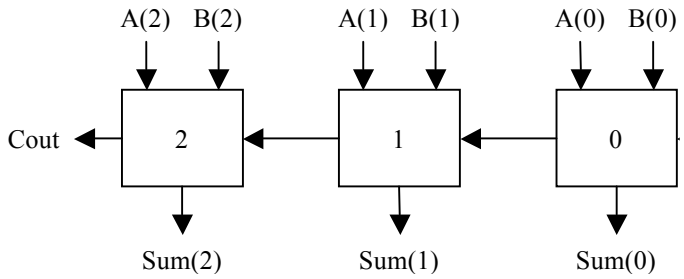
Gegeven is een hardware component met de volgende entity en architecture beschrijving:

```
entity onbekend is
      port (a, b, c: in bit; d: out bit);
end onbekend;

architecture gedrag of onbekend is
begin
      P1: process (b, c) is
      begin
            if (b='1') then d<='0';
            elsif (c'event and c='0') then d<=a;
            else d<=a;
            end if;
      end process;
end gedrag;
```

Welk van de onderstaande beweringen is juist?

a. De component is een JK-flipflop met c als klok die werkt op beide flanken en heeft tevens een asynchrone reset.
b. De component is een D-flipflop met c als klok die werkt op beide flanken en heeft tevens een asynchrone reset.
c. De component is een T-flipflop met c als klok die werkt op beide flanken en heeft tevens een asynchrone reset.
d. De component is een T-flipflop met c als klok die werkt op beide flanken en heeft tevens een synchrone reset.

Hieronder is een 3-bits ripple-carry adder gegeven door gebruik te maken van 1-bits full-adder elementen:



Hierna volgt de (onvolledige) VHDL beschrijving die u volledig moet maken:

```
ENTITY adder3b IS
      PORT (      A, B: IN bit_vector(2 DOWNTO 0);
            Cin : IN bit;
            Cout: OUT bit;
            Sum : OUT bit_vector(2 DOWNTO 0));
END adder3b;

ARCHITECTURE werking OF adder3b IS
COMPONENT full_adder
      PORT (A, B, Cin; IN bit; Sum, Cout: OUT bit);
END COMPONENT;
COMPONENT half_adder
      PORT (A, B; IN bit; Sum, Cout: OUT bit);
END COMPONENT;
signal tussen : bit_vector (1 DOWNTO 0);
BEGIN
      ...
      ...
      ...
END werking;
```

Vraag 18

Kies uit onderstaande vier mogelijkheden de juiste invulling van de drie stippellijnen in de architecture 'werking':

a.
```
een:  half_adder PORT MAP (A(2), B(2), Sum(2), Cout);
twee: full_adder PORT MAP (A(1), B(1), tussen(0), Sum(1), tussen(1));
drie: half_adder PORT MAP (A(0), B(0), Sum(0), tussen(0));
```

b.
```
een:  half_adder PORT MAP (A(0), B(0), Sum(0), tussen(1));
twee: full_adder PORT MAP (A(1), B(1), tussen(0), Sum(1), tussen(1));
drie: full_adder PORT MAP (A(2), B(2), tussen(1), Sum(2), Cout);
```

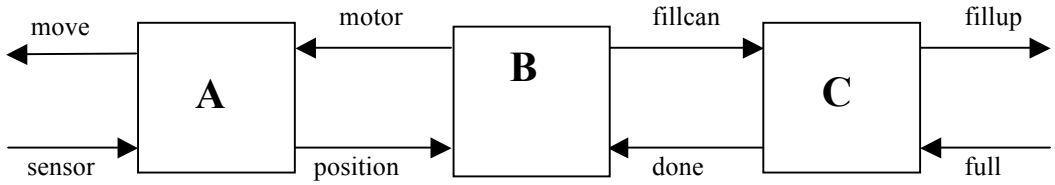
c.
```
een:  full_adder PORT MAP (A(2), B(2), tussen(1), Sum(2), Cout);
twee: full_adder PORT MAP (A(1), B(1), tussen(0), Sum(1), tussen(1));
drie: half_adder PORT MAP (A(0), B(0), Sum(0), tussen(1));
```

d.
```
een:  full_adder PORT MAP (A(2), B(2), tussen(1), Sum(2), Cout);
twee: full_adder PORT MAP (A(1), B(1), tussen(0), Sum(1), tussen(1));
drie: half_adder PORT MAP (A(0), B(0), Sum(0), tussen(0));
```

B. Open vragen (gewicht: 37%)

Vraag 19 (gewicht 20%)

Gegeven een machine voor het afvullen van verfblikken. De machine, bestaande uit een lopende band en een afvul machine, bevat 3 control units (A, B en C) zoals hieronder afgebeeld. Unit **A** bestuurd de lopende band, unit **C** bestuurt de afvulmachine en unit **B** verzorgt de centrale besturing.



Het gehele apparaat werkt als volgt. Het lege blik wordt door de lopende band onder de afvulmachine geplaatst. Vervolgens wordt het blik met verf gevuld tot het vol is. Hierna wordt het blik door de band afgevoerd.

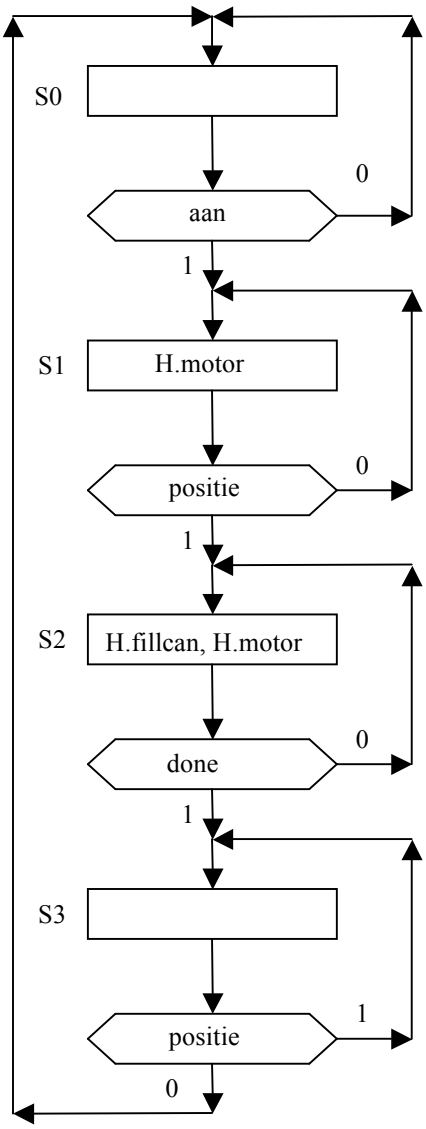
De ingang/uitgang signalen van unit C hebben de volgende betekenissen:

Signaal	Betekenis	Signaal	Betekenis
Fillcan = 1	Blik moet gevuld worden	Fillcan=0	Niets doen
Fillup = 1	Giet verf in blik	Fillup=0	Geen verf gieten
Full = 1	Huidig blik is vol	Full = 0	Leeg of geen blik
Done = 1	Vol blik onder afvuller	Done = 0	Geen vol blik onder afvuller

Het gedrag van control unit **B** is hiernaast d.m.v. een ASM weergegegven. Ontwerp nu zelf de ASM en hardware van unit **C** zodanig dat de machine correct functioneert (zie hieronder voor vraagstelling).

a. **(10 punten)** Specificeer het gewenste gedrag van het besturingssysteem voor control unit **C** mbv. een ASM. **Dit moet een *ASM* zijn!** Deze ASM vormt tevens de basis voor vraag 19 **b**. (Ook als uw ASM fout is, wordt *uw* ASM als uitgangspunt gehanteerd bij de beoordeling.)

b. **(10 punten)** Specificeer de hardware van control unit **C** d.m.v. logische expressies. Kies een state assignment die de states S_i afbeeldt op de state-variabelen $\mathbf{y_j}$. De state-opvolging hoeft niet progressief te zijn. Teken de K-maps voor $\mathbf{y_j}^+$, **fillup** en **done,** en leidt hun de expressies af.



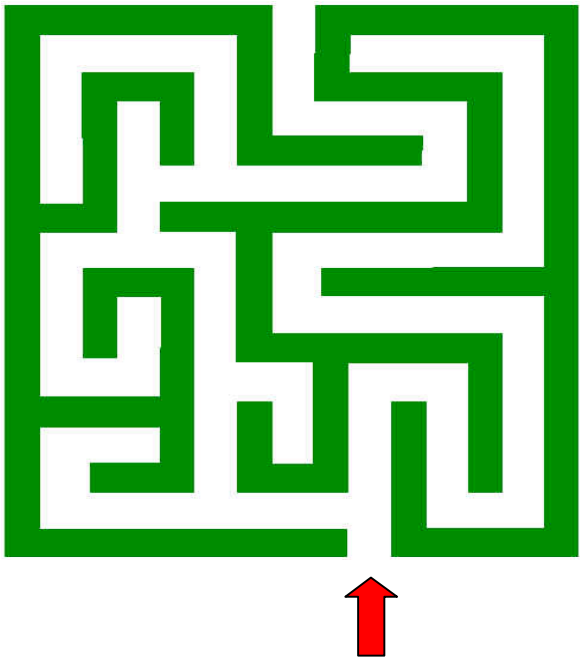
Control Unit B

Vraag 20 (gewicht 17%)

Een robot voertuigje moet zijn weg zien te vinden door een doolhof vanaf het aangegeven startpunt. Ontwerp een eenvoudige besturing voor de robot om deze taak te vervullen. Een eenvoudig algoritme om dit te doen is als volgt:

* Neem altijd alle rechter bochten indien mogelijk. Zo niet, rij dan rechtdoor. Is dat ook niet mogelijk, draai dan naar links en herhaal het algoritme.

Het robotje kan vooruit rijden, stoppen, 90° links of rechts draaien. De besturingssignalen zijn als volgt gecodeerd:



B1	B0	betekenis
1	1	Rechtuit rijden
0	1	90 draaien naar rechts
1	0	90 draaien naar links
0	0	geen opdracht

De besturing is gevoelig op transities (edge triggered) van signalen B0 en B1, d.w.z. dat B0 en/of B1 niet 1 hoeven te blijven tot het einde van de beweging. Voorbeeld: als B1 van 0 naar 1 wordt gezet dan begint het robotje links te draaien en stopt als hij 90 heeft gedraaid. Na elke (stukje rijden of draaien) stop de robot automatisch na een tijdje. Een besturingssignaal (B0, B1) kan alleen gegeven worden als de robot stilstaat.

Verder heeft het robotje drie sensor signalen:

RV	betekenis	RUV	Betekenis	ST	betekenis
0	rechts geblokkeerd	0	rechtuit geblokkeerd	1	robot staat stil
1	rechts vrij	1	rechtuit vrij	0	robot rijdt of draait

De robot mag pas beginnen als het signaal START (dit is niet de ST-signaal van de sensor) op 1 gezet wordt. Hierna is de robot niet meer te stoppen.

Opdracht:

Geef een VHDL beschrijving van de besturing van deze robot en teken de route die jouw besturing de robot laat doorlopen. (Neem aan dat de componenten ‘sensors’ en ‘besturing’ al bestaan en werken zoals hierboven beschreven zijn – echter, vergeet niet ze te declareren).

– o – o – o – o – o – o –

Uittreksel van "std_logic_1164"

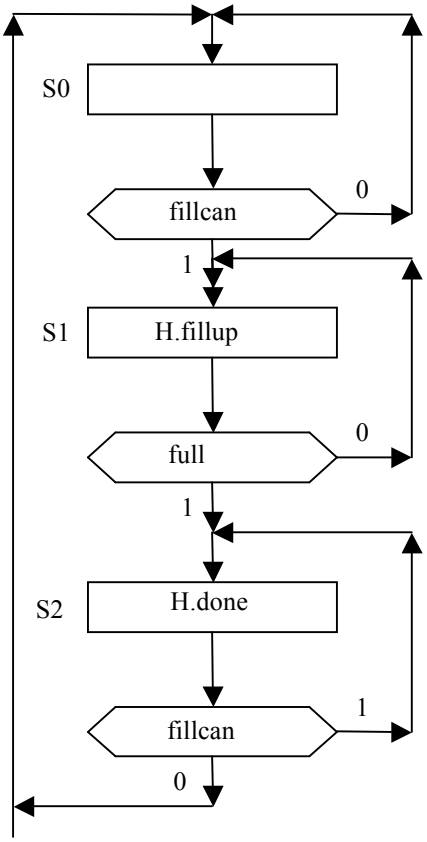
```
-------------------------------------------------------------------
-- tables for logical operations
-------------------------------------------------------------------
-- truth table for "and" function
CONSTANT and_table : stdlogic_table := (
--      ---------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -      |   |
--      ---------------------------------------------------
        ( 'U', 'U', '0', 'U', 'U', 'U', '0', 'U', 'U' ),  -- | U |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ),  -- | X |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  -- | 0 |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | 1 |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ),  -- | Z |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ),  -- | W |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  -- | L |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | H |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' )   -- | - |
);
-- truth table for "or" function
CONSTANT or_table : stdlogic_table := (
--      ---------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -      |   |
--      ---------------------------------------------------
        ( 'U', 'U', 'U', '1', 'U', 'U', 'U', '1', 'U' ),  -- | U |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' ),  -- | X |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | 0 |
        ( '1', '1', '1', '1', '1', '1', '1', '1', '1' ),  -- | 1 |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' ),  -- | Z |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' ),  -- | W |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | L |
        ( '1', '1', '1', '1', '1', '1', '1', '1', '1' ),  -- | H |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' )   -- | - |
);
-- truth table for "not" function
CONSTANT not_table: stdlogic_1d :=
--  ---------------------------------------------------
--  |   U    X    0    1    Z    W    L    H    -   |
--  ---------------------------------------------------
     ( 'U', 'X', '1', '0', 'X', 'X', '1', '0', 'X' );

-------------------------------------------------------------------
-- overloaded logical operators ( with optimizing hints )
-------------------------------------------------------------------

FUNCTION "and"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01 IS
BEGIN
    RETURN (and_table(l, r));
END "and";

FUNCTION "nand" ( l : std_ulogic; r : std_ulogic ) RETURN UX01 IS
BEGIN
    RETURN  (not_table ( and_table(l, r)));
END "nand";

FUNCTION "or"   ( l : std_ulogic; r : std_ulogic ) RETURN UX01 IS
BEGIN
    RETURN (or_table(l, r));
END "or";

FUNCTION "nor"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01 IS
BEGIN
    RETURN  (not_table ( or_table( l, r )));
END "nor";

FUNCTION "not"  ( l : std_ulogic ) RETURN UX01 IS
BEGIN
    RETURN (not_table(l));
END "not";
```

Uitwerkingen Eindtentamen Digitale Systemen 18/06/2007

MC-vragen:

1. d	**5**. a	**9**. b	**13**. c	**17**. b
2. c	**6**. b	**10**. c	**14**. c	**18**. d
3. a	**7**. c	**11**. a	**15**. d	
4. d	**8**. b	**12**. b	**16**. d	

Vraag 19



Control Unit C

y1	**y0**	**State**
0	0	S0
1	0	S1
1	1	S2
0	1	x

K-maps :

Fillcan –> C
Full – > F

y0$^+$:

CF \ y1y0	00	01	11	10
00	**0**	**X**	**0**	**0**
01	**0**	**X**	**0**	**1**
11	**0**	**X**	**1**	**1**
10	**0**	**X**	**1**	**0**

y0$^+$ = y0C + y1y0'F

y1$^+$:

CF \ y1y0	00	01	11	10
00	**0**	**X**	**0**	**1**
01	**0**	**X**	**0**	**1**
11	**1**	**X**	**1**	**1**
10	**1**	**X**	**1**	**1**

y1$^+$ = C + y1y0'

Fillup = y1 y0'
Done = y1 y0

Vraag 20

```
ENTITY robot IS
        PORT(START, clk: in bit);
END robot;

ARCHITECTURE gedrag OF robot IS

COMPONENT sensors
        PORT(RV, RUV, ST: out bit);
END COMPONENT;

COMPONENT besturing
        PORT(B1, B0: in bit);
END COMPONENT;

signal started, idle: bit;
signal RV, RUV: bit;
signal B1, B0: bit;
signal turn: bit
BEGIN
started <= '0';
turn <= '0'
map1: sensors port map (RV=>RV, RUV=>RUV,ST=>idle);
map2: besturing port map (B1=>B1, B0=>B0);

starten:        PROCESS(START)
                BEGIN
                        IF (START = '1') THEN
                                started <= '1';
                        END IF;
                END PROCESS;

opdracht:       PROCESS(clk, idle)
                BEGIN
                        IF (started = '1') THEN
                        IF (clk'event and clk = '1') THEN
                                IF (idle = '1') THEN
                                        IF (RV = '1') THEN
-- bij rechtsdraaien zal altijd na draaien de rechterkant (daarvoor achterkant
-- van de robot) altijd vrij zijn, dus moeten we weten bij het rechts vrij zijn
-- of dit voor (moet dus nog draaien) of na (nu nog rechtuit rijden) de draai is ->
-- hiervoor voor de signaal 'turn' gebruikt.
                                                IF (turn = '0') THEN
                                                        turn <= '1';
                                                        B0 <= '1';
                                                ELSE
                                                        turn <= '0';
                                                        B0 <= '1';
                                                        B1 <= '1';
                                                END IF;
                                        ELSIF(RUV = '1')
                                                B0 <= '1';
                                                B1 <= '1';
                                        ELSE
                                                B1 <= '1';
                                        END IF;
                                END IF;
                        ELSE
-- aangezien de signalen B0 en B1 niet hoog hoeven te blijven gedurende actie, worden ze
-- in principe bij elke negative edge van de clk 'gereset'.
                                B0 <= '0'; B1 <= '0';
                        END IF;
                        END IF;
                END PROCESS;

END gedrag;
```

– o – o – o – o – o – o –