UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	**INF3430/4431**
Eksamensdag:	**3. Desember 2018**
Tid for eksamen:	**09.00-13.00**
Oppgavesettet er på 6 sider.	
Vedlegg:	**1**
Tillatte hjelpemidler:	**Alle trykte og skrevne, samt kalkulator**

Kontroller at oppgavesettet er komplett før du begynner å besvare spørsmålene

Oppgaveteksten består av flervalgsoppgave 1–3, 6 og 8-9 som skal besvares på skjemaet som er vedlagt etter oppgaveteksten og oppgavene 4-5, 7 og 10 som besvares på vanlige ark.

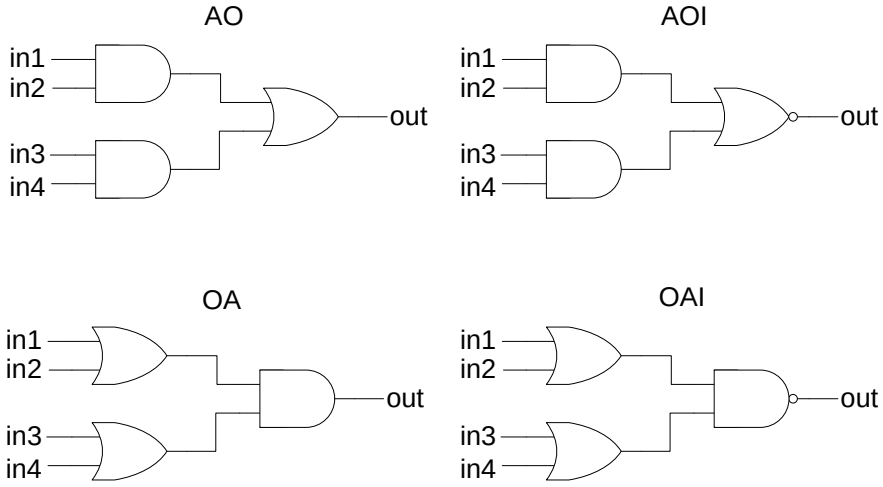
Oppgavenes vekt er vist i parentes bak oppgavenes nummer.

Det er ikke nødvendig å gjenta VHDL kode som allerede står i oppgaveteksten.

Generelt for flervalgsoppgavene:
Hver oppgave består av et tema og en del utsagn hver angitt med en stor bokstav. Oppgavene besvares ved å merke tydelige kryss (X) i rett kolonne for riktig svaralternativ (dvs. at et utsagn er sant) i skjemaet i vedlegg 1. Det er alltid *minst en* riktig avmerking for hver oppgave, men det kan være *flere* riktige avmerkinger. *For å få best karakter skal man sette flere kryss i en oppgave hvis det er flere riktige utsagn.* Det gis 1 poeng for hver avkrysning der det skal være avkrysning. Det gis -1 poeng for hver avkrysning der det ikke skal være avkrysning. Mangel på kryss der det skal være kryss gir også -1 poeng. Du kan benytte høyre kolonne i oppgaveteksten til kladd. **Skjema påført ditt kandidatnummer i vedlegg 1 er din besvarelse.**

Oppgave 1 (3 %)

Figuren under viser de kombinatoriske kretsene and-or (AO), and-or-invert (AOI), or-and (OA) og or-and-invert (OAI).



En 4-input Xilinx LUT med innhold **"F888"** (hex) realiserer en:	A	and-or (AO)	
	B	and-or-invert (AOI)	
	C	or-and (OA)	
	D	or-and-invert (OAI)	

Oppgave 2 (3 %)

FPGA teknologi	A	MicroBlaze er en soft prosessorkjerne	
	B	AXI4 er et eksempel på en delt prosessor bus.	
	C	AXI4Lite har uavhengige lese og skrive kanaler.	
	D	Etter en tid i metastabil tilstand vil alle flip-flop'er alltid gå til verdien '1'.	

Oppgave 3 (3 %)

FPGA teknologi	A	I en kombinatorisk process skal alle signaler som leses og tilordnes verdi (gis verdi) i process'en være med i sensitivitetslista.	
	B	I en sekvensiell process med asynkron reset skal bare reset og klokkesignalet være med i sensitivitetslista.	
	C	SRAM FPGA'er kan reprogrammeres etter leveranse til kunde.	
	D	En ASIC kan reprogrammeres	

Oppgave 4 (6 %)

Det skal i denne oppgaven lages en modul *compute_comb* som regner ut *a* multiplisert med *b* pluss *c* multiplisert med *d*, dvs. *result= (a*b) + (c*d)*. Entiteten til compute_comb er oppgitt under. Det skal utføres unsigned aritmetiske operasjoner på operandene a, b, c og d.

Utgangen *result* skal ha antall bit slik at summen alltid blir riktig, dvs. at overflow **ikke** kan inntreffe. Endre signalet *result* i *entity compute_comb* slik at deklarasjonen av std_logic_vector får det antall bit som kreves.

Implementer arkitekturen til modulen *compute_comb* vist under som en kombinatorisk prosess i syntetiserbar VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity compute_comb is
  port
    (a      : in  std_logic_vector(15 downto 0);
     b      : in  std_logic_vector(15 downto 0);
     c      : in  std_logic_vector(15 downto 0);
     d      : in  std_logic_vector(15 downto 0);
     result : out std_logic_vector(?? downto 0)); -- Change "??"
end entity compute_comb;

architecture rtl of compute_comb is
begin

  < Implementer compute_comb prosessen >

end architecture rtl;
```

Oppgave 5 (15 %)

Modulen *compute_comb* i oppgave 4 skal i denne oppgaven endres til en synkron modul *compute_seq* med resetsignalet *rst* og klokkesignalet *clk* som regner ut result= (a*b) + (c*d). Inngangene a, b, c og d er synkrone med klokkesignalet clk. Utgangen *result* skal også nå ha antall bit slik at summen alltid blir riktig, dvs. at overflow ikke kan inntreffe. Entiteten til compute_seq er oppgitt under.

Signalet *vdata* bestemmer når inngangene *a, b, c* og *d* har gyldige («valid») data, dvs. *vdata='1'*. Utgangssignalet *vresult* skal settes lik '0' når utgangssignalet *result* **ikke** er valid og lik '1' når utgangssignalet *result* er valid. Når utgangssignalet *result* ikke er gyldig skal verdien settes til null.

Det viser seg at det blir timingfeil under implementasjonen i valgt teknologi og med den valgte klokkefrekvensen. Det må derfor innføres pipelining, slik at multiplikasjon og addisjon

utføres på ulike klokkeperioder for å oppnå timingkravet. Det kan utføres flere multiplikasjoner og addisjoner i parallell i hver klokkeperiode.

Implementer arkitekturen til modulen compute_seq som er vist under som en sekvensiell prosess i syntetiserbar VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity compute_seq is
  port (
    rst     : in  std_logic;
    clk     : in  std_logic;
    a       : in  std_logic_vector(15 downto 0);
    b       : in  std_logic_vector(15 downto 0);
    c       : in  std_logic_vector(15 downto 0);
    d       : in  std_logic_vector(15 downto 0);
    vdata   : in  std_logic;
    result  : out std_logic_vector(?? downto 0);
    vresult : out std_logic);
end entity compute_seq;

architecture rtl of compute_seq is

  < Implementer VHDL koden her >

end architecture rtl;
```

Oppgave 6 (4 %)

Hvor mange registere (dvs. flip-flop'er) får den implementerte pipelinede arkitekturen rtl i oppgave 5	A	96 registere	
	B	97 registere	
	C	98 registere	
	D	99 registere	

Oppgave 7 (15 %)

I VHDL-koden under er det oppgitt en *package mypack*, en *package somesubprograms*, den uferdige *function numbermonthdays* og den uferdige *prodecure days*.

a) (6 %)

Lag ferdig *function numbermonthdays* i *package body* så den gir ut antall dager i måneden. Når det er skuddår (engelsk: «leap_year») er det 29 dager i februar, ellers 28 dager. Det er 31 dager i januar, mars, mai, juli, august, oktober, desember og 30 dager i de resterende månedene. Legg merke til at month_type er deklarert i package mypack så den typen er ferdig definert.

b) (9 %)

Lag ferdig *procedure days* som **skal** bruke *function numbermonthdays* til å beregne antall dager i måneden og også til å beregne antall dager i det kvartalet som måneden er en del av. F.eks. er det 29 dager i februar når det er skuddår og i det kvartalet med januar, februar og mars blir det dermed 90 dager (31+29+31=90).

```
package mypack is
 type month_type is (JAN, FEB, MAR, APR, MAY, JUN,
                     JUL, AUG, SEP, OCT, NOV, DEC);
end mypack;


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.mypack.all;


package somesubprograms is

  function numbermonthdays (leap_year : boolean;
                            month     : month_type)
    return unsigned;


  procedure days (leap_year   : in  boolean;
                  month       : in  month_type;
                  monthdays   : out unsigned(4 downto 0);
                  quarterdays : out unsigned(6 downto 0));

end package;


package body somesubprograms is

  < Implementer function numbermonthdays >

  < Implementer procedure days >

end package body;
```

Oppgave 8 (3 %)

Hvilken type er parametere deklarert som **in mode** i procedure days i oppgave 7	A	constant	
	B	variable	
	C	signal	

Oppgave 9 (3 %)

Hvilken type er parametere deklarert som **out mode** i procedure days i oppgave 7	A	constant	
	B	variable	
	C	signal	

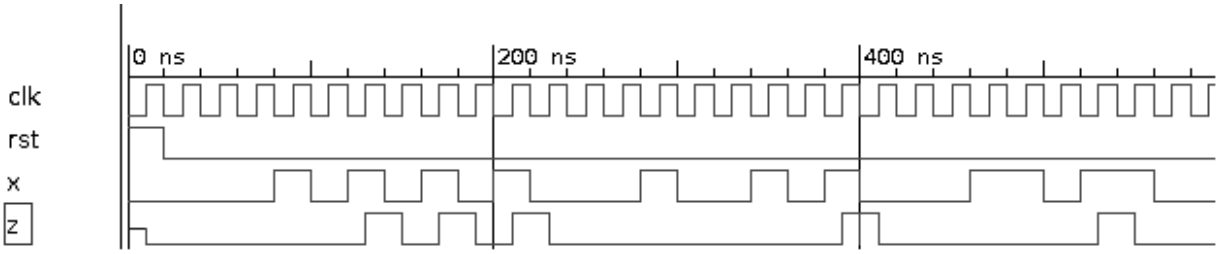
Oppgave 10 (45 %)

Det skal i denne oppgaven implementeres en **MEALY** type FSM tilstandsmaskin. Vi kaller tilstandsmaskinen en 101-detektor.

Tilstandsmaskinen skal fungere som følger:
Et høyt signal på z-utgangen skal gis når 101-mønsteret er detektert på inngangssignalet *x*.

Inngangssignalet x er synkront til klokkesignalet *clk*.

Diagrammet under viser et eksempel på inngangssignaler og utgangssignaler til tilstandsmaskinen:



a) Vekt 15 %

Tegn et ASM-diagram som viser tilstandsmaskinen beskrevet over.

b) Vekt 15 %

Implementer tilstandsmaskinen fra oppgave **11a)** som en to-prosess tilstandsmaskin i syntetiserbar VHDL.

Signalene skal være av type std_logic og det skal implementeres **synkron reset**.

c) Vekt 15 %

Implementer en VHDL testbenk. Testbenken skal **ikke** være selvsjekkende.

Bruk 0x55 («01010101»), 0x4a («01001010») og 0x6c («01101100») som verdier for inngangssignalet *x* hvor MSB bit 7 sendes inn i 101-detektoren først og LSB bit 0 til slutt som vist i timingdiagrammet over.

Vedlegg 1.
INF3430/INF4431. Oppgavesvar for kandidat nr: _____

Oppgave	A	B	C	D
1				
2				
3				
6				
8				
9				