Midterm Examination II

EE 203 - Digital Systems DESIGN (Fall 2015)

MEF University

Assigned: 6:30pm on December 17, 2015.
Due: 8:00pm on December 17, 2015.
Instructor: Şuayb Ş. Arslan.

Name: _____

Student ID: _____

Instructions

1. For every design you make or the solution you present, please show every step you take. I am looking for clear development of your approach for the solution so that you will be able to get partial credit.
2. This is a closed textbook exam. You **may not** work on the exam with anyone else, ask anyone questions, or consult the digital version of the textbook or other sites on the Web for answers.
3. Single-sided one page cheat sheet is allowed.
4. You can use scratch paper and attach them to this hard-copy if you need more space.
5. Please make sure your hand writing is legible. Although you will not be penalised due to a potential disorganization in your submission, but it would be best if you can keep your solutions and paper organization at a certain quality.
6. Do not forget to staple or attach the pages of the hard copy you hand in.

I wish you the best of luck!

Question #	1	2	3	4	Bonus	Total
Subject	Carry Look-Ahead Adder	Comparators and Decoders	Sequential Circuit Analysis	Sequential Circuit Design	Multiplexers and Design	
Points	25	20	20	35	10	100(+5+5)

Problem 1 (Carry Look-Ahead Logic - **25 points**) Let us remember the full adder implementation using two half adders. When you attempt to implement an n-bit adder, the logic diagram of the i-th stage of the adder should look like something as shown below. There are two new variables we have defined in class which were $\mathbf{P}_i = A_i \oplus B_i$ and $\mathbf{G}_i = A_i B_i$. With these new variables, it is clear from the logic diagram that the sum and the carry-out shall be given by $S_i = \mathbf{P}_i \oplus C_i$ and $C_{i+1} = \mathbf{G}_i + P_i C_i$.

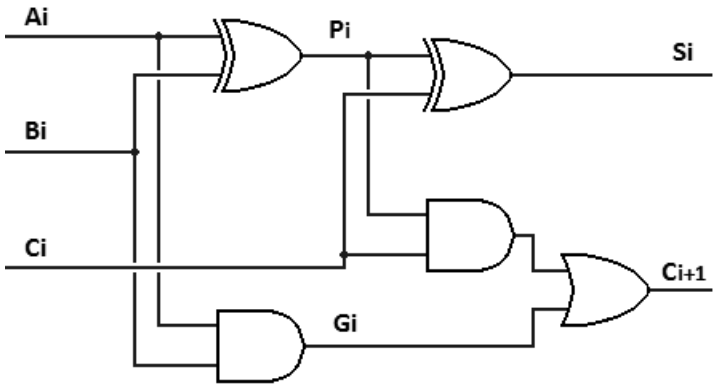

Figure 1: Full adder with **P** and **G** shown.

1. While designing a 2-bit carry look ahead logic, let us assume that we make an error i.e., a missing circle around the "plus" sign and write down $P_i = A_i + B_i$ instead of $\mathbf{P}_i = A_i \oplus B_i$ (I used bold case to differentiate Ps). This error shall lead to different sum and carry-out values in the full adder implementation which are named as $\tilde{S}_i$ and $\tilde{C}_{i+1}$. Please determine the extra logic (the gate names and how many you use) you need to express S_i in terms of $\tilde{S}_i$ and C_{i+1} in terms of $\tilde{C}_{i+1}$.
2. Express the carry-out values C_1 and C_2 in terms of P_0, P_1, G_0 and G_1.
3. Using the above logic (with the error), let us assume we generate the carry information using a carry look ahead generator (which has inputs P_1, G_1, P_0, G_0, C_0 and outputs C_2, C_1, C_0 - think of this as a block box, you do not need to draw the logic diagram) and we would like to design a 2-bit carry look ahead adder. Apart from the lookahead generator, how many extra gates (AND, OR, NOT and XOR gates) are needed to implement the correct adder? Justify your answer.

Problem 2 (Binary comparators - **20 points**) Consider a 1-bit comparator that has two inputs X and Y and three outputs where the first output is asserted if $X = Y$, second output is asserted if $X > Y$ and finally the last output is asserted if $Y > X$.

1. Design the 1-bit comparator using a decoder.
2. Synthesize a combinational circuit that can compare two 2-bit numbers A and B using hierarchical design methodology. Show your logic design in the picture provided below.
3. Can you use your 2-bit comparator to design a 3-bit comparator? Please show how.
4. **(Bonus+5)** Design the circuit of 2. using multiplexers. If you need to use extra gates, use XOR gates only.

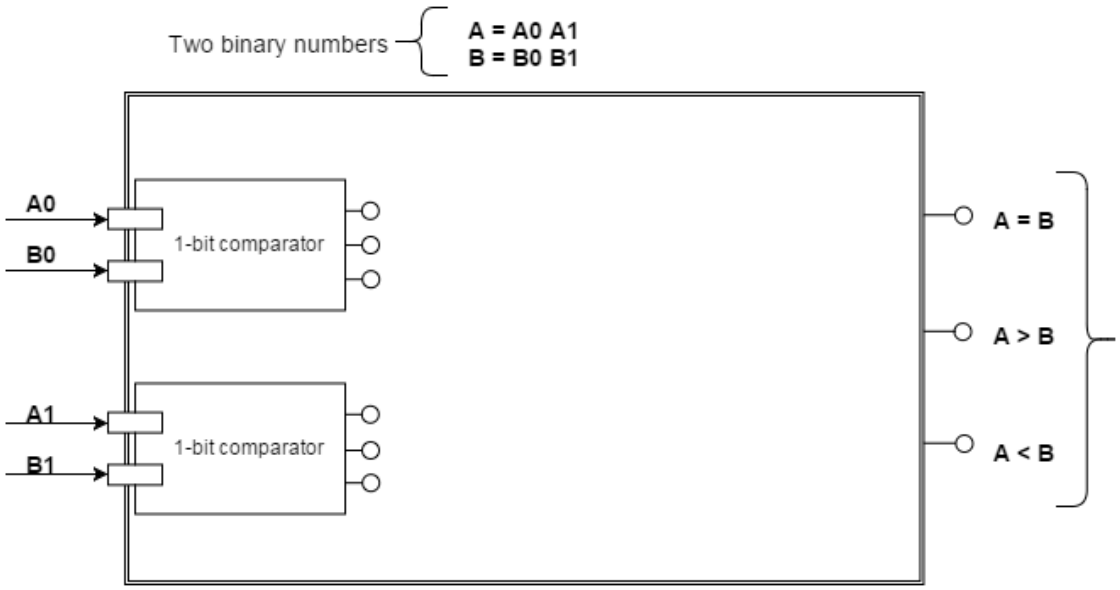

Figure 2: Hierarchical design of 2-bit comparator.

Problem 3 (Sequential Circuit Analysis - **20 points**) Consider the sequential circuit shown below.

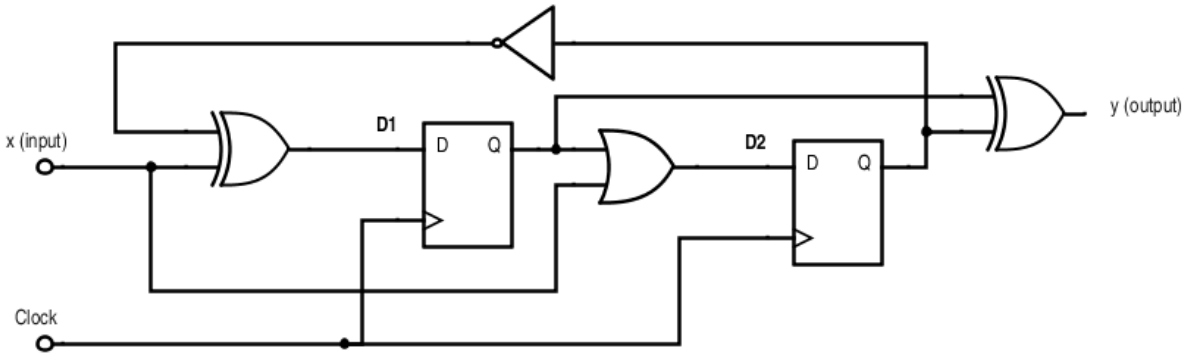

Figure 3: A sequential circuit diagram.

1. Label the outputs of flipflops D1 and D2 with A and B, respectively. Derive the state and output equations.
2. Draw state table and state diagram of the circuit. Label your states appropriately.
3. Is this circuit a Mealy or Moore machine? Justify your answer.
4. Can I use this circuit as a counter? If yes, how would you name it? What's the meaning of the output values y?

Problem 4 (Sequential Circuit Design - **35 points**) Digital storage or communication devices transmit or store bits according to predefined format. This format information usually includes a fixed sequence of bits to be used for synchronization (sync). This so called sync pattern is useful to determine the data frame and hence correct bit locations can be identified to start extracting the raw data. The raw data is usually transformed to a bit sequence such that it does not include the sync pattern anywhere in the bitstream. This way, the predefined sync patterns can be used to identify where the raw data is located. In this question, we shall design a 4-bit sync pattern "1001" detector in a given input bit stream using JK flip-flops. The picture below shows an output of the circuit for the given input bit stream (note that input and output bits may not be in sync due to delay, but we assume the input-output relationship to be time invariant).

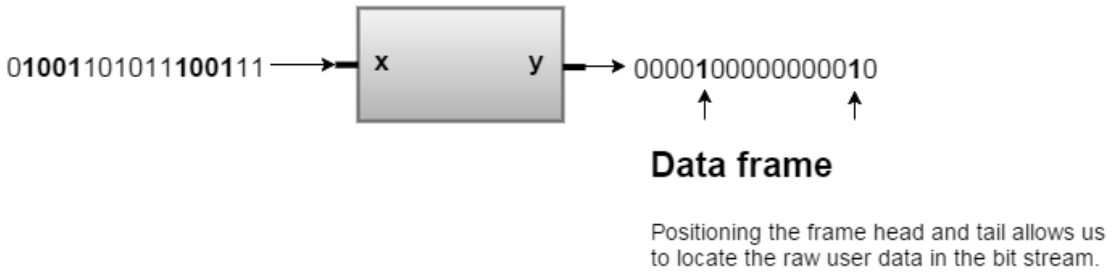

Figure 4: A sequence detector is a useful circuit for modern digital systems.

1. Draw the appropriate state diagram based on the above specification. Please reduce states if possible.
2. Choose an appropriate binary state labeling (it may not be necessarily optimal).
3. Drive the state table that includes current states, next states, circuit input x, circuit output y as well as JK flip-flop inputs Js and Ks. (Hint: Use excitation tables)
4. Use Karnough maps to minimize the Boolean expressions for y, Js and Ks.
5. Implement the final logic diagram of the circuit using only NAND gates.
6. **(Bonus+5)** Note that if the input bit steam was "010010011101…", the sequence detector shall produce the bit stream "000010010000…" which is not viable because it would mean there is no data between two sync patterns. Such inconsistency can be used to detect errors as well. Can you propose an error detection logic (not correction) to circumvent this problem? You do not need to implement it.