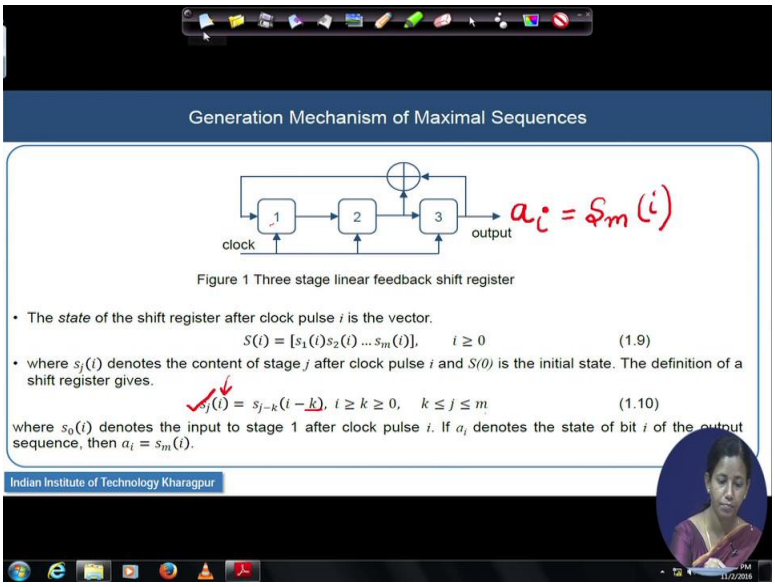
Spread Spectrum Communications and Jamming
Prof. Debarati Sen
G S Sanyal School of Telecommunications
Indian Institute of Technology, Kharagpur

Lecture – 10
Generation Mechanism of ML Sequence

Hello students, we have at in the last module, we were actually learning about the spreading sequences. And in this module, we will mainly consider inside the generation mechanism of the maximal sequences based on the linear feedback shift register architecture.

(Refer Slide Time: 00:45)



We saw this architecture in the last module, we discussed about it that we consider that we are having a three stage linear feedback shift register where there are three memory stages. And the output of the stage number three and stage number two are added, this is the modulo 2 operation which is basically an XOR operation is going on. So, it is modulo 2 added and we fed the output of this adder as an input to the stage number 1. The clock pulse was helping and between the arrival on each and every clock pulse, the stored data was moving from the one stage to the next. The output we took from the stage number 3.

We also saw the generation table spreading sequence table generation table in the last module, we are not going to repeat the table once again, rather we will try to give a

general form of any linear feedback shift register. And we will try to give a mathematical equation for generation of the shift register sequence at the output. Let us start slowly. Any state be of a linear feedback shift register after the clock pulse i has arrived we will be given by this. So, this is the total vector, vector is talking about the combination of what are the combination of all the stage values, the value of the each and every stage is written inside the brackets. I will say that it will not s 1 i if it is generalized then it should be s j i, where your j is the name of the stage, and i is the number of the clock pulse for which you are writing the value of the stage.

Of course, if I write capital s 0, he will tell what are the initial condition of each and every stage. So, with that concept remember the clock pulse can never have a negative value because we had starting at t equal to 0. So, I should have always value greater than or equal to 0. So, if this is the situation of at any moment after the arrival of the ith clock pulse, if I wish to see what are the values of the stage is storing then I will be able to see this.

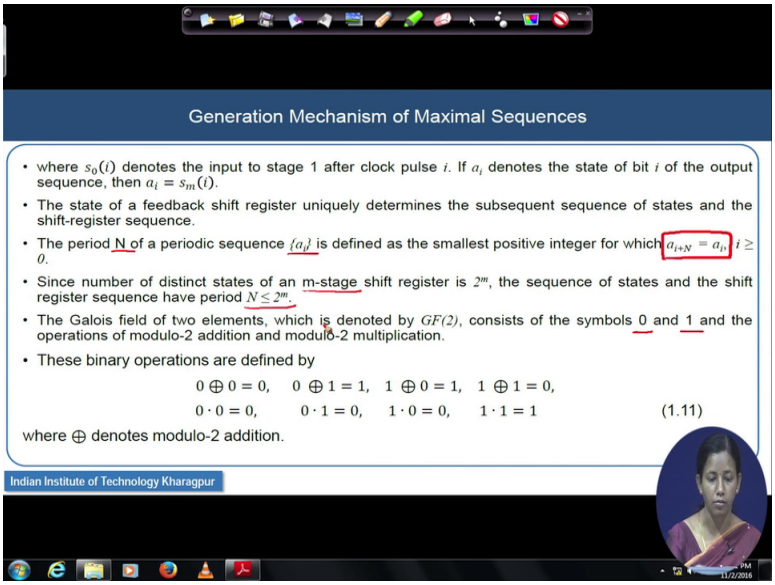
Now if I try to give mathematical formulation of the stage value of a independence stage, which is equal to s j i, what the way the way the value is stored in this typical stage is something like this. The value of s j at the i th clock pulse is nothing but the value which was stored in j minus k th stage at the moment of i minus k th time for i minus kth clock pulse. It means what; if I take an example if I wished to see what is the value stored in the stage number two. So, here it will be value of the stage number two say for after the arrival of the second clock pulse arrival of the second clock pulse, what will be it. Then I will get here it will be the value stored in the second minus k is actually the value kth timed if we I am thinking at is the instantaneous time earlier. So, whatever the value was here, and whatever the value was here during the clock pulse time of the first one.

So, what you are getting here is if you see at the second clock pulse, what am I getting here it is nothing but whatever was stored in the ith the earlier stage at the during the k minus during the k stage back during the kth clock pulse back. And situation is this equation holds good, if the k value is greater than 0; and definitely it should be less than i. So, when I am talking about the ith clock pulse I can gave the value stored between I minus k th clock pulses they can be never be greater than i, but k also it should be less than the j the number of the stages you are having. And definitely the stages where you

are asking that should be less than the maximum number of stages you are having in your design.

So, again s 0 i, it denotes that the input that this coming to the stage number one after the arrival of the clock pulse i. If I think that a i is the output that I am getting at the output of the whole structure at the bit time i, then the output a i is always equal to the value stored here in the last stage. So, it is always equal to s m i, I hope it is clear. So, we have already defined, what is that instantaneous value stored in each and every stage on the arrival of ith pulse. And we have also related this value with the preliminary stages and they are with the occurrence of the corresponding clock pulses going back in which is giving actually the trace of the previous clock pulses of the previous timing instance. Also we are getting what is the value stored in the ith, 0th position, and also what exactly is the output value how the output bits are related with the last stored value. We will relate all this points in the next derivation.

(Refer Slide Time: 07:10)

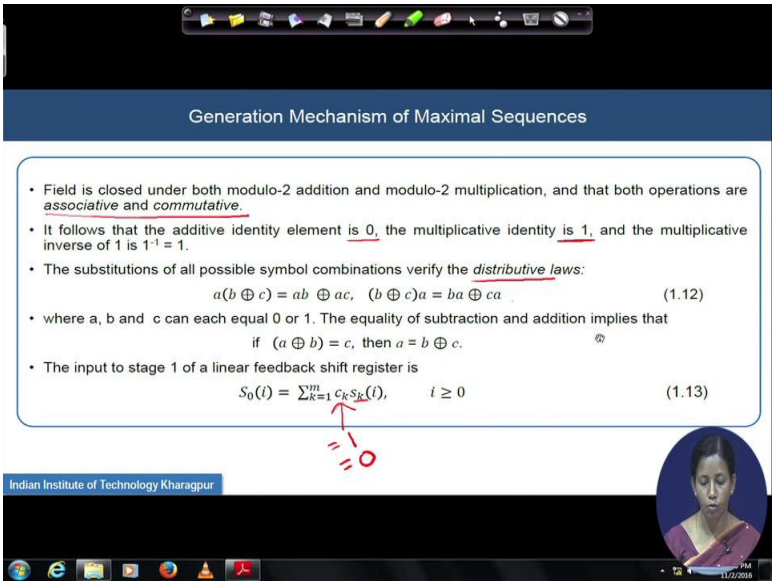


And see if we are having that the state of the feedback shift registers, they are uniquely determining the subsequent sequences of the states and the shift register sequences. If we consider that there are the period of the generated sequence a i, it is the period is equal to capital N then the smallest positive integer then the period can be defined as the smallest positive integer for which the value a i plus N is depicting a i value that we have seen in the last table. That if for the value of 7 for a three stage linear feedback shift registers,

after the seventh clock pulse, the value has been repeated in all the stages the value has been repeated. So, the period is defined as the smallest positive integer for which the a i plus N th value is repeating it is a i's value.

Now, if the m-stages shift registers, there are m number of the stages involved remember as we have already discussed, the period can never be greater than 2 to the power m, it should be always less than equal to 2 to the power m. If we are having a Galois field 2 elements that means, the symbols can have either 0 or 1, then we can have the operations like the modulo-2 addition or the modulo-2 multiplication. The binary operations would look like this. It is a XOR table we all understand that is the 0 XOR, 0 is equal to 0, 0 XOR 1 is 1, 1 XOR 0 will be 1, and 1 XOR 1 will be 0. And this the multiplicative operation that we understand that it will give equal to 1 when both inputs are equal to 1. Here this operation is a modulo-2 operation.

(Refer Slide Time: 09:14)



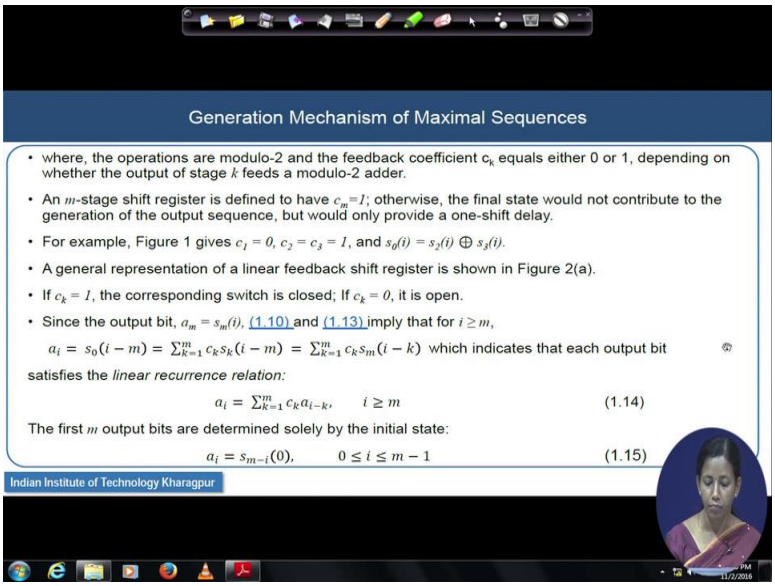
And the Galois field, so it is the field will be closed under this modulo-2 addition and modulo-2 multiplication and both the operations will be associative as well as the commutative. So, this property comes holds actually or can comes from the fundamental concept of the field and it is the closeness of that field. All that point we will be discussing when we will be go ahead with the Galois field mathematics. It also follows that this additive identity element for this Galois field-2, it will be the 0; multiplicative identity is equal to 1. If it is a Galois field-2, they are the identity element multiplicative

identity element and additive identity element. And by the virtue of the distributive law we understand that if we are having the elements a, b and c drawn from that field, if the field supports the distributive law then they are definitely a into b XOR c will be equal to a b XOR a c; and b XOR c into a it will be also given by b a XOR c a.

If there a, b, c can be either 0 or 1 and equality holds good between the subtraction as well as the addition, so you can also write that a XOR b is equal to c can be also holds also good for a is equal b XOR c. The input stage to the stage number one linear feedback shift register, we can write it that it is the combination of all the stages their independent c k into s k into i summation over all k to m is given you the s 0 i, why is it like so, c k is the feedback component. I mean if the stage is contributing to the feedback logic, then the value of the c k will be equal to 1; if the output of this s k is not contributing to the feedback logic then the value of that c k will be is equal to 0.

For example, if I go back to our earlier figure, here actually the stage number three and the stage number two are contributing to the feedback logic. So, c k value for c 3 and the c 2 they will be equal to 1, where the c 1 value will be 0. And every cases for each and every stage you are looking into the independent value of it getting multiplied with the corresponding contribution towards the feedback multiplying it and you are adding up such values from k is equal to 1 to k is equal to n, I mean one-by-one stage. The total combination is coming back as a feedback to the 0th stage, we are writing though writing it as s 0 basically it is the stage number 1 and the input to the stage number one, we are adding as S 0.

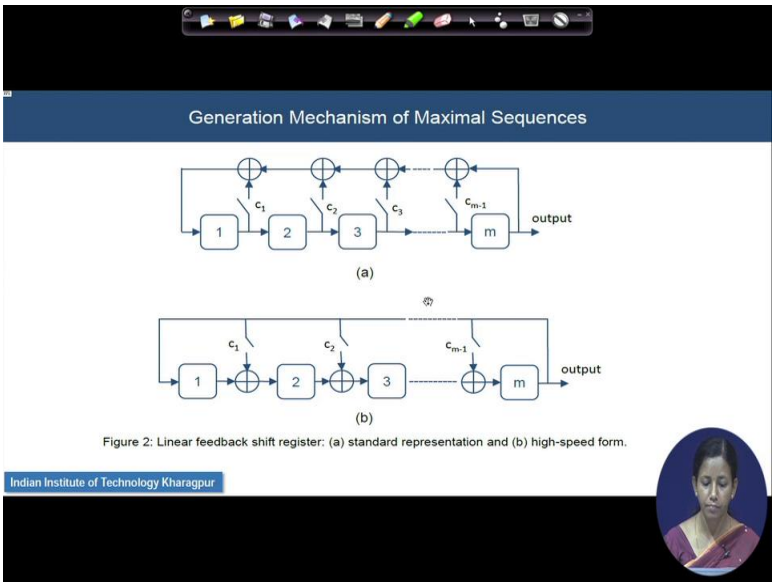
(Refer Slide Time: 12:44)



Once we have understood this fundamental concept, let us enter little bit in detail. As I have already mentioned the value of the c k is a contribution to the feedback it is a feedback coefficient we call, its value is either 0 or 1 depending upon whether that typical stage is contributing or not. Remember by default for the last stage which is the feedback coefficient always kept 1. If you do not keep it one then the final stage, you do not have any contribution to the generation of the shift register, it will have simpler delay one stage delay and that can be proved actually in the last figure that we were talking about in the slide number one. Where, as I showed that c 1 value is 0, c 2 and c 3 values are 1, and this hence your feed back to the input stage is the combination of the XOR operation of the stage number two and stage number three.

If your c 3 or last stage value is not supposed to be in 1, then you try to do the exercise at home, you will see that the final output is just shifted version. And there is no contribution of that final stage in the generation of the code happening. Now, the general representation of this linear feedback shift register with when we are giving general form of the mathematics, so the general form of the shift register we can also show.

(Refer Slide Time: 14:19)



In the next slide, we have shown the two different structure of the linear feedback shift registers, let us first concentrate on the upper one. So, now we have increased the stages up to the numbers small m. And here we are seeing that the c m value is always 1; and depending upon the logic scenario you wish to add some of the switches who are actually now controlling the contribution of the corresponding stage in the feedback architecture, they are either closed or they are opened. If they are closed, their value will be corresponding c values will be 1; and if they are opened that means, their corresponding value will be 0. And the corresponding stage will be contributing as XORed it will be XORed with the next previous one the final XORed output will be feedback as a input to the stage number 1.

Going by this expression and explanation also we can come back now. Now ready to develop slowly the output sequence, we understand the output value, output is equal to a m and which is basically the direct value stored in the s m i or the last stage of the value stored for the ith pulse in the last stage of feedback shift register. And this holds good always whenever the i is greater than or equal to m. When i is less than equal to m, you are basically getting the values which was stored in the initial situation. So, one-by-one the initial value will first come out, when i will go the number of the clock pulse will cross the number of stage is involved in the design definitely you can think that whatever last value stored is coming at the last stage last memory stage that will come out.

And if I try to see the structure then whatever the value coming in the equation coming at the output, so it will be also getting stored in the input to the first stage also. How is it coming, the concept of this comes from the fact that this upper figure is not implementable easily, and not hardware friendly. In order make the figure hardware friendly and to increase the speed of this implementation and processing of the data through the feedback shift register, we come down to this high speed firm. Where all the feedback path it is typically one value, and all the modulo-2 operators they are shifted towards the inside between the path of the processing between one stage to the next.

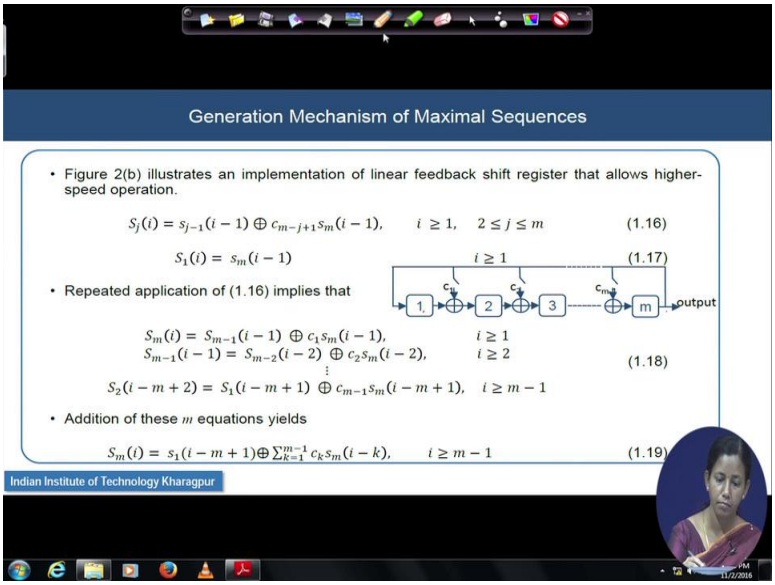
If this is the situation, then the value output here whatever we are seeing as a i and then a i will be basically whatever I am trying to see that a i will be directly connected also with this input. But never mind there we will come later, but currently what I am trying to see here is the value of the a i what we are getting at the output, when I is actually less than m for i less than or equal to m. What I am getting at the output is basically the initial value stored one-by-one in the stages. So, at the first clock pulse, the stored value in the last stage, initial value of the last stage will come out. Second clock pulse last, but one stages initial value will come out and like that it will go on till you are reaching the value of i is coming equal to m, then the first initial stage value initial stage value of the first memory or the firsts stage will come out.

If I utilize this concept then for i less than or equal to m, a i will be always s 0 i minus m. And if this is the situation, if the i is like this, and hence I can substitute the value of the s 0 because the value of the s 0 we have already seen in the last equation in the last slide. The value of my s 0 is given by this expression 1.13. So, if it is delayed now by i minus m, so it will be the same equation only changed by s k i minus m that we are substituting here in this place. And once this is there when we understand that the last stage value is basically the value of the coming output. So, s m i minus k is finally can be replaced by a i minus k. And now actually you are getting linear recurrence relation before i greater than equal to m, you are getting a linear relation always you are getting a linear relation that a i of this c k a i minus k. And whatever you are getting for i less than n minus 1 is for a i t c will be the value of the m i minus i of that stage. I hope it is clear.

So, I repeat then when the i is greater than or equal to m, you are getting that a m value stored whatever was stored in the m th stage, and then I understand that m th stage value whatever I got it was actually the value of the which was fed as the input to the stage

number one, m clock back from the current clock i. So, I am writing a i is equal to my s 0 i minus m. Then I substitute a value of s 0 from the previously developed equation, which is c k into s k i minus m. And definitely from the last stage it is equivalent to the output. So, we had coming down here. And this is the situation when i will be greater than equal to my number of the stage clock pulses coming exceeding the number stage. And here if we are not exceeding the stage in between the stage is we will get the m minus ith stored values for the 0, whatever it was store value for the 0 th pulse, I mean this is the initial situation or initial condition initial value stored one by one in all the stages. We will refer next onwards the high-speed form of this generation, and we will proceed for our derivation.

(Refer Slide Time: 21:27)



So, I have replicated that figure once again here for easy understanding of the derivation that is going on here. Remember that we understood that if we are changing the structure like this, bringing all the modulo-2 operator in between the stages and making the feedback path is equal to directly connected to the input. Then at any stage s j i, I am getting the value of s j i by the s j i minus 1 i minus 1th stage value getting convolved with the corresponding switch value. This switch value is such that it is c m minus j minus 1, you put a value. Suppose, if your a j is equal to 2, so 2 minus 1, you are getting the value stored in the stage number one for 2 minus 1 for ith for the first clock pulse. So, whatever the value was stored in stage number one during clock pulse number one

getting convolved with your j is equal to j is equal to 2 in our case, so c m minus 2 plus 1, so it is c m minus 1th value and here you are getting the value of c 1 of course.

And if you are having certain values here you are coming from this side to this side. So, basically you are clubbing or you are basically in taking the value of the last switch, each corresponding multiplication of this last s m value which was multiplied with this value of the switch. So, s m i minus 1, so everywhere you are computing the value of the second clock pulse, you are getting the input from the stage number stored in the first clock pulse also the value stored in the mth stage during the first clock pulse multiplied with the corresponding weight vector which is contributed by the switch. And then modulo 2 operation of this two, this equation holds good when your i is greater than equal to 1 and also for the j stage number stage j from the second stage onwards because for first stage this equation does not hold good first there is a direct coming an input from the last stage itself.

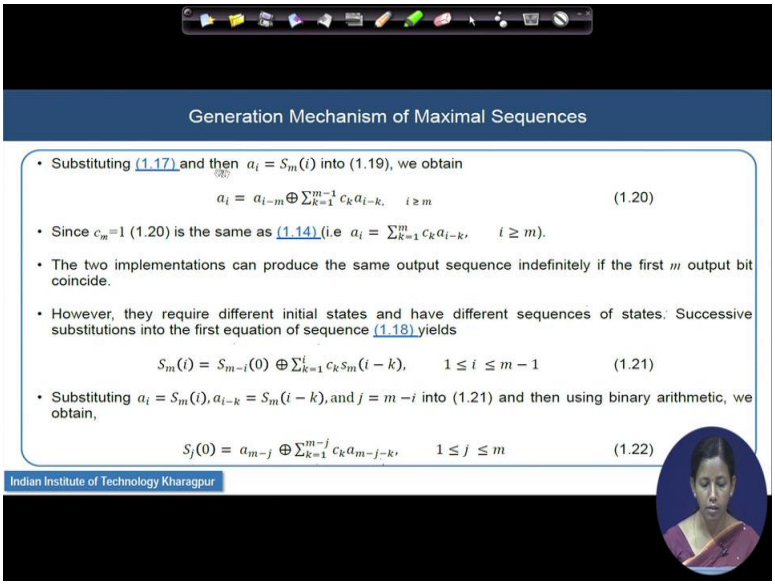
So, if I start like this. So, for s 1, for stage number one and the ith value this will be directly from s m i minus 1, correct. If I keep on repeating this first expression, so suppose I am trying to follow it for s m for s m i, I will get s m i minus 1 m minus 1 th stage for i minus 1 th clock pulse getting convolved with the first value of the c 1 and s m i minus 1. So, s m i minus 1 and the multiplication of this c 1 all that will be multiplied and coming beside.

Similarly, if you proceed for s m minus 1, it will be giving the same way the connection between m minus 2 and every time with the s m minus 2, you are getting the input from s m for the previous clock pulse and getting multiplied with the c 2. So, like that it is proceeding. And at last whenever you are ending up, whenever you are ending up you are coming with the expression like this. And if I keep on substituting the values here, suppose in the equation of s m, I am substituting from s m minus 1; and the inside that s m minus 1, I have the values s m minus 2.

And from s m minus 2, I have having values from s m minus v, if we go on like that you will be ending up with s m minus I getting substituted the value of s 1 i minus m plus 1 finally. And all these values will be added up. So, there is a XOR operation with the summing up term of the c k and s m i minus k. With c k and s m i minus k and then this guy it is moving from k equal to 1 to m minus 1. So, you are having the contributions

from c 1 to c n minus 1, k is varying from 1 to m minus 1. So, c k and s m i minus k, so that is way the whole part is getting formed.

(Refer Slide Time: 26:46)



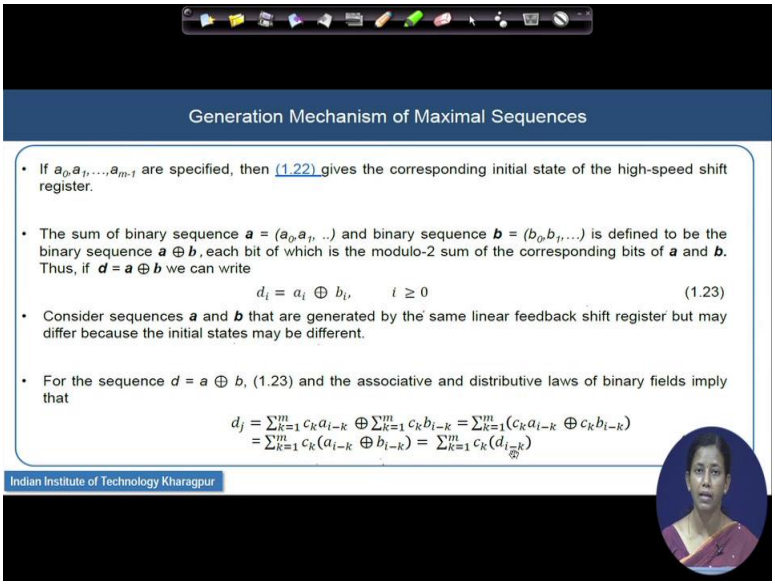
So, for each and every stage now I know how actually the data is getting accumulated. And it is processed also slowly one by one. We understand that a i will be the final stage output will be from s m i only. And if I now substitute from 1.17 like this from here and then you utilize, so a i will be always coming out from last stage, and if I substituted this value in the last developed equation here. So, where I will be ending up with this, this a i where what I have done this is s m i, I have substituted with a i because that is the understanding because the a i here is directly coming out from s m i. And this expression also can be substituted by some term of a i; and this is the another portion remaining in terms of a i minus k.

So, finally, I can relate the last state equation with the output sequence. And since I understand the c m is always should be equal to 1, so this equation basically is coming same as we have derived earlier. So, this is the proof that the high-speed form can also generate the same sequence like the previous non high-speed form, I mean this circuit and this circuit will give you the same sequence only difference is that this can generate the sequence at a very high-speed.

So, the two implementations now they are equivalent that is the demand is and however, this differential initial states and the difference sequence states, they are really required

for your successive generation. And if I keep on doing in the successive substitutions into the first equation one-by-one, so what we will end up with is when my clock pulse is varying between one to less to the number of the stages, I mean one to m, m minus 1, when this is varying then the expression will lead like this because you are basically getting XOR with the initial state values of the previous stage. And substituting a i here for s m, s m a i is equal to s m i and substituting a i minus k is equal to s m i minus k all that, so you will be ending with s j 0. And during some binary arithmetic also for j is equal to m minus i, we will be ending up with j 0, I mean the initial stage and this is equation ending up. So, s m i minus k will be governed by s m i s m minus j minus k with after all this substitutions, and here also I am coming m minus i. So, i to j transformation has happened nothing else.

(Refer Slide Time: 29:53)



And now in the last portion, we are trying to see whether we highlight little bit some properties of the generated sequence through this high-speed linear feedback shift registers. Suppose, you have generated two sequences from shift same shift register one sequence is a combination of a 0, a 1 dot dot dot; and the other binary sequence is there b, who is the combination of b 0 to certain value. And we define that there is a binary sequence which is now generated outside by doing the bit-by-bit modulo-2 operation of the sequence and sequence b. And we write that the newly generated sequence d is given by a XOR b. And the every component of the d, the ith bit is generated by the bit-by-bit modulo-2 operation as I told for i greater than equal to 0.

Now, consider that sequence a and b they are generated by the same linear feedback shift registers, but may be the initial condition for generation of a and the generation of b they are not same. If that is the situation for the sequence this d, and using this associative and distributive law of this binary field, we can show that the value each and every any bit value d j can be replied can be actually obtained by taking this c k i minus k XORed with c k d i minus k. And as this is the associative and distribution law we are applying, so the sum will come out and the independents, if I takes c k i minus k will be XORed with c k minus k i, I can take c k common. And then it is basically the XOR operation going on between a i minus k and b I minus k. So, a i minus k XORed with b i minus k is basically d i minus k.

What is the final observation is that we understand that any j th instant data output which is of this newly generated sequence is also considering and also maintaining a recurring relation of its previous sequences. Previous sequences means whatever you are getting at i minus k th clock pulse. So, fundamental j stages, if you are generating a linear feedback shift register, the generated sequence keeps a linear recurrence and occurrence of the generated sequence.

So, generated sequence all the bits will have linear recurrence relation. Even if you are generating the two different set of the sequences, who are having two different initial conditions; both of them individually will keep a linear recurrence, and a new sequence which is generated by the XOR operation of these two sequence a and b, he will also have a linear recurrence relationship with generated bits. All the generated bits will have a linear recurrence relationship with the previous bits.

Next module, we will see in detail little bit more about the properties of this maximum length sequences.