I will be replacing Professor Agarwal today because he is away. I am one of the recitation instructors for those of you who have not seen me.

We will talk today about a neat application of RC networks and expand those to application in MOS memory systems.

To connect with everything, we will get back to the basic circuit that we have been discussing so far.

And you recall the circuit that we have been studying, the canonical RC with an input voltage function of t.

And we had specified that we solved this problem for the case of a step input or a condition in which a t=0.

At t greater or equal to zero vI is equal to some capital VI value that for now on is constant.

And the other condition that we discussed was the value of the voltage on the capacitor that would exist at time t=0.

Let's call that vc(0). And in general there is some finite value here. It can be zero or it can be different from zero. Given that, we learned how to write down directly, without messing around with differential equations, the answer for the voltage on the capacitor vc(t), let me define also my vc right here, is equal to VI, the final value, plus vc(0), the initial value on the capacitor, minus the final value, e^-t/RC.

This is our standard equation to which we plug in, and it's either a rising exponential if VI is larger than VC or a decaying exponential if VI is a smaller value than VC.

This should all be familiar. And, again, as pointed out in the notes, the reading for today is 10.3 and for the new material you should look at Chapter 11 where we discuss memory.

This is where we stood as of last time.

Now, I would like to discuss a little bit more about the storage of charge in capacitors. And how we can take advantage of that for storing logic state. One of the things that I am sure you must be aware of is that one of the perhaps most massively produced chips is actually the so-called DRAM which you find in every PC and every computer that exists anywhere. This DRAM is dynamic random access memory in which we can store a state and come back and look at it at any time later, provided we don't power off our machine. The logic state in the basic memory elements, of which instead there are close to 1 giga elements per chip, are stored on capacitors.

And so we will play a little bit with that concept today.

And, although we're not going to discuss the specific example of the DRAM, the basic elements of the DRAM you

will see actually in a demo shortly. So that's the general response of this network that I have here to an input VI that happens at t=0. Now, the one thing that you recognize immediately is that it really doesn't matter what the value of VI was for t less than zero.

What really counts is the value of VI at t=0.

And that's the value that we're interested in.

Now, there is an implicit statement in that.

And that statement is that somehow that network appears like this at t=0. So, there has to be some switch there, and you will see that, that basically starts my condition to that at t=0. And so the history of VI really doesn't matter. The response following that equation that we have there will depend on the initial value which is vc(0) here. Now that is the voltage on the capacitor at that time. And then assuming that VI is a value that is larger than vc(0) will have a rising exponential that will come to this value. And this is the time constant RC and this is time. So, the capacitor starts with some voltage here and goes to a new voltage that is imposed by the input for time greater than zero.

We can define at any one time, say this time, this time, this time, this time the state of the capacitor. The state.

What is the state of the capacitor?

The state is the summary of all inputs that are relevant to predicting the future. If I know the state of the capacitor this time, I can predict what it is going to go given a response VI here in the future.

So, predicts the future. Now, what is the state variable on the capacitor? What is actually stored on the capacitor? You can say, well, what is stored is voltage.

The real physical quantity that is stored is the charge q which is for linear capacitors related to the voltage, let me actually write it correctly, vc like this.

So, the real state variable is this.

But for a linear capacitor, since there is one-to-one relationship between the two, v is also a state variable.

OK, so let's then go back to our original circuit.

What we have is -- -- vc(t), so that's the future value of the voltage on the capacitor, is a function of vc(0), the initial value and the variable input now in the future time.

And for the case of vI(t) being constant VI for t greater or equal than zero we have the equation that we just described.

Nothing new.

All the past inputs to the capacitor for time t less or equal to zero is summarized in this value.

And vi being constant the future is predicted from that.

So, that's the concept of the state.

There is an initial state on the capacitor.

And then there is a final state that will be reached when equilibrium actually is achieved.

There is a fair amount of discussion in the text, and we don't go in great detail here, but it is both convenient for analysis and also it's interesting in many cases to look at the response of a linear network for two different conditions. So, we're interested in two cases. One is the so-called zero state response. Now, what is the zero state response? It's the response to a condition in which we impose an input and impose also the condition that the initial value, initial state of the capacitor is zero. So then we ask how does it respond to vi(t)? So, starting with a capacitor at zero state what is the response?

It allows us to decouple the initial conditions from the response to the input. Now, you will see that this is actually very useful. The second condition to which we're also very interested is the so-called zero input response. What is that?

That is vi(t)=0. Now, it's the condition under which there is no input. vi(t)=0.

The question here is how does it relax?

We're starting with an initial state.

So, how this state relaxes out in the circuit.

Now, the zero state response, this one here is Z so called SR for our case, which I will write like this, vC, ZSR is simply a rising exponential.

We start from zero and we go to VI.

So, it's VI-VI e^-t/RC. So, that's the ZSR.

The ZIR, the zero input response is like this.

It's the decay of the initial voltage on the capacitor to zero or to equilibrium. Starting from vC(0) we're decaying like this. Now, do you see something that's rather obvious from what's on the board in terms of ZIR and ZSR and the final complete answer which is there?

They are specific cases, but how do they relate to the full answer? It's the sum.

It's the superposition of the two.

What basically we see here -- And that's actually a general statement, is that vC = vC,ZSR + vC,ZIR.

Now, you may say this is trivial because we started from that, ended back in that from some very simple observations.

However, we are not always solving networks for responses that are steps. The input voltage may be a ramp. We did that in recitation.

Or, it could be an impulse. Or, it can be a more complicated function. Having this observation in place actually allows us to solve the problem rather neatly.

If I have time at the end, I might come back to this.

So, this is the same equation as I started with, arrived at from a principle of superposition of two different solutions. One application of state which can be, since we have energy storage element here, the capacitor, which can be stored on the capacitor is in memory. And you may ask, so why do we need a memory node to perform logic?

Well, there are cases in which a result depends on previous results. So, a computation proceeds in time. In order to do that, we need to store intermediate results and proceed forward.

One good example is if you're doing a continuous summation, say, on your calculator, you keep putting things in the memory. The M+ button, right? And you keep adding a series of numbers. Every time we store the sum of the previous operation we add another number and so on.

Clearly we need some way of storing state.

For a complete computing system, we need combinational logic and we need memory. In fact, these are the two basic elements that are essential for any kind of computing system. We need to remember intermediate results. We need to remember transient inputs. And that's the role that all these enormous amount of memory that comes

to play in computers is doing. The basic memory abstraction is as follows.

Imagine a block which needs to be populated by transistor, resistor, capacitor, whatever elements.

And it has a control input, which we will call the store.

It has a state input that we will call dIN and has an output dOUT. When we're telling this element, OK, now it's time to store, it looks at the input dIN and stores it for, in principle, an infinite amount of time. If we were to make a drawing of this, of what this looks like, let's suppose, let me do all this in one axis. So, time moves this way.

Let's suppose that we have an input dIN that looks like this, and the store command comes in the form of a logic.

Let's actually suggest here this is logic one, this is logic zero. And, although this is not absolutely necessary, let's also define that the store command comes in the form of a logic one at this store input. Store, let's say, looks like this.

What does the output look like then in this particular case?

Assuming that the output was dOUT, the stored element was zero prior to the store, then the output would look like this. This is dOUT.

As you can see, it would remember the one that it saw at this point. In fact, it would do that irrespective of what was stored in this memory cell.

For example, suppose it was storing one and the output didn't change, it's still one.

If it was storing a zero, it would flip to a one.

If we had another store, let's say here, then what happens? Then it would go back down to zero because now we sampled an input that is zero and we flipped the state. That's what a memory -- -- element or cell would do for us.

It would remember the output state.

And, not only that, but in principle it should be undisturbable. In other words, I may do something to this dOUT but it should not flip the state. And that comes about quite a bit. Because in actual integrated circuit memory there is lots and lots and lots of nearest neighbors to this cell which, when they're flipped, have a cross-coupling to the cell.

The cell must be designed robust enough that it doesn't flip, that no coupling actually occurs.

All right. Now we're going to try to apply what we've learned so far to invent a basic memory element.

And, believe it or not, this is the key to the DRAM.

Let's implement this in a circuit.

Suppose I have a switch here like this.

And I will put a capacitor. I take my dOUT here.

This is dIN. And the switch is operated by a command here that we will call store.

When store is one it goes up. When store is zero it is down here. That's capacitor C.

This is the storage node.

What are we actually storing in this case?

Let's suppose that this voltage here is 5 volts.

I flip the switch up to one and I flip it back down to zero.

What's the voltage in this capacitor here?

5 volts. Now the capacitor is at 5 volts, I put dIN to ground, flip the switch back up and then back down to its known storing condition.

What's the voltage in the capacitor?

It's zero, exactly. So, it does store the value of the voltage that it saw, five or zero, high and low. It stores it because it stores charge. That's actually the physical quantity that's stored. It's manifested as a voltage, which we see. All right.

Now, is this, oh, before I move from here.

What is the basic cell in a DRAM, one that you go out and buy by the billions of cells? It's actually this.

The only difference is that this switch here is replaced with a MOSFET.

And that's all it is. So, a MOSFET plays the role of the switch. When the gate is high this is a resistor and connects the input to the capacitor.

And when the gate voltage is below the threshold voltage this is an open, as we've seen, and it isolates the transistor from the output. So, that's the basic memory element.

And, as I said, it's the key to a DRAM.

OK. Now let's consider a little bit the conditions of operation of this thing.

Let me draw the circuit in two conditions.

One in which it is storing, one in which it is sampling and one in which it is storing. Not to redraw this thing.

Assuming that I have a MOSFET there, I would have the on resistance in place here when store=1.

Now, in principle, the output is connected to -- -- some load resistance. We'll talk a little bit more about this load resistance in a minute.

This is the situation when we are at store=1 situation.

For example, let's suppose that dIN is 5 volts. Now, what is the situation for store=0? It's very simple.

We have the capacitor C and dOUT and here we have a resistance. The switch is open.

This is store=0 condition. What we have in this case is we have a problem similar to what I was discussing earlier.

It is a ZIR, if you like, situation. And this you can think of as a ZSR if we're starting with zero charge on the capacitor, but I'm interested in this part.

In this case, I am starting with a vC(0)=5 volts. And I'm asking myself how long will this cell hold the value? And, in fact, that is actually what happens in a dynamic RAM.

The value on the capacitor is not stored forever.

In fact, that's why we call it dynamic because we have to come back and restore it every once in a while.

For how long are we going to store the charge?

What's the response of vc for t greater than zero after the switch flicked? It's very simple.

It's vc is equal to 5 volts e to the minus t over RC, right? That's the response.

We have a decay. And applying to the things we know. We start from 5 volts, let's say here, I have a decay going down towards zero, at some point we are going to cross the threshold for high. The only period in which I have a

valid output, if the capacitor was storing a one, is this period here. This is the only period in which I have valid stored one because, once I go beyond capital T here, I have crossed the legal limit, threshold for discriminating a high output.

And from then on the output is no longer valid.

So, this memory is good provided time is less than capital T. It's not a case in which the capacitor can hold charge forever.

In fact, we can calculate, that is we can solve for T in this particular case. It's in your notes.

Nothing really profound. T is equal to minus RC log VOH over 5 volts. So, this is basically what the response is going to be. Now, there is an implicit assumption here, which is that the store pulse width is much, much larger than RON C.

In other words, when we want to store a one here starting from zero, we better charge it all the way up to 5 volts in the time that our switch is connected here.

And what is the relevant time constant?

It's going to be the RON C. In fact, it's actually the RON parallel RL with C. But typically RON is much, much less than RL so we don't have to worry about that.

Dominant time constant is RON C.

So, provided these things are happening, we have a memory.

Now, we can try to improve things a little bit.

We see here that we will have a decay to an invalid state in time T. How can we improve things?

One way to improve things are the buffer.

Here is our memory element again.

Here is the capacitor. This is the storing node.

Now I am going to put the buffering effect.

I am going to put two buffers here.

Two invertors, I should say, because if I am storing a one here I want to be able to see a one here as well. And, in this case, what I am looking at is the RIN of the buffer.

And, in principle, I have out here the RL.

Now, this is better because if RIN is much larger than RL then the time T, in this case, is much larger than the case without buffer. So, we buffer the effect of VL.

This could be one of these neat circuits we saw in recitation like a source faller, for example, or it can be just an inverter in which case you just see the input of a transistor. So, now this condition can be satisfied. Let me give you some cases which are some numbers that are typical for a dynamic RAM.

Typical times we're talking about is RIN on order of 1 gigaohm and storage node capacitor on order of 1 femtofarad to one picofarad. Now, if you can do the math in your head, which is just multiplication, you will see that the time constant, the RC is between 1 millisecond to 1 microsecond. And for DRAMs, actually, we try to be in the order of milliseconds.

These are the times we're talking about.

If I have this kind of circuit, somehow there has got to be additional circuitry that comes back, samples the voltage here and restores it. And that is actually what is happening in a DRAM. And my laptop is working there and its DRAM keeps getting refreshed every, say, millisecond or whatever the condition is.

But, in our case, we are going to do a slightly different case in which we will create a static memory.

Let's actually look at, first of all, the case of the discharge. Pay attention to, let me actually break the loop here.

This is my capacitor. This is a resistor that is in series with a capacitor like you see here.

Actually, I am going to keep that resistor in series with the capacitor, even in this case, because I have it for my second part of my example. I charge the capacitor to 5 volts. And you can see here this lights up, I hope everybody can see it, proportional to the voltage that I have here. From here on it's all logic levels. So, the intensity of light here will always be the same. It's either lit or it's not lit. Right now I am charging the capacitor. In fact, let's see.

Maybe I can discharge the capacitor first.

Here the capacitor is discharged.

As you can see, the input is zero, the output is a one, and then the output of this inverter here is a one. I have two

inverters in series.

And I am going to charge the capacitor.

I charged it to 5 volts and this lit up, this is off of course, that's an inverter, this is a valid zero, produce a valid one. And now I am going to take the input out. As you can see it's stored.

In fact, we have to wait for a very long time.

We don't have enough time to wait for this to discharge, so instead what I am going to do now is I am going to add also the resistor. Now I am going to flip the resistor in parallel with the capacitor to imitate what happens when we have an input resistance.

You saw that there was a discharge of the capacitor.

This input level went down. Voltage here flipped over to a one. Let me do it again now with a resistor in place. Storing charge on the capacitor. That's the store command.

Now, don't store. I have less, about a second. The element here is 20,000 microfarads and 100 ohms which gives me a time constant of two seconds. Assuming a VOC of the order of, let's say, I don't know what it is for this case, 2.5, the log would be about 0.5, so it cuts basically the time to about one. So, it lasts about one second, if my math is all correct. It's actually a little longer than a second, excuse me, but the point is that the charge is gone. Now, notice, however, that there is something I can do here, which is that suppose I take the switch or a switch and bring it back and provide a path from the output to the input here.

And this switch is open when this is closed and closed when this is open. So, this basically is the compliment of store. What I am doing now is I put a charge here, it produces a valid one at this point, and then I am feeding this valid one back to the input.

As you can see, this will now allow me, even though I have a high resistance, to store the value for a long time. In this case, what I am going to do is I am going to connect the output, as you can see here. And I have my resistor in.

And I am storing zero here, storing 5 volts.

Now I am going to flip the switch.

Basically, I mean the don't store, don't look case.

You notice this dims a little bit.

Sorry. No, I want the resistor in.

There. Yes.

OK, so the output remain value. This dimmed a little bit but the output has remained OK. All right.

So, we've provided a feedback. Now we've created a static memory. This will hold charge for as long as the circuit is powered up.

Now, there is still one little problem that I have with this kind of configuration. And that is if I disturb this output the charge may, the state may change.

So, for example, let's say that I have -- I disturbed it by coming close to it, so let's charge it again.

OK. I flipped the switch.

I flipped the state from the output.

That is an invalid condition. I shouldn't be able to do that.

How do I avoid that? How can I avoid this problem that you just saw?

Well, I need yet another buffer.

The answer is in your notes. If I don't take the output here but rather take the output here, or if I don't want an inverted output, if I don't want an inverted output, I could put yet another element there.

Then the situation would be fine.

In this case, let me do it again.

Charge.

Why isn't this lit?

A bad one?

Now, of course we disturbed the input.

Now, of course I can do anything I want here.

Nothing happens, but you may say this is a trivial case because this is already zero.

So, I am going to change the state.

Here's is the changed state. See.

I can show this. Nothing happens up there.

So, this is an interesting situation in which I am buffering the output so that the output does not feed back to the input. And, by and large, in designing circuits this is something that we do.

Now, in the remaining three minutes there is an example that we have. Can we put the laptop here?

OK, so here is an example of how memory can be put together now to create something a little bit more complicated.

And you can see the memory cells that we were discussing here. There's four of them, so this is a four bit memory. There is a decoder at the beginning here which decodes the address of each cell, so the input here will tell me which cell I need to address.

Let's look at the truth table. This is the truth table for the decoder. As you can see, depending on the address that I have here, this is zero, one, two and three in a binary system, only A, B, C or D is up, is high.

Which means that this end operation here only allows the input that is presented to all of the cells, what is going through the AND gate here to appear at the output. If, for example, we have a one, zero, the only end input that is going to be high is going to be this one.

And that means the only cell that will look at the input when the store comes up is going to be this one here.

At that point it will store whatever is on the input cell because that's an AND operation. That is a simple example of a memory. And following that simple arrangement you can build incredibly large memory systems.

So, that's all I had for today. And I will see you on Tuesday.