Tentamen Digitale Systemen (EE1410/ET1405) 22 juni 2011, 9.00 – 12.00 uur

Dit tentamen is **open boek** tentamen en bestaat uit 18 multiple choice (MC) vragen (63%) en 5 open vragen (37%). 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!) en uw handtekening te plaatsen.

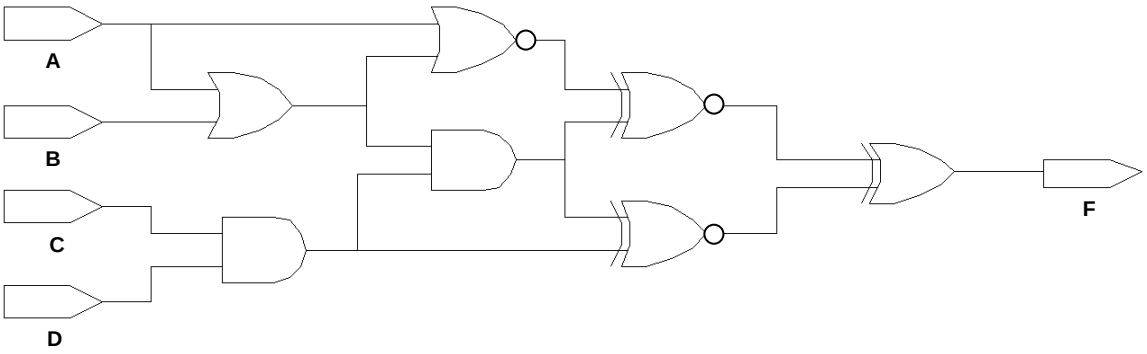
U mag het boek "Fundamentals of Digital Logic with VHDL Design", het VHDL boek (of alternatieve boeken hiervoor), het document RTL_Delta1 en eventuele prints van het **college** slides bij u hebben. Verder dus niets! Wij benadrukken dat u tijdens toetsen het tentamen dus **GEEN** gebruik mag maken van oude tentamens en toetsen (uitgezonderd tentamen en toets vragen die op de college slides staan). Gebruikt u deze toch dan zijn de tentamen fraude regels van toepassing.

Op grond van toetsresultaten kan het tentamencijfer met maximaal 1 punt verhoogd worden.

Succes!

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

Vraag 1

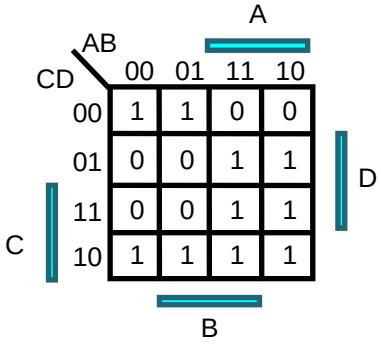


Voor bijgaand circuit gelden de volgende vertragingstijden:
NAND = AND = 3ns, NOR = OR = 5ns, XOR = XNOR = 7ns.
Oorspronkelijk geldt A = 0 en B = C = D = 1.
Dan wordt B = 0. Welke van de volgende uitspraken is correct?

a. Er treedt een 0-1-0 hazard op mbt. F
b. Er treedt een 1-0-1 hazard op mbt. F
c. F wordt 1 na 22 ns en verandert niet meer
d. Er treedt geen hazard op m.b.t. F

Vraag 2

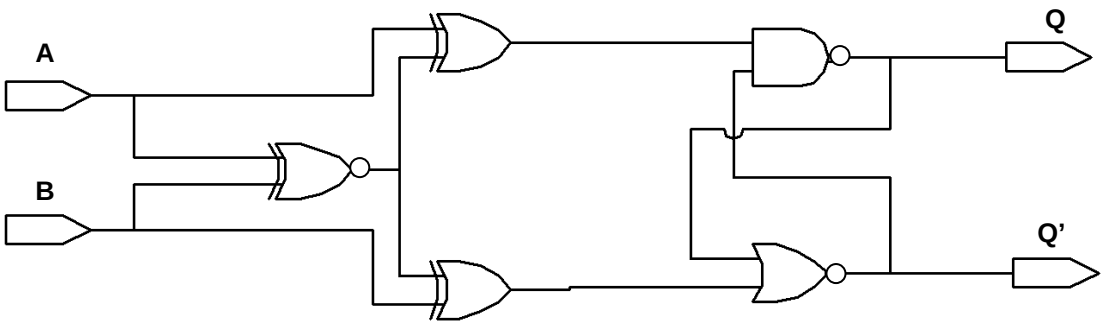
Gegeven de nevenstaande K-map (X betekent don't care). Hoeveel priemimplicanten bevat deze K-map en hoeveel daarvan zijn essentieel?



a. 3 priemimplicanten, waarvan 2 essentieel
b. 3 priemimplicanten, waarvan 3 essentieel
c. 4 priemimplicanten, waarvan 2 essentieel
d. 4 priemimplicanten, waarvan 3 essentieel

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

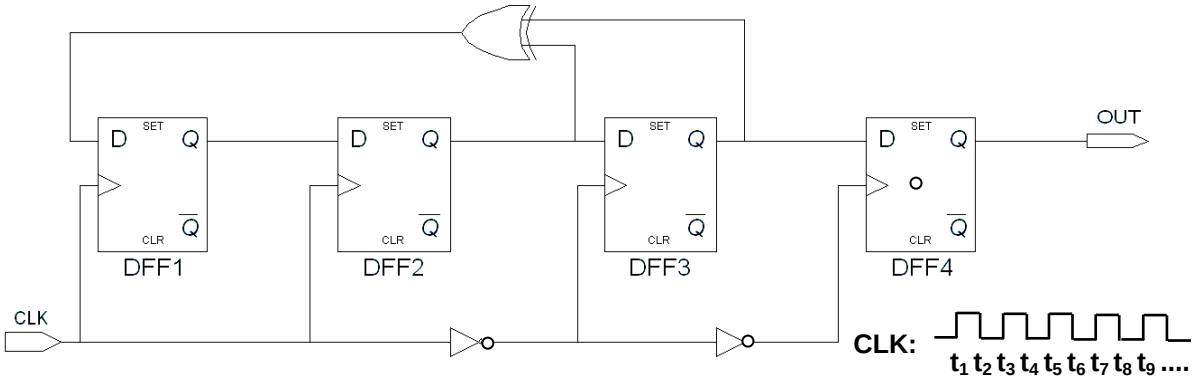
Vraag 4

Gegeven de berekening **C** = **A – B** in 2's complement notatie. Er geldt **C** = 110000 en **B** = 110100.
Wat is de waarde van **A**?

- **a**. 4
- **b**. -4
- **c**. -28
- **d**. Geen van bovenstaande antwoorden.

Vraag 5

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

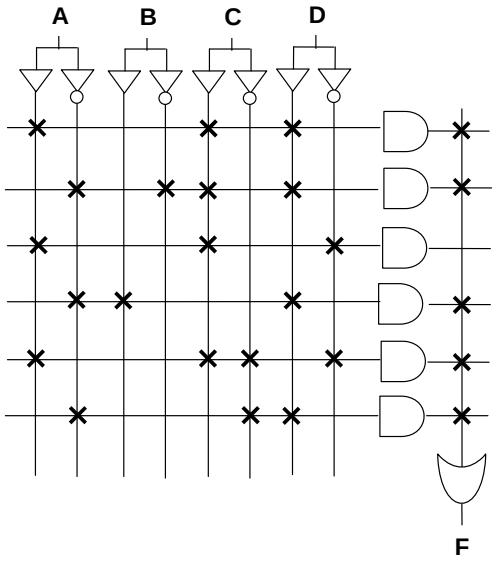


De klokperiode van de klok CLK is zodanig lang dat alle ingangen van de flipflops een tijd t_{setup} voor de volgende klokflank gestabiliseerd zijn.
Indien voor tijdstip t_1 alle Qs **1** zijn, wanneer wordt dan voor het eerst **0** op 'OUT' waargenomen? Op of direct na:

- **a.** t_4
- **b.** t_5
- **c.** t_6
- **d.** t_7 of later.

Vraag 6

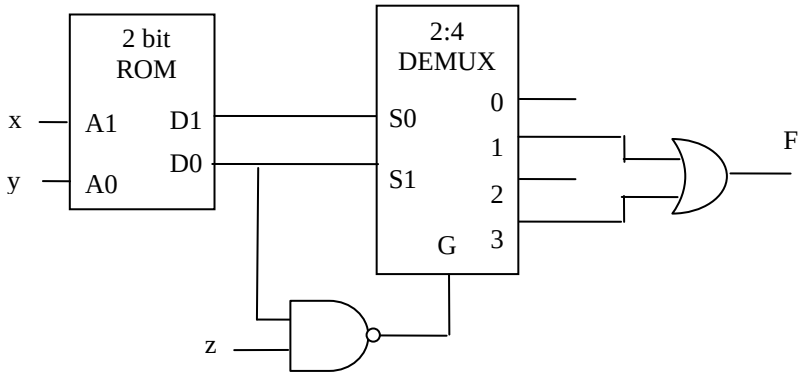
Welke van de volgende vereenvoudigingen voor de uitgang van nevenstaande PLA is correct?



- **a.** F = A'D + CD
- **b.** F = A'D + AC
- **c.** F = A'D + AD'
- **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	1
1	0	1	1
1	1	0	0

Welke van de volgende vereenvoudigingen is correct?

- **a.** F = yz'
- **b.** F = xy+z'
- **c.** F = xy'
- **d.** Geen van bovenstaande antwoorden.

Vraag 8

Gegeven de volgende Delta I assembly code:

```
        set  c
        ld   00001111b
        st   R1
        xor  01010101b
        add  R1
        bc   lbl1
        add  00000100b
lbl1:   add  00010000b
```

Wat is de waarde van A na de executie van de laatste instructie?

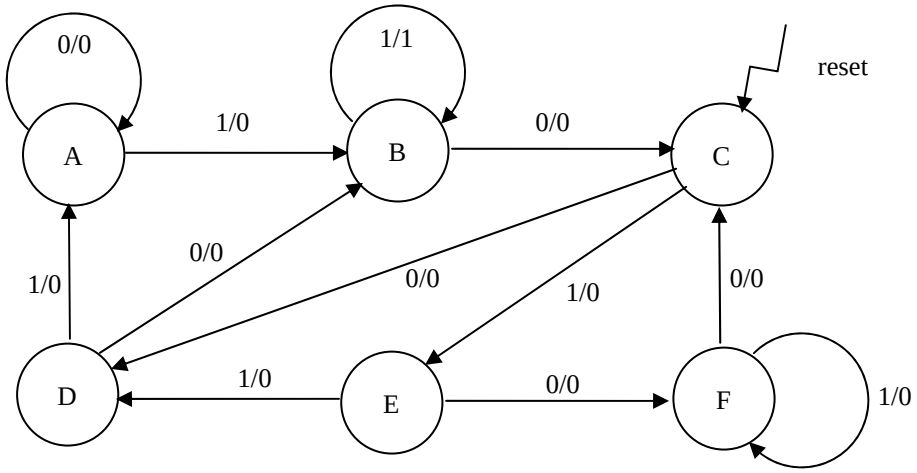
a. $A = 77_H$ **b.** $A = 7A_H$ **c.** $A = 7E_H$ **d.** $A = 7F_H$

Vraag 9.

De formule F = A.B.C + A.B'.D'+ A.C'.D'+ B.C.D' + A'.B.C.D is te vereenvoudigen tot:

- **a.** A.D'+ B.C
- **b.** A.(B + D')
- **c.** A + B.C
- **d.** A.C + B.C.D

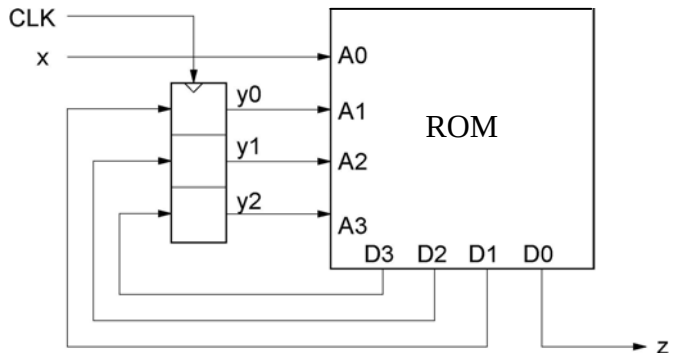
Vraag 10.



Gegeven bovenstaand toestandsdiagram van een machine met een ingang x en een uitgang z. Vertrekkende vanuit de reset toestand: welke ingangsreeks (laatste bit van de reeks staat rechts) geeft een 1 aan de uitgang ?

- **a.** 00001
- **b.** 00011
- **c.** 11001
- **d.** 11111

Vraag 11.



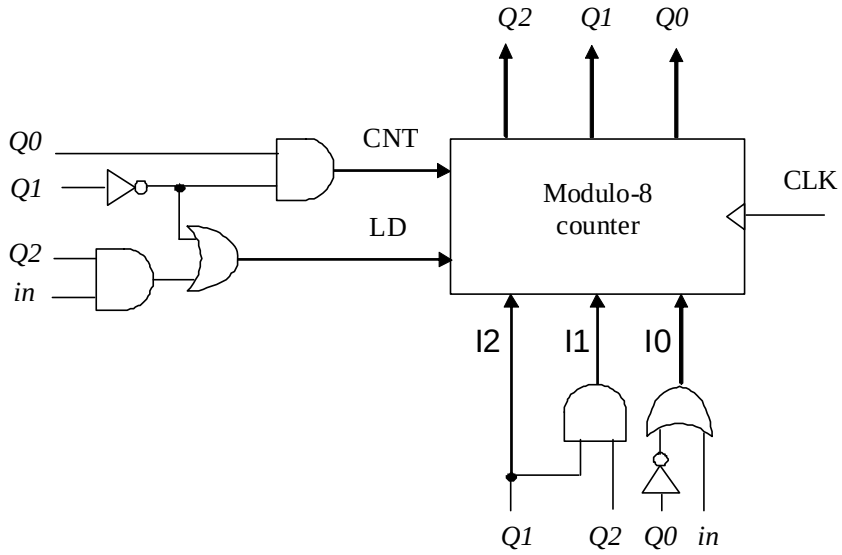
	y2	y1	y0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0
F	1	0	1

Als de toestandscodering uit de tabel rechts geldt voor het toestandsdiagram van vraag 10, en een realisatie m.b.v. een ROM wordt uitgevoerd zoals hierboven links is aangegeven, wat is dan de inhoud van de ROM op adres 1001 ?

- **a.** 1010
- **b.** 1100
- **c.** 0110
- **d.** Geen van bovenstaande antwoorden.

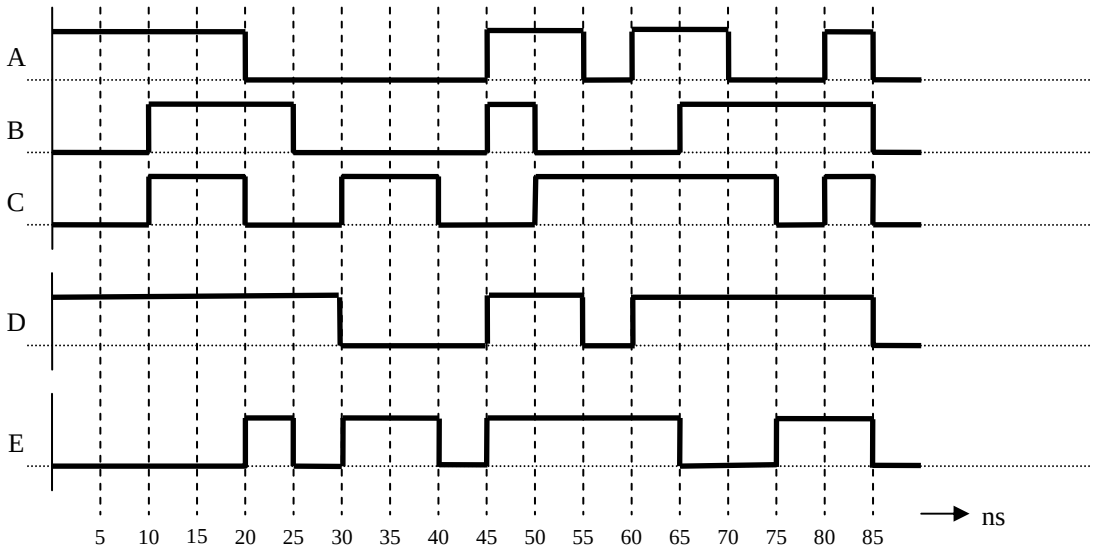
Vraag 12.

Nevenstaand schema is de schakeling van een FSM met ingangsignaal *in* en toestanden S0, S1, S2 … S7. De toestanden komen op de volgende manier overeen met de waarden van *Q2 Q1 Q0:* S0=000, S1=001, S2=010, S3=011, S4=100, S5=101, S6=110 en S7=111. De counter heeft 3 modi, nl. *LOAD* (LD=1), *COUNT* (CNT=1 en LD=0), en onthouden (CNT=0 en LD=0). Bij *I2 I1 I0* wordt de te laden toestand ingevoerd. Wat zijn mogelijke opvolgende toestanden voor *S5* ?



- **a.** S6 en S7
- **b.** S0 en S1
- **c.** S6, S0 en S1
- **d.** S5, S6 en S7

Vraag 13.



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

```
L1:    process (sensitivity-list) is -- de sensitivity-list ontbreekt
       begin
              D <= (A or B);
       end process;

L2:    process (sensitivity-list) is -- de sensitivity-list ontbreekt
       begin
              E <= (B xor C);
       end process;
```

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 14.

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

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

Vraag 15.

Gegeven zijn de volgende twee processen:

```
proc1:  process (x, y, z) is
        variable var_s1, var_s2: std_logic;
        begin
             var_s1 := x and y;
             var_s2 := var_s1 xor z;
             res <= var_s1 nand var_s2;  -- res is een port
        end process;

proc2:  process (x, y, z, sel) is
        begin
             w <= '1';
             if (sel = '0') then
                    w <= x xnor y;
             endif;
             v <= (x and y) and z;
        end process;
```

Met betrekking tot synthese en inference van latches, welk van de volgende uitspraken is correct?

- **a.** Bij proc1 wordt **niet** een latch geinferred en bij proc2 wordt **niet** een latch geinferred.
- **b.** Bij proc1 wordt **niet** een latch geinferred en bij proc2 wordt **wel** een latch geinferred.
- **c.** Bij proc1 wordt **wel** een latch geinferred en bij proc2 wordt **niet** een latch geinferred.
- **d.** Bij proc1 wordt **wel** een latch geinferred en bij proc2 wordt **wel** een latch geinferred.

Vraag 16.

```
entity D_flipflop is
       port (input, klok, set: in bit; output1, output2: out bit);
end D_flipflop;

architecture werking of D_flipflop is
begin
P1:    process (klok, set) is
       begin
              if (klok'event and klok = '1') then
                     if (set = '0') then
                            ...... (invulregel 1) ......;
                     else
                            ...... (invulregel 2) ......;
                     endif;
              endif;
       end process;
end werking;
```

De bovenstaande VHDL code beschrijft een D flipflop met een set mogelijkheid. Gevraagd wordt te beslissen wat er op de plaats van de invulregels 1 en 2 moet staan, door een keus te maken uit de onderstaande 4 mogelijkheden:

- **a.** op invulregel 1 moet staan: `output1 <= input; output2 <= not(output1);`
  op invulregel 2 moet staan: `output1 <= '0'; output2 <= not(output1);`
- **b.** op invulregel 1 moet staan: `output1 <= input; output2 <= not(output1);`
  op invulregel 2 moet staan: `output1 <= '0'; output2 <= '1';`
- **c.** op invulregel 1 moet staan: `output1 <= input; output2 <= not(input);`
  op invulregel 2 moet staan: `output1 <= '1'; output2 <= '0';`
- **d.** op invulregel 1 moet staan: `output1 <= input; output2 <= not(input);`
  op invulregel 2 moet staan: `output1 <= '1'; output2 <= not(output1);`

Vraag 17.

Gegeven de volgende VHDL code waarin een full adder wordt gemodelleerd. Hierin wordt gebruik gemaakt van twee half adders en een or-poort.

```
entity half_adder is
       generic (gate_delay : Time := 6 ns);
       port (a, b : in bit;  sum, carry : out bit);
end half_adder;

architecture my_half_adder_arch of half_adder is
begin
       carry <= a and b after 5 ns;
       sum   <= a exor b after gate_delay;
end my_half_adder_arch;

entity full_adder is
       generic (gate_delay : Time := 4 ns);
       port (a, b, c_in : in bit; sum, carry : out bit);
end full_adder;

architecture structural of full_adder is
       component half_adder is
              generic (gate_delay : Time := 6 ns);
              port (a, b : in bit; sum, carry : out bit);
       end component;
       signal s1, s2, s3 : bit;
begin
H1: half_adder      generic map (gate_delay => gate_delay)
              port map (a => a, b => b, sum => s1, carry => s3);
H2: half_adder      port map (a => s1, b => c_in, sum => sum, carry => s2);
c_out <= s2 or s3 after 5 ns;
end structural;
```

Wat is de langste delay (in ns) in de full_adder van de ingangen naar de uitgangen in de gegeven code?

- **a.** 10 ns
- **b.** 13 ns
- **c.** 14 ns
- **d.** 16 ns

Vraag 18.

In een architecture hebben we te maken met de vijf std_logic signalen a, b, c, d en e. Signaal b heeft de waarde ‘H’, c heeft de waarde ‘L’ en d heeft de waarde ‘L’. In de architecture komen de volgende twee concurrent statements voor:

```
e <= a and b;
e <= a or c or d;
```

Welke waarde krijgt e als signaal a veranderd naar de waarde ‘H’?

- **a.** e krijgt de waarde ‘X’.
- **b.** e krijgt de waarde ‘0’.
- **c.** e krijgt de waarde ‘1’.
- **d.** geen van de drie mogelijkheden is goed.

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";
```

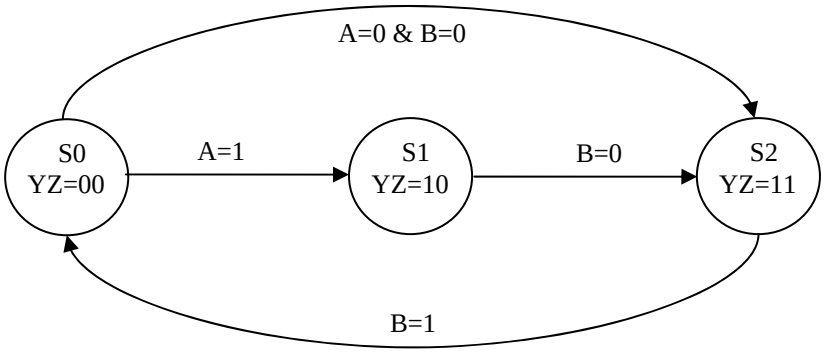
B. Open vragen (gewicht: 37%)

Vraag 19 (8 %)
Stel u bent elektrotechnicus bij een slopersbedrijf en u wordt gevraagd een elektronisch ontstekeningsmechanisme te ontwerpen voor het ontsteken van dynamiet. De besturing moet de volgende eigenschappen hebben:

- Wanneer een hendel wordt ingedrukt krijgt een signaal S de waarde 1. Wanneer de hendel daarna wordt losgelaten krijgt S de waarde 0 en dient een timer te gaan lopen.
- De timer wordt gereset met TS=1 en begint te lopen wanneer TS=0.
- Wanneer de timer afloopt geeft hij een uitgangsignaal TR=1 (anders TR=0)
- De besturing moet daarop een signaal F=1 geven van 1 klokperiode lang om het dynamiet te ontsteken.
- Wanneer de timer loopt, of afloopt, is het ten alle tijden mogelijk om met een signaal R=1 de besturing terug te brengen in de rusttoestand.

Gevraagd wordt om d.m.v. een Finite State Diagram (FSD) een beschrijving van de besturing te ontwerpen. Gebruik zo weinig mogelijk toestanden, maar zorg uiteraard dat de machine in alle gevallen correct zal werken: nodeloos te zeggen dat een klein foutje aanleiding kan geven tot rampzalige gevolgen ! De beschrijving moet een Finite State Machine volgens het Moore model zijn. Geef duidelijk de in en uitgangsignalen aan.

Vraag 20 (8 %)
Gegeven het volgende toestandsdiagram van een Finite State Machine met toestanden S0, S1 en S2, ingangen A en B en uitgangen Y en Z, en een bijbehorende state assignment.



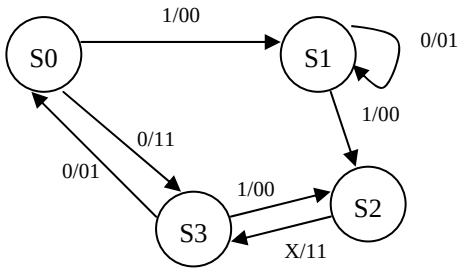
State	y1	y0
S0	0	0
S1	0	1
S2	1	0

Teken de K-maps voor de next-state variabelen $\mathbf{y_1}^+$ en $\mathbf{y_0}^+$, en leidt voor deze signalen minimale logische expressies af.

Vraag 21 (5 %)
Schrijf in de assembly taal van de Delta 1 processor een programma dat de waarde van register R2 aftrekt van de waarde van register R1, en de uitkomst opslaat in register R3. Dus: R3 := R1 – R2. Ga uit van een 2's complement representatie van de waarden in de registers. (Het kan in 5 instructies.)

Vraag 22 (10%)

Gegeven is het volgende toestandsdiagram.



Schrijf in VHDL (geef de entity en architecture) het model van deze controller. LET OP: De controller moet zijn geklokt (*negative edge-triggered*) er moet een *asynchrone* reset gegeven kunnen worden waardoor de controller weer naar state S3 teruggaat.

U kunt gebruik maken van het volgende template:

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

entity controller is
      port ();                          -- port declaraties zijn weggelaten
end entity controller;

architecture behavior of controller is
type statetype is (S0, S1, S2, S3);     -- declaratie van een nieuwe type
signal state : statetype;
...                         -- U kunt hier meer signalen/variabelen declareren
begin
label: ...                          -- architecture body is weggelaten
end architecture behaviour;
```

In dit template wordt gebruik gemaakt van het feit dat binnen VHDL nieuwe types kunnen worden gedeclareerd. Dit betekent dus dat signal `state` de waarden S0, S1, S2 en S3 kan aannemen. Namen voor ports en signals mag u verder zelf verzinnen.

Vraag 23 (6%)

Alle moderne processoren (een paar uitgezonderd) gebruiken een pipeline om hun prestatie te verbeteren. Het idee is om alle instructies op te delen in overeenkomstige stappen (English: pipeline stages), bijvoorbeeld 5. Dit zorgt ervoor dat als een instructie klaar is met stap 1, deze verder kan met stap 2. Op hetzelfde moment kan een nieuwe instructie worden gestart in stap 1. Het starten van stap 1 en het overgaan van de ene stap naar de andere wordt gecoördineerd door de klok. Je kan dit ook vergelijken met moderne autofabriek waarbij het bouwen van een auto (=instructie) in meerdere stappen wordt opgedeeld. Wanneer men klaar is met stap 1 (ook wel genoemd station), kan met stap 2 worden gestart terwijl men kan beginnen met het bouwen van een nieuwe auto in stap 1. Het equivalent van de processor klok is hier een echte tijdklok.

De eerste twee stappen van zo'n processor zijn vaak: fetch (ophalen van de instructie uit een geheugen/buffer) en decode (vertalen van instructie naar interne controle signalen - denk aan sequentiële systemen die controle signalen genereren voor combinatorische systemen). Een hardware ontwerper tracht een eerste opzet te maken van zo'n processor in VHDL en schrijft de volgende stukje VHDL als eerste begin – veel details zijn weggelaten:

```
fetch: process (clk) is
       begin
       if (clk'event and clk='1') then
            instructie <= geheugen(i);  -- een instructie wordt uit het geheugen gehaald
                                        -- met index i. de berekening van index i valt
                                        -- buiten de scope van deze vraag.
       end if;
       end process fetch;

decode: process (clk) is
       begin
       if (clk'event and clk='1') then
            case (instructie) is
                 when ins1 =>                    -- hier wordt aangenomen dat ins1, ins2,
                    controlesignaal_01 <= '0';   -- etc. voorgedefinieerd zijn.
                    controlesignaal_02 <= '1';   -- de controle-signalen zijn alleen ter
                                                 -- illustratie
                 when ins2 =>
                    controlesignaal_01 <= '1';
                    controlesignaal_02 <= '0';
                 -- etc.
            end case;
       end if;
       end process decode;
```

Neem hierbij aan dat het instructie-signaal een vector is van bits en dat een van te voren gedefinieerde codering bestaat van de instructies zoals (ins1, ins2, etc.).

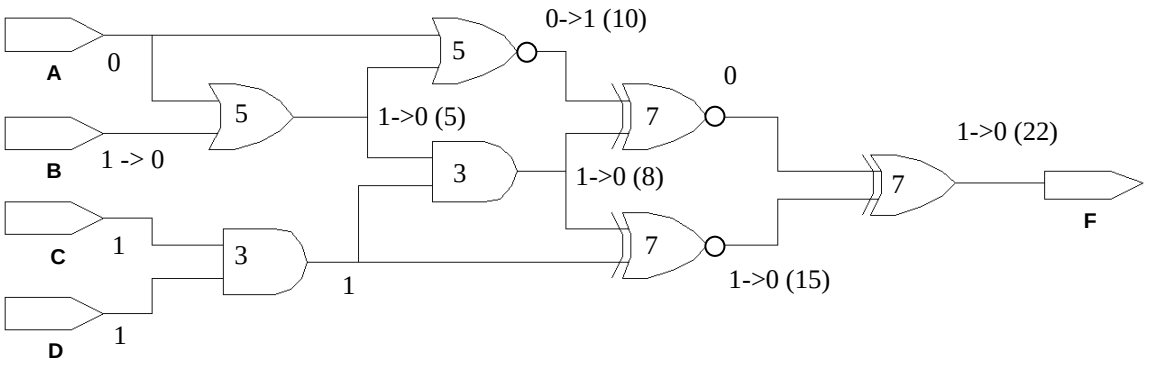
Let uit waarom deze aanpak wel of niet zal werken. Focus hierbij ook of de code wel of niet correct zal simuleren en leg duidelijk uit waarom.

– o – o – o – o – o – o – o –

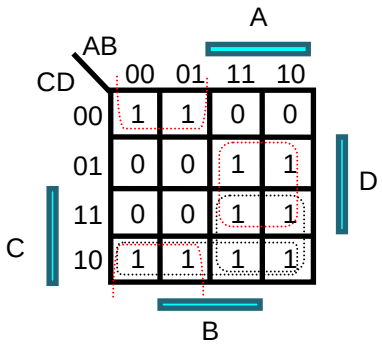
Uitwerkingen Tentamen Digitale Systemen (EE1410/ET1405) 22 juni 2011

MC-vragen:

Vraag 1: d

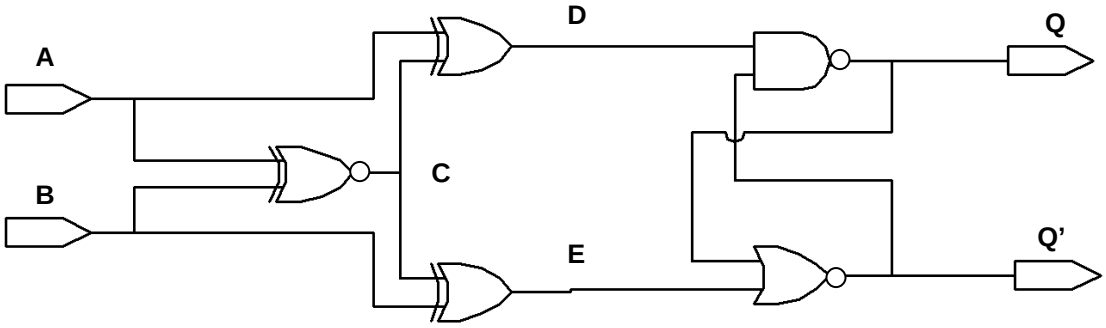


Vraag 2: c



4 priemimplcanten, waarvan 2 essentieel (in rood).

Vraag 3: b



A	B	C	D	E	Q	Q'	
0	**0**	**1**	**1**	**1**	**1**	**0**	**set**
0	**1**	**0**	**0**	**1**	**1**	**0**	**set**
1	**0**	**0**	**1**	**0**	**Q**	**Q'**	**hold**
1	**1**	**1**	**0**	**0**	**1**	**0**	**set**

Vraag 4: c

C = 110000 = -16 en **B** = 110100 = -12, dus -16 = A – (-12), oftewel A = -28

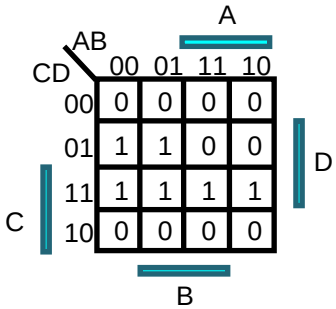
Vraag 5: b

Merk op dat FF1, FF2 en FF4 reageren op de positieve klokflank en FF3 op de negatieve klokflank

	D1 (Q2 ⊕ Q3)	Q1	D2	Q2	D3	Q3	D4	Q4
initieel	0	1	1	1	1	1	1	1
na t1	0	0	0	1	1	1	1	1
na t3	1	0	0	0	0	1	1	1
na t4	0	0	0	0	0	0	0	1
na t5	0	0	0	0	0	0	0	0

Vraag 6: a

Van PLA: F = ACD + A'B'CD + A'BD + A'C'D
Invullen in Karnaugh-map geeft:



Dus: F = A'D + CD

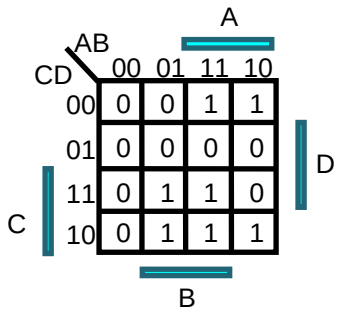
Vraag 7: d

S0 = D1 = A0' = y'
S1 = D0 = (A1.A0)' = (xy)'
G = (D0 . z) ' = ((xy)' z)' = xy + z'
F = (S1'.S0 + S1.S0).G = S0 . G = y'(xy + z') = y'z'

Vraag 8: c

```
                                    A          C
            set   c                 ?          1
            ld    00001111b         00001111b  1
            st    R1                00001111b  1
            xor   01010101b         01011010b  1
            add   R1                01101010b  0
            bc    lbl1              01101010b  0
            add   00000100b         01101110b  0
lbl1        add   00010000b         01111110b  0
```

Vraag 9: a



A.D'+ B.C

Vraag 10: d

Via toestanden E D A B B

Vraag 11: c

Adres 1001 wil zeggen, huidige toestand is E (y2,y1,y0 = 100) en ingangswaarde x is 1. Uit het teostandsdiagram volgt dan: de volgende toestand is D (011) en de uitgangswaarde z is 0, oftwel D3,D2,D1,D0 = 0110.

Vraag 12: b

Voor S5 geldt: *Q2 Q1 Q0* = 101.
In dat geval: CNT=1 en LD=1, wat betekent dat de counter zal inladen wat op *I2 I1 I0* staat. Afhankelijk van de waarde van in zal dat zijn 000 of 001. Wanneer deze waarden worden ingeladen is de volgende toestand dus S0 of S1
=> antwoord b

Vraag 13: b
Ga na welke events de processen zullen trigger-en en bepaal bij elk event de nieuwe waarde. Als dit correct wordt gedaan, dan zal blijken dat A en C in de sensitivity-lijst moeten staan.

⇨ Antwoord b

Vraag 14: d
Dezelfde redenering als bij vraag 13. Echter, bij antwoorden a, b, en c staat altijd of A of C en dus moet er op 80 ns een event plaatsvinden en dit zal resulteren de waarde ‘0’. Dit komt niet overeen met de getoonde waveform.

⇨ Antwoord d

Vraag 15: a

⇨ Antwoord a

Vraag 16: c

⇨ Antwoord c

Vraag 17: c
Het langste pad behelst de exor van H1, de and van H2, en de laatste or. Voor half_adder H1 wordt voor de exor poort een gate delay van 4 ns ingevuld. Voor de and poort van H2 geldt 5ns. Voor de laatste or poort geldt 5 ns. Dus: 4 + 5 + 5 = 14 ns.

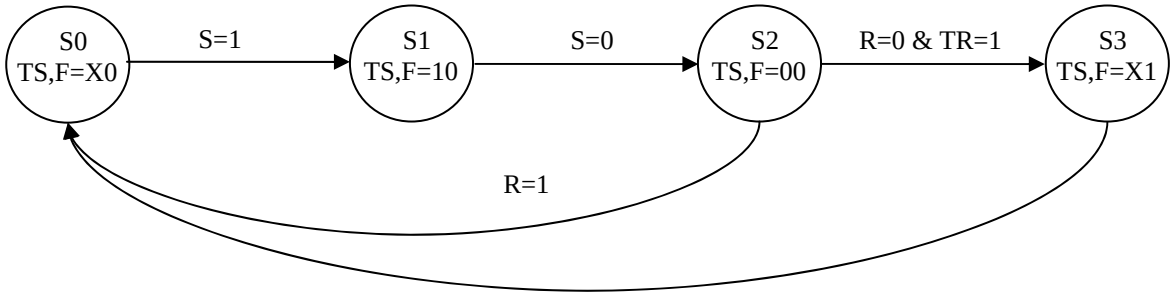
⇨ Antwoord c

Vraag 18: c
(‘H’ and ‘H’) resulteert in ‘1’ en (‘H’ or ‘L’ or ‘0’) resulteert in ‘1’. De resolutiefunctie van ‘1’ en ‘1’ resulteert in ‘1’.

⇨ Antwoord c.

Open vragen

Vraag 19

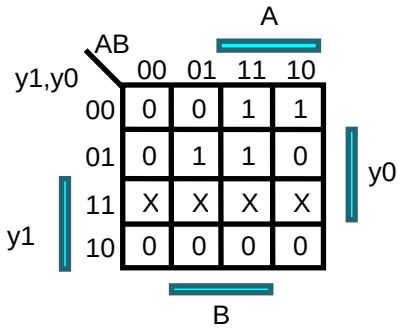


R kan ook gebruikt worden als reset signaal.

Vraag 20

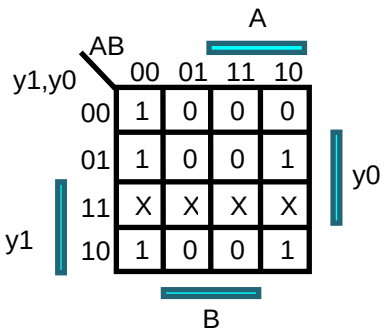
K-maps :

y0$^+$:



y0$^+$ = y0 B + y1' y0' A

y1$^+$:



y1$^+$ = A'B' + y0 B' + y1 B'

Vraag 21

```
set  c
ld   R2           : alle bits inverteren en er 1 bijtellen
xor  11111111b    : zorgt voor het optellen van een
add  R1           : negatieve waarde van R2 bij R1.
st   R3
```

Vraag 22

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

entity controller is
port ( reset, clk, x : in std_logic;
       z : out std_logic_vector (1 downto 0));
end entity controller;

architecture behavior of controller is
type statetype is (S0, S1, S2, S3);     -- declareren van een nieuwe type
signal state, next_state : statetype;
begin
comb_process:
       process (state, x) is
       begin
       case state is
              when S0 =>
                     if x = '0' then
                            next_state <= S3;
                            z <= '11';
                     else
                            next_state <= S1;
                            z <= '00';
                     end if;
              when S1 =>
                     if x = '0' then
                            next_state <= S1;
                            z <= '01';
                     else
                            next_state <= S2;
                            z <= '00';
                     end if;
              when S2 =>
                            next_state <= S3;
                            z <= '11';
              when S3 =>
                     if x = '0' then
                            next_state <= S0;
                            z <= '01';
                     else
                            next_state <= S2;
                            z <= '00';
                     end if;
              end case;
       end process comb_process;

clk_process:
       process(reset,clk)
       begin
       if reset = '1' then
              state <= S3;
       elsif (clk'event and clk = '0') then
              state <= next_state;
       end if;
       end process clk_process;
end architecture behaviour;
```

Vraag 23

De code zal werken omdat in beide processen de klok wordt gebruikt en de manier van opschrijven duidt op het gebruik van een flipflop waarin de instructies en de controlesignalen worden opgeslagen. Deze worden dan in de volgende stappen gebruikt. Waarom dit ook goed simuleert? Dit komt door de delta delay. De "assignment" van de nieuwe waarden van de instructies en controlesignalen worden pas een delta-delay uitgevoerd waardoor de processen altijd de "oude" waarden "leest" en niet gelijk de nieuwe waarden. Als dit niet zo was, dan zou het uitvoeren van bv. de process decode na de fetch process leiden tot het uitvoeren van de nieuwe instructie en niet eerst de oude.

– o – o – o – o – o – o – o –