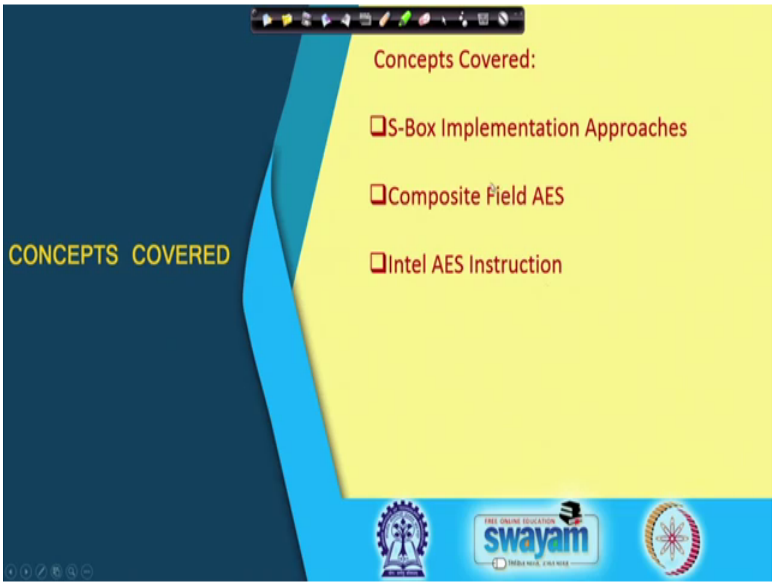
Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 13
Hardware Implementation of Advanced Encryption (Contd.)

So, welcome back. So, we shall continue with what we were discussing in the last class or rather we shall be trying to you know like look into the S-box design. We shall be trying to see how we can develop the AES S-box design and by applying the techniques of isomorphism that we were discussing in the last few classes.

(Refer Slide Time: 00:34)



CONCEPTS COVERED

Concepts Covered:

❑S-Box Implementation Approaches

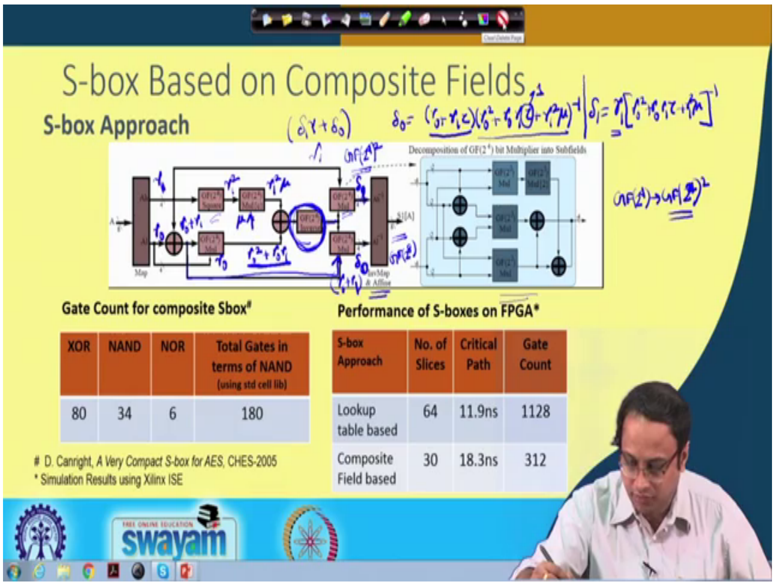
❑Composite Field AES

❑Intel AES Instruction

So, the concepts that I will be trying to cover in today’s class is rather to take a deep look into the S-box implementation. So, this is something that we shall be trying to cover in some of the next classes as well. And talk about totally I mean getting a composite field AES that means, where the entire AES is implemented in composite fields ok.

Then these are important thing, because you know like as you probably have understood by now that when you are doing a you know when you are doing the operations in composite field, then you also need to do a field transformation from the input field to the output field that means, from the field GF 2 to the power of 8 to GF 2 to the power of 4 square and vice versa. So, in order to diminish that we often implement the entire AES in composite fields ok and that is what we shall also be trying to look into. And finally, conclude with a quick comment about the Intel AES instruction.

(Refer Slide Time: 01:20)



S-box Based on Composite Fields

S-box Approach

$(\delta_1\gamma+\delta_0)$ $\delta_0=(\gamma_0+\gamma_1\tau)(\gamma_0^2+\gamma_0\gamma_1\tau+\gamma_1^2\mu)^{-1}$ | $\delta_1=\gamma_1[\gamma_0^2+\gamma_0\gamma_1\tau+\gamma_1^2\mu]^{-1}$

$GF(2^8)\to GF(2^4)^2$

Gate Count for composite Sbox#

XOR	NAND	NOR	Total Gates in terms of NAND (using std cell lib)
80	34	6	180

D. Canright, *A Very Compact S-box for AES*, CHES-2005

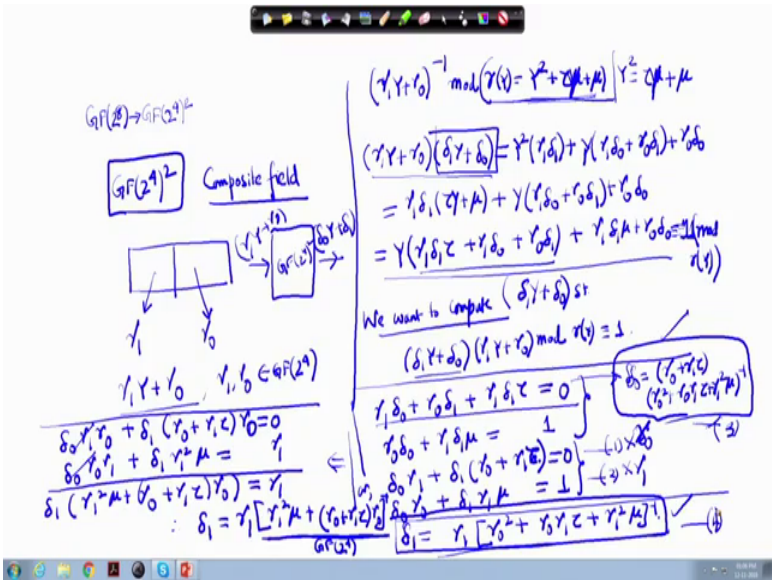
* Simulation Results using Xilinx ISE

Performance of S-boxes on FPGA*

S-box Approach	No. of Slices	Critical Path	Gate Count
Lookup table based	64	11.9ns	1128
Composite Field based	30	18.3ns	312

So, again let us come back to this slide where we were discussing about the S-box on composite fields, and we will try to take a quick look into this architecture ok. So, how essentially it works.

(Refer Slide Time: 01:38)



$GF(2^8)\to GF(2^4)^2$

$\boxed{GF(2^4)^2}$ Composite field

$\gamma_1\gamma+\gamma_0$, $\gamma_1,\gamma_0\in GF(2^4)$

$\delta_0\gamma_1\gamma_0+\delta_1(\gamma_0+\gamma_1\tau)\gamma_0=0$

$\delta_0\gamma_0\gamma_1+\delta_1\gamma_1^2\mu=\gamma_1$

$\delta_1(\gamma_1^2\mu+(\gamma_0+\gamma_1\tau)\gamma_0)=\gamma_1$

$\therefore\ \delta_1=\gamma_1[\underbrace{\gamma_1^2\mu+(\gamma_0+\gamma_1\tau)\gamma_0}_{GF(2^4)}]^{-1}$

$(\gamma_1\gamma+\gamma_0)^{-1}\bmod(r(\gamma)=\gamma^2+\tau\gamma+\mu)$ | $\gamma^2=\tau\gamma+\mu$

$(\gamma_1\gamma+\gamma_0)\boxed{(\delta_1\gamma+\delta_0)}=\gamma^2(\gamma_1\delta_1)+\gamma(\gamma_1\delta_0+\gamma_0\delta_1)+\gamma_0\delta_0$

$=\gamma_1\delta_1(\tau\gamma+\mu)+\gamma(\gamma_1\delta_0+\gamma_0\delta_1)+\gamma_0\delta_0$

$=\gamma(\gamma_1\delta_1\tau+\gamma_1\delta_0+\gamma_0\delta_1)+\gamma_1\delta_1\mu+\gamma_0\delta_0\equiv 1(\bmod\ r(\gamma))$

We want to compute $(\delta_1\gamma+\delta_0)$ st

$(\delta_1\gamma+\delta_0)(\gamma_1\gamma+\gamma_0)\bmod r(\gamma)\equiv 1$.

$\gamma_1\delta_0+\gamma_0\delta_1+\gamma_1\delta_1\tau=0$

$\gamma_0\delta_0+\gamma_1\delta_1\mu=1$

$\boxed{\delta_0=(\gamma_0+\gamma_1\tau)(\gamma_0^2+\gamma_0\gamma_1\tau+\gamma_1^2\mu)^{-1}}$ — (3)

$\delta_0\gamma_1+\delta_1(\gamma_0+\gamma_1\tau)=0$ — (1) $\times\gamma_0$

$\delta_0\gamma_0+\delta_1\gamma_1\mu=1$ — (2) $\times\gamma_1$

$\boxed{\delta_1=\gamma_1[\gamma_0^2+\gamma_0\gamma_1\tau+\gamma_1^2\mu]^{-1}}$ — (4)

So, in order to do that I will try to you know like tell you of you know like or rather go into the math of this design. So, in particular right let us try to see that suppose you have implemented or you have developed a design, where you essentially have broken up that you have got that got the design in GF 2 to the power of you know like 4 square that means, you are you are you are basically transformed the element from GF 2 to the power of 8 to GF 2 to the power of 4 square.

So, what I want now is that I want to calculate the final feet finite field inverse now in this field right, I want to calculate the finite field inverse in GF 2 to the power of 4 square, so that is my what I said as the composite field. So, I want to know how to calculate my finite field inverse in this particular composite field ok. So, how does an element in composite field look like? So you know like as I said that when you are implementing a field in GF 2 to the power of 4 square that means, there are two parts of this you know like polynomial. The first part is say you know like say let us tell it is gamma 1, and the second part is say gamma 0 ok.

So, therefore right the way you can read this element is a polynomial, we just say gamma 1 y plus gamma 0. So, note that each of these elements gamma 1 and gamma 0 that means gamma 1 comma gamma 0 are both elements in GF 2 to the power of 4 ok. So, they are elements in the finite in the composite in the field GF 2 to the power of 4.

So, now what I want is I want to calculate basically write this that is I want to calculate gamma 1 y plus gamma 0 whole to the power of minus one, because I want to calculate the finite field inverse. And I want to do this you know like modulo my irreducible polynomial. So, what is the reducible polynomial here? So, remember the irreducible polynomial here is r Y, which is denoted as something like Y squared plus tau Y plus mu ok.

So, remember that when we are talking about composite fields, then we were doing this modulo q y p x kind of thing in the last class. And that essentially is like one of these polynomials is this where you are where you are basically handling elements in GF 2 to the power of 4 square. So, for doing this right we have to also understand first that how you do multiplication in this field ok.

So, let us take this element like gamma 1 plus gamma 0, and try to multiply it with say delta 1 Y plus delta 0 ok. So, when you are trying to do this multiplication, remember that so let us try to do this multiplication ok. So, therefore if I just apply the schoolbook operation, then the first then I do this right, I get gamma I mean Y square. And the coefficient for this is gamma 1 delta 1. Likewise if I take Y, then the coefficients are gamma 1 delta 0 plus gamma 0 delta 1 plus gamma 0 delta 0 ok.

So, note that here I have exceeded the you know like the limit of this field that means, the degree is now Y square. And therefore, I do a modular reduction with this polynomial ok. So, in order do that again what I do is this right, but we basically substitute Y squared with tau mu sorry this is like so this would be Y ok. So, basically it is tau Y plus mu, so therefore I substitute this might you know like tau Y plus mu, so that means Y square will be substituted with gamma 1 delta 1 and Y square will be substituted with tau tau Y plus mu ok. And then you have of course you have got Y into gamma 1 delta 0 plus gamma 0 delta 1 plus gamma 0 delta 0 ok.

So, now if I just combine right, then and take out and bring Y other coefficients of Y at one place, then I have got gamma 1 delta 1 tau plus gamma 1 delta 0 plus gamma 0 delta 1 plus the constants which is gamma 1 delta 1 mu plus gamma 0 delta 0 ok. So, this is what we have you know when we are doing this multiplication operation.

So, remember that so now what we want is that we want we want to basically calculate. So, we basically we want to compute a value, which is say denoted as delta 1 Y plus delta 0 such that you know like delta 1 Y plus delta 0 multiplied with gamma 1 Y plus gamma 0 mod of mod of r Y mod of r Y should give me 1 ok. So, this implies that I basically want to calculate a delta 1 y plus delta 0 that means, essentially this part such that this result is equivalent to 1 mod of mod of r Y mod of r Y; r Y is essentially this Y square plus tau Y plus mu ok, so that means right what I want is since I want this to be equal to 1 that means, this to be congruent to 1 modulo this field. What I basically end up in is in having two equations.

So, the equation-1 is essentially gamma you know like gamma 1 gamma 1 delta 0 plus gamma 0 delta 1 plus gamma 1 delta 1 tau, which is essentially this part ok. So, this essentially is equal to 0, because I do not have any Y term here on the right hand side. Whereas, the other part which is gamma 0 delta 0 plus gamma 1 delta 1 mu right this is equal to 1. So, I have got two parts here ok.

So, you can imagine that I want delta 0 and delta 1, so basically I can just rewrite these right essentially and write that this is nothing but delta 0 gamma 1 plus I can take delta 1 common and I have got gamma 0 plus gamma 1 delta 1 I mean delta 1 is taken common.

So, I have got gamma 1 gamma 1 tau and, this is equal to 0. And I have got the other part, which is delta 0 gamma 0 plus delta 1 gamma 1 mu and this is equal to 1.

So, now I have got you know like two equations, and I have got two variables to solve that is gamma 0 and delta 1. So, how do I essentially solve them? We just apply simple you know like simple technique, where I basically you know like basically I just multiply this equation. So, there are two equations here 1 and 2. So, I just multiply one by because I want to eliminate say delta 0. So, I multiply this by you know like I just multiply this by I just multiply this by gamma 0, and I multiply this by gamma 1 ok.

So, therefore if I do this, then what I get is this right I get so if I just multiply this, then I get delta 0 gamma 1 gamma 0 plus delta 1 gamma 0 plus gamma 1 tau multiplied with gamma 1, and that is equal to 0 ok. So, I have multiplied sorry this would be gamma 0. So, I have multiplied gamma 0 here. And the first term is delta 0 gamma 0 gamma 1, because I am multiplying gamma 1 here plus delta 1 gamma 1 square mu ok, and that is equal to gamma 1.

So, therefore now if I add up these terms, remember it is in GF 2 power of 8, so basically I end up getting gamma 0, and then these two terms gets cancelled. So, I have got gamma 1 square mu plus gamma 0 plus gamma 1 tau ok. And essentially I have got I basically I have taken delta 1 common, so this essentially multiplies with gamma 0 ok. And finally, I essentially get gamma 1 on the right side.

So, therefore I can you know do a simplification, where I essentially I mean you can bring this to the right hand side, and therefore this is nothing but delta 1 is gamma 1 multiplied with gamma 1 square mu plus gamma 0 plus gamma 1 del tau gamma 0, and this I do an inverse. So, basically I calculated an inverse. But, note that this inverse is in GF 2 to the power of 4 ok. So, therefore this inverse is done in the in the in a smaller field, which is GF 2 to the power of 4.

So, therefore write I mean what we get here is that delta 1 is gamma 1 ok, and essentially write you can also just do as few simplifications here and you basically can get here this is nothing but you know like gamma 1 into gamma 0 square plus if I multiply this, you get gamma 0 gamma 1 tau plus gamma 1 square mu ok, and this is we do an inverse ok.

So, therefore write I mean this is essentially the equation that you get for gamma 1. So, likewise right you can take this gamma 1, and you can plug into this equation and from there you should be able to get delta 0 ok. So, like we have got delta 1 here, so you should be able to get delta 0 and it turns out that delta 0. If we just apply this and just plug in over there, you will get delta 0 is gamma 0 plus gamma 1 tau multiplied with again this inverse that is gamma 0 squared plus gamma 0 gamma 1 tau plus gamma 1 square mu whole to the power of minus 1 ok. So, this gives me the two parts which I want, so I am able to calculate gamma delta 0 and delta 1.

So, therefore write now I essentially take this input that means I take gamma 1 gamma 1 y plus gamma 0 as an input, and the inverse I mean essentially is done now in GF 2 to the power of 4 square. And the final result is also in GF 2 the power of 4 square. And the result is denoted as delta 0 Y plus delta 1 ok. And the equations or transformations are essentially given by these two equations say equation-3 and equation-4. So, you have got the equation-3 and equation-4 to tell you what are these transformations.

So, once you have done this right, then we can now go back to our slide. And we can see that essentially that is being ok, so that is being exactly done here ok. So, you can see here that essentially its being a done here, where you would like to you know like do this inverse calculation now in GF 2 to the power of 4 square. And if you observe here that these are your inputs, this is an input right these are input A which is in GF 2 to the power 8 we do a map, and then get the result in GF 2 to the power of 4 square. So, this higher part is or a h as written over here is essentially nothing but your gamma 1 ok. So, this is your gamma 1 and this is your gamma 0.

So, therefore what I do here is, I now you know like I can annotate here, you know like based on the previous discussions. And let me see yeah. So, what I can do is I can try here, and write these as say gamma 1. So, this is my gamma 1 part ok. So, this is my gamma 1, and this is your gamma 0 ok. And now I calculate delta 0 and delta 1. So, this is my delta 0, and this is my delta 1 ok.

So, now note that what you basically if you remember like what the equations that we had was as follows, so we had delta 0 which we had could so this is your delta 0, so we had delta 0 as equal to gamma 0 plus gamma 1 tau multiplied with gamma 0 square plus gamma 0, gamma 1 tau plus gamma 1 square mu whole to the power of minus 1 ok. So, now you can note that that exactly is essentially this part here ok. For example, this is the GF 2 to the power of 4 inverse. So, this is this inverse which is being computed.

So, therefore write the input to these inverse is essentially your so you have got gamma 1 as this input. So, you do a squaring here. So, this becomes gamma 1 square ok, and then you do your multiplication. So, note that this multiplication is with the constant ok. So, and this constant is nothing but mu. So, you basically calculate gamma 1 square mu here ok. And here you do a so here you basically calculate you know like. So, this is your nothing but gamma 0 plus gamma 1, and that is passed here and this is this input is gamma 0.

So, therefore when you multiply this, you get gamma squared plus gamma 0 gamma 1 ok and that is your, you know like this part. So, here you know like what we have assumed is this tau, we have assumed to be 1 ok. So, you can do that and if you if you do that, then your this is your part gamma 0 square plus gamma 0 gamma 1, and that you add with gamma 1 square mu. And then you pass it to the input of this, and therefore you get GF you pass it to the GF 2 the power of 4 inverse, and you get this result ok.

So, now you multiply this with this part, which is essentially you know like your gamma 0 plus gamma 1 ok. And gamma 0 plus gamma 1 is therefore passed here, and you get the corresponding you know like output. So, likewise right you can see that when you are doing this with the other part also can be observed here. So, again this is the other part of the inverse, you again do a multiplication.

So, therefore the other part; if you write remember that it is delta 1, and delta 1 was a noted as gamma 1 multiplied by this inverse. So, again the same thing is written here that is gamma 0 square plus gamma 0 gamma 1 again tau is 1 so plus gamma 1 square mu whole inverse. So, you multiply this by only with gamma 1 ok, so that is essentially this part where you are so in one case you are multiplying with gamma 0 plus gamma 1, and that is this part which ever you are multiplying with gamma 0 plus gamma 1.

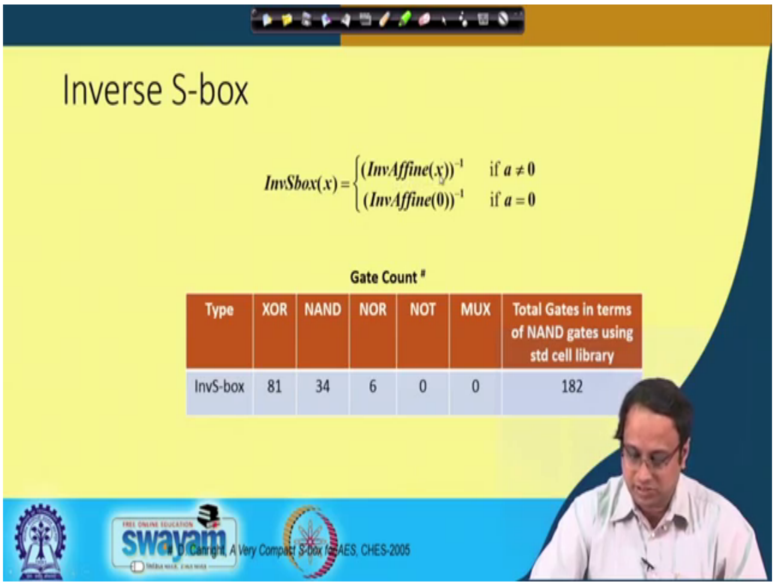
And this is the other part we are whether you where you are multiplying with only gamma 1 ok, so therefore this is you are multiplying with only gamma 1. So, so in anyway so therefore write this turns out to be that this is your rather I would say delta 1, and this is your delta 0 ok. And that is essentially the two parts of the output that you get.

So, finally when you have got this output which is in GF 2 to the power of 4 square. So, GF 2 to the power of 4 square the output that you get here is nothing but delta 1 Y plus delta 0. So, these are the two parts here. So, this is the higher part, so this is your delta 1, and this is your delta 0. So, you get this delta 1 and delta 0. And then you apply the inverse affine transformation to get the result back in GF 2 to the power of 8 ok.

So, how do you replace how do you get this GF 2 to the power of 4 inverse. So, you can actually apply this freak in a recursive manner. Therefore, if you apply this trick in a recursive manner, you again can get a sink another alternative architecture. And therefore, now you basically decompose GF 2 to the power of 4 to GF 2 to the power of 4 square ok. And therefore, write this operation now I mean GF 2 square.

So, therefore when you are doing this, then you can note that the inversions will be done in a smaller field ok. And therefore, you can actually you know like do this GF 2 to the power of 4 multiplications GF 2 to the power of 4 squares, and all of them can be decomposed into further subfields and you can do it at do them in smaller fields ok. So, so with this background right you can see that if you can see the benefits because you can see that when you are doing this operation, you can observe that the gate counts that essentially happens right it is significantly reduced. And therefore there is a lot of saving, when you are adopting this kind of tricks.

(Refer Slide Time: 18:52)



Inverse S-box

$$InvSbox(x) = \begin{cases} (InvAffine(x))^{-1} & \text{if } a \neq 0 \\ (InvAffine(0))^{-1} & \text{if } a = 0 \end{cases}$$

Gate Count #

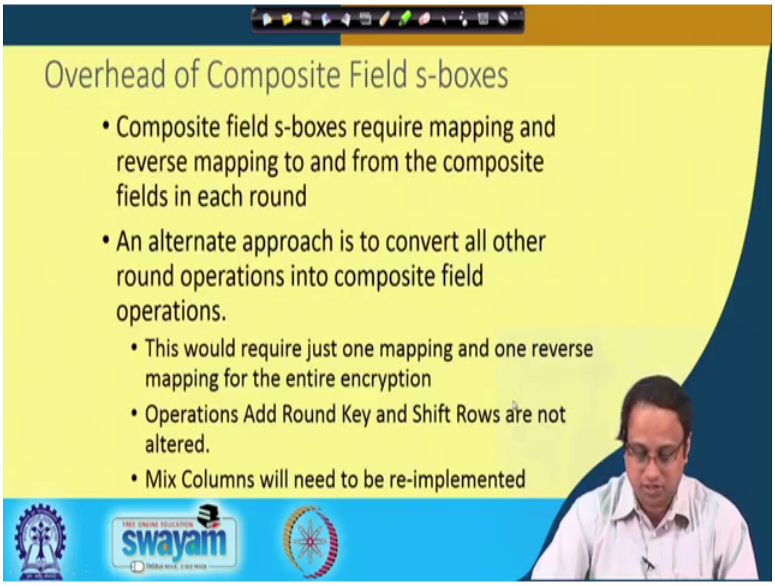
Type	XOR	NAND	NOR	NOT	MUX	Total Gates in terms of NAND gates using std cell library
InvS-box	81	34	6	0	0	182

D. Canright, A Very Compact S-box for AES, CHES-2005

So, now of course like when I apply the same freak for the inverse S-box, and here are some savings for the some cost for the inverses box. So, the inverse S-box when you are doing the inverse S-box, when a is not equal to 0, then you have to again apply the same trick.

But, you have to first apply the inverse affine, and then you have to do the do that do the inversion computations. When a is equal to 0, of course you have to do that 0 on the inverse affine. And then you have to calculate the corresponding inverse. So, when you are doing this inverse S-box in you know like composite fields, then again the gate count requirement in terms of NAND gates is around 182. And therefore, you have got similar amount of resource requirement for the inverse S-box as well.

(Refer Slide Time: 19:30)



Overhead of Composite Field s-boxes

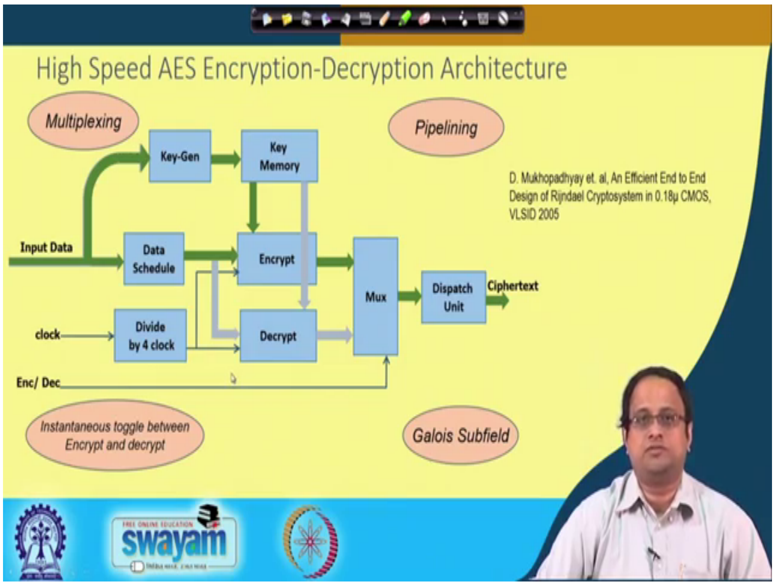
- Composite field s-boxes require mapping and reverse mapping to and from the composite fields in each round
- An alternate approach is to convert all other round operations into composite field operations.
  - This would require just one mapping and one reverse mapping for the entire encryption
  - Operations Add Round Key and Shift Rows are not altered.
  - Mix Columns will need to be re-implemented

So, therefore right when you are trying to adopt this technique for representing or implementing a composite field S-box, so remember that you have to do a mapping and reverse mapping from GF 2 the power of 8 to the composite field and back. So, an alternative so because of if you want to redo and you do it palace box computation ok, and therefore this conversion can add to a bottleneck.

If you really want to develop an ordinary approach where this bottleneck is alleviated one alternative approach would be to convert all round transformations into the composite field. So, you basically do a transformations at the only beginning, like when you have got the plaintext and the key, you convert the plaintext and the key to the composite field, and then write the entire AES in composite fields ok.

So, you basically design the entire AES in composite fields, and finally you get the result in GF 2 to the power of 4 square or your chosen composite field and in just do an inverse transformation to get the result. So, in that case that transformation over it is only at two point two places at the beginning, and an end, but you are not doing it per S-box or per computation ok. And that can significantly reduce the bottleneck. So, therefore these who just required one mapping and one reverse mapping for the entire encryption or decryption. And the operations like a drowned key and shift rows are of course not altered, but the mix columns have to be reinvented implemented like the S-box ok.

(Refer Slide Time: 20:48)



High Speed AES Encryption-Decryption Architecture

Multiplexing

Pipelining

Key-Gen

Key Memory

Input Data

Data Schedule

Encrypt

Decrypt

clock

Divide by 4 clock

Mux

Dispatch Unit

Ciphertext

Enc/ Dec

D. Mukhopadhyay et. al, An Efficient End to End Design of Rijndael Cryptosystem in 0.18μ CMOS, VLSID 2005

Instantaneous toggle between Encrypt and decrypt

Galois Subfield

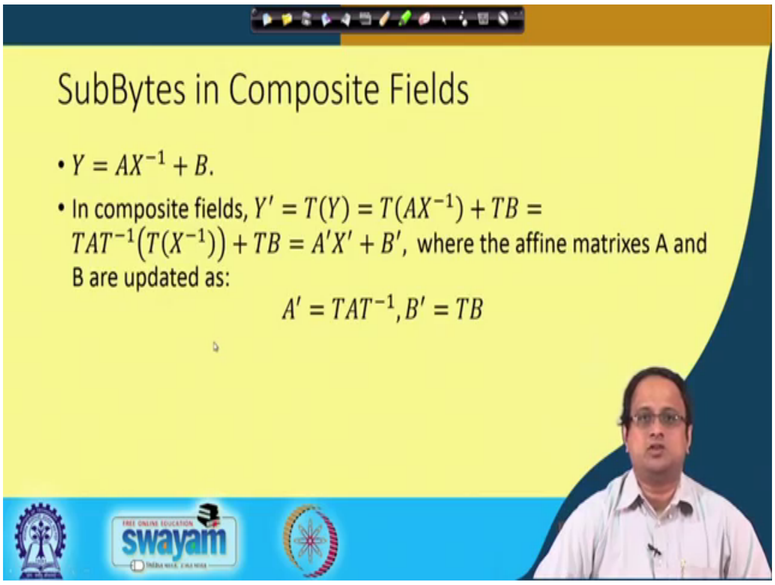
So, let us try to take a look it. So, so you know like about overall architecture in that case. So, when you are doing this, this is probably one architecture where you are doing both encryption and decryption on the same hardware. So, as I say that if you want to improve the throughput, then there are several techniques that you can try to adopt.

For example, you can try to multiplex the data the input data, so that you know like that so that it is because you remember that it is a pin count description. So, if you really want to save on to that, then you can multiplex the input to the encryption engine and with a decryption engine. You can also try to adapt pipelining, so that if you want to pipeline say the encryption, you need in the round by round manner. But, remember that when you are doing pipelining, yes the throughput will increase, but you will not be able to support sudden modes of encryption.

For example, right you can reflect on this point, then when you do a pipelining then there is no advantage, when you are wanting to have say a cipher block chaining or what is called as CBC ok. On the other end there may be some modes of encryption where pipelining would give advantage, so that depends upon what or how you want to use your AES block cipher, you can choose to apply pipelining or not to apply, but yes at the same time when you are applying pipelining, there is an increase in area, but there is also an advantage in terms of throughput.

Likewise, you we would also probably want an instantaneous toggle between the encryption unit and the decryption unit, and also as long as we discussed about the Galois subfield if you do that, then there is a compact implementation. So, so therefore right I mean you can actually you know like depending upon this you may probably want to you know like do these operations, so that when you are doing encryption, then essentially one part of the circuit gets active. But, when you are doing decryption, then there is another part of the then the modes get reverse, because now the input will be a cipher text, and the decryption you need should be able to decrypt on the fly and give you the result, which is your plain text. But, one important thing as I mentioned is that the entire thing should have now has to be represented as in the composite field ok.

(Refer Slide Time: 22:37)



SubBytes in Composite Fields

- $Y = AX^{-1} + B$.
- In composite fields, $Y' = T(Y) = T(AX^{-1}) + TB = TAT^{-1}\big(T(X^{-1})\big) + TB = A'X' + B'$, where the affine matrixes A and B are updated as:

$$A' = TAT^{-1}, B' = TB$$

So, so for example let us try to look into the sub bytes first. So, in the sub byte this is your transformation in the plain and simple way. You are doing a finite field inverse, then you are doing an affine mapping with A and B. So, these are your feed chosen vectors and I mean vector is B, and matrix is A. So, when you are trying to implement in composite fields, remember there is an initial transformation. So, therefore you take in your element Y, apply the T matrix which you are choosing or which you essentially they were up for converting your element 2 GF 2 to the power of 4 square say or GF 2 to the power of 2 power of 2 your chosen composite field. And then you get the result in Y dash, so you do your computation in Y dash ok.

So, therefore right what you want is this Y is nothing but if you plug in this Y equal to A X inverse plus B into this equation, then you are basically doing this right. You are basically doing A X inverse plus B, and then you are applying a T map on that. And remember that T is a linear mapping. So, I can distribute this or like this. So, I can get T A X inverse plus T B ok.

So, basically what I am trying to explore here is that I am trying to explore whether I can combine the, you know like the composite field inverse with the affine mapping in one single shot ok. So, if I do that, then you will see that I have got T A X inverse plus T B ok. So, what I can do is I can write this X inverse as T X inverse ok. And I can essentially bring in a T inverse factor and I can remove it by again multiplying with T. So, T inverse T is I ok.

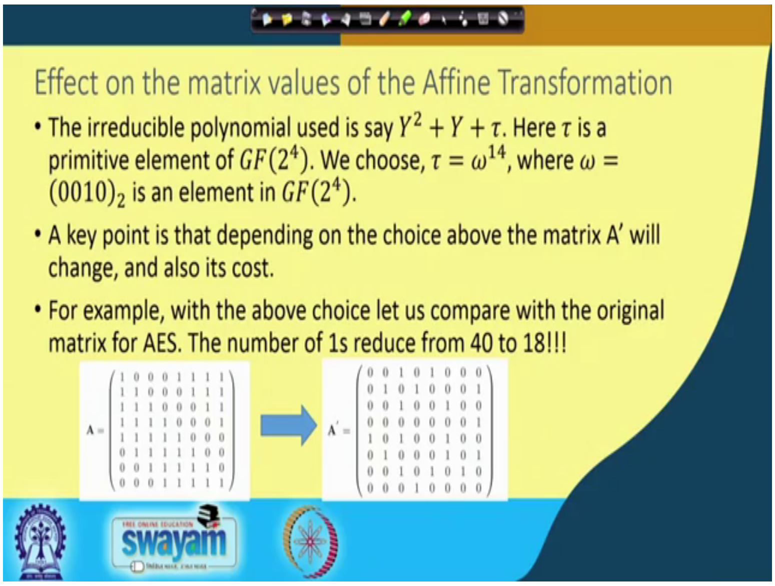
So, I can equivalently write this in this form. But, if I do that, then essentially you can see that of course, I have got T B also. But, then what I can do is now, I can think right that this tat inverse is my new matrix A dash is a new matrix A dash, and TB is a new vector denoted as B dash ok. So, now this internal part is what is T X inverse ok, so that means, it is nothing but I am doing the inverse X inverse, and then I am applying a T mapping onto it ok.

So, as me as if right this X inverse is an element in GF 2 to the power of 8, and then I have applied the T output or apply the T matrix on it, and converted into GF 2 to the power of 4 square, so that means now I can do my computation if my so what I can do now is that suppose my computation of the finite field inverse in this is in GF 2 to the power of 4 square ok.

So, rather than you know like bringing the result back and forth from GF 2 to the power of 8 to GF 2 to the power of 4 square or GF 2 to the power of 4 square back to GF 2 to the power of 8. What I can do is that I can simply compute the entire thing in GF 2 to the power of 4 squared that means, the S-box is in GF to the power of 4 squared.

And finally, when I am doing the affine mapping, rather bringing the result back to GF 2 to the power of 8, I do this computation in GF 2 to the power of 4 square. But, when I am doing the mapping in GF 2 to the power of 4 square, then I cannot apply A and B as my affine vector choice affine vector matrix choices. But, rather I have to replace this vector matrix choices by these matrices T A T inverse and T B that means, A dash and B dash ok.

(Refer Slide Time: 25:59)



Effect on the matrix values of the Affine Transformation

- The irreducible polynomial used is say $Y^2 + Y + \tau$. Here τ is a primitive element of $GF(2^4)$. We choose, $\tau = \omega^{14}$, where $\omega = (0010)_2$ is an element in $GF(2^4)$.
- A key point is that depending on the choice above the matrix A' will change, and also its cost.
- For example, with the above choice let us compare with the original matrix for AES. The number of 1s reduce from 40 to 18!!!

$$A = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \Rightarrow A' = \begin{pmatrix} 0&0&1&0&1&0&0&0 \\ 0&1&0&1&0&0&0&1 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&0&0&1&0&1 \\ 0&0&1&0&1&0&1&0 \\ 0&0&0&1&0&0&0&0 \end{pmatrix}$$

So, now once you have done that right then you can do this computation here, and there are some interesting observations that you can make at this point to understand the advantages of these chosen parameters. For example, the irreducible polynomial which I chose was Y squared plus Y plus tau. So, remember I made this one also he when he was discussing about the S-box.

So, here tau is a primitive tau is often a primitive element of GF 2 the power of 4, and we choose this tau as often say omega power of 14, where omega is say X which is you say essentially nothing but an element on GF 2 to the power of 4. So, what it means is just simply put these are some you know like some chosen parameters that I choose for this irreducible polynomial say r Y ok.

A key point here is that if you choose this for example, then these choices will actually have a bearance on these matrices A dash and B dash ok. So, interestingly here is a comparison, so this is your original matrix A, which I have which I had for the affine computation. And when I transform them that means when I transform them by using this equation T A T inverse, then my resultant matrix is as follows A dash.

If you compare the number of one's here, the number of one's is something like 40 and it has reduced to only 18 values ok, so that means now if I want to use gates for representing you know for implementing this matrix versus you know like implementing this matrix, you can easily understand that the number of XOR's will be significantly reduced ok. If I want to implement this 0 1 multiplication, I just need an XOR ok.

So, this you should try to you should be able to think about, I just (Refer Time: 27:30) you to think that if I want to implement this matrix with the vector multiplication, then the entire circuit will operate only with XOR's there is not no other gate, which is required. So, therefore here the number of gates, which will be required will be proportional to this number of ones in this matrix, which is 40 compare it with this with this implementation, where it will be proportional with only 18 ok.

And therefore, there will be a significant amount of saving, where you are you know where you are choosing the proper choices, when you know this is essentially making proper choices on this irreducible polynomial coefficients ok. And here is ones a nice choice ok, but there may be other bad choices as well. So, you need to explore that space and from there you should be able to you know like come up with a design strategy, where you essentially optimize the affine transformation in a much more effective way ok.

So, for the sake of time I will stop here ok. And we shall continue with the AES descriptions in the next class, where we will first try to you know like look into the overall AES design. And in particular, I will be trying to look into the S-box in slightly more details so ok.

Thanks for watching.