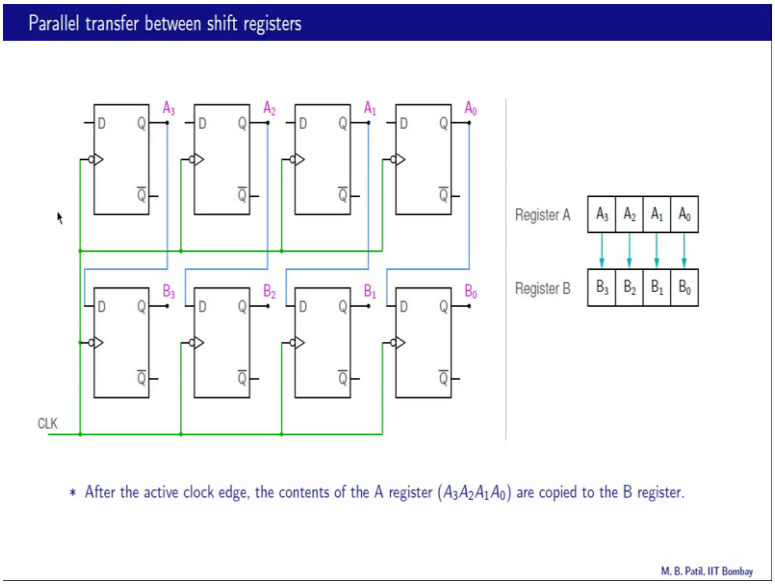
Basic Electronics
Prof. Mahesh Patil
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Lecture - 65
Shift registers

Welcome back to Basic Electronics. In this class we will continue with the shift register, we will look at a bidirectional shift register, we will also see how a parallel transfer between 2 shift registers can be implemented, we will look at how parallel in serial out data transfer can be achieved. Next we will see how shift registers together with an adder can be used to multiply 2 binary numbers. Finally, we will start with a new topic namely counters. So, let us start.

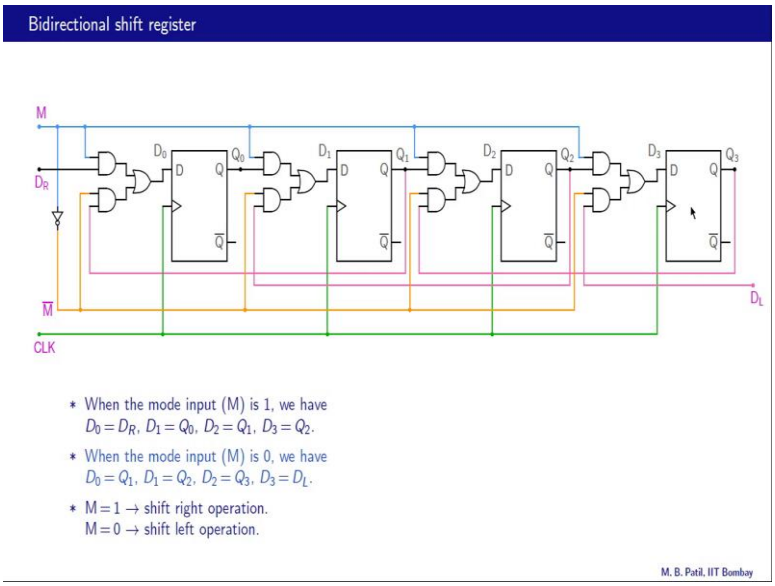
(Refer Slide Time: 00:53)



Let us suppose now that we have 2 shift registers this is A 3, A 2, A 1, A 0 are the bits of the first shift register and B 3, B 2, B 1, B 0 are the bits of the second shift register. What we want to do is to transfer the data which is in this first shift register into the second one. So that is easy to achieve, what we need to do is to connect this A 3 as the D input of this B 3 flip flop; A 2 over there and so on. And now after the active clock edge the contents of the a register will be copied to the B register, because this A 3 would show up as Q here which is B 3 A 2 will show up as B 2 and so on.

So, that is a very easy operation and it is represented schematically over here.

(Refer Slide Time: 02:04)



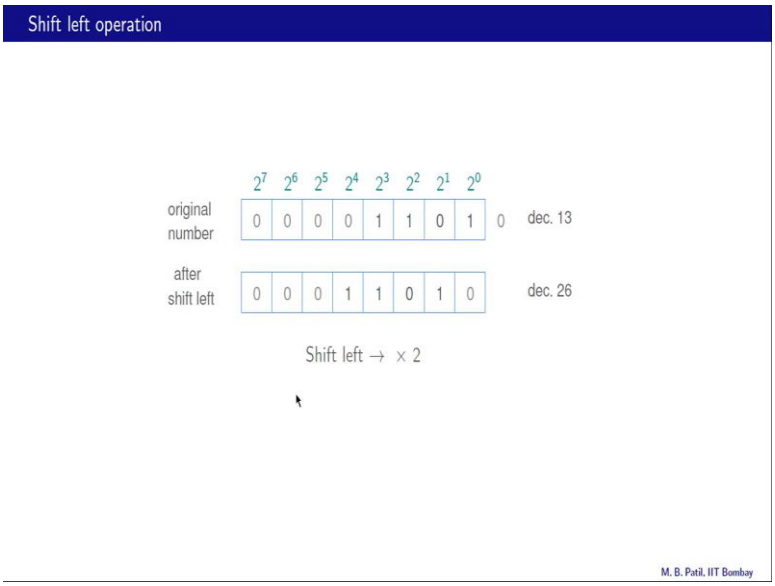
The shift register that we saw earlier was shifting data from left to right, but we can also make a bi directional shift register shown here which can shift data either from left to right or from right to left depending on the value of this input called M or mode. When the mode input is 1 then the data gets shifted to the right and if it is 0 then it gets shifted to the left in that direction. So, let us see how this works.

Let us take M equal to 1 first: in that case what happens is this M bar is 0 and the output of this and gate is 0 and essentially D is then the same as this input here. And similarly this D is the same as this input etcetera. And then you will notice that we have exactly the same connection that we saw in the earlier example, and that is the shift right operation. With each clock edge the data will shift from its flip flop to this 1 from this to this and so on.

What about M equal to 0? If M is 0 then the output of this and gate is 0 and then this D in that case is the same as Q 1, and similarly we will find that this D 1 is the same as Q 2; D 2 is the same as Q 3 and D 3 is the same as D l. So, that is the connection that we are talking about now. And with an active clock edge this D l gets loaded into this flip flop whatever was Q 3 will now come here whatever was Q 2 will now come here and so on.

So, in summary with M equal to 1 we have the shift right operation and with M equal to 0 we have the shift left operation.

(Refer Slide Time: 04:27)



Let us consider a binary number 1 1 0 1 here, and see what happens to it when we perform a shift left operation. This number is 1 plus 4 plus 8 that is decimal 13 this 1 has a weight of 2 raised to 0, this 1 has a weight of 2 raised to 2 and so on. And now let us see what happens when we shift it left. And here is the result of the shift left operation, this entire number has got shifted by 1 position shifted left, and this 0 has got padded over here.

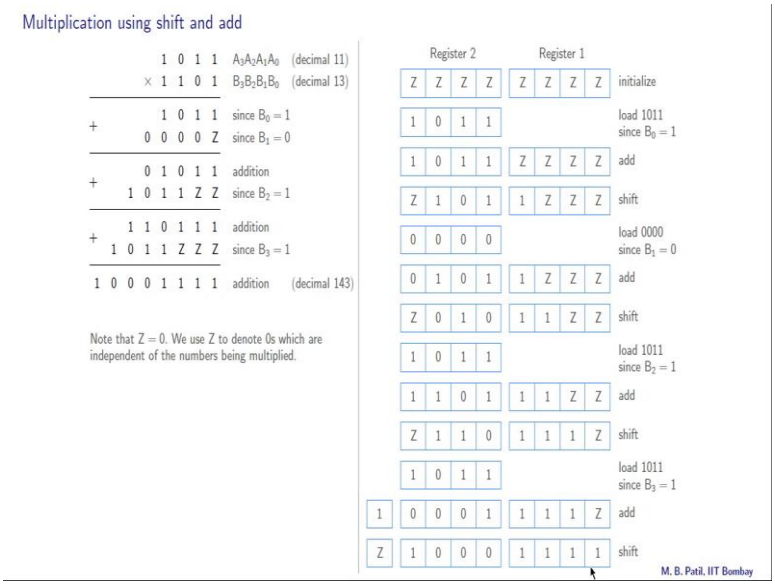
The new number now is 2 plus 8 plus 16 that is decimal 26 and that is of course twice the old number 13. And this is not surprising, because if we consider this bit for example it is old weight for 2 raised to 0 and its new weight now is 2 raised to 1; that is twice the pole weight. This 1 add a weight of 2 raised to 2 and now it has got a weight of 2 raised to 3 twice again and so on. So therefore, this entire binary number has got doubled by shifting left.

So, in conclusion the shift left operation is equivalent to multiplying the binary number by 2. And similarly shifting right is equivalent to dividing a binary number by 2. For example, if this was our original number and we shift it right to get this number then the new number is half the original number.

Now the question is how do we implement these operations? These shift registers are available commercially as ICS and we have already seen how to load a given number such as this 1 into a shift register. And we also seen, how a shift left or shit right

operation is performed on a shift register. So, we have discussed all the relevant mechanisms we just need to put it all together and achieve these operations

(Refer Slide Time: 06:51)



Let us now look at multiplication using the shift and add process. And here is an example; 1 0 1 1 that is decimal 11 that is our number a A 3 A 2 A 1 A 0. A 3 is this and A 0 is this one; so A 0 is the LSB, A 3 is the MSB. Here is our number B and that is decimal 13 and we want to multiply these two. So, this is our multiplicand and this is our multiplier. Where do we begin? We look at the LSB that is 1. So, we multiply the multiplicand by the LSB, and in this case it means we just reproduce this number as it is 1 0 1 1.

Next, we look at the next higher bit and that is 0 and when we multiply the multiplicand by A 0 we get 0 0 0 0 and that is what we write over here. And we have to shift it left because that gets a higher weight. So therefore, there is A 0 coming in from the right here. Now a comment about the notation this Z is actually the same as 0, but we use a different symbol Z to denote 0s which are independent of the numbers being multiplied. So, this Z is always going to be 0 no matter what these numbers are.

After that we add these 2 numbers and that is the result. Next we look at the next higher bit that is 1, so we reproduce the multiplicand. And now we shift it left 2 times and therefore we get these two 0s here. We add again then we look at the last bit the MSB that is 1 therefore, we reproduce this multiplicand again like that and shift it 3 times. So,

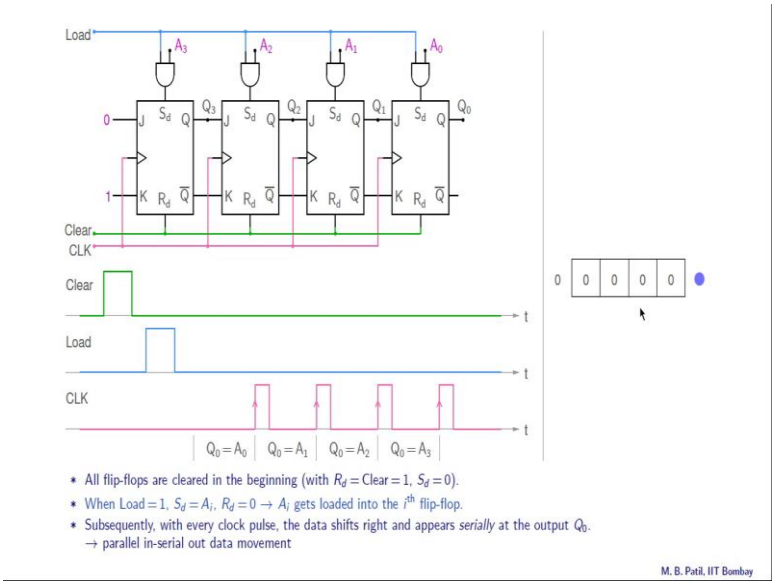
we get these three 0s here. And then finally, we add these 2 numbers to get our final product and that is decimal 143.

Let us see how this method can be implemented. We start with 2 registers 4 bit registers register 2 and register 1. This is going to be our LSB, that is going to be our MSB; and we initialize both these registers with all 0s. Now step number 1 is to load 1 0 1 1 the multiplicand this 1 in this other register. Since B 0 is 1 and now we are going to add these two 4 bit binary numbers to get this result here. And this binary number is simply going to be copied to this part. So, we have got 1 0 1 1, in this case of course these are all 0s so we have the same number over here that is the result of the addition.

Next we shift this result right by 1 position, so this number 1 0 1 1 has appeared here now 1 0 1 1 and 1 0 has got padded from the left. Now, the next bit, since B 1 is 0 we load 0 0 0 0 we are talking about this step now. And now we are going to add these 2 numbers and now you see that this corresponds to adding these 2 numbers here. And this 1 is going to get added to this 0 that is anyway happening because this 1 is simply going to get copied over there.

So, that is the result of this addition. And we keep repeating this process until we reach B 3. So, next step shift right 1 bit, so this 0 1 0 1 1 has come here 0 1 0 1 1, 1 0 has got added from the left. Now we load 1 0 1 1, since B 2 is 1 that corresponds to this step here add shift right. And finally, we load 1 0 1 1 because B 3 is equal to 1; that is this step here and then add and shift. So, that is our final result and that is indeed the same as this number here.

(Refer Slide Time: 11:57)



Let us look at this circuit and try to figure out what it is doing. We have 4 j k flip flops and they are all positive edge triggered. Here is the direct set input S d this is the direct reset input R d. And what we notice is that although we are using j k flip flops here we are really using them as D flip flops, because you see that j and k are inverse of each other in each of these cases; we have 0 here 1 here, you have Q 3 here Q 3 bar here, Q 2 here Q 2 bar here. Q 1 here and Q 1 bar here. So although these flip flops are j k flip flops the functionality is like those of D flip flop. So, essentially it is a shift register that we are talking about.

Now, let us look at the connections: the clock is common to all of them, and the clear input is also common that is going to the direct reset input of each flip flop. Now to S d we are connecting this variable called load anded with some other variable here that variable is A 3 here A 2 here A 1 here A 0 here. And A 3, A 2, A 1, A 0 represents a binary number. And what we are going to do is to load this binary number into these flip flops to begin with.

These are the signals being applied; a single pulse here for the clear signal, single pulse for the load signal. And this clock signal has been derived from a free running clock and some logic. So, the first pulse starts here and there are a total of 4 clock pulses. Now let us see how this works. At this time we have load equal to 0, so the output of this and gate is 0 and so are others; so therefore S d is 0. The clear input is high, so therefore R d is 1.

So, we have S d equal to 0 and R d equal to 1 and that will clear the flip flop; that means Q 3 will be made equal to 0 Q 2 Q 1 Q 0 will also be made equal to 0.

After the clear pulse the load pulse comes along and let us looks at the situation somewhere here. So, the load signal is 1 clear is 0, so all the R d inputs are 0 now. This is 1, so the output of this gate that is S d for this flip flop is equal to A 3. Now let us take 2 situations A 3 equal to 0 first; that means, we have S d equal to 0 and R d equal to 0. So, nothing is really going to happen the flip flop will work normally; its output if you recall is already 0. And since there is no clock edge yet the output will continue to be 0. So, that is the situation if A 3 was 0.

Now, consider A 3 to be 1 in that case S d will be 1 and R d is 0. So therefore, the flip flop output Q 3 will get set to 1. In other words if A 3 is 0 then Q 3 would be 0 at this time and if A 3 was 1 then Q 3 would be 1. So, this bit here is simply getting loaded into this flip flop and similarly A 2 will get loaded here etcetera. So, this entire binary number has now got loaded into this shift register.

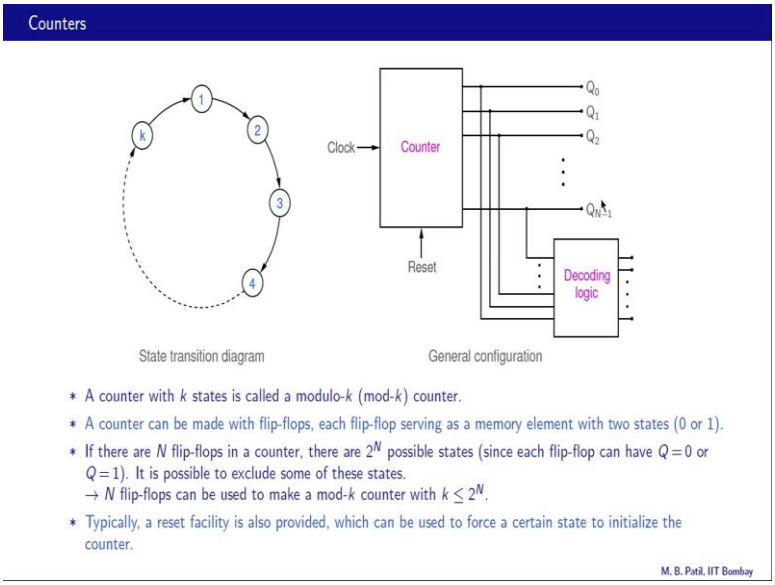
Now, the clock pulse is come along. What happens when the first active edge comes? This 0 will get transferred to Q 3, whatever Q 3 was will go to Q 2 and so on. So, that is a shift right operation by 1 position. And with each additional active edge these the data will simply shift right 1 more time. So, that is how this circuit works. To summarize all flip flops are cleared in the beginning with R d equal to clear equal to 1, this part and S d equal to 0.

When load is equal to 1 this part S d equal to a i; that means, A 3 or A 2 or A 1 or A 0 and R d is 0; that means, the binary number gets loaded into the shift register. And subsequently with every clock pulse the data shifts right and appears serially at the output Q 0 over there. So, first this A 0 will appear at Q 0 and after that this A 1 which has got loaded in this flip flop initially will appear at Q 0 and so on. And this corresponds to parallel in serial out data moment.

Let us now look at these schematic pictures showing this parallel in serial out data moment. What is parallel in this is our data A 3, A 2, A 1, A 0; and that is getting loaded parallely into these flip flops, like that. So, that happens in this phase, and now the clock pulse comes along the first clock pulse we are going to have a shift right operation because of that. A 0 appears in the first flip flop because we have j equal to 0 here, and

whatever was here gets shifted to the next flip flop and so on. With 1 more clock pulse we have 1 more shift right and so on.

(Refer Slide Time: 18:45)



Let us now start with counters. Counters are circuits with several states; and in this picture or the state transition diagram each state is represented by 1 circle here, this is state 1, this is state 2, this is state 3, 4 etcetera up to k. And after this state the counter goes back to state 1. And this kind of counter with k states is called a modulo k or mod k counter.

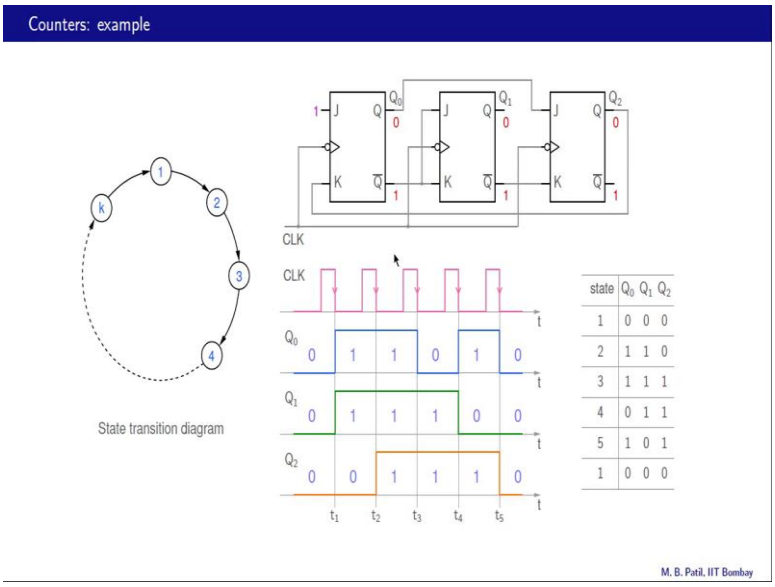
And what is inside a counter? Let us see: a counter can be made with flip flops each flip flop serving as 1 memory element with 2 states 0 or 1. The output of the flip flop Q can be 0 or 1. And if there are N flip flops in the counter we have 2 raised to N possible states, because each flip flop can have Q equal to 0 or Q equal to 1. And therefore, let us say we have 3 flip flops then we can have a maximum of 8 possible states here; so our k can be 8.

But we can also exclude some states, and therefore N flip flops can be used to make a mod k counter with k less than or equal to 2 raised to N. So, that is what these picture shows, inside this counter there are flip flops and this Q 0, Q 1, Q 2 etcetera are the outputs of those flip flops. And how does the counter go from 1 state to the next for example, from 1 to 2 or 2 to 3 that is done by the clock pulses. So, every time there is a clock pulse the counter will advance by 1 state from 1 to 2, 2 to 3, 3 to 4 and so on and

this goes on. In addition a research facility is also provided this input here which can be used to force a certain state to initialize the counter.

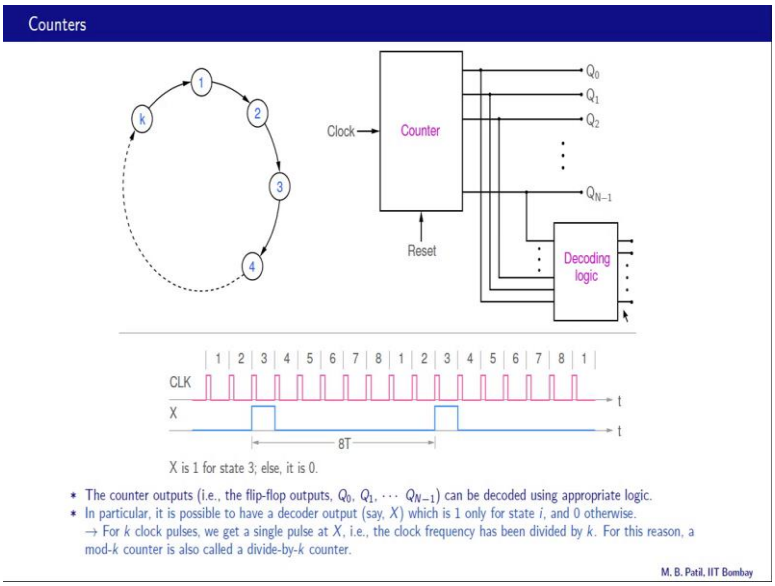
For example, let us say we have 3 flip flops inside this counter so the outputs are Q 0, Q 1, Q 2 and we might want the counter to start in the state 0 0 0. So, that can be achieved using the reset facility. Now in addition there is also this box for decoding logic shown over here, we will look at that a little later.

(Refer Slide Time: 21:29)



Here is an example of a counter: this is the general state transition diagram and this is our specific counter. And in fact this is a circuit we have already looked at earlier. So, the sequence for this counter is shown over here; if you start off with Q 0, Q 1, Q 2 equal to 0 0 0 then this is how it proceeds. And here is a table showing Q 0, Q 1, Q 2, so we have 0 0 0 to begin with that is our state 1 that one. Then we have 1 1 0: 1 1 0 that is our state 2, then 1 1 1: 1 1 1 state 3, then 0 1 1: 0 1 1 state 4, then 1 0 1 state 5 and then after that we come back to 0 0 0 back to state 1. That means, our last state this 1 is 1 0 1. So, k here is 5. So, it is a modulo 5 or mod 5 counter.

(Refer Slide Time: 22:47)



Let us now look at this decoding logic. This logic is nothing but some combinatorial circuit suitably designed so as to get some desired outputs here could be 1 or more outputs. Now since the counter is cycling through k states, these outputs will also be periodic that is after k clock cycles the output at any of these pins is going to repeat.

Now in particular let us take this example. Let us say we have a mod 8 counter; that means k is 8; that means, we have 8 states 1, 2, 3, 4, 5, 6, 7, 8 and then back to 1, this is our clock signal. Now it should be possible for us to design our decoding logic so as to get this kind of waveform at 1 of the output pins here; for example the first one.

Now this signal X is 1 when the counter is in state 3, otherwise it is 0. And since state 3 comes once in every 8 clock periods, if the clock period is t this interval is going to be 8 times t. Or in other words the frequency of this signal X is going to be the clock frequency divided by 8; so let us make that remark. It is possible to have a decoder output say X which is 1 only for state i in this case the third state and 0 otherwise. So, for k clock pulses we get a single pulse at X; that is the clock frequency has been divided by k. For this reason a mod k counter is also called a divided by k counter. So, this counter for example, would be called a divided by 8 counter.

Let us summarize: we looked at some operations related to shift registers and also an application namely; multiplication of two binary numbers. We have seen the

functionality of a counter. In the next class we will look at some examples of counters. So see you next time.