Afspraken:

- Vermeld op elk blad naam en voornaam. Dit is zeer belangrijk voor de schriftelijke vraag! Alle pagina's nummeren.
- De voorbereidingstijd voor de eerste twee vragen samen is 0,5 uur, voor de derde vraag 2 uur, waarvan maximum 1 uur voor deel (a). De mondelinge bespreking ervan begint respectievelijk 0,75 en 2,5 uur na het begin van het examen. De vierde vraag wordt schriftelijk opgelost en hiervoor is 1 uur voorzien. De vierde vraag wordt ten laatste 4,25 uur na het begin van het examen afgegeven.
- Toiletbezoek is pas mogelijk na de de mondelinge bespreking van de eerste twee vragen.
- Papier en appendix A zijn voorzien. Papier kan je bijvragen. Enkel eigen schrijfgerief en eten/drank mogen bijkomend gebruikt worden. Dus geen rekenmachine of communicatiemiddelen zoals gsm.
- Vul alvast het aanwezigheidsformulier in. Je laat dit aanvullen bij elke mondelinge bespreking en geeft dit af samen met vraag 4. Het aantal kladbladen moet niet vermeld worden.
- De opgave en appendix A worden samen met de kladbladen apart afgegeven. Ook het niet gebruikte papier wordt teruggenomen. De kladbladen worden na het examen door de docent vernietigd.

Vragen:

1. Welke methoden kan je gebruiken om een combinatorische schakeling in meer dan twee lagen te ontwerpen?

2. Wat is 'clock skew', hoe ontstaat het, wat zijn de gevolgen en hoe kan je het voorkomen?

3. Ontwerp een FSMD die od = 3 cos(id) berekent zoals dit op de achterkant van dit blad in VHDL beschreven is. Bij de realisatie mag je enkel bouwblokken gebruiken waarvan de hardware implementatie in de les bestudeerd is.
   Alle `real` getallen worden in hardware voorgesteld als getallen met een vaste komma. Voor de ingang `id` wordt 8 bits na de komma gebruikt en voor de uitgang `od` wordt 16 bits na de komma gebruikt. Verder worden voor alle getallen zoveel bits gebruikt als nodig om de uitgang met de gewenste nauwkeurigheid te berekenen.

   a) Teken een ASM-schema voor deze schakeling, die de vereisten van de VHDL-beschrijving qua tijdsgedrag respecteert. Probeer zo weinig mogelijk toestanden te gebruiken. Indien één enkel ASM-schema te ingewikkeld lijkt, mag je dit opsplitsen in meerdere ASM-schema's.

   b) Ontwerp het datapad tot op RTL-niveau. Minimalisering is niet expliciet nodig, maar probeer wel zo weinig mogelijk hardware te gebruiken, zonder evenwel nog het ASM-schema te wijzigen. Vergeet ook niet het aantal bits bij iedere verbinding te vermelden.

   c) Beschrijf het controlegedeelte met een toestandsdiagram. Als er meerdere ASM-schema's gebruikt worden volstaat het om enkel het diagram met de meeste toestanden te beschrijven.

4. Maak de goedkoopst mogelijke IC realisatie van de FSM, beschreven in nevenstaande tabel, met ingangen I & J en uitgangen X & Y. Je mag enkel gebruik maken van JK-flipflops en NAND- en/of NOR-poorten met 2 ingangen. Maak gebruik van Karnaugh-kaarten om alle functies te bepalen. Vergeet de nuttige bijkomende informatie (zoals het waarom van een keuze) niet te vermelden!

Huidige toestand	Volgende toestand / XY		
	$IJ = 00$	$IJ = 01$	$IJ = 10$
S_0	$S_0/10$	$S_4/01$	$S_1/10$
S_1	$S_0/01$	$S_1/10$	$S_4/10$
S_2	$S_2/10$	$S_4/01$	$S_1/10$
S_3	$S_3/10$	$S_3/01$	$S_2/10$
S_4	$S_2/01$	$S_3/10$	$S_0/10$

```
library ieee; use ieee.math_real.all;     -- definieert de constante math_deg_to_rad

entity fsmd is
  port (clk, rst : in bit;  LD : in integer range -1 to 3;
        id : in real range -30.0 to 30.0;
        od : out real;  oa : out bit);
end entity fsmd;

architecture behav of fsmd is
  signal n, rsti : natural;
  signal x : real;
begin
  p1: process is
    variable x2, y : real;
  begin
    wait until clk = '1';
    while n <= 0 loop
      wait until clk = '1';
    end loop;
    oa <= '0';
    x2 := (x**2)/2.0; y := 0.5;
    if n < 3 then y := 3.0; end if;
    for i in 1 to n-1 loop
      exit when rsti > 0;
      y := 3.0 - y*x2;
      wait until clk = '1';
    end loop;
    n <= 0;
    if rsti = 0 then
      od <= y; oa <= '1';
    end if;
  end process p1;

  p2: process (clk, rst, id) is
    variable r : real;
  begin
    r := id * math_deg_to_rad;
    if rst = '0' then
      n <= 0; rsti <= 1; oa <= '0';
    elsif clk'event and clk = '1' then
      rsti <= 0;
      if n = 0 then
        case LD is
          when -1     => x <= r;
          when  0     => null;
          when others => n <= LD;
        end case;
      end if;
    end if;
  end process p2;
end architecture behav;
```