THE UNIVERSITY OF ALABAMA IN HUNTSVILLE
Department of Electrical and Computer Engineering

CPE 322 Digital Hardware Design Fundamentals
Spring Semester 2022: Homework Assignment #1
Basic Sequential Digital Design Methodology Review
Design of a Two's Complement Sequential Subtraction Network

Purpose
In this homework assignment you are asked to utilize traditional Mealy and Moore sequential Finite State Machine, FSM, design methodologies that were introduced in the EE 202, Digital Logic Design class and are reviewed in Chapter 1 of the course text to create a binary subtraction network. You will convert from the FSM representations into multiple realizable logic-level designs. These designs will differ from one another by the manner in which the combinational logic portion of the network is implemented (based upon the use of logic gates, or ROM type lookup tables, or multiplexers as the building combinational logic building blocks) Implementing combinational logic using lookup tables and multiplexers is a common method to implement such logic in programmable logic devices such as FPGAs.

Assignment
This assignment requires that students develop multiple Mealy and Moore Finite State Machine implementations of a functional unit that performs two's complement subtraction of two numbers that are of arbitrary length.

Background
This homework assignment focuses on the design of a sequential Boolean network whose function is to perform a two's complement subtraction of two arbitrary length binary numbers **A** and **B** producing as difference **D** of the same binary size. These two inputs are assumed to be sent bit-serially in time starting with the least significant bit first. The output, **D** is also assumed to produce the difference (**A** - **B**) in a bit serial manner in least to most significant bit ordering with increasing time as shown in Figure 1. The network also contains a **Reset** and **Enable** input. The **Reset** input is the means by which an external controller can reset the design to its original state by clearing the flip-flop memory elements so the network can be used to compute the difference of another arbitrary sized binary numbers. The **Enable** input allows the external controller to pause your design (in effect skipping clock cycles) so that it ignores the **A** and **B** inputs during times that they are not valid. It is the job of the controller to make sure the Boolean network has been reset when a new number is sent and to enable the flip-flops in your network when the inputs **A** and **B** are valid. Your job is to develop the sequential network in a manner the considers only the inputs **A**, and **B**, and the output **D**. In the portions of this homework that require flip-flop state assignments, you should assume that the initial state occurs when all of the flip-flops are zero. This would be the case when all the reset and enable signals associated with the flip-flop portions of your design are cascaded from flip-flop to flip-flop as shown in the following figure.

.

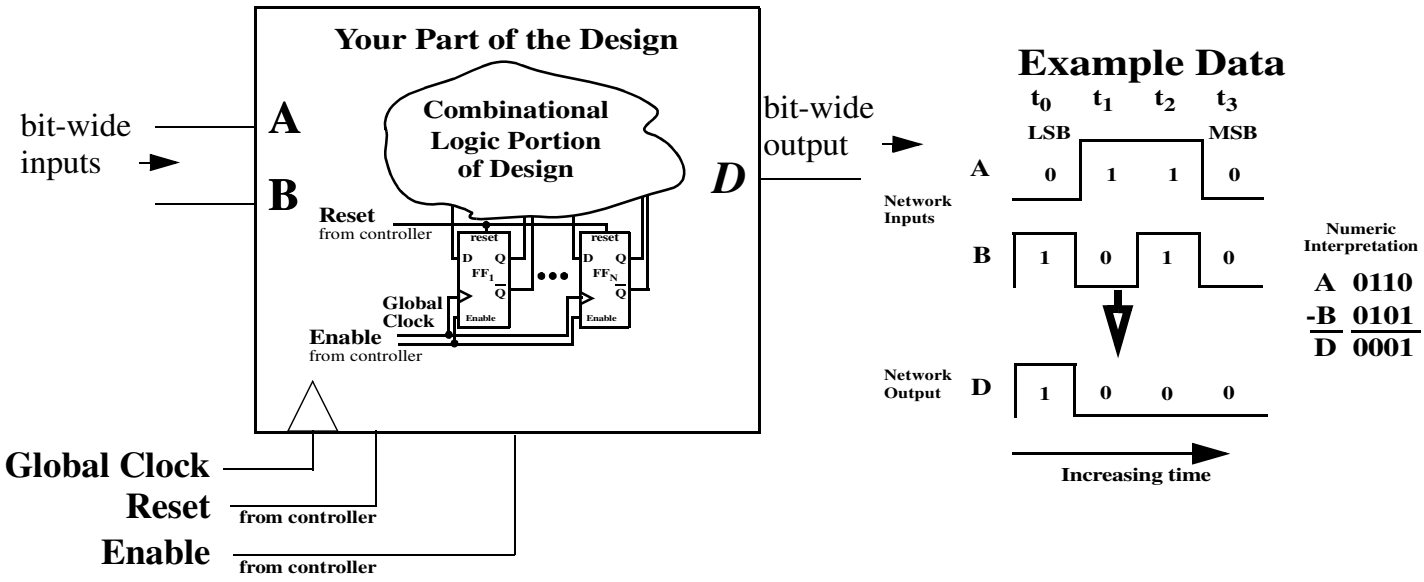
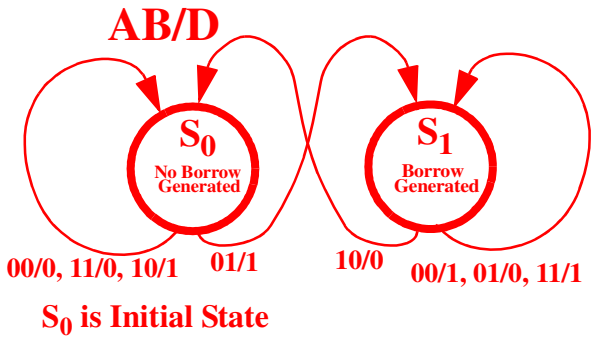


Figure 1: Functional Overview of Two's Complement Subtractor

Assignment Part 1

Develop a Mealy State Graph (SG) for the Subtractor design shown in Figure 1. The design should contain a minimum number of states to implement the desired function. Identify the initial state and fully label all states in the diagram. Also label the input(s) and output of the design. Using this state graph develop an input sequence which will transverse all arcs and nodes of the SG. Since the Reset and Enable functionality is contained in the Flip-Flop section of the design do not include this functionality inside your SG.



Example Input Sequence for an 8-bit Two's Complement Addition of $A=166_{10}$ - $B=108_{10}$ → $D=58_{10}$

	LSB							MSB		
A =	0	1	1	0	0	1	0	1	10100110_2	166_{10}
B =	0	0	1	1	0	1	1	0	- 01101100_2	- 108_{10}
D =	0	1	0	1	1	1	0	0	00111010_2	58_{10}
	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7		

Increasing Time

Assignment Part 2

Implement the SG design in Part 1 using the minimum set of discrete logic gates (AND, OR, and NOT) and D-flip-flops for the state assignment that you have chosen. Show all step in this process. Evaluate the correctness of your design by neatly comparing the output of your final implementation with that which is predicted by the SG for the sequence that you developed in Part 1

State Table

Current State	Next State A=0 B=0	A=0 B=1	A=1 B=0	A=1 B=1	Output, D A=0 B=0	A=0 B=1	A=1 B=0	A=1 B=1
S_0	S_0	S_1	S_0	S_0	0	1	1	0
S_1	S_1	S_1	S_0	S_1	1	0	0	1

State Assignment

Q_0

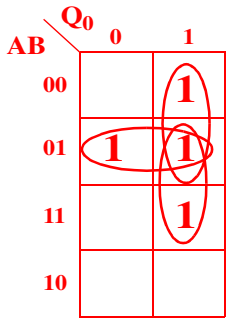
$S_0 = 0$

$S_1 = 1$

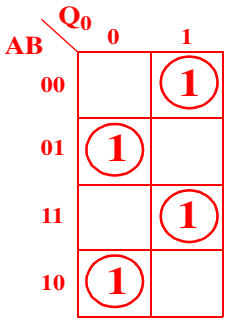
(Only one flip-flop required. It should be at a Logic 0 when the flip-flop is reset. This means its output $Q_0 = 0$ should indicate when the logic network is in State S_0)

Transition Table

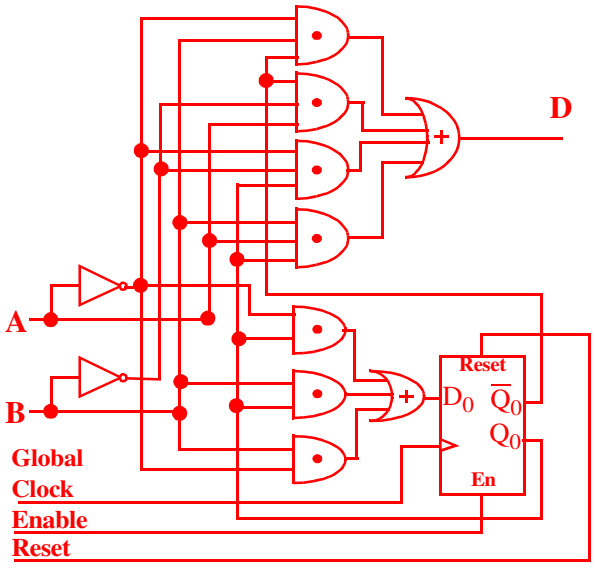
Current State Q_0	Next State A=0 B=0 Q_0^+	A=0 B=1 Q_0^+	A=1 B=0 Q_0^+	A=1 B=1 Q_0^+	Output, D A=0 B=0	A=0 B=1	A=1 B=0	A=1 B=1
0	0	1	0	0	0	1	1	0
1	1	1	0	1	1	0	0	1



$Q_0^+ = D_0 = \overline{A}B + \overline{A}Q_0 + BQ_0$



$D = \overline{A}\,\overline{B}\,Q_0 + \overline{A}\,B\,\overline{Q}_0 + A\,B\,Q_0 + A\,\overline{B}\,\overline{Q}_0$

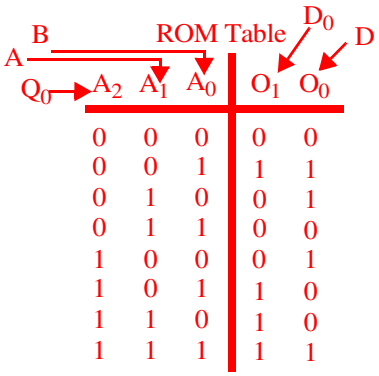


$Q_0^+ = D_0 = \overline{A}B + \overline{A}Q_0 + BQ_0$

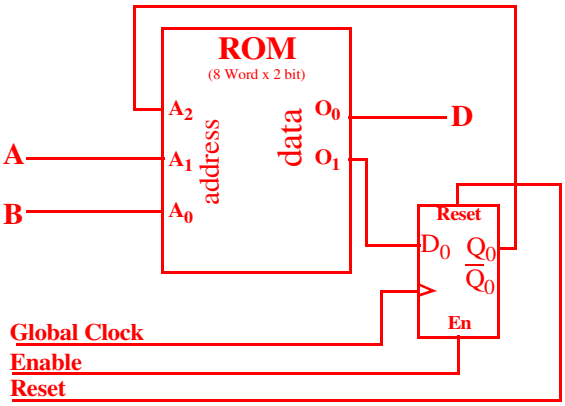
$D = \overline{A}\,\overline{B}\,Q_0 + \overline{A}\,B\,\overline{Q}_0 + A\,B\,Q_0 + A\,\overline{B}\,\overline{Q}_0$

Assignment Part 3

Replace the combinational logic potion (AND, OR and NOT gates) of Part 2 with a single ROMs that is of minimal size to implement the combinational logic portion of the SG of Part 1 with a minimal number of D-flip-flops. Clearly identify the Address and Data buses of the ROM element and specify its complete ROM Table.



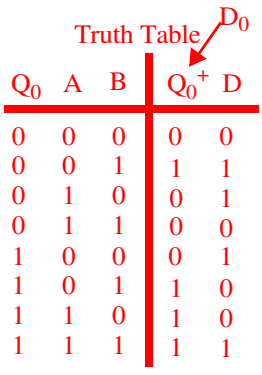
A_2	A_1	A_0	O_1	O_0
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Assignment Part 4

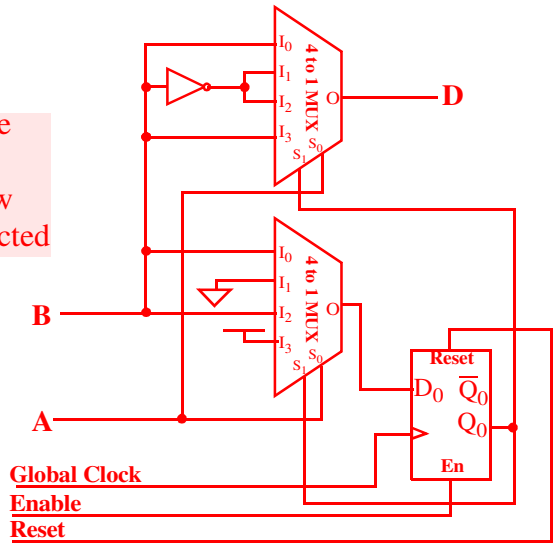
Replace the combinational logic potion (AND, OR and NOT gates) of Part 2 with one or more 4-to-1 multiplexers that will implement the combinational logic portion of the SG of Part 1 with a minimal number of D-flip-flops. Clearly identify the inputs and outputs of the multiplexers. Assume positive logic with a Logic 1 representing VCC and a Logic 0 representing GND. You may also use inverters (NOT gates) as needed.

.



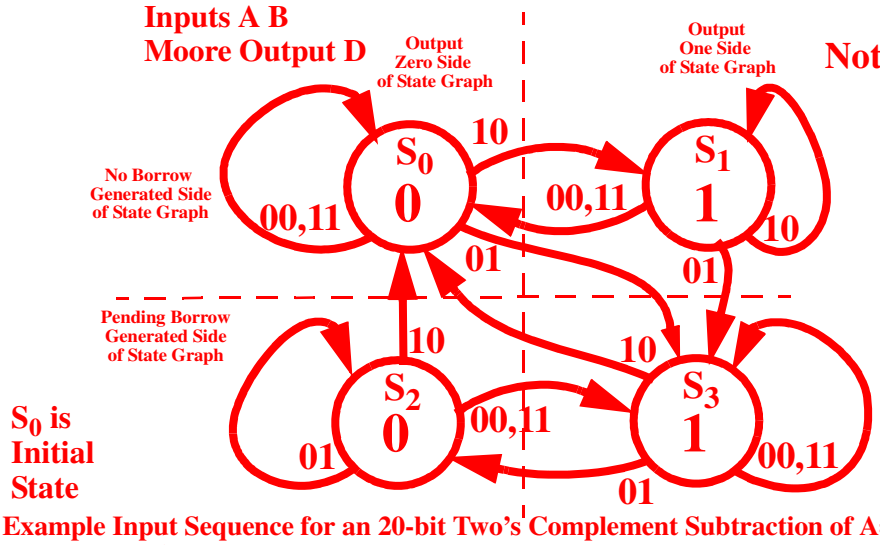
Q_0	A	B	Q_0^+	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Note: Many possible correct solutions depending upon how the MUX are connected



Assignment Part 5

DDevelop a Moore State Graph, SG, for the Two's Complement Design shown in Figure 1. Since the Reset and Enable functionality is contained in the Flip-Flop section of the design do not include this functionality inside your SG. The design should contain a minimum number of states to implement the desired function. Identify the initial state and fully label all states in the diagram. Also label the input(s) and output of the design. Using this state graph develop an input sequence which will transverse all arcs and nodes of the SG.



Example Input Sequence for an 20-bit Two's Complement Subtraction of $A=593086_{10}$ - $B=505112_{10}$ → $D=87974_{10}$

(all edges are transversed -- many sequences are possible)

	LSB																			MSB
A = 0	1	1	1	1	1	0	1	0	0	1	1	0	0	0	0	1	0	0	1	1001000011001011111_2
B = 0	0	0	1	1	0	0	0	1	0	1	0	1	1	0	1	1	1	1	0	- 0111101101010001100_2
S = 0	0	0	1	0	1	0	1	0	1	1	1	1	0	1	0	0	1	1	0	0001010101111010011_2
	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	

Increasing Time

Assignment Part 6

Implement the SG design in Part 5 using the minimum set of discrete logic gates (AND, OR, and NOT) and D-flip-flops for the state assignment that you have chosen. Show all step in this process. Evaluate the correctness of your design by neatly comparing the output of your final implementation with that which is predicted by the SG for the sequence that you developed in Part 5.

State Table

Current State	Next State A=0 B=0	Next State A=0 B=1	Next State A=1 B=0	Next State A=1 B=1	Output, D
S_0	S_0	S_3	S_1	S_0	0
S_1	S_0	S_3	S_1	S_0	1
S_2	S_3	S_2	S_0	S_3	0
S_3	S_3	S_2	S_0	S_3	1

Minimum Number of Flip/Flops needed = 2 for 4 states

State Table

Current State	Next State A=0 B=0	A=0 B=1	A=1 B=0	A=1 B=1	Output, D
S_0	S_0	S_3	S_1	S_0	0
S_1	S_0	S_3	S_1	S_0	1
S_2	S_3	S_2	S_0	S_3	0
S_3	S_3	S_2	S_0	S_3	1

highest priority

Rule 1 -- states which have the same next state under the same input conditions should have adjacent state assignments.
$(S_0, S_1), (S_2, S_3)$

Rule 2 -- states are next states to the same state should be given adjacent state assignments.
$(S_0, S_1, S_3), (S_0, S_2, S_3)$

Rule 3 -- states that have the same output under the same input conditions should be given adjacent state assignments.
$(S_0, S_2), (S_1, S_3)$

If S_0 is to be defined as the case where both flip/flops are reset to a logic zero then two minimal state assignments are possible by using these rules.

State Assignment

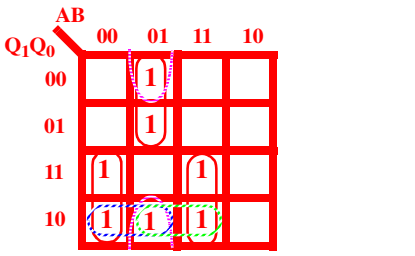
Q_0 \ Q_1	0	1
0	S_0	S_3
1	S_1	S_2

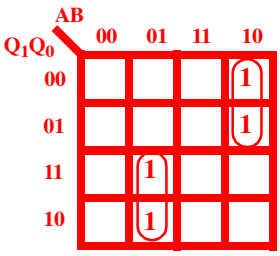
state assignment table

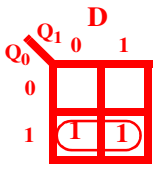
$S_0 = 0\ 0$
$S_1 = 0\ 1$
$S_2 = 1\ 1$
$S_3 = 1\ 0$

Transition Table

Current State Q_1Q_0	Next State A=0 B=0 $Q_1^+Q_0^+$	A=0 B=1 $Q_1^+Q_0^+$	A=1 B=0 $Q_1^+Q_0^+$	A=1 B=1 $Q_1^+Q_0^+$	Output, D
00	00	10	01	00	0
01	00	10	01	00	1
10	10	11	00	10	0
11	10	01	00	10	1







$Q_1^+ = D_1 = \overline{A}\,\overline{B}\,Q_1 + \overline{A}\,B\,\overline{Q}_1 + A\,B\,Q_1 + \overline{A}\,B\,\overline{Q}_0$

or

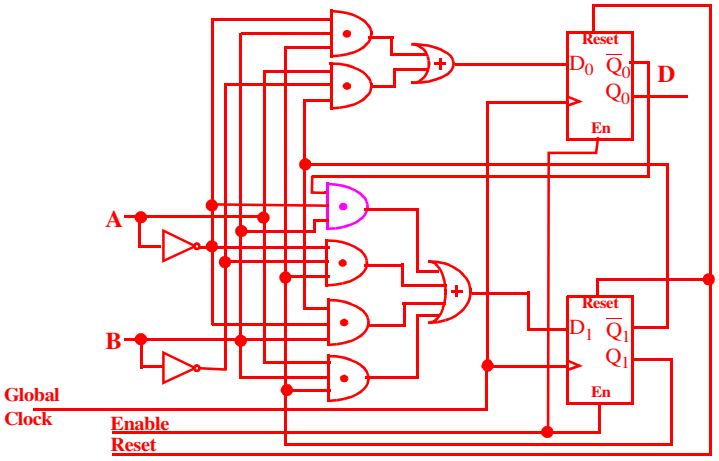
$= \overline{A}\,\overline{B}\,Q_1 + \overline{A}\,B\,\overline{Q}_1 + A\,B\,Q_1 + \overline{A}\,Q_1\,\overline{Q}_0$

or

$= \overline{A}\,\overline{B}\,Q_1 + \overline{A}\,B\,\overline{Q}_1 + A\,B\,Q_1 + B\,Q_1\,\overline{Q}_0$

$Q_0^+ = D_0 = \overline{A}\,B\,Q_1 + A\,\overline{B}\,\overline{Q}_1$

$D = Q_0$



Assignment Part 7

Implement the SG design in Part 5 using again the minimum set of discrete logic gates (AND, OR, and NOT) but this time implement your state assignments using one-hot encoding. In this encoding scheme each state has a corresponding D-flip-flop which is at a Logic 1 whenever the design is in that state and is a Logic 0 otherwise. Assume in this case that the D-flip flops have both presets and clears which will be used to make sure that the state is initialized with only the flip-flop associated with the initial state being a Logic 1 and the remaining flip-fops being initially set at Logic 0. Show all step in this process. Evaluate the correctness of your design by neatly comparing the output of your final implementation with that which is predicted by the SG for the sequence that you developed in Part 5

State Table

Current State	Next State A=0 B=0	A=0 B=1	A=1 B=0	A=1 B=1	Output, D
S_0	S_0	S_3	S_1	S_0	0
S_1	S_0	S_3	S_1	S_0	1
S_2	S_3	S_2	S_0	S_3	0
S_3	S_3	S_2	S_0	S_3	1

For one-hot state assignment*
(4 flip/flops -- one for each state)

	Q_3	Q_2	Q_1	Q_0
S_0 =	0	0	0	1
S_1 =	0	0	1	0
S_2 =	0	1	0	0
S_3 =	1	0	0	0

*Flip/Flop associated with a particular state is high when the design is in that state

By inspection from either the State Table or the State Graph the following 4 next state equations and one output equation can be directly written without further analysis.

$$Q_0^+ = D_0 = \overline{A}\,\overline{B}\,Q_0 + A\,B\,Q_0 + \overline{A}\,\overline{B}\,Q_1 + A\,B\,Q_1 + A\,\overline{B}\,Q_2 + A\,\overline{B}\,Q_3$$

$$Q_1^+ = D_1 = A\,\overline{B}\,Q_0 + A\,\overline{B}\,Q_1$$

$$Q_2^+ = D_2 = \overline{A}\,B\,Q_2 + \overline{A}\,B\,Q_3$$

$$Q_3^+ = D_3 = \overline{A}\,B\,Q_0 + \overline{A}\,B\,Q_1 + \overline{A}\,\overline{B}\,Q_2 + A\,B\,Q_2 + \overline{A}\,\overline{B}\,Q_3 + A\,B\,Q_3$$

$$D = Q_1 + Q_3$$