ECE 742 - Digital System-on-Chip Design

Course Summary Report - Day 1

By: **Gouthami Channakeshavamurthy**

A20445435

Instructor: Dr. Jafar Saniie

Class Date: 09-07-2019

Due Date: 09-20-2019

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced. I did not receive any assistance from anyone or copy other sources and internet for completing this assignment.

Signature : Gouthami . C

Abstract - *This paper presents the summary of what was taught on Day-1 of Digital System-On-Chip Design course. It includes the basic introduction to the System-On-Chip designs and other logic design fundamentals, along with other references and examples that were shown in the class. Moving forward in the paper, we will understand why it is important to study the System-On-Chip designs for an engineer. We also study some Boolean expressions, digital circuits and their specifications.*

1. Introduction

Digital systems play an essential role in everyday life; the present technological world is called digital age. Digital systems are used in many communication traffic control, health monitoring and many other enterprises.one of the most crucial character of digital systems is their ability to represent and manipulate discrete element of information, there are fundamental reasons that the commercial products are made with digital circuits because by changing the program in programmable device same hardware can be used for many different applications. As the number of transistors on silicon increments to deliver complex capacities, the expense per unit diminishes and computerized gadgets can be purchased at an undeniably lessened cost. The one of the most important digital design is field programmable gate array (FPGA). This is a VLSI circuit that can be programmed at the user's location. An FPGA consists of an array of many logic blocks, surrounded by programmable input and output blocks and connected via programmable interconnections .this report consists of both combinational and sequential circuits which are used in digital design, and it also contains some applications in digital design.

2. Technical Contents

We first start with the discussion of System-On-Chip applications in modern world, how it has helped in the technological growth. Imagine present world without phones and laptops! That discomfort life is cured by the invention of SoC systems. These inventions have helped our life to be much more easier. The improvements in this field has given us devices that are more compact, light, speed and great power backup. Further we see the development in this field that were provided by our fellow IIT students. We were introduced to various ways of implementing these SoC design concepts in real-time applications.

A. Below are some of the projects which were demonstrated and discussed in the class:

- **Embedded Image and Video Processing Application on Virtex-5 FPGA,** *by C.Desmouliers, E. Oruklu, S. Aslan, J. Saniie, F. Martinez*

This video demonstrates the implementation of an Image and Video Processing Platform (IVPP) on FGPAs using PICO based HLS. This hardware/software co-design platform has been implemented on a Xilinx Virtex-5 FPGA. The video interface blocks are done in RTL and the initialization phase is done using a MicroBlaze processor allowing the support of multiple video

resolutions. This video discusses the architectural building blocks showing the flexibility of the proposed platform. This flexibility is achieved by using a new design flow based on PICO. IVPP allows custom-processing blocks to be plugged-in to the platform architecture without modifying the front-end (capturing video data) and back-end (displaying processed output). This paper presents several examples of video processing applications, such as a Canny edge detector, motion detector and object tracking that have been realized using IVPP for real-time video processing
[1][2].

- **Computer vision based Autonomous Robotic Arm**

In this video, a robotic sorting arm based on color recognition technique was presented. In this system, when a new frame is captured by the camera, the object will be detected using color-base image processing technique. The position of the object in real-world will be calculated by its mass center in image. Using Inverse Kinematics algorithms, the control input for the robotic arm will be calculated and then sent to Arduino microcontroller. Then, the microcontroller will drive the motors on the robotic arm to sort and position the objects according to their color. Using the proposed technique, the sorting robotic arm system can distinguish and sort different objects successfully with properly tuned parameters for both machine vision and 3D mobility of the robotic arm[1][3][4][7].

- **Real-time Video Transmission and Processing using Xilinx ZYNQ SoC,** *by Tianwei Yan and Yang Xu*

Real time processing of image and video applications is essential for functions such as image segmentation, object recognition, and feature tracking. Those functions help in making critical decisions in many applications. In this video, real time video Transmission and Processing using Xilinx ZYNQ SoC is demonstrated. The video explains, how the video HDMI and image processing modules are built/assembled on the hardware which enables the high performance video transmission and processing system. The hardware is divided into two parts i.e., *Green* - Central Hardware, where Zync based Zed board where the control system and image processing module is implemented. It also has video direct memory which can access every pixel by the program. *Red* - FMC expansion HDMI board which has HDMI input/output interfaces. Here, the laptop acts as a real time video input and the processed output is displayed on the monitor. There are 3 image processing Chanels implemented as part of this project:
- Edge detection channel
- Corner detection channel
- Original video channel

B. Logic Design Fundamentals

1. Combinational Logic Circuits

Some of the basic gates used in logic circuits are shown in Figure-1. Unless otherwise specified, all the variables that we use to represent logic signals will be two-valued, and the two values will be designated 0 and 1. We will normally use positive logic, for which a low voltage corresponds to a logic 0 and a high voltage corresponds to a logic 1. When negative logic is used, a low voltage corresponds to a logic 1 and a high voltage corresponds to a logic 0.[8]

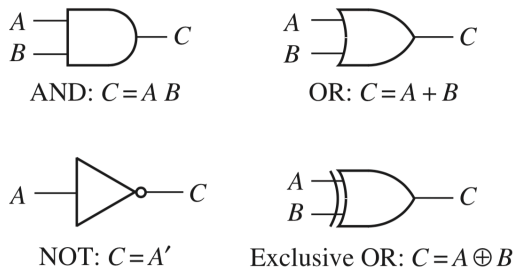


Figure - 1: Basic Gates

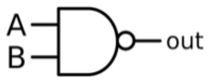
A. **AND Gate:** AND gate is an electronic circuit which gives logic 1 (high) as its output when all of its inputs are logic 1 (high). The symbol used for AND operation is a . (dot)

B. **OR Gate:** OR gate is an electronic circuit which gives logic 1 (high) as its output when one of its inputs is logic 1 (high). The symbol used for OR operation is a + (plus).

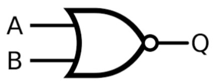
C. **NOT Gate:** NOT gate is an electronic circuit which inverts the input. NOT gate gives logic 1 (high) as its output when its input is logic 0 (low).

D. **XOR Gate:** XOR gate is an electronic circuit which gives logic 1 (high) as its output if either but not both of its two inputs are high.

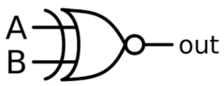
E. **NAND Gate:** NAND gate is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.



F. **NOR Gate**: NOR gate is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if any of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion.



G. **EXNOR gate:** The Exclusive-NOR gate circuit does the opposite to the EOR gate. It will give a low output if either, but not both, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.



The NAND and NOR gates are called universal functions as by using either AND gate or OR gate and NOT gate they can be generated.

Table-1 is a summary truth table of the input/output combinations for the NOT gate together with all possible input/output combinations for the other gate functions. Also note that a truth table with 'n' inputs has 2^n rows. You can compare the outputs of different gates.

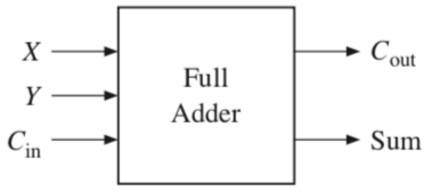
NOT gate	
A	$\overline{A}$
0	1
1	0

INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	EXOR	EXNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Table 1: Logic gates representation using the Truth table

- **Full Adder:**

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM. A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to the another.



(a) Full adder module

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(b) Truth table

Figure - 2: Full Adder

$$Sum = X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in} \text{ ——— } (minterms)$$

$$C_{out} = X'YC_{in} + XY'C_{in} + XYC'_{in} + XYC_{in}. \text{ ——— } (minterms)$$

$$C_{out} = (X+Y+C_{in}).(X'+Y+C_{in}).(X+Y'+C_{in}).(X+Y+C'_{in}) \text{ -- } (maxterms)$$

2. Boolean Algebra and Algebraic Simplification

The basic mathematics used for logic design is Boolean algebra. The below table summarizes the laws and theorems of Boolean algebra. They are listed in dual pairs.

	Duality Concept
Operations with 0 and 1:	
$X + 0 = X$	$X \cdot 1 = X$
$X + 1 = 1$	$X \cdot 0 = 0$
Idempotent laws:	
$X + X = X$	$X \cdot X = X$
Involution law:	
$(X')' = X$	
Laws of complementarity:	
$X + X' = 1$	$X \cdot X' = 0$
Commutative laws:	
$X + Y = Y + X$	$XY = YX$
Associative laws:	
$(X + Y) + Z = X + (Y + Z) = X + Y + Z$	$(XY)Z = X(YZ) = XYZ$
Distributive laws:	
$X(Y + Z) = XY + XZ$	$X + YZ = (X + Y)(X + Z)$
Simplification theorems:	
$XY + XY' = X$	$(X + Y)(X + Y') = X$
$X + XY = X$	$X(X + Y) = X$
$(X + Y')Y = XY$	$XY' + Y = X + Y$
DeMorgan's laws:	
$(X + Y + Z + \cdots)' = X'Y'Z' \cdots$	$(XYZ \ldots)' = X' + Y' + Z' + \cdots$
$[f(X_1, X_2, \ldots, X_n, 0, 1, +, \cdot)]' = f(X_1', X_2', \ldots, X_n', 1, 0, \cdot, +)$	
Duality:	
$(X + Y + Z + \cdots)^D = XYZ \cdots$	$(XYZ \cdots)^D = X + Y + Z + \cdots$
$[f(X_1, X_2, \ldots, X_n, 0, 1, +, \cdot)]^D = f(X_1, X_2, \ldots, X_n, 1, 0, \cdot, +)$	
Theorem for multiplying out and factoring:	
$(X + Y)(X' + Z) = XZ + X'Y$	$XY + X'Z = (X + Z)(X' + Y)$
Consensus theorem:	
$XY + YZ + X'Z = XY + X'Z$	$(X + Y)(Y + Z)(X' + Z) = (X + Y)(X' + Z)$

3. Exclusive-OR Theorem

The following theorems apply to exclusive-OR:

Here, we are inverting the complement

- $X \oplus 0 = X$
- $X \oplus 1 = X'$
- $X \oplus X = 0$
- $X \oplus X' = 1$
- $X \oplus Y = Y \oplus X$ – –– ––– –– –– – –– ––– – –––*(commutative law)*
- $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$ –– -- -(associative law)
- $X(Y \oplus Z) = XY \oplus XZ$ – –– – ––-- ––– - ––––-(distributive law)
- $(X \oplus Y)' = X \oplus Y' = X' \oplus Y = XY \oplus X'Y'$

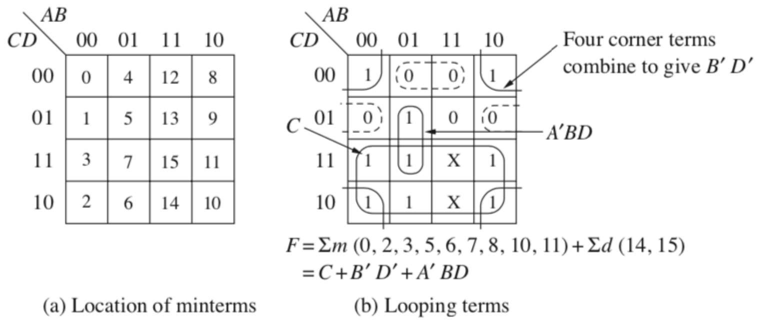
4. Karnaugh Map

A K-map is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized. Since any Boolean function can be expressed as a sum of minterms, it follows that a Boolean function is recognized graphically in the map from the area enclosed by those squares whose minterms are included in the function. In fact, the map presents a visual diagram of all possible ways a function may be expressed in standard form. By recognizing various patterns, the user can derive alternative algebraic expressions for the same function, from which the simplest can be selected[8].

The simplified expressions produced by the map are always in one of the two standard forms: sum of products or product of sums. It will be assumed that the simplest algebraic expression is an algebraic expression with a minimum number of terms and with the smallest possible number of literals in each term. This expression produces a circuit diagram with a minimum number of gates and the minimum number of inputs to each gate.

The Karnaugh map uses the following rules for the simplification of expressions by grouping together adjacent cells containing ones.

- Groups may not include any cell containing a zero
- Groups may be horizontal or vertical, but not diagonal.
- Groups must contain 1, 2, 4, 8, or in general 2^n cells.
- Each group should be as large as possible.
- Each cell containing a *one* must be in at least one group.
- Groups may overlap.
- Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.
- There should be as few groups as possible, as long as this does not contradict any of the previous rules.


AB
CD
00 01 11 10
Four corner terms combine to give $B'D'$
$A'BD$
C
$F = \Sigma m\,(0, 2, 3, 5, 6, 7, 8, 10, 11) + \Sigma d\,(14, 15)$
$= C + B'D' + A'BD$
(a) Location of minterms
(b) Looping terms

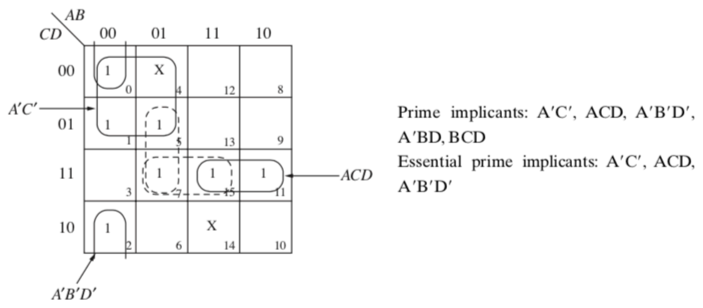

AB
CD
A'C'
ACD
A'B'D'
Prime implicants: A′C′, ACD, A′B′D′, A′BD, BCD
Essential prime implicants: A′C′, ACD, A′B′D′

Figure - 2: 4-Variable Karnaugh Map

5. Designing With NAND and NOR Gates

NAND and NOR are called universal gates because all the other gates like AND, OR, NOT, XOR and XNOR can be derived from it. In many technologies, implementation of NAND gates or NOR gates is easier than that of AND and OR gates. The below Figure-3, shows the symbols used for NAND and NOR gates. The bubble at a gate input or output indicates a complement. Any logic function can be realized using only NAND gates or only NOR gates.

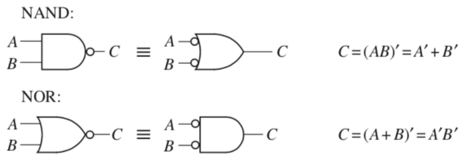
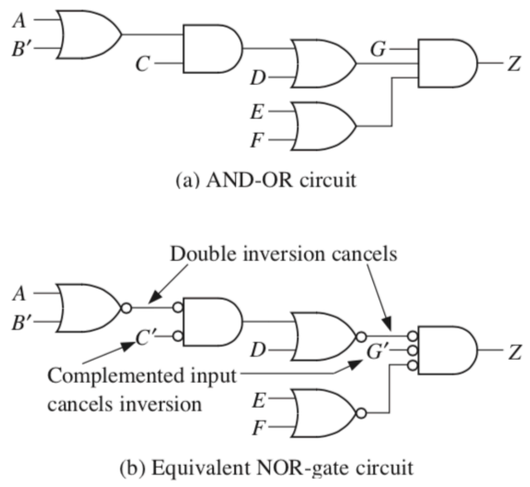


Figure - 3: NAND and NOR Gates


(a) AND-OR circuit
Double inversion cancels
Complemented input cancels inversion
(b) Equivalent NOR-gate circuit

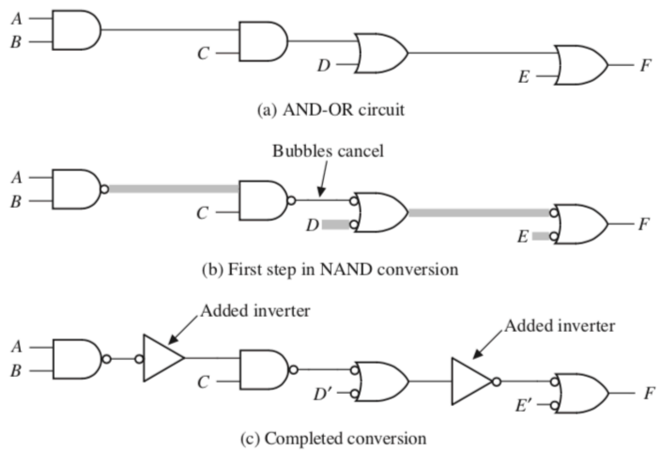


Figure - 4: Conversion to NOR gates

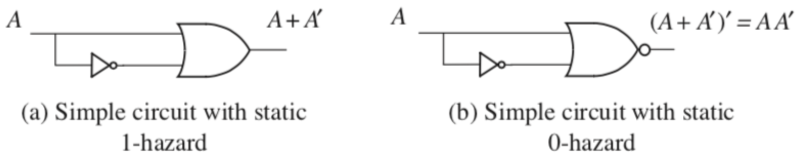
Figure - 5: Conversion of AND-OR circuit to NAND gates

6. Hazard Circuits

When the input to a combinational circuit changes, unwanted switching transients may appear in the output. These transients occur when different paths from input to output have different propagation delays. If, in response to an input change and for some combination of propagation delays, a circuit output may momentarily go to 0 when it should remain a constant 1, we say that the circuit has a **static 1-*hazard***.

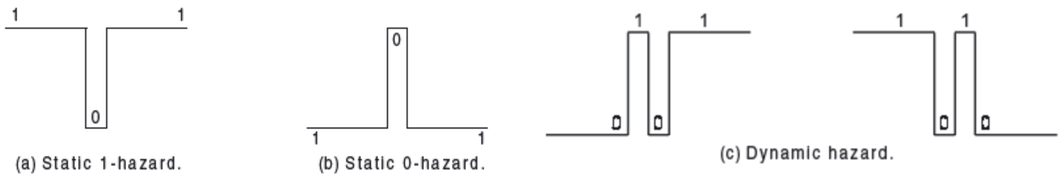
Similarly, if the output may momentarily go to 1 when it should remain a 0, we say that the circuit has a **static 0-*hazard***.

If, when the output is supposed to change from 0 to 1 (or 1 to 0), the output may change three or more times, we say that the circuit has a ***dynamic hazard***.



(a) Simple circuit with static 1-hazard
(b) Simple circuit with static 0-hazard

For larger circuits, we can rectify this problem by combining the different groups of 1's that are not combined earlier in the K-map, i.e., the minterms that are adjacent but not combined more prior This helps in terms of rectifying hazard



(a) Static 1-hazard. (b) Static 0-hazard. (c) Dynamic hazard.

7. Sequential Circuits (Flip-Flops and Latches)

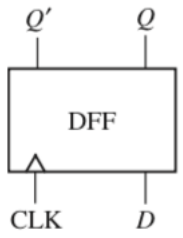
The sequential circuits consist of combinational circuits to which storage elements are connected to form a feedback path. The output depends on both present and past inputs. There are two types of sequential circuits synchronous and asynchronous.They usually contain a clock to synchronize their operation. Sequential circuits commonly use flip-flops as storage devices. There are several types of flip-flops, such as Delay (D) flip-flops, J-K flip-flops, Toggle (T) flip-flops, and so on.

A. Clocked D flipflop

The D Flip-flops are used as a part of memory storage elements. these are introduced as delay in timing circuits .In a D flipflop, if the clk is low, there is no effect on the output. And if the clk is high the output changes according to the input.In the above diagram D is the input and Q, Q+are output signals. If the CLK is high and D=0, then Q+ =0 and if D=1 Q+=1. The present output is same as input irrespective of the previous state of the output.

When D=’0’ Q^+=0

When D=’1’ Q^+=1



D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

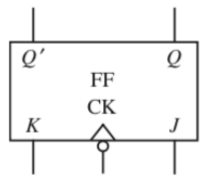
Figure - 6 Clocked D Flip-Flop with Rising-Edge Trigger

B. ***JK flipflop***

In the above table Q(t) represents the present state and Q(t+1) is the next stage one clock period later. The characteristic table above represents that if j and k are zero, then next state is equal to present state This can be given as Q(t)= Q(t+1), indicating clock produce no change .Nowadays , JK flipflop is not widely used. It is the extended D flipflop. It has lot of flexibility and reduces the logic for circuits.

When J=0 K=0	Q^+=Q Same as D flipflop
When J=0 K=1	Q^+=0 Reset
When J=1 K=0	Q^+=1 Set
When J=1 K=1	Q^+=Q′ Toggle (we can create clk/2 pulse in this mode)

Q^+=JQ′+K′Q



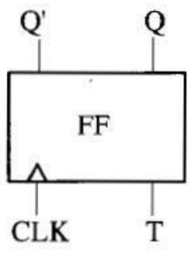
J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Figure - 7: Clocked J-K Flip-Flop

C. ***T-flipflop***

The T-flipflop is called a toggle flipflop. The T-flip-flops a complimenting flipflop and can be obtained from a JK flipflop when inputs J and K are tied together. The characteristic table of T flipflop contains only two cases if T=0 then the lock edge does not change the state, i.e., if T=0; Q+ =Qand if T=1 the clock edge compliments the output state Ie., T=1; Q+ =Q.

When T=0 Q^+=Q   When T=1 Q^+=Q′ Q^+=T $\oplus$ Q



T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

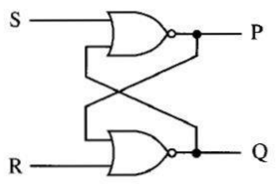
Figure - 8: T-Flipflop

D. <u>SR Latch</u>

SR Latch is a circuit with two cross-coupled NOR gates.The SR latch has two important states, i.e., when Q=1, Q+=0 the latch is said to be inset state and if Q=1, Q+=0 then it is in the reset state. In SR Latch

```
if S=0, R=0  then, Q+=Q
   S=0, R=1  then, Q+=0
   S=1, R=0 then , Q+=1
```

There is a fourth state, i.e., if both the inputs are one, then the output goes to zero, and the oscillatory condition occurs so normally we make sure that both input states are not one.



S	R	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

Figure - 9: SR Latch

E. <u>Mealy Sequential Circuit Design:</u>

There are two basic types of sequential circuits: Mealy and Moore. In a Mealy circuit, the outputs depend on both the present state and the present inputs. In a Moore circuit, the outputs depend only on the present state. A general model of a Mealy sequential circuit consists of a combinational circuit, which generates the outputs and the next state, and a state register, which holds the present state (see below figure -10). The state register normally consists of D flip-flops. The normal sequence of events is

1. the X inputs change to a new value;
2. after a delay, the corresponding Z outputs and next state appears at the output of the combinational circuit; and

3. the next state is clocked into the state register and the state changes. The new state feeds back into the combinational circuit and the process is repeated.

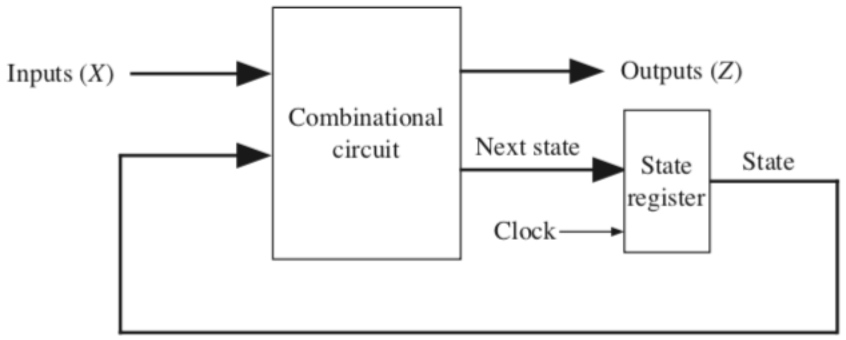


Figure -10: General Model of Mealy Sequential Machine

- *<u>Mealy Machine Design Example 1: Sequence Detector:</u>*

A sequence detector is a sequential state machine which takes an input string of bits and generates an output 1 whenever the target sequence has been detected.In a Mealy machine, output depends on the present state and the external input (x). In this model of FSM, the output values are determined both by its current state and the current inputs. The state diagram of a mealy machine associates an output value with each transition edge. Here 1/1, 1/0, 0/1, 0/0 represent input/output and S0, S1, S2 represent the states. Consider a part of the state diagram S1 -> S2 where “0/1” is written on the arrow. This can be interpreted as, when the current state of the system is S1 and when input “0” is applied, the system goes into next state - S2 and the output of the system is “1”.

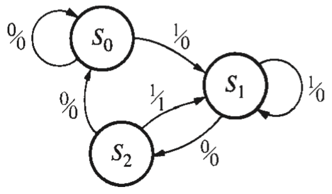


Figure -11: Mealy State Graph for Sequence Detector

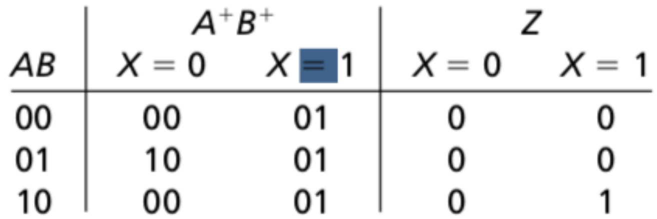
Present State	Next State $X = 0$	Next State $X = 1$	Present Output $X = 0$	Present Output $X = 1$
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

Table- 2: State Table for Sequence Detector

As an example of mealy sequential machine, let us see sequence detector. The sequence to be detected is 101. Thus if 101 occurs one-after-other at the input X sequentially, the output Z=1. In other cases the output is 0.

Initially, the state is s0, if the next input is 0, then the output is 0 and the next state is s0.

If next input is 1, the output is 0, and the next state is s1. The output is only 1when we get an input 1 while in state s2. Likewise, the entire state graph is constructed. After this process, we need to put the k=map for A+B+Z to get the Boolean relation. Then the hardware can be realized.



	A^+B^+		Z	
AB	$X = 0$	$X = 1$	$X = 0$	$X = 1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

Table - 3: Transition Table for Sequence Detector

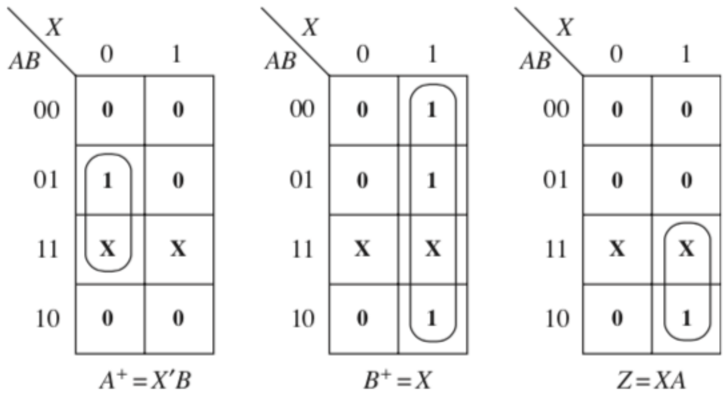


Table - 5: K-Maps for Next States and Output of Sequence Detector

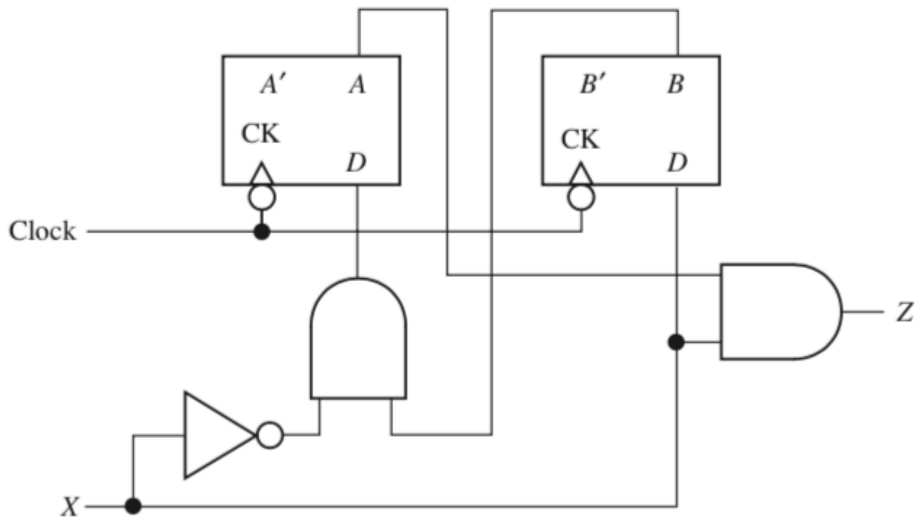
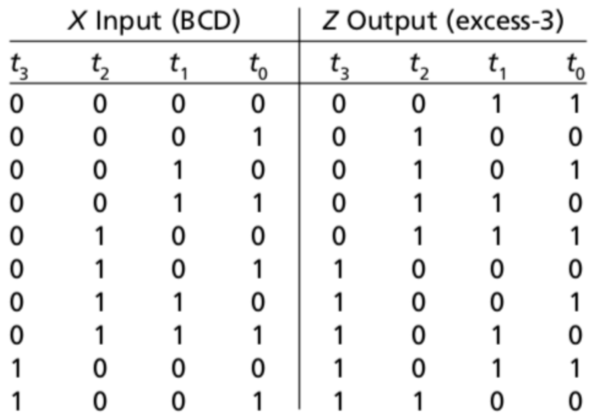


Figure - 12: Circuit for Mealy Sequence Detector

- *Mealy Machine Design Example 2: BCD to Excess-3 Code Converter*

The primary advantage of Excess-3 coding over BCD coding is that a decimal number can be nine's complimented (for subtraction) as easily as binary number can be one's complemented; just invert all the bits. Designing a serial code converter that converts an 8-4-2-1 binary-coded-decimal (BCD) digit to an excess- 3-coded decimal digit. The input (*X*) will arrive serially with the least significant bit (LSB) first. The outputs will be generated serially as well. Table 1-5 lists the desired inputs and outputs at times t_0, t_1, t_2, and t_3. After receiving four inputs, the circuit should reset to its initial state, ready to receive another BCD digit.



X Input (BCD)				*Z* Output (excess-3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Table -6: Code Converter

If all of the BCD bits are available simultaneously, this code converter can be implemented as a combinational circuit with four inputs and four outputs. However, the bits arrive sequentially, one bit at a time. Hence we must implement this code converter sequentially.

At t_0, we add 1 to the least significant bit, so if X =0, Z =1 (no carry), and if X =1, Z =0 (carry 1).Let us use S_1 to indicate no carry after the first addi- tion, and S_2 to indicate a carry of 1 after the addition to the LSB.

At t_1, we add 1 to the next bit, so if there is no carry from the first addition (state S_1), X= 0 gives Z =0 + 1 + 0 =1 and no carry (state S_3), and X =1 gives Z =1 + 1 + 0 =0 and a carry (state S_4). If there is a carry from the first addition (state S_2), then X0 gives Z= 0+1+1=0 and a carry (S_4),and X1 gives Z=1+1+ 1= 1 and a carry (S_4).

At t_2, 0 is added to X, and transitions to S_5 (no carry) and S_6 are determined in a similar manner. At t_3, 0 is again added to X, and the circuit resets to S_0.

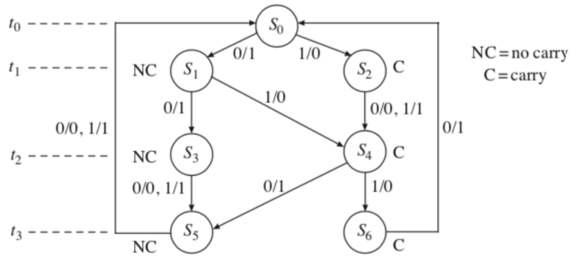
In order to reduce the amount of logic required, we will make a state assignment using the following guidelines [9]

I. *States that have the same next state (NS) for a given input should be given adjacent assignments (look at the columns of the state table).*

II. *States that are the next states of the same state should be given adjacent assignments (look at the rows).*

III. *States that have the same output for a given input should be given adjacent assignments.*

FIGURE 1-23: State Graph and Table for Code Converter



(a) Mealy state graph

	NS		Z	
PS	$X=0$	$X=1$	$X=0$	$X=1$
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

(b) State table

FIGURE 1-25: **Karnaugh Maps for Code Converter**

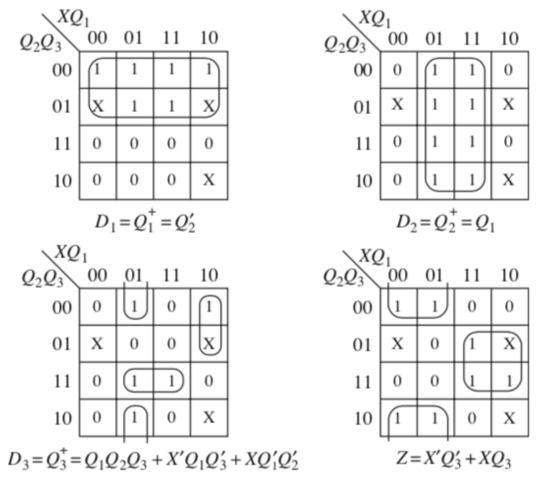
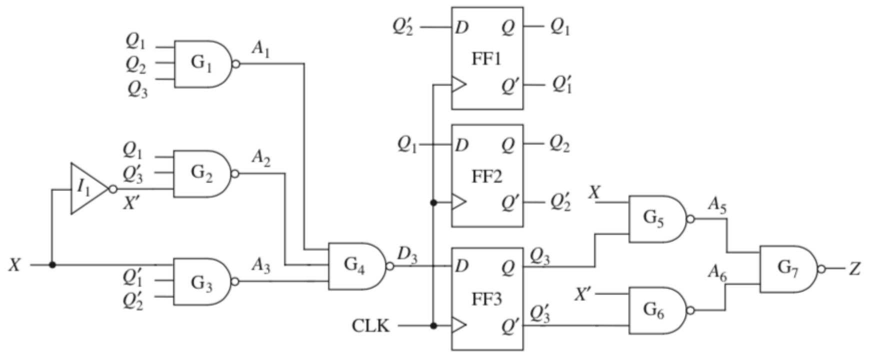


FIGURE 1-26: **Realization of Code Converter**



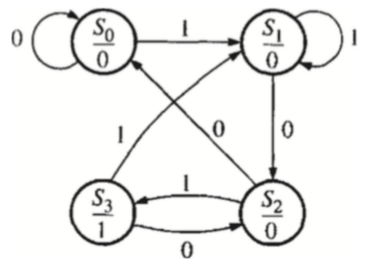
F. Moore Sequential Circuit Design:

In a Moore circuit, the outputs depend only on the present state. Moore machines are typically easier to design and debug compared to Mealy machines, but they often contain more states than equivalent Mealy machines. In Moore machines, there are no outputs that happen during the transition. The outputs are associated entirely to the state[7].

- *Moore Machine Design Example 1: Sequence Detector*

The circuit will examine a string of 0's and 1's applied to the X input and generate an output Z 1 only when the input sequence ends in 101. The input X can change only between clock pulses. The circuit does not reset when a 1 output occurs.

FIGURE 1-28: State Graph of the Moore Sequence Detector



Reset state designated So, If a 0 input is received, we can stay in state S_0 as the input sequence we are looking for does not start with 0. However, if a 1 is received, the circuit goes to a new state, S_1. When in S_1, if we receive a 0, the circuit must change to a new state (S_2) to remember that the first two inputs of the desired sequence (10) have been received. If a 1 is received in state S_2, the circuit should go to a new state to indicate that the desired input sequence is complete. Let us designate this new state as S_3. In state S_3, the output must have a value of 1. The outputs in states S_0, S_1 and S_2 must be 0's. The sequence 100 resets the circuit to S_0. A sequence 1010 takes the circuit back to S_2 because another 1 input should cause Z to become 1 again.

TABLE 1-7: State Table for Sequence Detector

Present State	Next State		Present Output (Z)
	$X = 0$	$X = 1$	
S_0	S_0	S_1	0
S_1	S_2	S_1	0
S_2	S_0	S_3	0
S_3	S_2	S_1	1

TABLE 1-8: Transition Table for Moore Sequence Detector

	A^+B^+		
AB	$X = 0$	$X = 1$	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

5. Applications

- **Basic Logic Circuits:** Logic circuits are found in several devices including multiplexers, arithmetic logic units, computer memory and registers. They are also used in microprocessors, some of which can contain over 100 million gates.
- **Flip flops and Latches:** Application of the flip flop circuit mainly involves in bounce elimination switch, data storage, data transfer, latch, registers, counters, frequency division, memory, etc. *Example:* Main real life applications of Flip flops are counter display at bank, token counter, microwave oven timer, because counter circuits acts only based on flipflop function and it will count value based on number of flip flops used. If 4 flip flops means count only 16 numbers (2 power number of flip flops,) there is no real time use of one individual flip flop because it is 1 bit storage element. Hence it used only with some combination circuit to make real time applications.
- **Mealy and Moore Machines:** Moore/Mealy machines, are FSM (Finite state machines) that have also output at any tick of the clock. Modern CPUs, computers, cell phones, digital clocks and basic electronic devices/machines have some kind of finite state machine to control it.
  - *Example1: Candy Machine* – Inputs: N (nickel received), D (dime received) – Outputs: C (dispense candy), R (give refund) – Should dispense candy after 15 cents deposited, + refund if overpaid. Then await next customer.
  - *Example2: Traffic light* - It is a system that consists of multiple subsystems, such as the different traffic lights, that work concurrently.
- **System on chip:** It is an integrated circuit that contains all the required circuitry and components of an electronic system on a single chip. It can be contrasted with a traditional computer system, which is comprised of many distinct components. A desktop computer, for example, may have a CPU, video card, and sound card that are connected by different buses on the motherboard. An SoC combines these components into a single chip.
  - *Example: Apple Watch* - The Apple "W" series is a family of "System on Chip" (SoC) and wireless chips with a focus on Bluetooth and Wi-Fi connectivity. The Apple W2 is a wireless chip from Apple used in the Apple Watch Series 3. It is integrated into the Apple S3 SiP. Apple released that the implementation of the chip makes Wi-Fi 85% faster and Bluetooth and Wi-Fi 50% more power efficient than the previous model's chip design.[10]

6. Conclusions

The fundamental concepts of digital design like gates, small modules like adders and reduction of Boolean expression using k-maps are studied in detail in this class. The difference between combinational circuit and sequential circuit is studied. In particular, mealy sequential circuit is analyzed with an example. The basic concepts and functioning of various flip-flops are studied and relationship between flip-flops are examined. Finally, digital application in the field of embedded computing is pictured. Towards the end, the conclusion is that digital systems are faster and more efficient that could be used for many real-time applications.

7. References

1. http://ecasp.ece.iit.edu/
2. https://doi.org/10.1109/EIT.2010.5612173
3. https://doi.org/10.1109/EIT.2017.8053385
4. https://www.youtube.com/watch?v=mMCpGTEKtmM
5. ecasp.ece.iit.edu/video/summer2019/china_fpga_2019.mp4
6. http://ecasp.ece.iit.edu/video/summer2019/china_fpga_2019.mp4
7. http://ecasp.ece.iit.edu/video/summer2019/ecasp_robotic_arm_2019_summer.mp4
8. Digital Design With an Introduction to the Verilog HDL, M. Morris Mano [Pearson 5th Edition]
9. Roth, *Fundamentals of Logic Design*, 5th Ed. [Thomson Brooks/Cole, 2004]
10. https://en.wikipedia.org/wiki/Apple-designed_processors