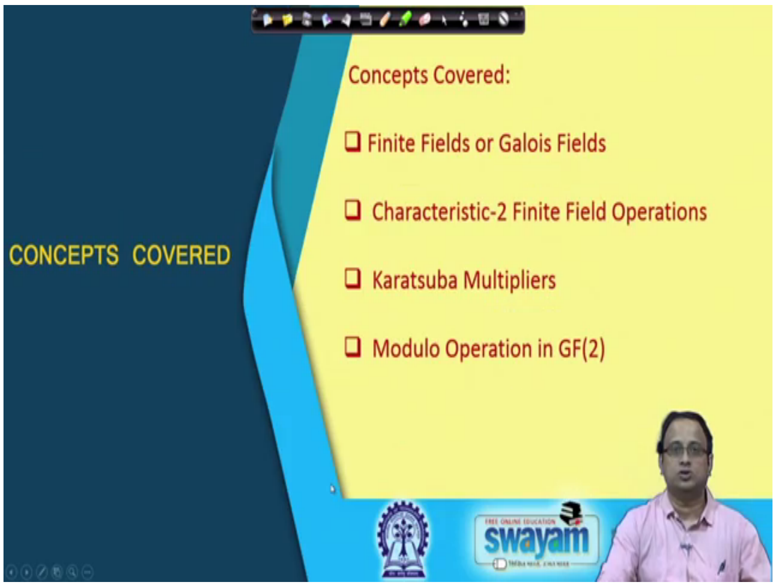
Hardware Security

Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 04
Finite Field Architectures – 1

So, welcome to this fourth lecture on Hardware Security. So, today we shall be discussing about certain topics on Finite Fields and also trying to understand about how you can develop hardware architectures for some important finite field circuits.

(Refer Slide Time: 00:29)



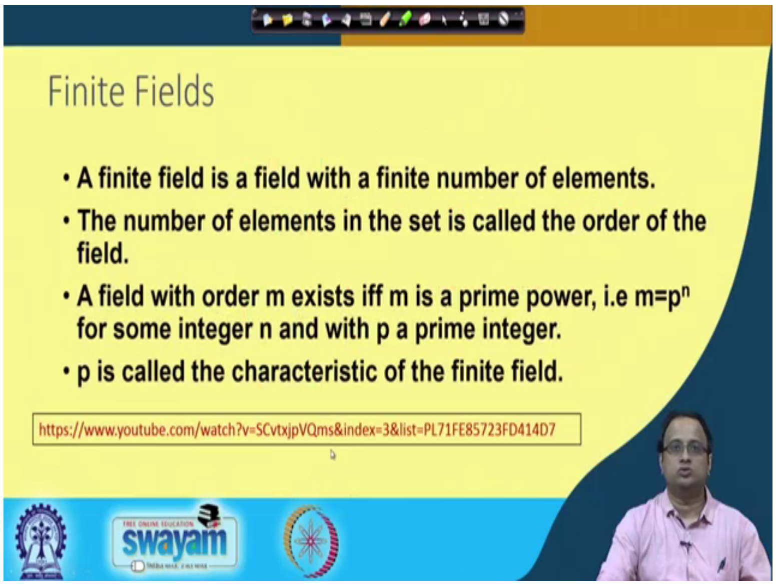
CONCEPTS COVERED

Concepts Covered:

- Finite Fields or Galois Fields
- Characteristic-2 Finite Field Operations
- Karatsuba Multipliers
- Modulo Operation in GF(2)

So, today we will be covering these concepts. So, we will be starting with a brief introduction into finite fields or what are called as Galois fields and then in particular will be talking about characteristic two finite field pre-operations, which are well suited for arithmetic and hardware circuits. In particular will be focusing on an important design on multiple of multipliers which are called as Karatsuba multipliers and discuss also about how to perform a modulo operation in GF 2 and which is a very common operations when we are dealing with such kind of finite field primitives.

(Refer Slide Time: 01:07)



Finite Fields

- A finite field is a field with a finite number of elements.
- The number of elements in the set is called the order of the field.
- A field with order m exists iff m is a prime power, i.e $m=p^n$ for some integer n and with p a prime integer.
- p is called the characteristic of the finite field.

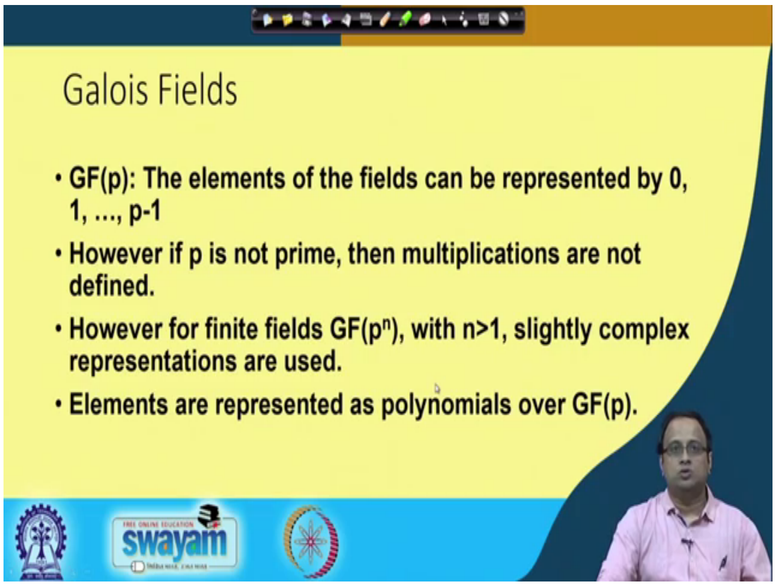
https://www.youtube.com/watch?v=SCvtxjpVQms&index=3&list=PL71FE85723FD414D7

So, to start with brief introduction into what are called as finite fields. So, a finite field is essentially a field which has got finite set of numbers and the number of elements in the set are is often called as the order of the field. And there is a result which says that a field with ordered m exists if and only if m is a prime power; that means, m is of the form of p power of n for some integer n with p which is a prime integer.

This p is often also called as the characteristic of the finite field. Another way of defining a characteristic of the finite field is if we take a nonzero element in the finite field and the characteristic of the field is the minimum number of times when we add the result to itself; that means, we do a plus; a plus a so until k times and k is the minimum such integer value for which the result becomes equal to 0. So, we have got different kinds of different characteristic fields. For example, we have got prime fields and we have got binary fields in which case p about the value of p is 2.

So, if you are interested for more introduction or more discussions on finite fields. So, you can click into this YouTube clip where you can get slightly more details about what fields in general or finite fields imply.

(Refer Slide Time: 02:27)



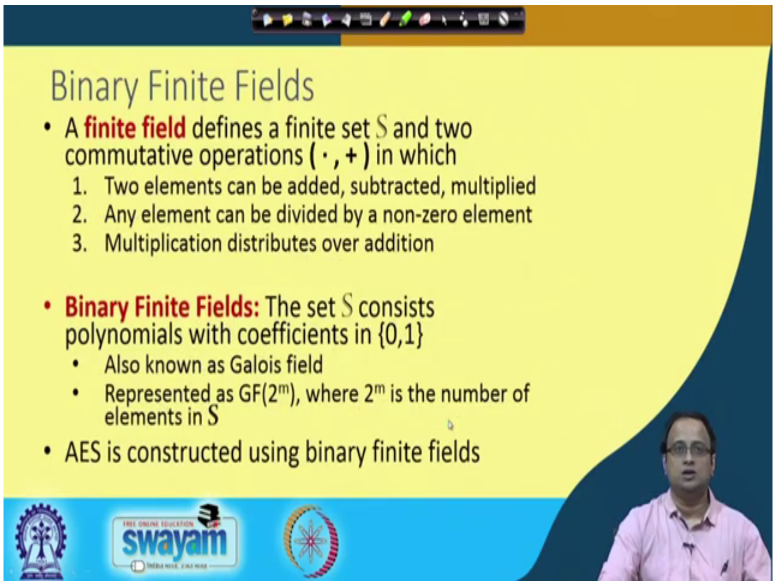
Galois Fields

- GF(p): The elements of the fields can be represented by 0, 1, …, p-1
- However if p is not prime, then multiplications are not defined.
- However for finite fields $GF(p^n)$, with n>1, slightly complex representations are used.
- Elements are represented as polynomials over GF(p).

So, so, in particular as I said Galois fields are the elements of fields and elements of the are where the elements of the finite fields can be represented by the numbers or elements from 0 to p minus 1; that means, 0, 1, till so until p minus 1. However, if p is not prime then the multiplications are not defined. So, therefore, it is important that p in general has to be prime there is a concept of extension field where you take GF p and you extend it to what are called as a extension fields and often denoted as GF p power of n with n which is greater than 1.

And, it is the representation is slightly more complex and the elements in this particular field are often represented as polynomials over GF p. So, I will be detailing this in my doc. So, Galois fields form a very important class of fields which are used in several cryptographic operations. So, it is important to have a have an understanding about how Galois fields look like and how they can be represented and in particular how we can do operations on such kind of field elements.

(Refer Slide Time: 03:33)



Binary Finite Fields

- A **finite field** defines a finite set S and two commutative operations (· , +) in which
  1. Two elements can be added, subtracted, multiplied
  2. Any element can be divided by a non-zero element
  3. Multiplication distributes over addition
- **Binary Finite Fields:** The set S consists polynomials with coefficients in {0,1}
  - Also known as Galois field
  - Represented as GF(2^m), where 2^m is the number of elements in S
- AES is constructed using binary finite fields

So, as I said that the characteristic can be prime and also it can be 2 in which case the field is often called as a binary finite field. So, a binary finite field or a field or a finite field with characteristic 2 is where the set S consists of polynomials with coefficients in 0 and 1. So, here the field is often represented as GF 2 power of m as I said that the general extension form field is GF p power of m.

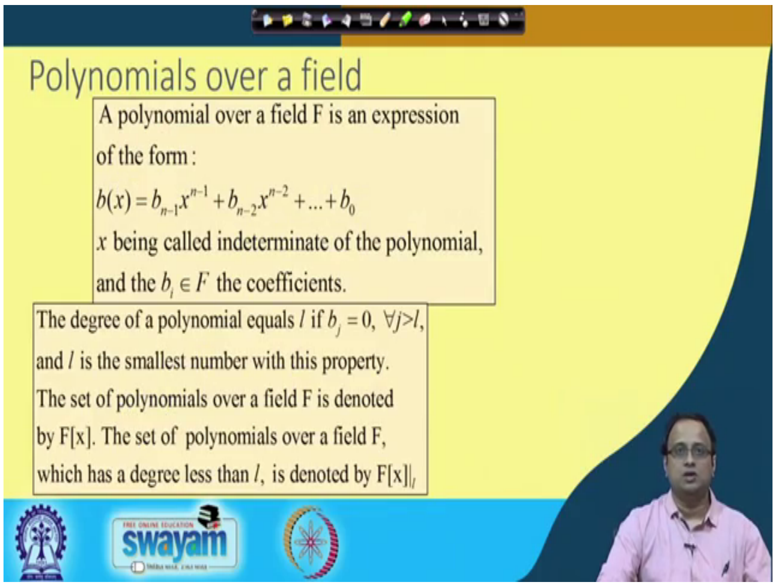
In this case p is equal to 2. So, therefore, this is called or this represent representation becomes GF 2 power of m. So, 2 power of m is the order of the field; that means, it is the number of elements which are in the field. So, this forms a very important class of a composite important class of finite fields because of the simple reason that the elements or the underlying elements can be represented by values 0 and 1. And this essentially makes it very conducive for arithmetic circuits because as we know that in arithmetic we are often convenient we have convenient structures when we have got binary representations.

So, in particular when you are talking about GF 2 power of m you as I will mention that you can actually represent them by a register of dimension m where each element in the register is either a 0 or 1. So, in this case the representation or the hardware essentially which is required to represent this element is very optimal I mean there is no wastage. On the other hand and you can actually compare it for example, with GF 3 power of m which is another finite field. If you take GF 3 power of m then the underlying elements will be in GF 3; so, that means, they can be either 0, 1 or 2.

Now, if I want to represent elements 0, 1 and 2 then I need two bits. So, in two bits I can actually or I should be able to represent four elements out of which I am using only three values like 0, 1 and 2. So, therefore, the representation is not very optimal. On the other end right when you talk about GF 2 power of m the representation is optimal and therefore, the hardware right which you get is often much more compact compared to the other fields.

In particular as we will be seeing in the next classes is that the famous advanced encryption standard is constructed using binary finite fields. So, it is important to have an understanding about how binary finite field a arithmetic looks like.

(Refer Slide Time: 05:57)



Polynomials over a field

A polynomial over a field F is an expression of the form:

$b(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + ... + b_0$

x being called indeterminate of the polynomial, and the $b_i \in F$ the coefficients.

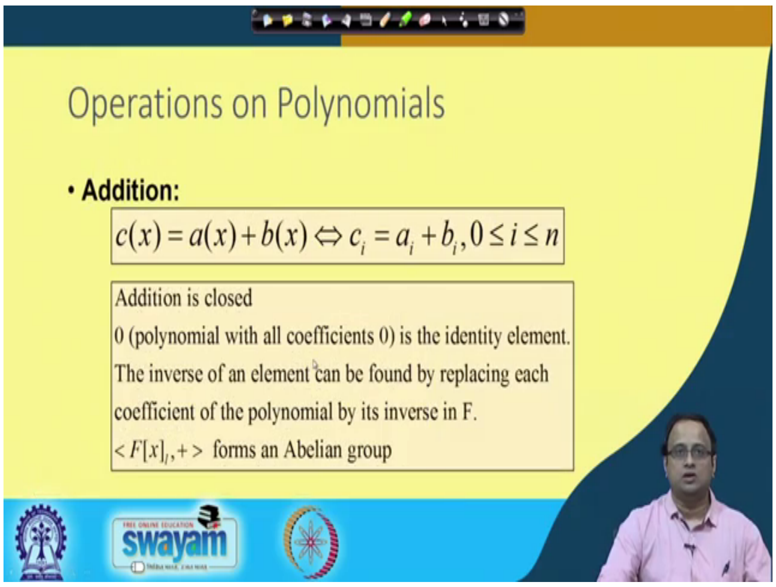
The degree of a polynomial equals l if $b_j = 0, \forall j > l$, and l is the smallest number with this property. The set of polynomials over a field F is denoted by F[x]. The set of polynomials over a field F, which has a degree less than l, is denoted by $F[x]|_l$

So, as I say that how do we it is important to understand how do you represent elements in a field or in a finite field. So, the customary way of doing that is by using this concept of polynomials. So, what we do is that, we take polynomials over a field F and this is often represented as an expression of the form of b n minus 1 x to the power of n minus 1 plus b n minus 2 x to the power of n minus 2 plus so on till b 0. So, in this case x is called the indeterminate of the polynomial and the elements b n minus 1 till b 0 these are belong to the belong to the field.

So, in that case we say that this polynomial b x is defined over a field f now what is the degree of this polynomial now degree of this polynomial right is equal to l if the value of b j becomes 0 for all j which are greater than l; that means, that l is the smallest number with this property. So, the set of polynomials or what a field is denoted often and by this symbol F x and the set of polynomials over a field which is a degree which is less than l is denoted often as F x with the suffix of l so.

So, therefore, right when we talk about binary finite fields then he would imply that the polynomial that I get here b x will have it is coefficients which are either 0 or 1; that means, the coefficients belong to the field GF 2 which has got only 2 elements either 0 or 1.

(Refer Slide Time: 07:21)



Operations on Polynomials

- **Addition:**

$$\boxed{c(x) = a(x) + b(x) \Leftrightarrow c_i = a_i + b_i, 0 \le i \le n}$$

Addition is closed
0 (polynomial with all coefficients 0) is the identity element.
The inverse of an element can be found by replacing each coefficient of the polynomial by its inverse in F.
$< F[x]_l, + >$ forms an Abelian group

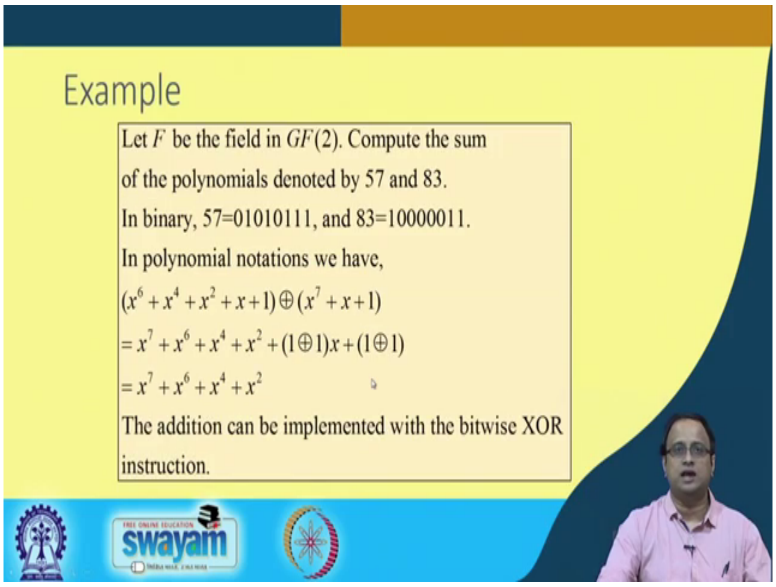
So, now we would like to define operations on these polynomials or operations on these finite field elements. So, the more the most fundamental operation as we know right in fields is addition. So, we would like to take two a polynomial two elements a x and b x which are represented as polynomials, which are field elements and we would like to define an addition on them.

So, the usual way of doing that is we do why to do it coefficient by coefficient. So, so we have got two polynomials like a x and b x which has got which has got coefficients starting from a 0 to n minus 1. So, what we do is that we basically start adding them coefficient by coefficient so. So, therefore, write the addition as is very easy to understand is that it is close; that means, if I take two polynomials and if I add them then it also belongs to an element in the field. So, in that sense the addition is closed.

And, 0 would stand with 2 I mean will essentially the 0 in this field right is a polynomial where all the coefficients are 0. So, therefore, if I for example, consider a binary find a binary finite field if I you know like just take a x and if I add a x with a x then I would get that all the elements will have coefficient 0. So, therefore, that is essentially the identity element in the field.

So, the inverse of an element can be found by replacing each coefficient of the polynomial by it is inverse in F; that means, like for example, if I take a polynomial a x then inverse of ax will be b x where each of these coefficients are replaced by its inverse. For example, if I take the GF 2 j to power of n field; for example, then if and consider a polynomial a x where the coefficients are either 0 or 1. So, as we know I mean in Galois 2 right or GF 2 each elements is self inverse. So, therefore, write I mean if I take a polynomial a x and if I add it with a x then I would get the 0 polynomial; that means, all the coefficients would be 0 in that case.

(Refer Slide Time: 09:27)



Example

Let F be the field in $GF(2)$. Compute the sum of the polynomials denoted by 57 and 83.

In binary, 57=01010111, and 83=10000011.

In polynomial notations we have,

$(x^6+x^4+x^2+x+1)\oplus(x^7+x+1)$

$=x^7+x^6+x^4+x^2+(1\oplus 1)x+(1\oplus 1)$

$=x^7+x^6+x^4+x^2$

The addition can be implemented with the bitwise XOR instruction.

So, so likewise right when we as we did addition. So, here is an example of how we can do you add. For example, the field is in this case GF 2 and if I take two polynomials which are represented by these numbers 5 7, 8 3. So, in by binary right this 5 7 is the hexadecimal representation. So, we take 5 and represent it as 0 1 0 and 7 as 0 triple 1. And, likewise we take 8 3 3. So, 8 will stand as 1 triple 0 and 3 will be 0 0 1 1.

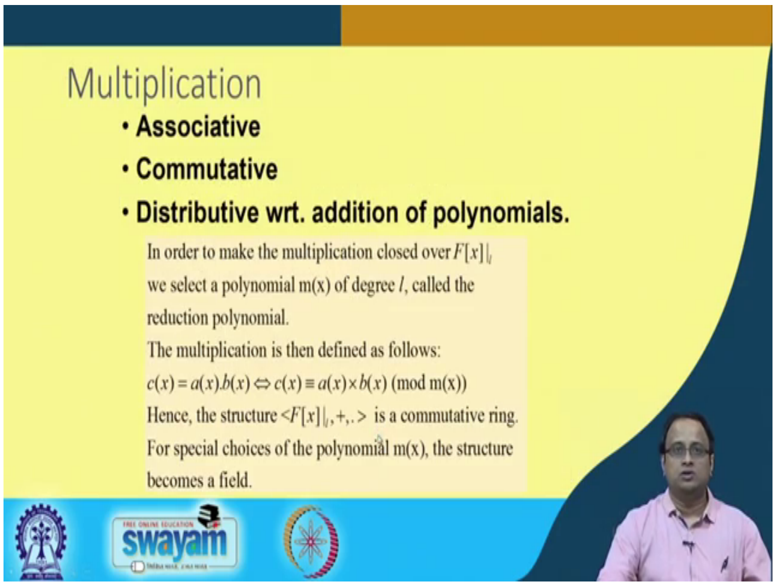
So, therefore, now in the polynomial rotation therefore, I will be starting with 1 here. So, this is 1, so, this is x. So, the coefficient of x is 1 likewise the coefficient of x square is 1, but x 3 right there coefficient is 0. So, I do not have that in the polynomial, but I have got x 4. So, I have got x 4 and then I have got x power of 6. So, this essentially stands for the polynomial or the number 5 7. Likewise we have got 8 3 and 8 3 has got 1 plus x plus x power of 7. So, therefore, the polynomial is x power of 7 plus x plus 1.

Now, if I want to add these two elements then what I will do is that I will add them coefficient by coefficient. So, therefore, in this case I see that 1 and will be added with 1. So, one will be XOR with 1. So, remember that the or note that the addition in GF 2 is nothing, but a Boolean XOR. So, therefore, for example, if I want to add a finite field element 1 with 1, then I basically do an XOR or exclusive or between 1 and 1. So, likewise right if I take x, then the coefficients are 1 and 1 on both sides. So, I do an XOR with 1 and 1, ok.

So, if I take for example, the other coefficients I essentially do not need to do any operation because there are no overlaps. So, now, we can do an XOR with in 1 and 1, so, I get 0. So, therefore, this term vanishes likewise 1 and 1 0. So, therefore, the coefficient x also is not there in the final polynomial and therefore, the final polynomial is nothing, but x squared plus x power of 4 plus x power of 6 plus x power of 7.

So, one can note that the addition can be implemented with the bitwise XOR instruction and there are no complicated operations which you need to do like for example, carry which you often encountered in other cases. So, that is this completely a carry less addition.

(Refer Slide Time: 11:41)



Multiplication

- **Associative**
- **Commutative**
- **Distributive wrt. addition of polynomials.**

In order to make the multiplication closed over $F[x]|_l$ we select a polynomial m(x) of degree l, called the reduction polynomial.

The multiplication is then defined as follows:

$c(x) = a(x).b(x) \Leftrightarrow c(x) \equiv a(x) \times b(x) \pmod{m(x)}$

Hence, the structure $<F[x]|_l, +, .>$ is a commutative ring.

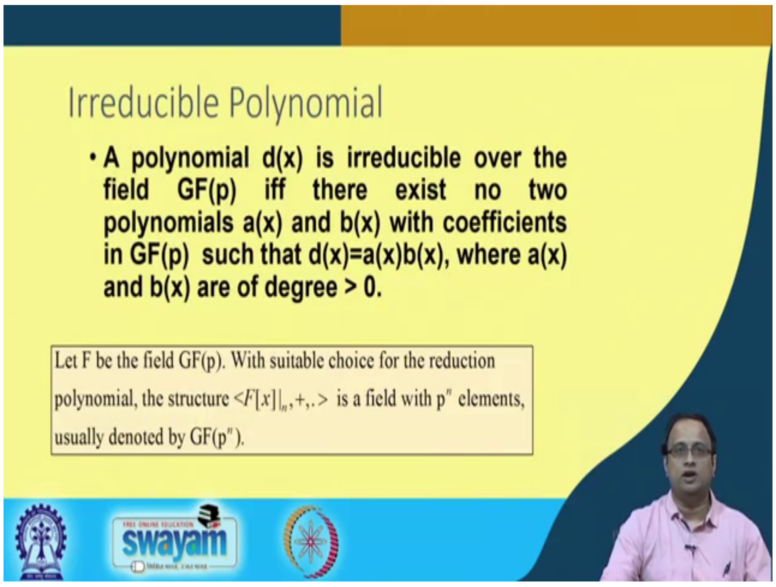
For special choices of the polynomial m(x), the structure becomes a field.

So, ah, but the; however, right multiplication is slightly more complex. So, of course, multiplication has got the properties of associativity and commutative and also distributivity with respect to addition of polynomials. So, this derives from it is fundamental field properties. So, therefore, if I take two polynomials a x and b x and want to multiply them then one can observe that if I take for example, a field GF 2 power of n that field is for a finite field. So, when I multiply a x with b x there is a chance that the coefficients will become more than what the field allows.

So, therefore, what we need to do is we need to do a modular operation so that the final result is brought back to the field. So, therefore, right we often have got once we have defined these two operations like both the addition as well as a multiplication then we have got this as what is called as a commutative ring and in particular right where each element or each non-zero element in this field has got it is own multiplicative inverse right; that means, if I have or if I can define an element which if I multiply with that element gives me the identity element in with respect to multiplication then I call that as a field.

So, therefore, this structure becomes a field now a ring will become a field for certain choices of this polynomial which I do have with which I do this modulo operation. And, with that with that restriction or with that restriction of the polynomial the structure qualifies as a finite field or becomes a finite field.

(Refer Slide Time: 13:13)



Irreducible Polynomial

- A polynomial d(x) is irreducible over the field GF(p) iff there exist no two polynomials a(x) and b(x) with coefficients in GF(p) such that d(x)=a(x)b(x), where a(x) and b(x) are of degree > 0.

Let F be the field GF(p). With suitable choice for the reduction polynomial, the structure $<F[x]|_n,+,.>$ is a field with p^n elements, usually denoted by $GF(p^n)$.

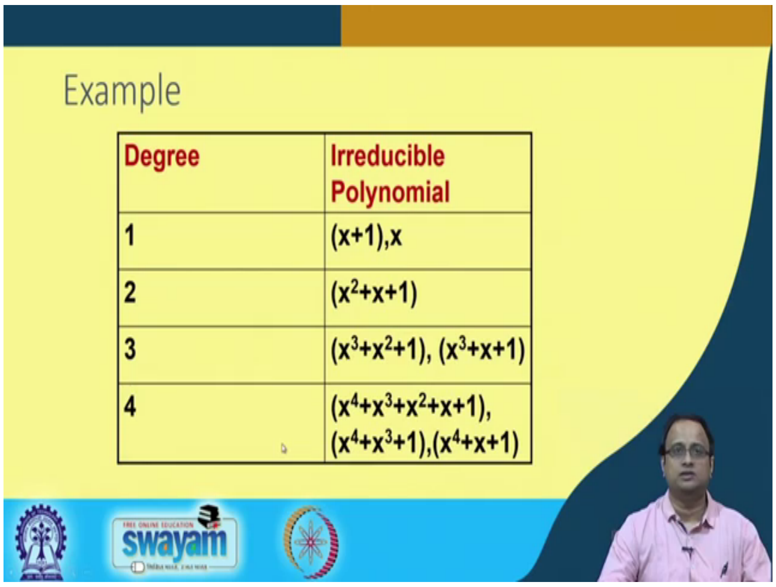
So, therefore, right I mean that that the qualifier or the property which the polynomial mean needs to satisfy is often called as a irreducibility property of the polynomial. So, this you can imagine as an analogy to what are called as prime numbers when we deal with numbers.

So, an irreducible polynomial or a polynomial d x is irreducible over the field say GF p or Galois field p if and only if there exists no two polynomials a x and b x with coefficients in GF p such that the product of a x and b x is equal to d x; that means, d x factorizes or factors into a x and b x where a x and b x are of degrees greater than 0; that means, I do not get any non trivial factors of d x.

So, therefore, right let a if your field is GF p then with suitable choices for this reduction polynomial; that means, if this reduction polynomial satisfies this property of irreducibility then I get a field. So, therefore, if you for example, take that this particular polynomial essentially satisfies the properties and say off irreducibility then finally, I get a finite field which has got p to the power of n element. So, p is that characteristic of the field and n is an integer which is the defined by the degree of my irreducible polynomial.

For example, like if my if the degree of my irreducible polynomial is say m; that means, I have got a polynomial of the form of x to the power of m plus so on that means, any element which is there in the field will have a degree which is lesser than m. So, therefore, the write the number of elements that I will find in that field is p to the power of m, and this is often customary represented as GF p power of m or GF p power of n if m be n and then write GF p power of n is called as the extension field of GF p.

(Refer Slide Time: 15:05)



Example

Degree	Irreducible Polynomial
1	$(x+1), x$
2	(x^2+x+1)
3	(x^3+x^2+1), (x^3+x+1)
4	$(x^4+x^3+x^2+x+1)$, (x^4+x^3+1), (x^4+x+1)

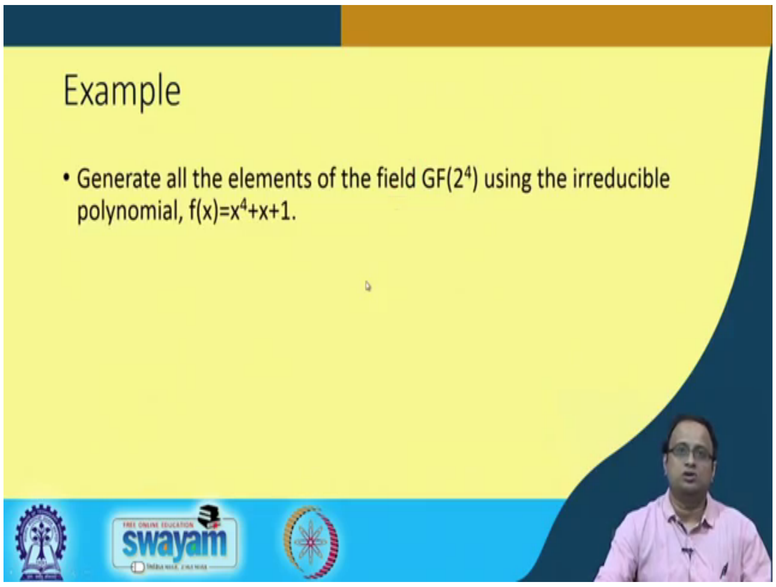
So, now here are some possible examples, but we essentially can have bigger tables for example, with degree 1 this is an irreducible polynomial, degree 2 we have an irreducible polynomial, degree 3 this is an irreducible polynomial, degree 4 this is another irreducible polynomial. So, these are all irreducible polynomials to define fields like GF GF 2 power of 2, GF 2 power of 3, GF 2 power of 4 and so on.

One can observe that these are all either trinomials or pentanomials, but you cannot have an irreducible polynomial with even number of terms. One of the reasons is if you do that in this case for example, like if you have like a x power of 4 plus 1 as an alternative choice of an irreducible polynomial then you can trivially see that x plus 1 is a factor of that polynomial and therefore, that does not qualify as an irreducible polynomial. So, therefore, right x power of 4 plus x plus 1 or x power of 4 plus x cube plus 1 on the other end does not have such trivial factors and therefore, that qualifies as an irreducible polynomial.

So, what I can now do is that I can take my I can take arbitrary polynomials I can take arbitrary polynomials where the coefficients are in GF 2 and once I take such polynomials and I start reducing them or I do modulo operation with either or with any of these you know like polynomials like say GF like x power of 4 plus x plus 1, then I can construct the field GF 2 to the power of 4; that means, what I will try to do is I will take any polynomial which has got 0 1 coefficients and then I will divide that polynomial with x power of 4 plus x plus 1 as a choice of my individual polynomial take the remainder the remainder becomes an element in my field.

So, if I start constructing such fields then the maximum number of elements that I can have in the field is 2 to the power of 4. So, I can have you know like 16 possible elements in the field, but at the same time right this essentially is just one possible I would say like one possible annotation of the field if you take another polynomial then you will have another annotation of the field. And, one can show that all these fields right are equally humorous, but they are not equal. They are essentially what are called as their equivalent in some sense because they are isomorphic to each other.

(Refer Slide Time: 17:17)

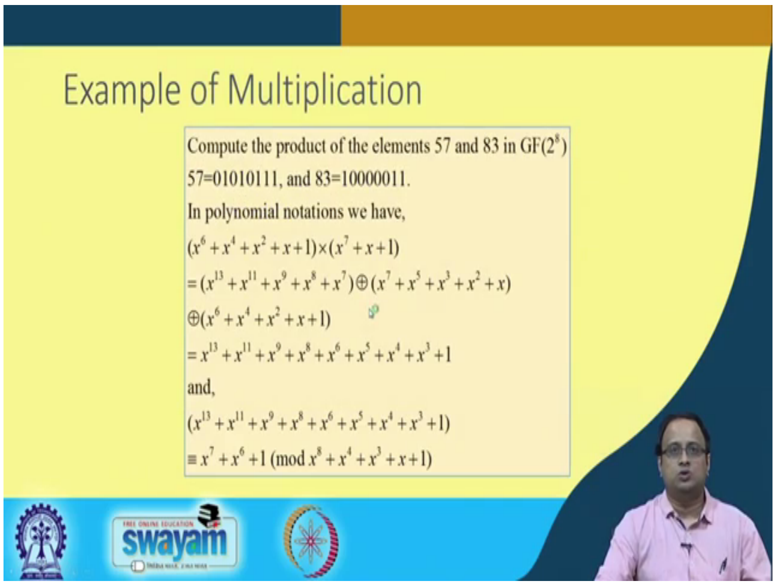


Example

- Generate all the elements of the field GF(2^4) using the irreducible polynomial, $f(x)=x^4+x+1$.

So, so, with this background right I mean I just give you I will not go or solve this particular problem, but rather I will just ask you with this question; that means, how you can construct a GF 2 power of field using the polynomial say x to the power of 4 plus x plus 1 is something which I leave to you at this point as an exercise.

(Refer Slide Time: 17:39)



Example of Multiplication

Compute the product of the elements 57 and 83 in GF(2^8)

57=01010111, and 83=1000011.

In polynomial notations we have,

$(x^6+x^4+x^2+x+1)\times(x^7+x+1)$

$=(x^{13}+x^{11}+x^9+x^8+x^7)\oplus(x^7+x^5+x^3+x^2+x)$

$\oplus(x^6+x^4+x^2+x+1)$

$=x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1$

and,

$(x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1)$

$\equiv x^7+x^6+1 \pmod{x^8+x^4+x^3+x+1}$

So, so, now let us give me let us give another an example of how we can do a multiplication as we have seen addition for example, which was very easy, but multiplication would be slightly more complex. For example, if I take elements like 5 7 and 8 3 which are elements in GF 2 power of 8. So, 5 7 is as represented here and 8 3 as represented here and if I take the polynomial notations, then I have got elements the first element as I said before and likewise this is my second element.

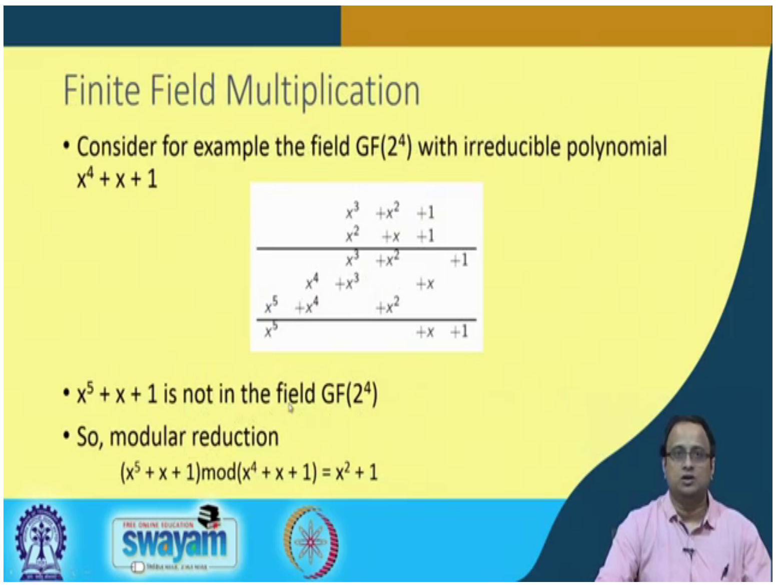
Now, when I am multiplying these two elements then I can observe that for example, I have got a degree here x power of 6 and I have got a degree here x power of 7. So, when I multiply them then I have got my degree which becomes x power of 13, ok. Now, x power of 13 right is not there in the field because my field which has got GF 2 power of 8 in this case can have maximum degree of 7, ok. So, it cannot have x power of 8 or more than that as its degree. So, what about what I need to do is therefore, I take the final result.

So, for example, this is the product of my two of the two polynomials or the two elements and then I need to reduce it or I need to take a modulo operation with this polynomial; that means, with the corresponding with the corresponding irreducible polynomial for GF 2 power of 8. So, in this case I have taken the irreducible polynomial as suggested in standard tables which is x to the power of 8 plus x to the power of 4 plus x to the power of 3 plus x plus 1.

So, therefore, if I take this polynomial and if I reduce it, then or take the modulo with this polynomial then I get x to the power of 7 plus x to the power of 6 plus 1. So, note that this polynomial is in GF 2 power of 8 because the maximum degree of this polynomial is 7 and therefore, this is an is a field element I get an element in the field.

So, likewise right I can show that multiplication is also closed, and therefore, I essentially I have got a nice field structure on which I can define my arithmetic both addition as well as product.

(Refer Slide Time: 19:33)



Finite Field Multiplication

- Consider for example the field GF(2^4) with irreducible polynomial $x^4 + x + 1$

$$\begin{array}{cccccc}
 & & x^3 & +x^2 & +1 & \\
 & & x^2 & +x & +1 & \\
\hline
 & & x^3 & +x^2 & & +1 \\
 & x^4 & +x^3 & & +x & \\
x^5 & +x^4 & & +x^2 & & \\
\hline
x^5 & & & & +x & +1
\end{array}$$

- $x^5 + x + 1$ is not in the field GF(2^4)
- So, modular reduction

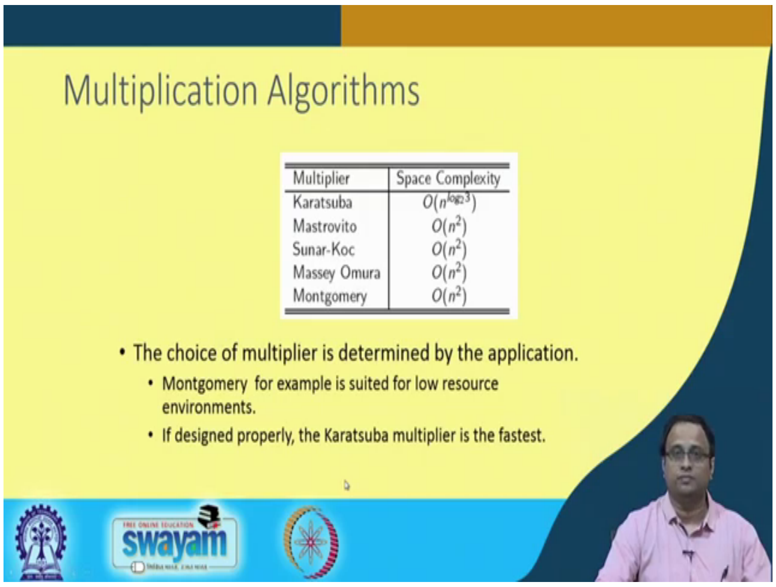
$(x^5 + x + 1) \bmod (x^4 + x + 1) = x^2 + 1$

So, so, therefore, right the finite field is therefore, you know like here is another example of how we can do it or how we can do it for it. So, this example is with slightly smaller field. For example, the field is GF 2 power of 4. So, there are two elements like x power of 3 plus x square plus 1 there is another elements x square plus x plus 1, when I am doing multiplication the first step is a normal school book operation where I elaborate or get the partial products and then I start adding them.

Note that the addition is in done in GF 2; that means, if you have got 2 x square elements like x square and x square then you do not get that reflected in the result because when I am adding x square with x squared; that means, that the coefficients are XORed; that means, one gets XOR with 1. So, you get 0 here.

So, therefore, in this case the result is x to the power of 5 plus x plus 1. Again like before this is not a field element because this does not belong to GF 2 power of 4. So, therefore, I need to again do a modulo reduction operation. So, the modulo the irreducible polynomial for GF 2 power of 4 is say x to the power of 4 plus x plus 1. So, basically what I do is, I take this polynomial divided by x to the power of 4 plus x plus 1 and I take the remainder the remainder in this case is x squared plus 1.

(Refer Slide Time: 20:43)



Multiplication Algorithms

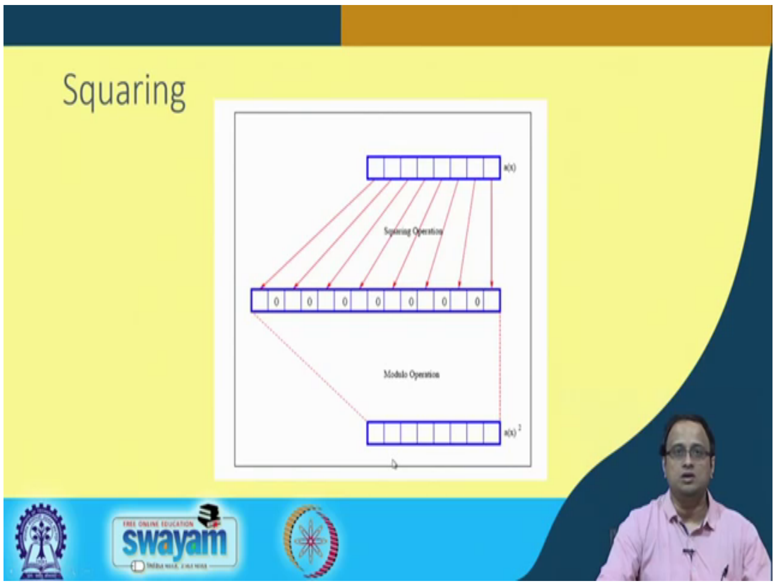
Multiplier	Space Complexity
Karatsuba	$O(n^{\log_2 3})$
Mastrovito	$O(n^2)$
Sunar-Koc	$O(n^2)$
Massey Omura	$O(n^2)$
Montgomery	$O(n^2)$

- The choice of multiplier is determined by the application.
  - Montgomery for example is suited for low resource environments.
  - If designed properly, the Karatsuba multiplier is the fastest.

So, there are different choices of multiplication algorithms, ok. So, for example, I have listed few of the important ones like Karatsuba, Mastrovito, Sunar-Koc, Massey Omura, Montgomery and so on. You can observe that most of these algorithms have got space complexity which is quadratic with the bit length of the arguments. On the other hand Karatsuba seems to be interesting because it has got which is something lesser than that. So, it is O n into the power of log 3 base 2; that means, it is more efficient compared to the other ones in terms of their space complexity.

However, the choice of the multiplier is often defined by or determined by applications for example, like if we have got you know like characteristic p fields I mean; that means, GF p fields then often montgomery multiplication is a better choice. On the other hand right if you design properly then a Karatsuba a multiplier has got its own merits and therefore, right it is an important and interesting class of algorithm to study.

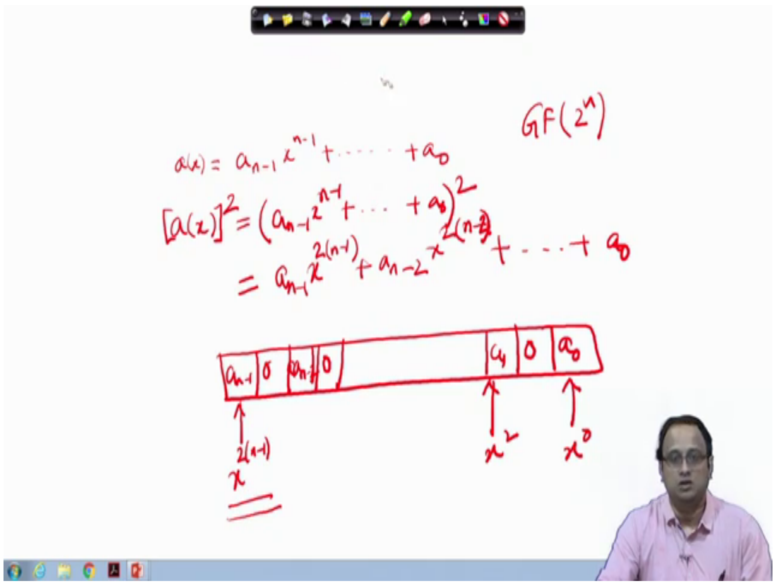
(Refer Slide Time: 21:39)



So, there are there are several other operations also right which you will probably see when you are dealing with finite fields. So, the other important operation which we also encounter along with multiplication is squaring. So, in particular when we are doing squaring in GF 2 arithmetic, then the computations are very efficient. So, this slide shows how one can do squaring.

So, for example, the input polynomial which is a x is represented in this register and I say that for example, this is an element in GF 2 power of n where n is some integer value and then when you are trying to do a squaring operation then what you just need to do is you need to interpose the corresponding bits by 0's. So, this you can understand easily by if we just considered this calculation.

(Refer Slide Time: 22:27)



$GF(2^n)$

$a(x) = a_{n-1}x^{n-1} + \cdots + a_0$

$[a(x)]^2 = (a_{n-1}x^{n-1} + \cdots + a_0)^2$

$= a_{n-1}x^{2(n-1)} + a_{n-2}x^{2(n-2)} + \cdots + a_0$

For example, if you take a polynomial say a x which is essentially nothing, but a n minus 1, x to the power of n minus 1, plus so on till a 0 and then you want to do a squaring. So, basically what I mean is I need to do a x and I would like to square that. So, the square of this is nothing, but a x to the n minus 1 x to the power of n minus 1 plus. So, on till a 0 and you would like to square.

So, now, I just leave it to you with this result here, but without going into details is that when you do a squaring in normal operations right you get often coefficients which are multiples of 2. But, in this case since you are dealing with this field that is GF 2 power of n then all those elements will go to 0 because in this case 2 is nothing, but which is congruent to 0.

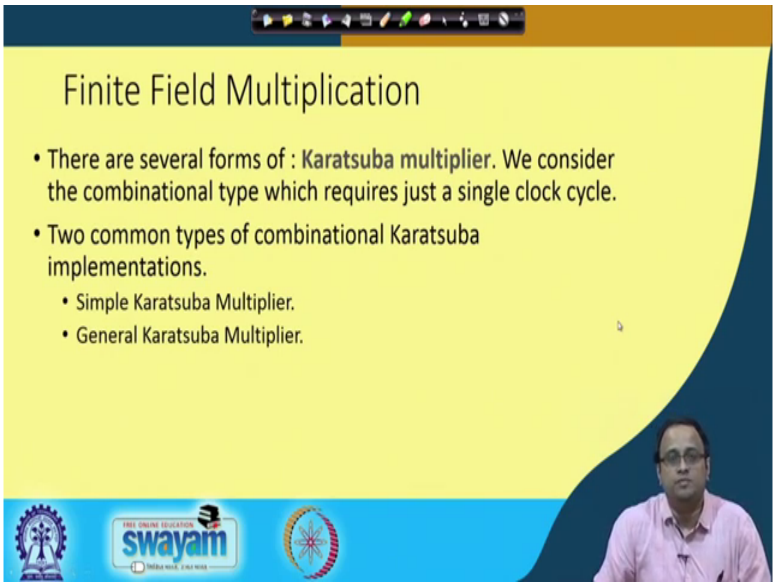
So, therefore, what you will get is basically a n minus 1 square, but again a n minus 1 is an element in GF 2. So, n minus 1 square is also an minus 1 and n minus 1 x to the power of 2 times n minus 1 plus say an minus 2 x to the power of 2 n minus 1 which is like you know like the next element. So, you will get this us say another n minus 2 into 2 plus so on till a 0 which is the constant term of the polynomial.

So, therefore, right you can imagine that if you try to note the result right the result essentially is nothing, but copying the original elements like a 0 and a 1 and the final one is a n minus 1, but the coefficient here is x to the power of 2 n minus 1 and this is the coefficient of this is x power of 0, the coefficient of this is x square. So, these are all even degree terms whereas, the in between 1's are nothing, but 0. So, this is your a n minus a n minus 2, ok.

So, therefore, right what you basically get is nothing, but if these that you take the original polynomial coefficients, but you just interpose them with zeros and that is nothing, but your a x square. Of course, as you can see that x to the power of 2 n minus 1 is more than 2 to the power of n. So, therefore, this number is not an element in the field. So, you need to do a reduction operation and bring the result back to the field.

So, so, with this background right I mean we can you know like go back and ok. So, therefore, what you would still need to do is you need to bring the result back to the field and the final result essentially should be back to the field means you need to do a modulo operation. So, that the result becomes back or comes back to GF 2 power of n.

(Refer Slide Time: 25:29)



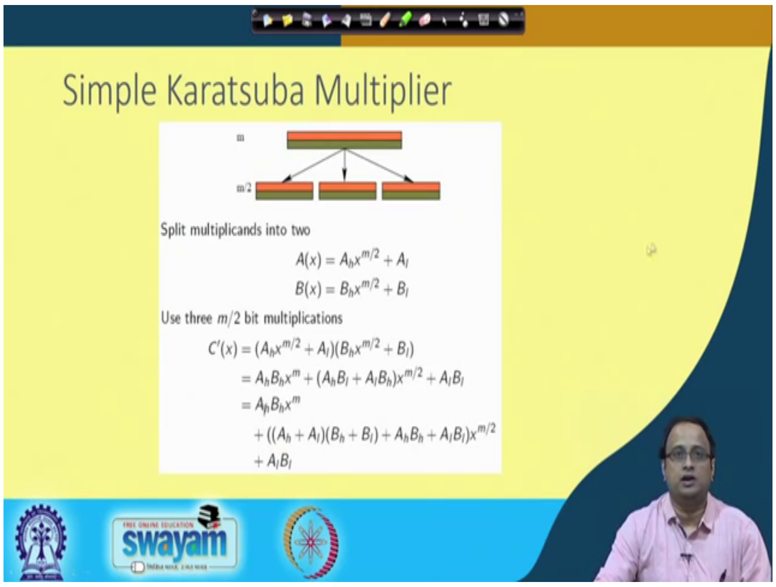
Finite Field Multiplication

- There are several forms of : Karatsuba multiplier. We consider the combinational type which requires just a single clock cycle.
- Two common types of combinational Karatsuba implementations.
  - Simple Karatsuba Multiplier.
  - General Karatsuba Multiplier.

So,. So, so, the other important operation as I said is the finite field multiplication. So, in finite field multiplication right there are different forms of multipliers, but I in particular in specific right I will be trying to explain the Karatsuba multiplication which is a very common form of combinatorial multiplication. That means the result when you are giving the inputs a x and b x and the corresponding irreducible polynomial it gives you the result a x into b x mod p x in just one single clock cycle.

And, there are different forms of commontoriel combinational Karatsuba implementations. Two very a common forms of algorithms are what are called as a simple Karatsuba multiplier and the general Karatsuba multiplier.

(Refer Slide Time: 26:09)



Simple Karatsuba Multiplier

Split multiplicands into two

$$A(x) = A_h x^{m/2} + A_l$$
$$B(x) = B_h x^{m/2} + B_l$$

Use three $m/2$ bit multiplications

$$\begin{aligned} C'(x) &= (A_h x^{m/2} + A_l)(B_h x^{m/2} + B_l) \\ &= A_h B_h x^m + (A_h B_l + A_l B_h) x^{m/2} + A_l B_l \\ &= A_h B_h x^m \\ &\quad + ((A_h + A_l)(B_h + B_l) + A_h B_h + A_l B_l) x^{m/2} \\ &\quad + A_l B_l \end{aligned}$$

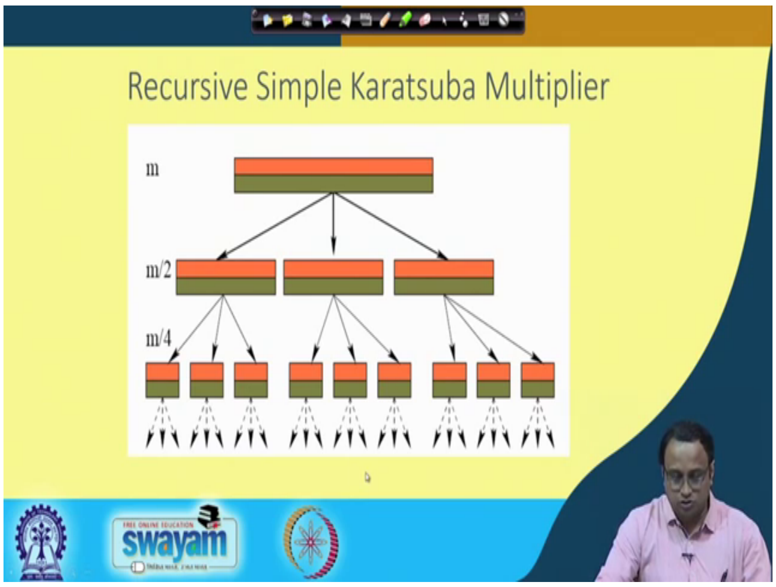
So, in particular right let us see how the simple Karatsuba multiplier works. So, in simple Karatsuba multiplier you basically take the input A x and B x which you would like to multiply and you split each of the polynomials into two parts like A h and A l and likewise B x into B h and B l.

So, now when you want to do this operation that is A x into B x, in a normal school book multiplication right what do you do? We basically have got four sub multipliers. You have got to multiply A l with B l A h with B l and likewise A h with B h and A l with B h, ok. Now, in the Karatsuba setting what you do is you basically do the multiplications or some multiplications A h and B h and A A h and A l and B l, ok, but the other part that is A h into B l plus A l into B h you write them write them in this form.

So, you basically elaborate this term and write it as A h plus A l into B h plus B l. Note that if you do this then there are some extra terms which are you know getting incorporated which are A h into which is A h into B l plus A l into B h, you would like to subtract that, and in GF 2 arithmetic subtraction is nothing, but addition because both of them are XORs. So, therefore, in order to eliminate them or cancel out them you basically add A h into B h plus A l into B l and therefore, this is equivalent to A h into B l plus A l into B h.

If you do that right then one can observed that the some multiplications that we need to do are A l into B l into B h which you are again reusing here and the third multiplication which is like A h plus A l into B h plus B l. Of course, you need to do some extra additions, but we have reduced four multiplications to three multiplications, and that gives you saving.

(Refer Slide Time: 27:55)



Recursive Simple Karatsuba Multiplier

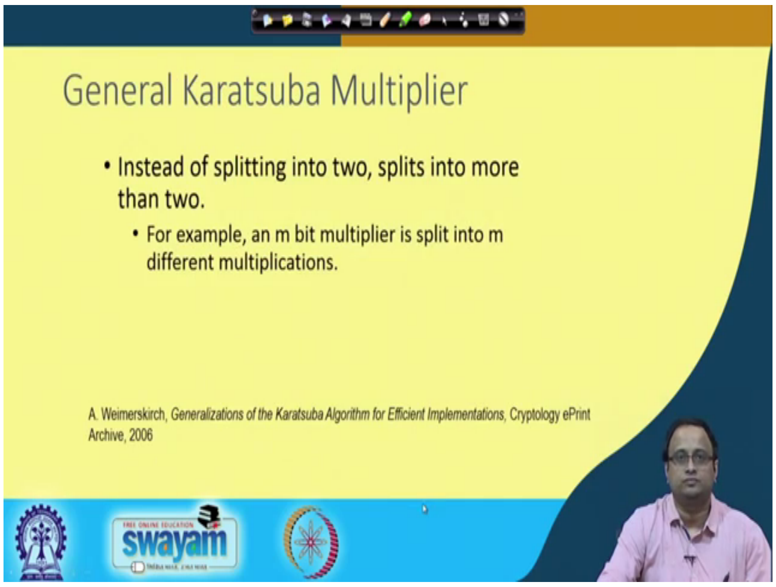
So, you can apply this in a recursive manner. So, you can take m and assuming that m is a power of 2 you can decompose into m by again m by 2 blocks which you can again sub decompose and you can carry on applying this simple Karatsuba to a multiplication in a recursive manner, ok. But, at the same time it is important to understand that this Karatsuba multiply becomes efficient if you do a proper thresholding.

For example, right if you go down two bits right then you what you are basically saving is AND gates because in terms of with respect to bits right a multiplication is nothing, but an AND operation and the addition is nothing, but essentially an OR operation on an XOR operation, ok.

So, therefore, right when you are what you are basically saving is multiplications. That means if you just go down to one or single bits then you are saving AND gates, but you are expending other gates, ok. So, therefore, that may not be very you know like efficient. So, basically you have to stop this applying the simple Karatsuba technique at a suitable point so that you essentially can leverage the amount the saving you know like the saving which you are getting at the cost of something.

So, therefore, right thresholding is a very key k key step when you if you want performance when you are implementing Karatsuba multiplications.

(Refer Slide Time: 29:05)



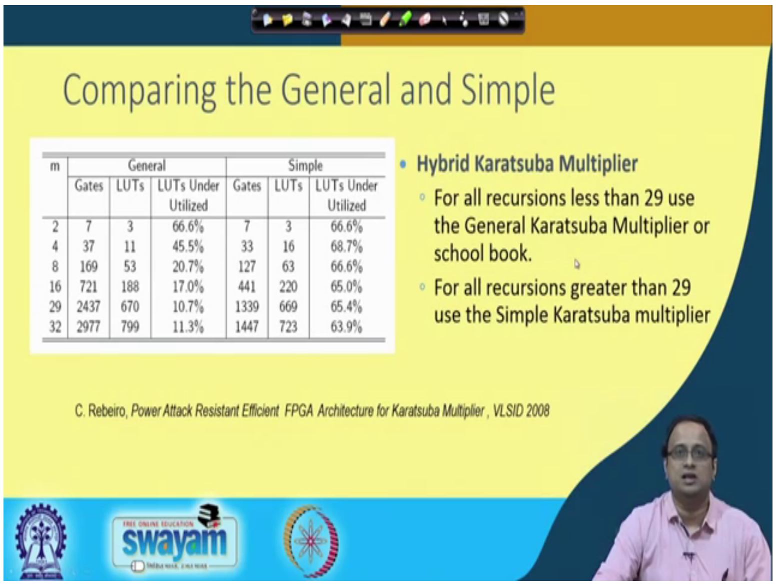
General Karatsuba Multiplier

- Instead of splitting into two, splits into more than two.
  - For example, an m bit multiplier is split into m different multiplications.

A. Weimerskirch, *Generalizations of the Karatsuba Algorithm for Efficient Implementations*, Cryptology ePrint Archive, 2006

So, there is there is another flavor of Karatsuba multiplication which are called as general Karatsuba multiplication. In general Karatsuba multiplication instead of splitting into two we split into more than two, ok. For example, an m bit multiplier is split into m different multiplications and that essentially leads to another flavor of multiplication which are called as general Karatsuba.

(Refer Slide Time: 29:23)



Comparing the General and Simple

m	General			Simple		
	Gates	LUTs	LUTs Under Utilized	Gates	LUTs	LUTs Under Utilized
2	7	3	66.6%	7	3	66.6%
4	37	11	45.5%	33	16	68.7%
8	169	53	20.7%	127	63	66.6%
16	721	188	17.0%	441	220	65.0%
29	2437	670	10.7%	1339	669	65.4%
32	2977	799	11.3%	1447	723	63.9%

- **Hybrid Karatsuba Multiplier**
  - For all recursions less than 29 use the General Karatsuba Multiplier or school book.
  - For all recursions greater than 29 use the Simple Karatsuba multiplier

C. Rebeiro, *Power Attack Resistant Efficient FPGA Architecture for Karatsuba Multiplier*, *VLSID 2008*

So, I will come into that in slightly more details, but before I go into that let us remember the LUT structure whichever FPGA has, ok. So, if I showed or discussed in the last class. So, now, if I take these two multiplications which we discussed like the simple Karatsuba which I elaborated and a general which I just mentioned right as another topology of the Karatsuba multiplication and if you observe the lookup table and the gates right then you see an important observations.

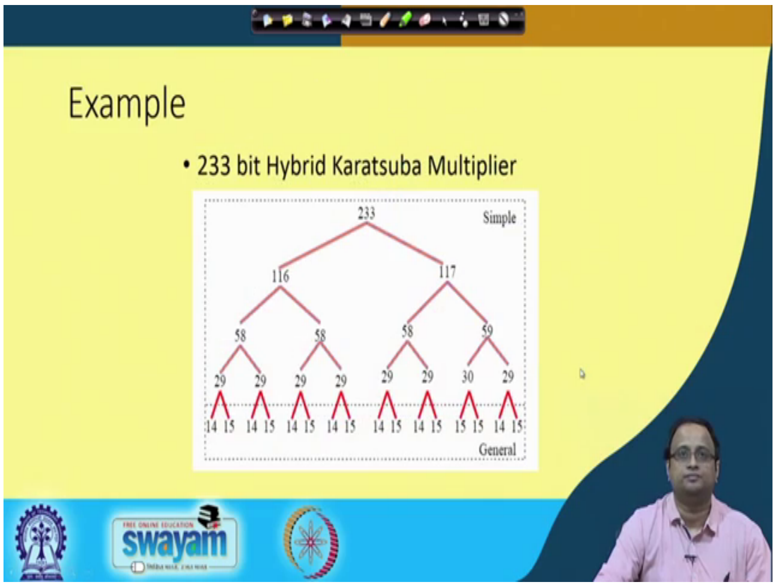
For example, light if you see the when we are increasing the finite or the m or value of m or the other size of your finite field then in the context of general Karatsuba then you see that the gate count increases significantly compared to the simple Karatsuba, ok; that means, that general Karatsuba consumes more amount of resources. But, if you go down with smaller field sizes like for example, field sizes of 4 then you see that the gate count right of the general Karatsuba is essentially 37. But, interestingly if you see the look up table counts then you see that the lookup table count is less compared to that of the simple Karatsuba. For example, here you see it is 11 whereas, here it is 16.

So, this is because the general Karatsuba for smaller dimensions or in general has got a bit add LUT underutilization. For example the LUT underutilization for the general Karatsuba is something like 45 percentage when you are having a field value of 4 whereas, the in the underutilized for simple Karatsuba is always around 65 percent or 66 percent.

So, therefore, right you can actually adopt a strategy where you can actually apply the simple Karatsuba when you have got a higher dimension of m. But, as you go down when you are doing this recursive operation you switch from the simple Karatsuba to the general Karatsuba because for smaller field sizes the general Karatsuba would require lesser number of LUTs and therefore, would be more efficient. So, this particular structure is something which we call as the hybrid Karatsuba and therefore, it right it can give you two more compact designs.

So, basically what we do is for all regressions which are less than say some specific threshold. So, in this case in our case it was 29, but it will vary from platform to platform. So, we one can use the general Karatsuba multiplication or the school book or even the school book multiplier, but for all higher recursions right we use the simple Karatsuba as essentially as a word the broad Karatsuba operation.

(Refer Slide Time: 31:45)



Example

- 233 bit Hybrid Karatsuba Multiplier

So, for example, like in this case we give an example with 233 bit. 233 is because you know like many of our cryptographic operations works on 233 bits as I will be telling in the next classes. So, 233 bit is something which is not an even number, so, we cannot decompose into equal partition. So, what we do is we decompose into say party one partition of a 116 bits another partition of 107 bits 17 bits and then carry on this decomposition and I say that there is a specific threshold below which I switch my algorithm.

So, I essentially apply a different flavor of multiplication. In this case I have applied general Karatsuba, but even a school book multiplication would be interesting to see. The main thing is that you have to change from the standard Karatsuba because otherwise right there will be an there will be wastage of the look up tables.

So, what we will do is we will basically right we will I we will stop here and in the next class right we will try to go into more details, where we will be trying to go into or we will try to understand how to implement this basic structure. So, we will try to see how we can realize the structure on Verilog and we will see that how we can develop a design or rather develop and complete into and Karatsuba or multiplication algorithm on hardware. So, with this I would like to thank you.