Université d'Ottawa
Faculté de Génie

École de Science Informatique
et de Génie Électrique

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

uOttawa

L'Université canadienne
Canada's university

CEG3155/CEG3555 Digital Systems II

Name (surname, first name):______________________________________

Student Number:____

Examiners: Professor Miodrag Bolic
Professor Rami Abielmona

Final Examination: December 10, 2015 (14:00-17:00) Time allowed: 3 hours

Note:

- This is a closed-book examination and you are required to abide by the University's regulations regarding the conduct of exams
- Answer ALL questions.
- Use the provided space to answer the following questions. If more space is needed, use the back of the page.
- Read all the questions carefully before you start.

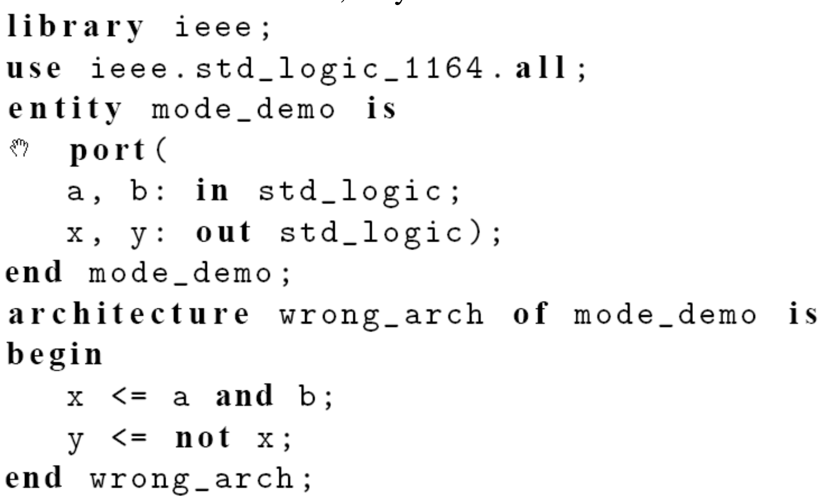
Good luck

Question	Max. Points	Points
1	20	
2	20	
3A or 3B	20	
4A or 4B	20	
5A or 5B	20	
Total	**100**	**100**

Question 1 (20 marks)
Part A (5 marks):

Short questions.

a) Is this code correct? If not, why.



```
library ieee;
use ieee.std_logic_1164.all;
entity mode_demo is
   port(
   a, b: in std_logic;
   x, y: out std_logic);
end mode_demo;
architecture wrong_arch of mode_demo is
begin
   x <= a and b;
   y <= not x;
end wrong_arch;
```

b) Explain the difference in the output of these two pieces of code. Assume that *a*, *b*, *c* and *y* are signals. Note that *tmp* is a signal in the code on the left and a variable in the code on the right.

process(a,b,c,tmp)
begin
tmp <= '0';
tmp <= tmp **or** a;
tmp <= tmp **or** b;
end process;
	process(a,b,c)
	variable tmp: std_logic;
	begin
	tmp := '0';
	tmp := tmp **or** a;
	tmp := tmp **or** b;
	y <= tmp;
	end **process**;

c) Let us consider a 4-bit binary counter that counts from 0 to 15 and then resets to 0 and starts up again. Do you need additional logic to go from 15 to 0 such as checking if the output of the counter is 15? Why?

d) Let us consider a 4-bit shift register that only supports shift right operations. How many D flip-flops do we need to implement this shift register? If this register was modified to become a bi-directional shift register, how many D flip-flops would it take then to implement the new shift register?

e) Combinational logic functions are sometimes implemented using look-up tables (like in Altera FPGAs). The look-up table contains results for all possible combinations of the input. If the maximum number of inputs per look-up table was 5, how many look-up tables would it take to implement a 7-variable function?

Part B (5 marks):

This question deals with bus-based systems.

a) Draw a block diagram of a circuit that connects outputs of 4 blocks to a shared bus. All the lines are 16 bits wide. The output of the logic blocks are connected to the bus using tri-state buffers. Two bit select input are used to select/open one of the 4 tri-state buffers.

b) Derive VHDL code for your block diagram.

Part C (5 marks):

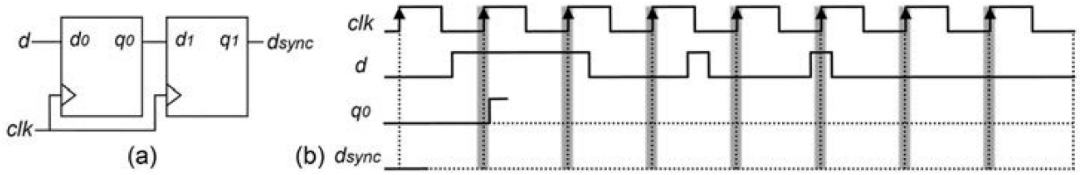
A synchronizer is presented in the figure below along with an illustrative timing diagram.

a) What do the gray areas in the figure on the right represent?

b) Why must d remain stable during that time interval? What is metastability?

c) Given the asynchronous input d shown in the figure, draw the waveforms for q0 and d_{sync}. (The initial part of q0 was already drawn).

d) Two short pulses (lasting less than one clock period) are included in the *d* waveform. Are they always detected? Justify your answer.



Part D (5 marks):

Using a conditional signal assignment, write VHDL code for an 8-to-3 priority encoder.

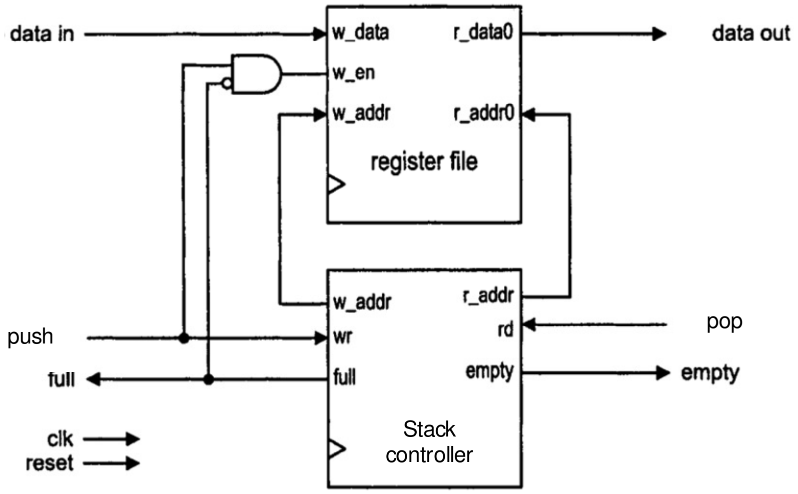
Question 2 (20 marks)
Part A (10 marks)

A stack is a buffer in which the data is stored and retrieved in first-in-last-out fashion. A synchronous stack should consist of the following I/O signals:

- *data_in* and *data_out*: data to be written (also known as pushed) into and read (also known as popped) from the stack.
- *push* and *pop*: control signals to enable the push or pop operation.
- *full* and *empty*: status signals.
- *clk* and *reset*: the clock and reset signals.

We can use a register file, as in the figure below, to construct this circuit.

(a) Consider a stack of four 16-bit words. Draw the FSM that includes control logic for implementing the stack controller.



Question 2 Part B (10 marks)

Derive the VHDL code for generating the *full* and *empty* status signals.

Please choose only one of Question 3A and Question 3B.

Question 3A Part A (10 marks):

This question deals with the synthesis of synchronous sequential circuits.

Use the method of partitioning minimization procedure to minimize the state table below, and show the sets P1 to P5 and your reduced table (hint: the reduced table contains 5 states).

Present	Next states		Output Z	
State	X = 0	X = 1		
$y_2y_1y_0$	$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	X = 0	X = 1
A	B	C	0	0
B	C	D	1	0
C	B	E	0	0
D	F	E	0	0
E	G	A	0	0
F	F	H	1	0
G	A	D	1	0
H	D	F	1	1

P1 = (ABCDEFGH)

P2 =

P3 =

P4 =

P5 =

Present	Next states		Output Z	
State	X = 0	X = 1		
$y_2y_1y_0$	$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	X = 0	X = 1

Question 3A Part B (10 marks):

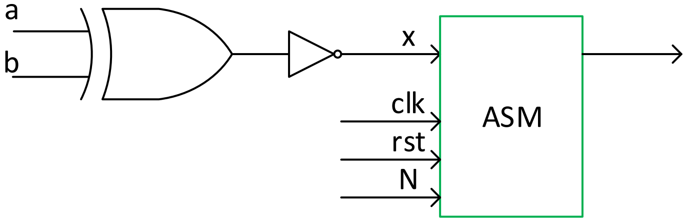
For the reduced table in Part A, determine the implementation equations for the next state and output variables by using a sequential state assignment.

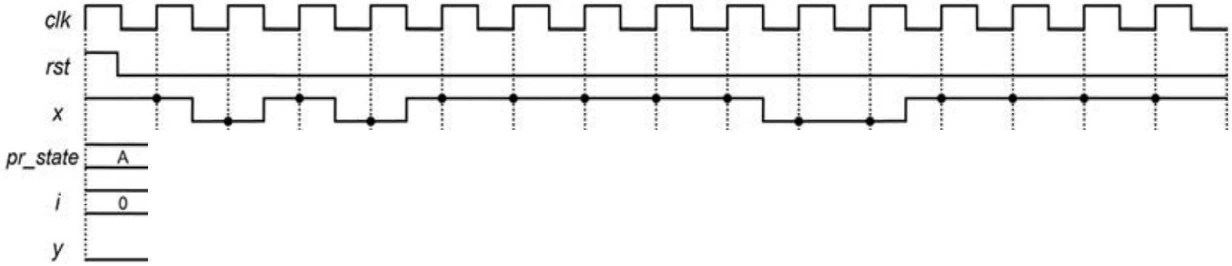
Compare the output equation with that of an output generated by using a Gray state assignment (avoiding two variable changes). Use the circuit cost measure for your comparison.

Question 3B (20 marks)

Design an ASM capable of sequentially comparing two arbitrarily long serial bit streams. The machine must determine whether the last *N* bits are pairwise equal (this means that the effect of the oldest pair of bits must be discarded when a new pair is received).
The circuit ports are depicted in the figure below. The inputs (serial bit streams) are *a* and *b*, while the output is *y* (= ‘ 1 ’ if all last *N* pairs of bits are equal). The comparator, in this case, is just a two-input XNOR gate, also depicted in the figure, which produces *x* = ‘1’ when the inputs are equal. This signal (*x*) will be the actual input to the FSM.

a. Design an ASM that would work for any value of *N* (unsigned 8 bit number).
b. Draw timing diagram for *N*=3. Please note that timing diagram also needs to show present state. If your ASM has a counter then present the value at the output of the counter (symbol *i* represents the counter). Please note that *i=0* is shown in the timing diagram (but you do not need to start from 0).





Please choose only one of Question 4A and Question 4B.

Question 4A (20 marks):

For the following flow table:

Present State	**Next State**				Output z
	w_2w_1=00	**01**	**10**	**11**	
A	(A)	G	E	—	0
B	K	—	(B)	D	0
C	F	(C)	—	H	1
D	—	C	—	(D)	0
E	A	—	(E)	D	1
F	(F)	C	J	—	0
G	K	(G)	—	D	1
H	—	—	E	(H)	1
J	F	—	(J)	D	0
K	(K)	C	B	—	0

Question 4A Part A (5 marks):

- Show the reduced table using the partitioning procedure
- Show the merger diagram for the table

Question 4A part B (5 marks)

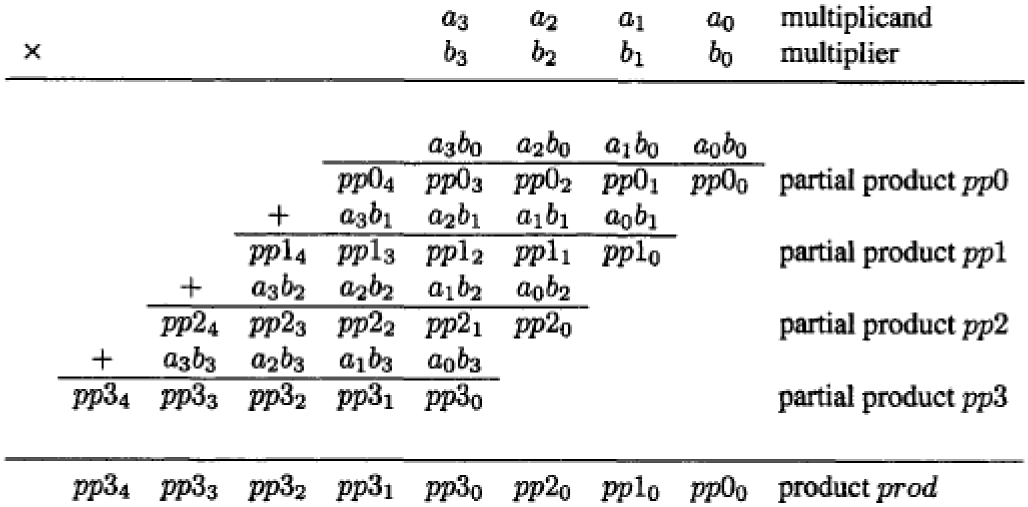
- For the reduced table in part A, , convert the table into the form of Mealy model and merge the compatible states
- Show the reduced flow table
- Show the relabeled flow table

Question 4A part C (10 marks)

- For the flow table in part B, show the initial transition diagram
- Modify the transition diagram to avoid any diagonals
- Show the final flow table, from the modified transition diagram
- Show the final excitation table, using a suitable state assignment.

Question 4B (20 marks)

An example of a 4-bit multiplication procedure is shown below.



$$\begin{array}{ccccccccl}
 & & & & a_3 & a_2 & a_1 & a_0 & \text{multiplicand} \\
\times & & & & b_3 & b_2 & b_1 & b_0 & \text{multiplier} \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 & \\
 & & & pp0_4 & pp0_3 & pp0_2 & pp0_1 & pp0_0 & \text{partial product } pp0 \\
 & & + & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & & \\
 & & pp1_4 & pp1_3 & pp1_2 & pp1_1 & pp1_0 & & \text{partial product } pp1 \\
 & + & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & & \\
 & pp2_4 & pp2_3 & pp2_2 & pp2_1 & pp2_0 & & & \text{partial product } pp2 \\
+ & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & & \\
pp3_4 & pp3_3 & pp3_2 & pp3_1 & pp3_0 & & & & \text{partial product } pp3 \\
\hline
pp3_4 & pp3_3 & pp3_2 & pp3_1 & pp3_0 & pp2_0 & pp1_0 & pp0_0 & \text{product } prod
\end{array}$$

As can be seen, multiplication can be implemented by performing additions of shifted partial-products (pp). Now, let us consider an 8-bit multiplier. Let pp_7, pp_6, . . . , pp_1, pp_0 be the shifted partial products of an 8-bit multiplier. The final product can be expressed as

$y=pp_7+pp_6+\ldots+pp_1+pp_0$

(a) Arrange the additions as a tree to exploit parallelism and show the block diagram of your implementation. Make sure that you take care about the number of bits in each adder.

(b) Assume that only one adder is provided. Derive the ASMD chart to perform multiplication based on just simple sequential additions.

(c) Show block diagram of the datapath (including status signals).

(d) Show the block diagram of the FSM.

(e) What is the minimum and maximum number of clock cycles required to perform multiplication? What is the total number of bits in each register?

Please choose one of Question 5A and Question 5B.

Question 5A (20 marks):
Question 5A Part A (10 marks):

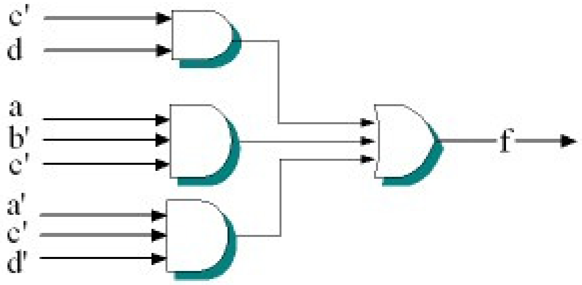
All transitions between stable states in the following flow table have a Hamming distance of 2, since we are utilizing a one-hot state assignment for the table rows. We can add unstable states to try and reduce the Hamming distance by 1 for each transition from a stable state to another, passing through an unstable state. Your task is to determine the encodings of the unstable states, and the new values for the next state and output variables to reflect the addition of the unstable states. Complete the table below with your answers.

Present State	**Next State**				Output z
	w_2w_1=00	**01**	**10**	**11**	
A → 0001	(A)	B	D	(A)	1
B → 0010	C	(B)	(B)	D	0
C → 0100	(C)	(C)	B	A	0
D → 1000	A	C	(D)	(D)	1

Present State	**Next State**				Output z
	w_2w_1=00	**01**	**10**	**11**	
A → 0001	(A)			(A)	1
B → 0010		(B)	(B)		0
C → 0100	(C)	(C)			0
D → 1000			(D)	(D)	1
E → ____					
F → ____					
G → ____					
H → ____					
J → ____					
K → ____					

Question 5A part B (10 marks)

For the following logic circuit f(a, b, c, d), determine the static hazards. Redesign the circuit to be hazard-free and having the same output (i.e. *f*). Show the final circuit. Note that the apostrophe in the figure indicates the complemented form of the variable.



Question 5B (20 marks)

a) Show a block diagram of a counter that counts from 30 until 1 and then again from 30 using methods that we studied in the section on sequential design (without using FSM, ASM or ASMD). The counter should generate 1 for one clock cycle (or less) every time the value at its output is 1.

b) Show the ASM for this counter.

c) Derive the VHDL code for part b).