Information

Question	Question title	Marks	Question type
□	Information 2021		Document

Basic VHDL

Question	Question title	Marks	Question type
1	Libraries and type conversion	10	Programming
2	Process and subprogram	7	Programming

Basic diagrams and tables

Question	Question title	Marks	Question type
3	State diagrams	4	Multiple Choice
4	State tables	4	Multiple Choice

Knowledge and principles

Question	Question title	Marks	Question type
5	Digital design knowledge	20	Multiple Response

Advanced design

Question	Question title	Marks	Question type
6	Pipelining	10	Programming

Complete design and verification

Question	Question title	Marks	Question type
7	ASM diagram	15	Upload Assignment

8	State machine implementation	15	Programming
9	Test bench	15	Programming

Information 2021

Written home exam in IN3160, IN4160
2021 Spring
Duration: 2021-06-10, 09:00 to 2021-06-10, 13:00

It is important that you read this cover page carefully before you start.

General information:
• Important messages during the exam are given directly from the course teacher on the course's semester page. It is therefore important that you check the course's semester page regularly.

• Your answer should reflect your own independent work and should be a result of your own learning and work effort.

• All sources of information are allowed for written home exams. If you reproduce a text from books, online articles, etc., a reference to these sources must be provided to avoid suspicions of plagiarism. This also applies if a text is translated from other languages.

• You are responsible for ensuring that your exam answers are not available to others during the exam period, neither physically nor digitally.

• Remember that your exam answers must be anonymous; do not state either your name or that of fellow students.

• If you want to withdraw from the exam, press the hamburger menu at the top right of Inspera and select "Withdraw".

Collaboration during the exam:
It is not allowed to collaborate or communicate with others during the exam. Cooperation and communication will be considered as attempted cheating. A plagiarism control is performed on all submitted exams where text similarities between answers are checked. If you use notes that have been prepared in collaboration with others before the exam, this might be detected in a plagiarism control. Such text similarities will be considered an attempt at cheating.

Cheats:
Read about what is considered cheating on UiO's website.

Multiple choice exercises:
There is no penalty for answering wrong, but the amount of boxes that can be checked cannot be greater than the number of correct answers.

Digital hand drawing / file upload:
You have been given 30 min. extra time for uploading files (e.g. digital hand drawings). Check out how to submit digital hand drawings

Contact information: User support exam

Good luck!

1 Libraries and type conversion

Below is a VHDL entity for a simple subtraction module.

```
entity subtract is
  generic(size: integer := 16);
  port(
      a, b:    in std_logic_vector(size-1 downto 0);
      result:  out std_logic_vector(size downto 0)
  );
end entity subtract;
```

Both "*a*" and "*b*" are unsigned. "*result*" shall be in two's complement form, and shall contain the result of the calculation "*a*-*b*".

- Define libraries needed for the entity and the architecture.
- Write a synthesizable architecture that implements the subtraction as described.
    - Create a signed signal "*i_result*" that shall contain the result of the calculation
    - Make sure to convert all input and output as needed to complete the task.

Fill in your answer here

```
1
```

Maximum marks: 10

2 Process and subprogram

Below is a VHDL entity for a simple subtraction module (the same as in previous exercise).

```
entity subtract is
  generic(size: integer := 16);
  port(
      a, b:    in std_logic_vector(size-1 downto 0);
      result:  out std_logic_vector(size downto 0)
  );
end entity subtract;
```

Both "*a*" and "*b*" are unsigned. "*result*" shall be in two's complement form, and shall contain the result of the calculation "*a-b*" (as with the previous exercise).

In this exercise you do not need to declare libraries, they are assumed to be as in the previous exercise. (No points will be awarded for those here).

In both tasks (a and b) you shall define a signed, intermediate result, *i_result*, for the calculation. Intermediate results shall be kept locally within the process or subprogram. All code shall be synthesizable.

a) Write an architecture "*process_arch*" that uses a process to perform the calculation.

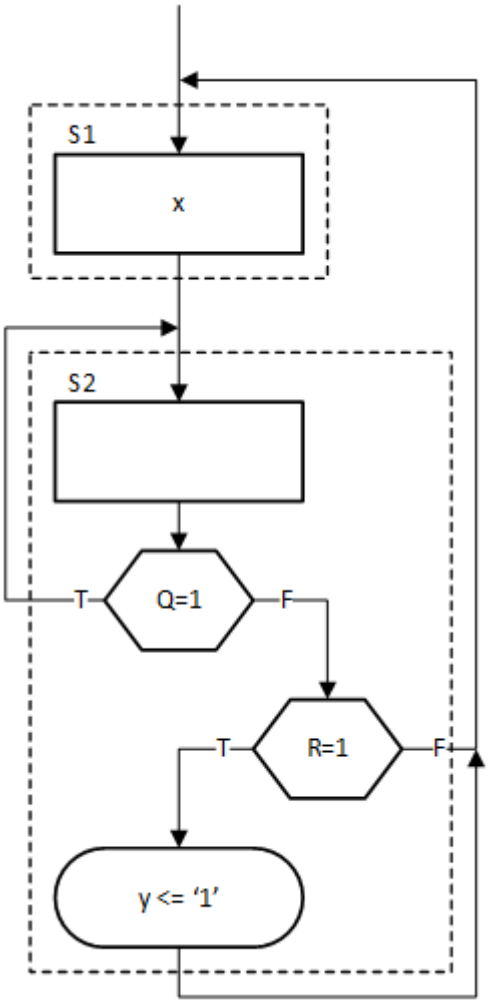
b) Write an architecture "*subprog_arch*" that uses a subprogram to perform the calculation. The subprogram shall return the calculation result as a signed.

Fill in your answer here

```
1
```

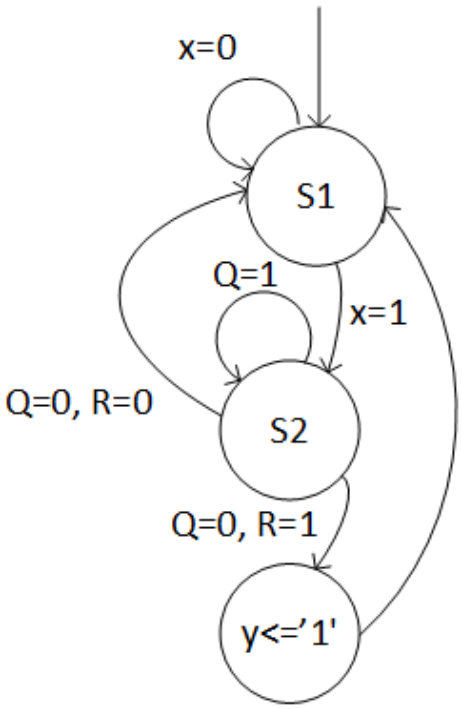
Maximum marks: 7

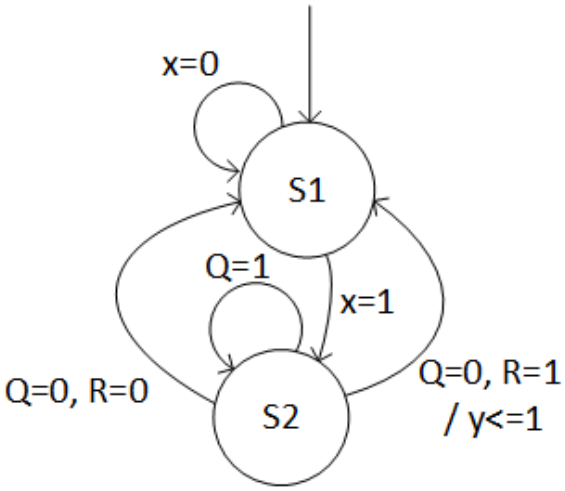
3 State diagrams

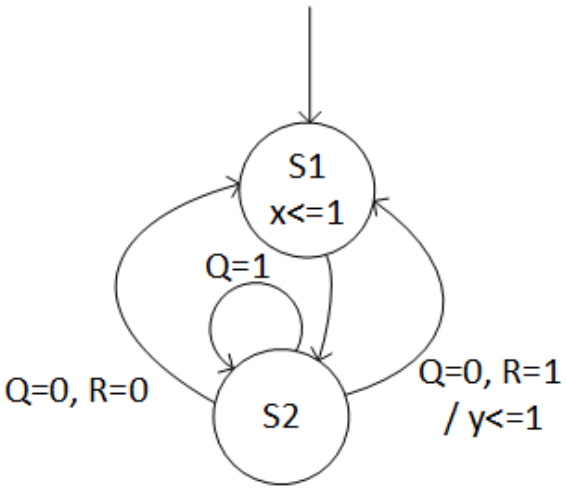


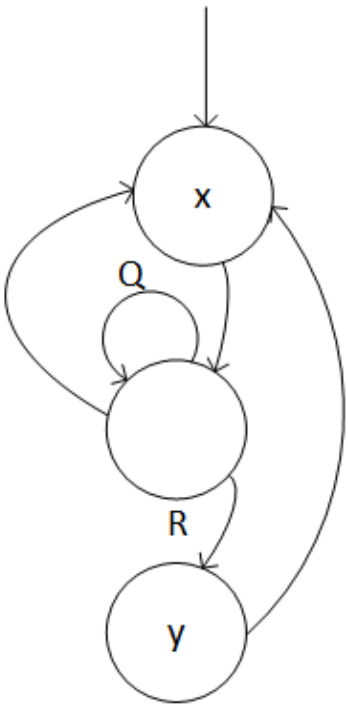
Select the state diagram that corresponds to the ASM diagram above.

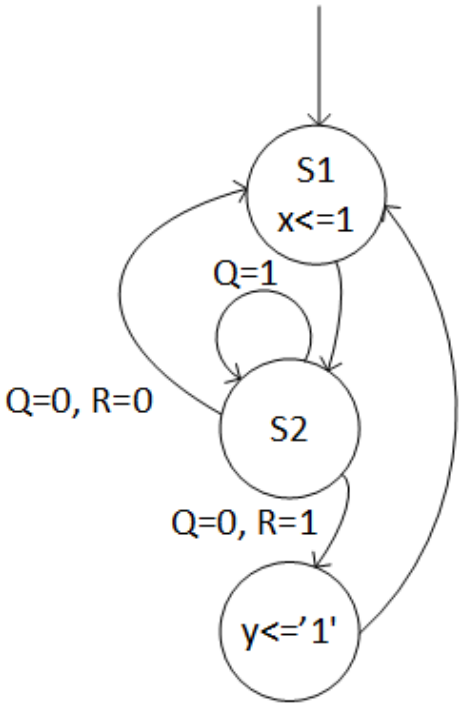
Select one alternative:





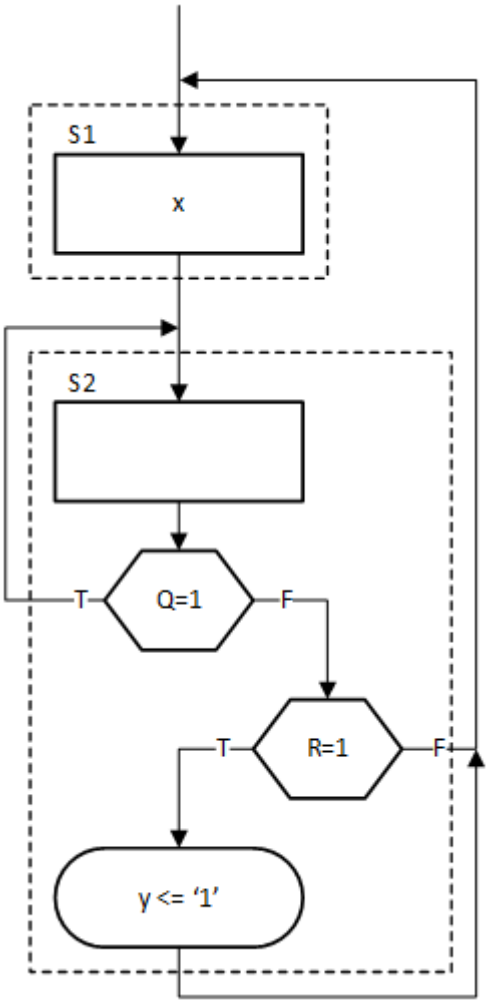






Maximum marks: 4

4 State tables



Select the state table that corresponds to the ASM diagram above.

Select one alternative:

State	QR	Next state	x	y
S1	XX	S2	1	0
S2	00	S1	0	0
S2	01	S3	0	0
S2	10	S2	0	0
S2	11	S1	0	0
S3	XX	S1	0	1

State	QR	Next state	x	y
S1	XX	S2	1	0
S2	00	S1	0	0
S2	01	S1	0	1
S2	10	S2	0	0
S2	11	S2	0	0

State	QR	Next state	x	y
S1	XX	S1	0	0
S1	XX	S2	1	0
S2	00	S1	0	0
S2	01	S1	0	1
S2	10	S2	0	0
S2	11	S2	0	0

State	QR	Next state	x	y
S1	XX	S2	1	0
S2	00	S1	0	0
S2	01	S3	0	0
S2	10	S2	0	0
S2	11	S2	0	0
S3	XX	S1	0	1

State	QR	Next state	x	y
S1	XX	S1	0	0
S1	XX	S2	1	0
S2	00	S1	0	0
S2	01	S3	0	0
S2	1X	S2	0	0
S3	XX	S1	0	1

Maximum marks: 4

5 Digital design knowledge

In this exercise, there are several statements with various precision level. Your task here is to mark true statements.
Statements that are not always true, or does not make sense in the context of digital design, shall be considered false.

- Generic example: Assume we know Ole is the owner of a blue hat.
  - Which statement is true and which is false:
    - *Ole wears a blue hat* (False – owning one does not mean he wears it)
    - *Ole has a hat* (True, he may own several, but he does at least have one)
    - *Ole has hats* (False, although he may have several, we can only confirm that he has one)

Out of the following 40 statements, there are exactly 10 statements considered true, and you will only be able to select 10.

Select one or more alternatives:

- [ ] A mealy machine does not use state registers
- [ ] RAM is synchronous
- [ ] Bus interfaces does not use control signals
- [ ] Pipelining is typically implemented using RAM
- [ ] A behavioral model shall be synthesizable
- [ ] Hazards occur in every digital design
- [ ] A shifter contains a shift register
- [ ] Flow control is only used for clock domain crossing
- [ ] The selected statement in VHDL is always the best option
- [ ] Double buffering reduces delay
- [ ] SRAM has interleaved memory
- [ ] Configurable logic blocks has four types of AD-converters
- [ ] Divide and conquer is bad practice in digital design.
- [ ] All states are legal in a microcoded FSM
- [ ] Sequential circuits should not be used as FSMs
- [ ] If we use “if” in VHDL we will avoid creating latches
- [ ] A crossbar switch has at least two baffles
- [ ] A behavioral model shall not be synthesizable
- [ ] A brute force conqueror uses hack and slash for partitioning designs
- [ ] Formal verification cannot be achieved
- [ ] A built-in self-test is synonymous with a self-testing testbench
- [ ] DRAM cells must be rewritten after read ✔
- [ ] Synthesizeable VHDL is executed sequentially

- [ ] A self-testing test bench can be used for verification ✔
- [ ] Flip-flop inputs should switch during setup time to avoid metastability
- [ ] If we use “if” in VHDL we will avoid creating hazards
- [ ] A look-up-table is digitally equivalent to a ROM or RAM device ✔
- [ ] A brute force synchronizer uses double buffering (double registers) ✔
- [ ] A microcoded FSM can implement a Mealy machine ✔
- [ ] A mealy machine does use registers for all outputs
- [ ] It is generally bad practice to use brute force synchronizers for multiple bits ✔
- [ ] Combinational logic does not contain flip-flops ✔
- [ ] VHDL means Verilog Hardware Description Service Language
- [ ] Seven segment displays contain Linear Feedback shift registers
- [ ] A microcoded FSM with a sequencer can be seen as a microcontroller or microproc ✔ sor.
- [ ] A FIFO does not use double buffering (double registers)
- [ ] Periodically valid signals is better than always valid signals
- [ ] ROM can be asynchronous ✔
- [ ] Moore Machines never create hazards
- [ ] Hazards occur when output changes more than once after input has changed ✔

Maximum marks: 20

6 Pipelining

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity pipelined is
 port (
  clk : in std_logic;
  rst : in std_logic;
  a : in std_logic_vector(7 downto 0);
  b : in std_logic_vector(7 downto 0);
  c : in std_logic_vector(7 downto 0);
  result : out std_logic_vector(9 downto 0);
  start : in std_logic;
  result_valid : out std_logic
  );
end entity pipelined;
```

The pipelined module entity is described above.

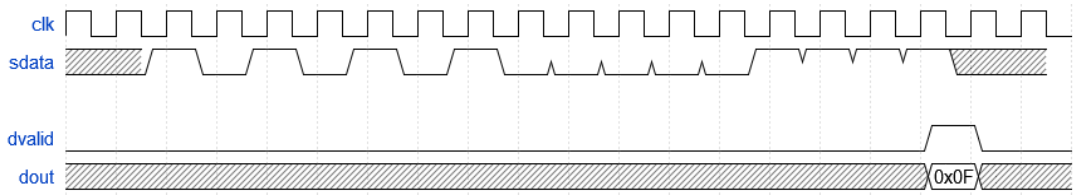
In this assignment a module which calculates *result* = *a*+*b*+*c* shall be implemented.
In order to meet timing closure at the required frequency, pipelining shall be used.
All input are synchronized to the clock signal *clk*.
Reset can be either synchronous or asynchronous.
The implementation shall be synchronous to the clock signal *clk*.
All output should be driven by registers to avoid propagating hazards.
The computation shall use unsigned arithmetic operation on the operands *a, b* and *c*.
When the start signal is high the computation shall start, and the result_valid signal shall be high when valid data is present on the result signal.

Implement the architecture for the pipelined module.

Fill in your answer here

```
1
```

Maximum marks: 10



The timing diagram (above) shows an example of input and output values of the finite state machine module.

In this section, a module using a Moore type finite state machine shall be implemented.

The state machine detects the sequence 0xAA transmitted on a serial line (sdata). After the sequence 0xAA is detected, the following byte received shall be set on an output (dout), and a data valid (dvalid) signal shall be set high.

7 ASM diagram

Draw an ASM-diagram which depicts the finite state machine described in the section document (above).
Counters and shift registers does not need to be included in the ASM-diagram.

Upload your image file (.png or .jpg) here. Maximum one file.

All file types are allowed. Maximum file size is **2 GB**

Select file to upload

Maximum marks: 15

8 State machine implementation

Implement the finite state machine module from assignment 7 as a two-process (or three-process) state machine in synthesizable VHDL.
All input and output shall be of type *std_logic* or *std_logic_vector*.
The implementation shall use synchronous reset.

Fill in your answer here

```
1
```

Maximum marks: 15

9 Test bench

Implement a VHDL test bench for the finite state machine module.
Use 0x0F («0b00001111»), 0x4A («0b01001010») and 0x6C («0b01101100») as input values to sdata.
The test bench shall verify correct output on dout and dvalid and report the results.
In the figure, the timing pattern shows 0x0F beeing transferred.

Fill in your answer here

```
1
```

Maximum marks: 15