CONCEPT 9.1 TYPES OF NON-VOLATILE OR READ ONLY MEMORY

ROM or read only memory, devices store data that is not erased when its power is turned on or off. These devices are said to be “non-volatile”. Program data stored in read only memory is called “Firmware” because it is not corrupted by fluctuations in power or even by powering down the computer itself. When the computer first turns on, it relies on firmware in its ROM (Read Only) memory to teach it how to read the keyboard, display things on the computer screen and to load the other software from the hard drive. These basic essential programs that “boot up the computer” or teach it how to do all of its basic functions are called “BIOS” or basic input/output system.

Early versions of read only memories were factory programmed but this was expensive and final. Mistakes or changes in the program were not allowed.
Equipment manufacturers and end users demanded a device they could program as they needed. It was accomplished by adding fused links to each of the bits of the memory cells. A portable programmer was used to burn the links open. Mistakes caused only a few devices to be discarded rather than an entire production run.

Some manufacturers still needed read only memory chips that could be reprogrammed multiple times. Chip manufacturers responded by replacing the fuses with transistors. Various circuits were tried but none really worked. Almost by accident, it was found that by having a “buried gate” in an MOS transistor, the buried gate would pick up a capacity coupled charge and then hold it almost indefinitely. The only problem was that the charge could not be removed and the project was put aside. When it was revisited some months later, it was found that the part of the device that was exposed to the florescent lights in the room had been erased while the part of the memory that had been covered still had the original data in tact. This discovery led to the “Erasable Programmable Read Only Memory” or “EPROM”. This device could be programmed and then erased by exposing the surface of the memory device to ultra-violet light.
CONCEPT 9.2  TYPES OF RANDOM ACCESS MEMORY (RAM)

Random access memory usually refers to memory that can be read or written to on the fly. In reality, read only memory (ROM) can be randomly accessed so the name is a little misleading. The basic memory structure is like the ROM memory organization and decoding but the cells can be written to as well as read from.

The simplest RAM cell uses a single transistor just like an EPROM but the transistor gate is not buried. Since it is in electrical contact with the circuitry that controls it, the gate charge can leak off and the transistor can lose its data. If it is periodically refreshed by reading the cell and feeding the cell state back to the gate, this action restores its gate charge and output state. The time interval to lose the gate charge is short, only 10 ms. Ten milliseconds is an eternity to a computer so it is only a minor inconvenience. This type of memory is popular because it is easy to manufacture, requires less physical space for memory cells, and uses less energy. It is called “Dynamic RAM” or “DRAM”.

![Diagram of Erasable Programmable Read Only Memory Cell](image-url)
Early DRAM had to be refreshed externally to the memory chip. Each computer would periodically stop normal activity and would go out and refresh the memory. Modern DRAM memory is still used, but it has the circuitry on the chip that internally refreshed the memory. The computer can use it like any other memory device without worrying about refreshing the memory.

A flip flop is a memory cell. “Static RAM” or “SRAM” uses flip flops for each memory cell in place of the single transistor. SRAM has the advantage of being very fast and not requiring refreshing internally or externally. A single small battery can keep the memory alive without any external activity. SRAM memory is still used to keep vital startup information in computers although it is gradually being replaced by still another type of non-volatile memory called FLASH memory. (FLASH RAM is discussed below.) SRAM is not used in computer for large blocks of memory like DRAM is used but is used for a block of memory that is very fast and used to make computers execute code at a high rate of speed. This special computer memory block is called CACHE MEMORY and is normally made of SRAM type memory. SRAM memory is typically ten times faster than DRAM memory.
“FLASH MEMORY” is a cross between DRAM and EPROM memory. It uses a single transistor in each memory cell like DRAM and each transistor has a buried gate like EPROM. The gate is manufactured to be within a few atoms distance from a second gate used to write to the buried gate. FLASH memory is actually the slowest form of RAM but it is the fastest form of PROM, hence the name FLASH. It is being used to replace the SRAM memory in computers used to hold startup information that was kept alive with a small battery. FLASH RAM does not need an external power source to always be connected to retain its memory. Like EPROM, the memory is trapped as charge on the buried gate until it is physically changed by reversing the polarity on the program gate. This type of memory also has another name, “EEPROM” or “Electrically Erasable, Programmable, Read only Memory”. It is sometimes also called E²PROM or now FLASH memory.

**CONCEPT 9.3  ADDING MEMORY IN THE FORM OF FLIP FLOPS TO LOGIC CIRCUIT**

Up until now, each circuit we designed put out a definite response to a set of input conditions. This type of logic circuitry is called combinational logic. By adding memory to a logic circuit, the circuit now puts out a response based on input variables, and the history of what has happened before in the circuit.
The outputs are fed back into the circuit and used to make decisions about how the circuit should respond to input conditions. These output signals that are fed back into the circuit are called “State Variables”. The control signals that come into the circuit from the outside are called “Input Variables”.

Since this type of circuit depends on both input signals and state variables to determine how it changes, it is called a “State Machine”. The simplest of all state machines is a counter. It does little more than progress through a count sequence each time the circuit is triggered by an external clock pulse. The sequence can be a straight binary count sequence, or it can be a special count order. A state machine can be designed to progress through any conceivable order and can have outside control signals that change the order to any number of still different sequences. State machines are like special little computers that can be programmed to step through a process while looking for outside conditions to change the order of the steps at any time. Counters, combination locks, process control circuits, and alarm systems are just a few of the millions of applications for state machines.

**CONCEPT 9.4  SYNCHRONOUS AND ASYNCHRONOUS COUNTERS**

“Synchronous” means clocked while “Asynchronous” mean without clock. Most state machines are synchronous because the circuit relies on the information fed back from previous sequences. If things just rippled through the circuit without the synchronizing influence of a common clock, the circuit would become unstable data would move around the circuit in a random, unpredictable way.

There is one counter that is asynchronous however and it does work well. It is called a ripple counter and does not require a common clock. Each subsequent counter stage is clocked by the previous stage. When an event triggers the first stage, the result ripples through the subsequent stages. For this reason an asynchronous counter is also called a “Ripple Counter”.

A synchronous counter uses the same clock signal to trigger each stage and relies on the outputs from each stage to control how the counter transitions or counts. This is now a state machine because it feeds the outputs or states back to control the count sequence. The same simple three bit asynchronous binary counter can be designed as a synchronous counter.
In the asynchronous counter or ripple counter above, the clock changes the first stage and then the next stage must wait for the clock to transition through that stage before the clock arrives at the next stages and so on. The clock is delayed from stage to stage. The amount of delay is approximately 10 ns from stage to stage. It is still so fast that the delay is not a problem for most applications.

In the synchronous design, all of the flip flops are clocked in parallel or at the same time from the same source. When an event toggles the counter, each flip flop reads its inputs and transitions to the next count sequence. This requires the inputs to be set up to transition to the desired state prior to the clock pulse. When the clock comes, every stage then changes at the same time.

CONCEPT 8.5 SYNCHRONOUS COUNTER DESIGN

Synchronous counter design is straightforward. First you must graph the desired count sequence on a graph called a “State Graph”. Next the state graph transitions are transferred to a State Table, and from the state table, the design is realized. The process is briefly described below but is covered in great detail in Module 10.
A state graph has a circle for every possible state. If you have three state variables, your state graph will have eight circles if you have four, you will have sixteen circles and so on. Each circuit has S0 through S8 for the three variable graph or S0 through S15 for the four variable graph. Next you trace the desired transition path on the map with unused states going to the reset or beginning state which is usually S0.

The “State Transition Table” or “State Table” for short has the state variables in columns to the left in a straight binary sequence just like a truth table. The columns on the left side are called the “Present State” and they map each possible output combination just prior to a clock transition. Immediately to the right is a group of columns called the “Next State” and they show where each of the present states go after a clock transition occurs. The State → Next State transition follows the state graph. For example, if S0 goes to S5, the next state row to the right of S0 (0 0 0 ), would have S5 (1 0 1 ).

If “D” flip flops are used as the memory devices on the output of the state machine, the “D” inputs must have the next states present when the transition clock signal occurs. Then the “D” flip flops will transition to that next state. The next state becomes the new present state when it is transferred to the output as the flip flop is clocked and the state machine is then ready to transition to the next state in the state graph. The example below shows how this is accomplished.

If you were going to design this and burn it into a programmable device, it would not be necessary to simplify the design. The programmable device will have more than enough gates and inputs to easily accommodate the unsimplified design.
**DESIGN OF A SEQUENTIAL COUNTER**

1. Draw a state graph showing how you want the machine to transition.

   ![](state_graph.png)

2. Transfer the state graph to a State table.

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State A B C</th>
<th>Output or Control Variables DA DB DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 0 0 0</td>
<td>S1 0 0 1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>S1 0 0 1</td>
<td>S2 0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>S2 0 1 0</td>
<td>S3 0 1 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>S3 0 1 1</td>
<td>S4 1 0 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>S4 1 0 0</td>
<td>S5 1 0 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>S5 1 0 1</td>
<td>S6 1 1 0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>S6 1 1 0</td>
<td>S7 1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>S7 1 1 1</td>
<td>S0 0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

3. Use K maps to simplify the “D” input design equations

   - \( D_\alpha = AC + AB + ABC \)
   - \( D_e = B\overline{C} + \overline{BC} \)
   - \( D_c = C \)

4. Design the State Machine using the equations you just simplified

   ![](state_machine.png)

Common Clock
CONCEPT 8.6  MONOSTABLE MULTIVIBRATORS OR “ONE SHOTS”

One shots are devices that will put out a single pulse when triggered but an external transition. They can be designed be triggered by either a rising edge or a falling edge trigger. The device puts out both a Q and /Q output so you can have a low to high to low, or a high to low to high pulse. The width of the pulse is determined by an external capacitor and resistor that connects externally to the device. The width of the output pulse is roughly equal to the value of the resistor times the value of the capacitor in seconds.

CONCEPT 8.7  APPLICATIONS OF “ONE-SHOTS”

One-shots are used in a variety of places. They can be used to “debounce” switches or to remove electrically noise that occurs at the instant a switch closes.

There are two types of one shots, a “retriggerable” type that restarts its time out time with each trigger pulse and a non-retriggerable type that completes its time out before allowing itself to be triggered again. The retriggerable one-shot is used in computers as a “watch dog timer”. A watch dog timer keeps testing to see if the computer program is running normally and if it is not, it resets the computer. This is done by programming the computer to repeatedly send out a trigger pulse to a retriggerable one-shot. The output of the one shot goes to the computer reset. If the program fails to trigger the one-shot before it can time out, it resets the computer thus prevent it from going into a non-sense loop and hanging up.

Since the pulse width is always the same for a one-shot, it can be used to make a simple tachometer. The input to the one-shot is from a circuit that gives out a pulse each time a motor shaft turns. The output pulses then go to a capacitor that receives the same quantity of charge each time the one-shot pulses. The more frequently the one-shot pulses, the higher the voltage will be accumulated across the capacitor. This voltage will be proportional to the number of revolutions per minute of the motor.

There are several monostable multivibrator or one-shot chips out there. In the 7400 series, there is the 74121, 74122, 74123, and the 74221. These devices are available in most of the families, not just TTL but also LS, HC, HCT, etc. The data sheets for the LS series are on your CD in PDF format.

The LM555 will also work well as a one-shot if configured with the trigger pin 2 externally clocked.
CONCEPT 8.8  THE 555 TIME CHIP

A 555 timer is a small device that can be made into a one-shot, or a “free running multivibrator” or clock generator. The device requires several resistors and capacitors to be connected to it externally to work. The operation depends on how the trigger is connected. It can be self retriggering, or can require and external pulse to trigger it. When triggered, the capacitor charges at a rate determined by its value and the value of resistors in series. When the capacitor reaches a threshold voltage, it trips and internal flip flop and switches itself into a discharge configuration until the charge is dumped off of the capacitor. During this time, it outputs a pulse. If it is connected in a self retriggering configuration, it then repeats the charge discharge cycle, putting out a string of pulses that can be used to clock digital circuits and events.

There are several good sources on how to use the 555 timer. The manufacturer specification sheet that is included with your CD has the PDF files on the LM555. Another good publication on the 555 timer was written in 1984 by Forrest M. Mims III for Radio Shack. I could not find it in print any longer however. The reference is:

“Engineer’s Mini-Notebook, 555 Timer IC Circuits” by Forrest M. Mims III, Published by Tandy Corporation, Fort Worth TX 76-102.
Radio Shack Cat. No. 276-5010A