**Note:** Please post your homework to ICS232 D2L on or before the due date.

If you do not post your homework on or before the due date, please post your late homework when complete (Late Homework: -15% Penalty).

1. (5 pts) Define the following
   1. Combination Logic:

**Combinational logic** (sometimes also referred to as **time-independent logic**) is a type of [digital logic](https://en.wikipedia.org/wiki/Digital_logic) which is implemented by [Boolean circuits](https://en.wikipedia.org/wiki/Boolean_circuit), where the output is a [pure function](https://en.wikipedia.org/wiki/Pure_function) of the present input only. <https://en.wikipedia.org/wiki/Combinational_logic>

* 1. Sequential Logic:

**Sequential logic** is a type of [logic circuit](https://en.wikipedia.org/wiki/Logic_circuit) whose output depends not only on the present value of its input [signals](https://en.wikipedia.org/wiki/Digital_signal_(electronics)) but on the [sequence](https://en.wikipedia.org/wiki/Sequence) of past inputs, the input history as well. This is in contrast to [*combinational logic*](https://en.wikipedia.org/wiki/Combinational_logic)*,* whose output is a function of only the present input. That is, sequential logic has [*state*](https://en.wikipedia.org/wiki/State_(computer_science)) (*memory*) while combinational logic does not. <https://en.wikipedia.org/wiki/Sequential_logic>

* 1. How are sequential circuits different than combinational circuits?

**Combinational Circuit –**

1. In this output depends only upon present input.
2. Speed is fast.
3. It is designed easy.
4. There is no feedback between input and output.
5. This is time independent.
6. Elementary building blocks: Logic gates
7. Used for arithmetic as well as boolean operations.
8. Combinational circuits don’t have capability to store any state.
9. As combinational circuits don’t have clock, they don’t require triggering.
10. These circuits do not have any memory element.
11. It is easy to use and handle.

**Examples –** Encoder, Decoder, Multiplexer, Demultiplexer

**Sequential Circuit –**

In this output depends upon present as well as past input.

Speed is slow.

It is designed tough as compared to combinational circuits.

There exists a feedback path between input and output.

This is time dependent.

Elementary building blocks: Flip-flops

Mainly used for storing data.

Sequential circuits have capability to store any state or to retain earlier state.

As sequential circuits are clock dependent, they need triggering.

These circuits have memory element.

It is not easy to use and handle.

**Examples –** Flip-flops, Counters

<https://www.geeksforgeeks.org/difference-between-combinational-and-sequential-circuit/>

|  |  |  |
| --- | --- | --- |
| Comparison | Combinational circuit | Sequential circuit |
| Basic | The output is discovered by the present state of the inputs. | Both the present input and past state output are used to identify the output. |
| Storage capability | Does not store data. | Can store a small amount of data. |
| Application | used in adders, encoders, multiplexer, etcetera. | Flip-flop and latches. |
| clock | Circuits do not rely on the clock. | Clock is utilized for performing triggering functions. |
| feedback | No requirement of the feedback. | Feedback is required. |

1. (5 pts) Add the following 8-bit two’s complement numbers (i.e. one sign bit and seven data bits) AND indicate “Overflow” if it occurs.

1110 1100 Carry bits

Example: 0101 0110 86

+ 0011 0110 + 54

Answer: 0 1000 1100 (-116) 140

Overflow: A Positive Number + a Positive number cannot be a Negative number

1110 1110

a.  1111 0111

+ 1101 0101

Answer: 1 1100 1100 No Overflow (Sign bit did not changed)

1111 1110

   b.  1111 1111

+ 0111 0101

Answer: 1 0111 0100 No Overflow (Can’t have Overflow adding different signed numbers)

1111 1110

c.  0110 1111

+ 0101 0101

Answer: 0 1100 0100 Overflow: A Positive Number + a Positive number cannot be a Negative number

1. (5 pts) Construct a truth table for the following:

xyz + x(yz)' + x'(y+z) + (xyz)'

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | z | F(xyz) |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |

1. (5 pts) Construct a truth table for the following:

(x + y')(x' + z')(y' + z')

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | y | z | x + y' | x' + z' | y' + z' | F(xyz) |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

1. (5 pts) Construct the XOR operator using only NAND gates. Hint: x XOR y = ((x′ y)′(xy' )' )'

x

x′

(x′y)′

((x′y)′(xy′)′)′ = x⊕y

(xy′)′

y′

y

x

y

((x(xy)’)’ (y(xy)’)’)’  (x NAND (x NAND y)) NAND (y NAND (x NAND y))

((x(x’ + y’)’ (y(x’ + y’)’)’ DeMorgan’s

((xx’ + xy’)’ (yx’ + yy’)’)’ distributive

((0 + xy’)’ (yx’ + 0)’)’ complement

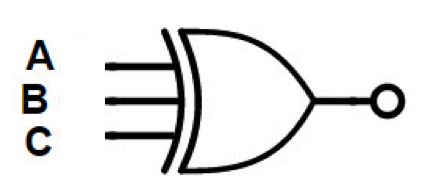
((~~0~~ + xy’)’ (yx’ + ~~0~~)’)’ identity

((xy’)’ (yx’)’)’

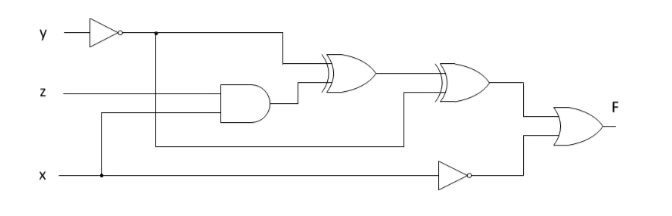
((x’y)’ (xy’)’)’ commutative

1. (5 points) Construct a truth table for a (3) three input XOR gate.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | Q |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |



1. (5 pts) Given the following circuit:



* 1. Define the boolean equation for the output function F(xyz)

F(xyz) = ((y'⊕xz)⊕y')+ x'

* 1. Define the truth table for the output function F(xyz)

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | z | F(xyz) |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

1. 5(5 pts) Given the truth table below:

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | Q |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

* 1. Write out the SOP (Sum-of-Products) minterm Boolean equation.

Answer: A’ B’ C + A’ B C + AB’ C + A B C

* 1. Construct the K-map for this circuit.

AB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

1. Pg 187 deterministic finite automata (DFA) (5 points) Define the deterministic finite automata (DFA) computational model and each term in the quintuple M=(Q, S, ∑ , δ, F)

DFA model:   
A finite-state machine that accepts or rejects strings of symbols and only produces a unique computation (or run) of the automaton for each input string.

[*Hopcroft, John E.*](https://en.wikipedia.org/wiki/John_Hopcroft); [*Motwani, Rajeev*](https://en.wikipedia.org/wiki/Rajeev_Motwani); [*Ullman, Jeffrey D.*](https://en.wikipedia.org/wiki/Jeffrey_Ullman) (2001). [*Introduction to Automata Theory, Languages, and Computation*](http://www.pearsonhighered.com/educator/product/Introduction-to-Automata-Theory-Languages-and-Computation/9780201441246.page) (2 ed.). [*Addison Wesley*](https://en.wikipedia.org/wiki/Addison_Wesley). [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*0-201-44124-1*](https://en.wikipedia.org/wiki/Special:BookSources/0-201-44124-1)*. Retrieved 19 November 2012*

Q:

A finite set of states that represents every configuration the machine can assume.

S:

An element of Q that represents the start state, which is the initial state of the machine before it receives any input.

∑:

An input alphabet or set of events that the machine will recognize.

δ:

A transition function that maps a state in Q and a letter from the input alphabet to another (possibly the same) state in Q.

F:

A set of states (elements of Q) designed as the final (or accepting) states.

(All the definitions without references are taken from the textbook).

1. (5 pts) Define the following terms as they pertain to Latches and Flip-Flops.

Note: The book does not cover these definitions very well. Please visit: <https://en.wikipedia.org/wiki/Flip-flop_(electronics)#JK_flip-flop>

* 1. Metastability

If two or more inputs to a flip-flop are changing at about the same time then the output can oscillate. It can take some time to settle into its state, causing the output to be unpredictable.

* 1. Propagation Delay

In general, this is the amount of time it takes for a signal to go from a source to a destination. In terms of flip-flops it is the time it takes the flip-flop to change its output after the clock edge.

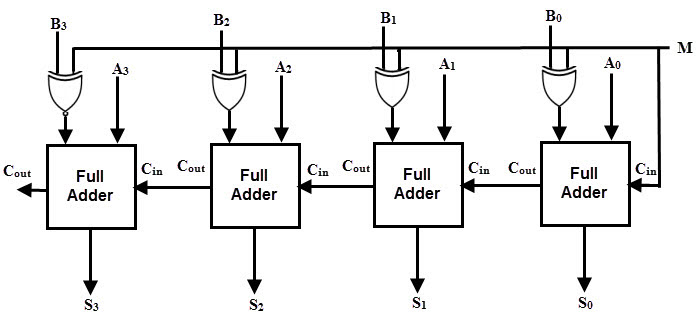
* 1. Setup Time

The Setup Time is the minimum amount of time the date input should be stable, before the clock edge, in order for it to be latched properly.

* 1. Hold Time

Hold time is the minimum amount of time after the clock’s active edge during which the data must be stable

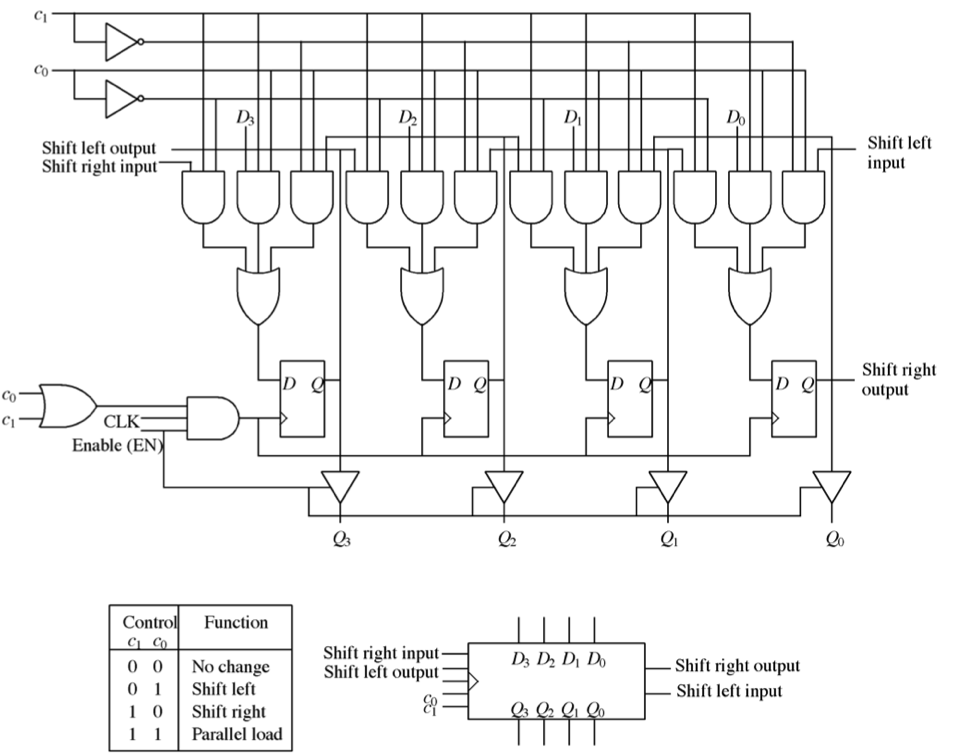
1. 5(5 pts) Construct the truth table for given the circuit below:



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| B0 | M | A0 | Cin | B0 ⊕ M | S0 | Cout |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

1. (5 pts) Given the left-right shift register shown below answer the following questions. Place an ‘x’ for don’t care conditions on the non-relevant outputs.

Note: Each Control input ‘c0 and c1’ as listed in the function table is shown twice in the diagram.



Enable

1. Fill in the shift output table below for the clocks Clk1-to-Clk4 given the following parallel load input to the shift register on clock Clk0. Place an ‘x’ for don’t care conditions on the non-relevant outputs.

Clk0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D3 | D2 | D1 | D0 | C1 | C0 |
| 1 | 1 | 0 | 1 | 1 | 1 |

Clk1-to-Clk4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Clk1** | **Clk2** | **Clk3** | **Clk4** | c1 | c0 |
| Shift Right Output | 1 | 0 | 1 | 1 | 1 | 0 |
| Shift Left Output | x | x | x | x |

for Clk0: C1 C0 Control bits set for a parallel load of latches from D0 through D3.

For Clk1-to-Clk4: C1 C0 Control bits set for a Shift Right Output; Don’t Care on Shift Left Output

1. Fill in the state output of Q0-to-Q3 on clock Clk4 given the input ‘Enable=1’ AND the following Input serial load of the shift-register in clocks Clk0-to-Clk3

Clk0-to-Clk3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Clk0 | Clk1 | Clk2 | Clk3 | C1 | C0 |
| Shift Right Input | 0 | 1 | 0 | 0 | 1 | 0 |
| Shift Left Input | 1 | 0 | 1 | 1 |

Clk0 -> clk 3: C1 C0 set for a Shift Right Input

Clk4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q3 | Q2 | Q1 | Q0 | C1 | C0 | Enable |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |

1st four clocks control bits set to ‘shift right’ so use the shift right input bit sequence to input values into the latches. Clk4 control bits are set to ‘no change’ and enable provides latch outputs to Q0 through Q3