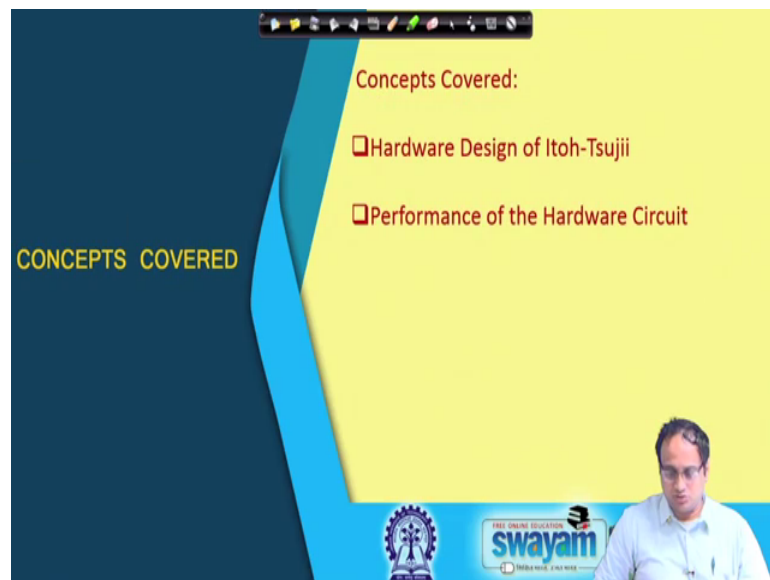


Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 07
Hardware Architecture for Finite Field Inverse

So, welcome back. So, today I like what we have discussed about the Itoh-Tsujii inversion algorithm. So, what we will do today is that we will try to see about a corresponding hardware architecture for it ok. Like suppose you know like once we have understood the algorithm, this is a very general problem that we will encounter is that we have to now translate that algorithm into hardware. So, we have already seen a flavor of that when we discussed about the GCD algorithm and we thought about how we can discuss into the data path and the control path so that was an example. So, here is a more I would say specific case where we are trying to implement a finite field circuit which is often used in several cryptographic combinations.

(Refer Slide Time: 00:55)



So, to start with right let us so this is what we will discuss today. Like we will discuss about the hardware design of Itoh-Tsujii, and we will discuss about the performance of the hardware circuit like how we can choose the parameters, so that the performance is kept nice ok. So, whether it is optimal in terms of its performance.

(Refer Slide Time: 02:15)

Estimating the number of clock cycles

- #Multiplications: $l-2+3=l+1$. Note that 2 multiplications are needed initially for pre-computation, and one for final squaring.

$$\# \text{Clock - cycles} = (l+1) + \sum_{i=2}^{l-1} \left\lceil \frac{u_i - u_{i-1}}{u_s} \right\rceil$$

- Difference in clock cycle: $\left\lceil \frac{u_i - u_{i-1}}{u_s} \right\rceil - 1$

This can be as high as $(m-1)/2$ for $GF(2^m)$

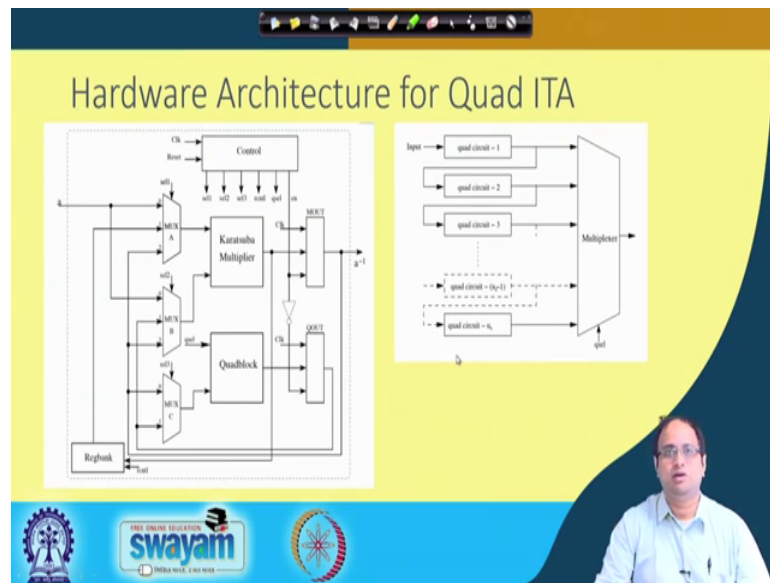
swayam

THE ONLINE EDUCATION

INDIA WISE. LEAD WISE.

So, what we can do here is that if we again observe the clock cycles just to recapitulate about that. You have essentially got 1 plus 1 number of clock cycles, because you have 1 minus 2 to do the multiplication to the 1 minus 2 essentially is what you do here is in this table, but then you have got two extra multiplications at the top to calculate this pre computation plus the extra multiplication to do the squaring that is three multiplications. So, therefore, we have got 1 minus 2 plus 3, so you have got 1 plus 1 number of multiplications, so that essentially is the clock cycles accounts for this clock cycle value. Here 1 plus 1. And this is this summation is for the remaining parts, where you are doing u_i minus u_{i-1} , you are dividing by u_s . But note that now we since we are doing quadding instead of squaring, we also need a corresponding architecture where we do not have a cascade of squarings, but we have a cascade of quads.

(Refer Slide Time: 03:09)



So, therefore, right I mean here is a corresponding architecture for that. So, let us take some time to understand how these architecture works. So, again we have got a data path and we have got a control path. So, you can see that this is my control circuit denoted as control, whereas this part essentially is my data path. So, in the data path, what are the two most important operations that we need? Here we need a multiplier, because we have essentially we need several multiplication operations. So, we need an optimized multiplier ok; we need an efficient multiplier.

So, in order to do that we will we can use the Karatsuba multiplier, but we designed in the last class, and then we also need a quad block. So, how do you realize a quad block? So, you can observe that when you are doing quadding in GF 2 arithmetic then you are basically doing squarings followed by squarings ok. So, now, squaring as we discussed right is a linear operation in GF 2 to the power of m arithmetic ok. So, therefore, right you can actually realize a squaring operation by just using only XORs, you do not need any AND gates to realize that ok. Moreover you can if you want to have powers of 2, like powers of 2 means like 2 squared, 2 cube and so on all of them can be realized as that ok.

So, basically it is just by using exhaust you can realize it. So, you can actually have a very efficient architecture for the quad block. And as we discussed right that the quad block essentially write in terms of delay is kind of equivalent to the squaring circuit ok.

So, it is probably more optimal or better at least it utilizes the look up tables more efficiently.

So, now, we see that we have got the multiplier and there are two inputs to the multiplier, but the one which we will be choosing like which one which you will say or the data that you will be sending as an operand to the multiplier is being selected by a multiplexer. So, there are two multiplexers at the input of the multiplier. And likewise right for the quad block whatever you want to compute, whatever inverse you want to compute that data is also being selected by a multiplexer ok. So, you have got three multiplexers here ok.

There are two registers at the output. So, for example, I register the multiplier by using an amount registered. Likewise I register a quad block and I denote it as Q out. So, note that in my architecture here, I will be either using a multiplier or I will be using a quad block ok. So, therefore, in what essentially what I mean to say is that I will be enabling either this register or I will be enabling this register ok. So, therefore, at any clock cycle, I will be either calculating this value here, or I will be either using this value, or I will be using either this value.

So, you can note that the control generates an enable signal, and the enable signal passes here, but it is toggled here, that means, if it is 1, then it is selecting this register. But if it is 1, then this is set at 0. Likewise if it 0, then this one is selected, and this one is not selected ok. So, therefore, I am basically switching between the multiplier and the inversion ok.

And then if I just want to take a look into my quad block as we have seen in the square at circuit right, what we will be having here is we will be having a cascade of quad circuits. So, here again I have got u s quad circuits which I have cascaded all of them are quads individually. And then there is a multiplexer at the output of these q s stages, that means like if I if the number of quaddings that I want to do is less than the number of stages here u s then I can multiplex that that output correspondingly ok.

However, if the number of quaddings is more, then I need to take this output, feed it back and I need to cycle it. So, I need to spend more than one clock cycles to do that operation ok. How many clock cycles, exactly as we have seen the number of clock cycles divided by u s, and I take a $c \log$ that ok, so that gives me the approximation of the number of clock cycles that I will encounter or incurred in that case ok.

So, now once you have essentially understood the data path, so you also need to and also understand that the data right essentially which you are computing is essentially going through transition that means you are essentially continuously modifying those data ok. So, therefore, the data that you essentially are encounter or computing on essentially stored often in the form of a register bank. So, there is a register bank where you have registers where you are storing them.

Now, the number of registers which you essentially will be having in the register bank will actually depend upon your choice of the addition chain ok. For example, you will see that in the addition chain, if you need to have you know like when you are calculating the alphas right, and you need to resort or you know you need to take an alpha which previously was there or previously was used then that data should be available right because you do not want again to compute that ok, otherwise the entire advantage will be lost. So, therefore, that old data needs to be stored in the register bank and needs to be suitably addressed to get that value ok. So, therefore, that depends completely upon their addition chain ok.

(Refer Slide Time: 08:03)

Control Word Design

The diagram shows a control unit with inputs clk , $reset$, and en . It outputs control signals $sel1$, $sel2$, $sel3$, $sel4$, $sel5$, $sel6$, $sel7$, $sel8$, $sel9$, $sel10$, $sel11$, $sel12$, $sel13$, $sel14$, $sel15$, $sel16$, $sel17$, $sel18$, $sel19$, $sel20$, $sel21$, $sel22$, $sel23$, $sel24$, $sel25$, $sel26$, $sel27$, $sel28$, $sel29$, $sel30$, $sel31$, $sel32$, $sel33$, $sel34$, $sel35$, $sel36$, $sel37$, $sel38$, $sel39$, $sel40$, $sel41$, $sel42$, $sel43$, $sel44$, $sel45$, $sel46$, $sel47$, $sel48$, $sel49$, $sel50$, $sel51$, $sel52$, $sel53$, $sel54$, $sel55$, $sel56$, $sel57$, $sel58$, $sel59$, $sel60$, $sel61$, $sel62$, $sel63$, $sel64$, $sel65$, $sel66$, $sel67$, $sel68$, $sel69$, $sel70$, $sel71$, $sel72$, $sel73$, $sel74$, $sel75$, $sel76$, $sel77$, $sel78$, $sel79$, $sel80$, $sel81$, $sel82$, $sel83$, $sel84$, $sel85$, $sel86$, $sel87$, $sel88$, $sel89$, $sel90$, $sel91$, $sel92$, $sel93$, $sel94$, $sel95$, $sel96$, $sel97$, $sel98$, $sel99$, $sel100$. The diagram also shows a Register Bank, Karasuba Multiplier, and Quadblock.

Step	Clock	sel1	sel2	sel3	sel4	en
$\alpha_1(n)$	1	0	0	x	x	1
$\alpha_2(n)$	2	0	2	x	x	1
	3	x	x	0	1	0
$\alpha_3(n)$	4	1	1	x	x	1
	5	x	x	0	1	0
$\alpha_4(n)$	6	1	1	x	x	1
	7	x	x	0	3	0
$\alpha_5(n)$	8	2	1	x	x	1
	9	x	x	0	1	0
$\alpha_6(n)$	10	1	1	x	x	1
	11	x	x	0	7	0
$\alpha_7(n)$	12	2	1	x	x	1
	13	x	x	0	14	0
$\alpha_8(n)$	14	2	1	x	x	1
	15	x	x	0	1	0
$\alpha_9(n)$	16	1	1	x	x	1
	17	x	x	0	14	0
$\alpha_{10}(n)$	18	x	x	1	14	0
	19	x	x	1	1	0
$\alpha_{11}(n)$	20	2	1	x	x	1
	21	x	x	0	14	0
$\alpha_{12}(n)$	22	x	x	1	14	0
	23	x	x	1	14	0
$\alpha_{13}(n)$	24	x	x	1	14	0
	25	x	x	1	2	0
$\alpha_{14}(n)$	26	2	1	x	x	1
$\alpha_{15}(n)$	27	2	2	x	x	1

Logos for IIT Bombay and Swamyam are visible at the bottom.

So, now I would like to design the control word. So, if I want to design the control word this is how what I do here is. So, essentially right what I start doing is I start to you know like calculate the multiplications and the inversions one by one ok. So, if you remember right, what we started with was we started with calculating this alpha 1 ok. And alpha 1

was initialized to a cube ok. So, if you remember right alpha one was initialized to a cube.

So, if you go back right, you will see that alpha 1 was initialized to a cube, a to the power of 3. So, how we how can we calculate this a to the power of 3 is being elaborate here in terms of the clock cycle. So, as I said that you are basically spending two clock cycles here. And what do I do is first, I do a multiplication and then I get the corresponding result which is a squared. I again feed this back and I calculate a to the power of 3 ok. So, what I do first is that I first need to ensure that the multiplier here gets a that means, this select line should select 0 that is why the select 1 is set to 0.

What about this multiplexer? I again need to take this a. So, I again want to make this select also 0. So, this select is also set to 0. So, now I get a square. So, the select 3 and the essentially does not matter. So, I have set don't cares here ok, because what I allow it to optimize more, because the choice of these values does not matter in terms of functionality. So, I can I should set them as don't cares. And then finally, write this en is set to 1 that means, I am essentially using the result of the multiplier output ok. So, therefore, I get a square here. And now what I do is that I take this a square and I would like to feed that back right, because I would like to compute a square multiplied with a.

So, therefore, in the next clock cycle, you will see that select 1 is still set to 0 which means select 1 being 0, it takes in a. So, therefore, this multiplier has got one of its operand as a in the next clock cycle, whereas in the next clock cycle the second multiplexers select line has been set to 2 ok. Now, if it is set to 2, then that means, I am choosing this line ok. So, I am choosing this line which is nothing but the output which I have derived from here ok. So, therefore, I have got square and if I am multiplying squared with a, so I am multiplying a square with a to get a cube. Again you see that the en is being set to 1, because I am again observing this output ok.

Now, what about alpha 2 a? So, in alpha 2 a again if you observe right if you go back and see the computation of alpha 2 a. You basically are doing this you basically are multiplying alpha 1 and you are you are essentially passing alpha 1 and raising it to the power of 4 and you are multiplying this with alpha 1 ok. So, therefore, that therefore, alpha 1 needs to be sent to a quad circuit and the result needs to be multiplied with alpha 1. So, therefore, what do I do in the next clock cycle is I essentially raised alpha 1 to the

power of 4 ok. So, therefore, this data I essentially get from the registered bank ok, and therefore, what I do is that I take this data and I send it to the to the quad block ok.

So, now if you observe here that in this third clock cycle select 1 and select 2 are not important, because the multiplier is not working here, rather what is important to select three. So, in select 3 right I have set it to 0 which means right I am getting these data, so that essentially right I am this is the corresponding data which I am passing it. And this data essentially is being taken from if you observe this line is being taken from here ok, so that means, right here I had a cube calculated.

So, this a cube data is directly being fed back to the quad block ok. And once it is being fed back to the quad block, I need to observe the output in q out. So, therefore, q out right essentially is where I should get the corresponding output data ok. And therefore, right I essentially also make q select 1 and I get the corresponding result ok. Likewise right you will see that the enable signal is set to 0 here because if the enabled signal is set to 0 here, this being 0, this becomes 1.

So, therefore, this becomes functional ok. So, in the next clock cycle, that means, in the clock cycle 4, I will multiply this with a right. So, therefore, what I have to do is, I have to take this and I have to multiply this back. So, therefore, what I do here now is again I get I activate my select 1 and select 2 signals. So, therefore, I activate and I make both of them 1 so; that means, like this ways 1 as well as this is 1. So, you see that if this is 1, then that means I am getting it from the I am if you observe you trace this back, then I am getting this from here ok. So, I am getting this from here and the other data I am getting from the registered bank ok.

So, therefore, I am now multi I am essentially multiplying and I am getting alpha 1 to the power of 4 and I am multiplying that with alpha 1. So, essentially right we what we have is. So, if you observe right I mean this is clear right. So, you have got one of the input from the registered bank, whereas the second input from the multiplexer is actually taken from the output of the quad block. So, you see that if you follow this line right, then this essentially is from the quad block ok.

So, therefore, this is the data where you had alpha 1 to the power of 4. So, now, you are multiplying this with alpha 1 to get the corresponding next output. So, exactly this is what has been done subsequently. So, you basically again need to calculate alpha 3, you

need to calculate alpha 6, alpha 7 and so on. And essentially finally, alpha 116, so you can see that in alpha 116, you have to do several quaddings. So, you will see that whenever this enable signal is 0 that means, you are doing the corresponding quadding ok.

So, you are doing so many quaddings and finally there is a multiplication. For every section, you see that there are some quaddings followed by a multiplication ok. So, there is some squaring follow some quaddings followed by a multiplication ok. And finally, there has to be a squaring that means, the final result has to be squared, and this is done by a multiplier. So, therefore, again I am selecting 1, selecting 2 setting them to 2, which means I am essentially taking my prior result and I am just multiplying there ok.

So, again the multiplier enable is high, that means, I am using the result of the multiplier. So, you can observe that for all these computations, I have used the multiplier ok. So, therefore, right what essentially probably is the next thing that we need to think of is that once we have made a functionally correct design is how do we analyze the performance of it. So, therefore, the first thing which is important here is to understand what is the critical path of your circuit ok. And you will see that most often write the critical path will pass through the multiplier, because the multiplier is the most costly step that you encountered ok. And anyway right there is one clock cycle or there is one event when you are doing only a multiplication ok.

So, therefore, the critical path of your design is kind of constrained by the multiplier ok. So, you need a good multiplier to make your design efficient ok, and that is why we were starting good multipliers in the previous class. But at the same time it is also important to realize that how do you design the quad block right essentially should also be taken into account. So, therefore, if you make a very erratic design of the quad block, then it may happen that you make your critical path worse than that method of the multiplier so that should be a criteria for setting the number of cascades that you are doing.

(Refer Slide Time: 15:29)

The slide is titled "Number of Cascades" and contains the following bullet points:

- Number of quad circuits (u_s) has an influence on the clock-cycles, frequency, and area requirements.
- Decreasing u_s would reduce area, but also increase clock cycles.
- Let a quad circuit require l_p LUTs and have a combinational delay of t_p .
- A cascade of u_s quad circuits would require $u_s \cdot l_p$ LUTs and have a delay of $u_s \cdot t_p$.
- In order that the quadblock does not alter the frequency of operation u_s should be selected such that $u_s \cdot t_p$ is less than the critical path, which is of the multiplier circuit.
- So, we make $u_s \cdot t_p$ close to the multiplier delay.

The slide also features a video inset of a man speaking in the bottom right corner, and logos for "swayam" and "THE ONLINE EDUCATION" at the bottom left.

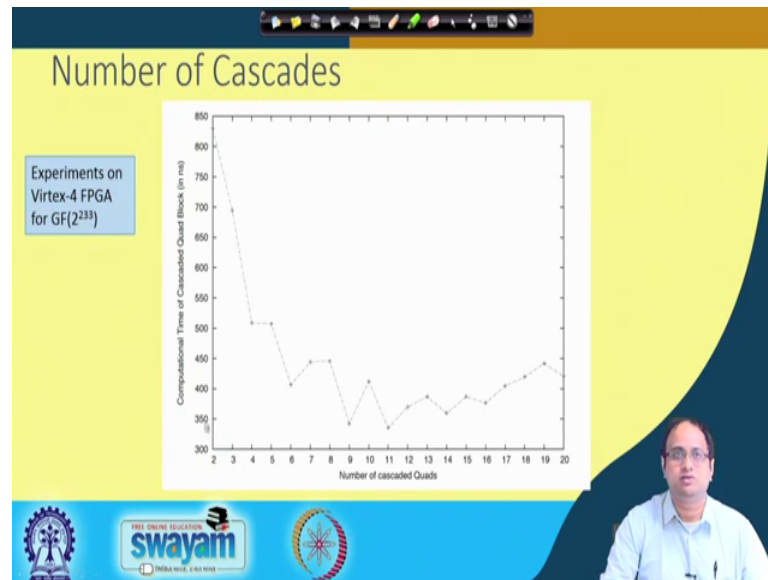
So, more seriously right that I mean this is a very important step that how do you decide the number of cascades and that will essentially make your performance, make the performance or set the performance of your design. For example, if the number of quads I mean the number of quad circuits, if you denote it as u_s , it will have any an influence of the clock cycles, frequency and area requirements. You can see that if I increase the number of quad operations, number of in increase the number of cascades, then the area requirement will increase ok. At the end, at the same time right it will also reduce the number of clock cycles.

But if you decrease it, then the number of clock cycles with increase, but the area will get reduced ok. So, there is a trade off. So, suppose you know like there is a quad circuit like one quad circuit which will take say l_p number of lookup tables, and there is a combinational delay of say t_p ok. So, therefore, right if I want to formalize the number of if there are u_s cascades, then what is the number of lookup tables that we will be consuming it will be just u_s multiplied by l_p and the delay will also be proportional to u_s into t_p ok.

So, as I said that since your multiplier is a very crucial block, and you do not want to make the delay what is then that of the multiplier. So, you will ideally like to keep u_s into t_p less than that of the multiplier critical path ok. At the same time you do not want to make u_s very small, because if you make u_s very small than number of clock cycles will

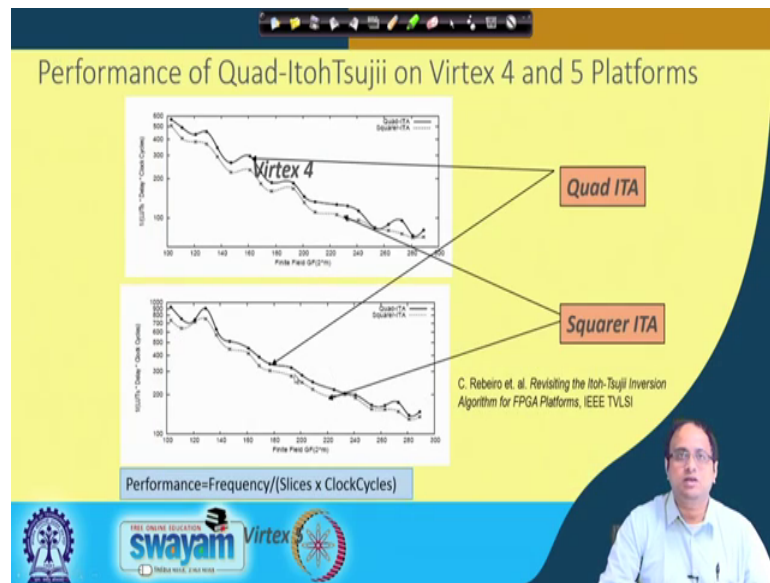
increase ok. So, therefore, the optimal way of design strategy would be probably to keep us in to t p close to that of your multiplier t s slightly less, but not more. So, this is the criteria that we followed.

(Refer Slide Time: 17:05)



And if you follow this you will see that and here is a plot to demonstrate that, if I increase the number of cascades that you are doing that is u s for example, and if you implement say the GF 2 to the power out 233 on a vertex-4 FPGA, this is probably how your graph will look like ok. On the y-axis, I have plotted out the computational time of the cascaded quad block. And you will see that there is if you make the cascade another number of cascade steps very small then also the time will be more, because you are spending more clock cycles. Whereas, if you make it high then also you are spending more there is a more delay the delay ok. And therefore, there is an optimal point for which you get more efficient designs. And you will see that the number of cascades here there is a region like something like 9 to 12, and we set something as 12 for example for our design ok.

(Refer Slide Time: 17:55)



So, therefore right if you take this and if you want to make or see the performance, you will see that with properly chosen such design parameters, and if you plot the quad Itoh Tsuji inversion with the squarer Itoh Tsujii inversion algorithm, you will find out that therefore all the field sizes ok. So, we these are kind of a plot where on the x-axis, you have got different field parameters ok. You will see that there is not always a significant advantage of using the quad with Itoh Tsujii algorithm over the square or Itoh Tsujii algorithm for LUT based FPGAs ok.

How did we get the performance? So, normally right the way that we basically calculate the performance is as follows. We divide the frequency, divide the slices and the clock cycles ok. So, therefore, right what my objective as the designer would be to improve performance which means that I should be able to optimize my design at a higher frequency, that means, the critical path of my design should be small. And also if I want to increase my performance, a number of clock cycles should be small; and if I want to improve my performance, my slice requirement should be small.

So, it is a nice tradeoff between the number of resource that we encounter or in use the number of clock cycles that you need and also the frequency at which your design operates ok. So, therefore, what we plot here is that performance in terms of different field size parameters, and we see that for all the field sizes this properly designed quad Itoh Tsujii algorithm should give an advantage compared to a squarer Itoh Tsujii

algorithm. Of course, right you can get into more details about trying to understand why quad and why not octet for example ok. And essentially also decide upon the you know like the corresponding tradeoffs which can encounter because of that. For time purpose, I will not go into all those tradeoff issues.

(Refer Slide Time: 19:41)

Performance of Inversion Hardware on Xilinx Virtex E

Implementation	Algorithm	Platform	Field	Slices	Frequency (MHz)	Clock Cycle (c)	Computation Time (c/f)	Performance (Equation 5.33)
Dormale [107]	Montgomery	XCV2000E	160	890	50	-	9.71 μ sec	115.7
		XCV2000E	256	1390	41	-	18.7 μ sec	38.4
Crowe [101]	Montgomery	XCV2000E	160	1094	51	-	6.28 μ sec	145.5
		XCV2000E	256	1722	39	-	13.17 μ sec	44.1
Henriquez [328]	ITA	XCV3200E	193	10065	21.2	27	1.33 μ sec	78
Henriquez [325]	Parallel ITA	XCV3200E	193	11081	21.2	20	0.94 μ sec	95.7
		XCV3200E	193	11911	36.2	20	0.55 μ sec	152.1

But rather you know like I will redirect you to my to the textbook that we have.

(Refer Slide Time: 19:43)

References:

- Debdeep Mukhopadhyay and Rajat Subhra Chakraborty, Hardware Security: Design, Threats and Safeguards, CRC Press

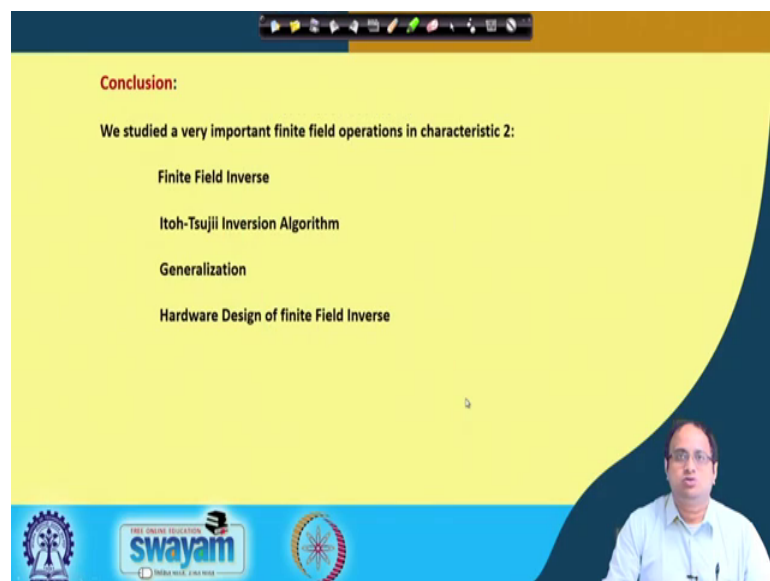
Hardware Security: Design, Threats, and Safeguards
 Debdeep Mukhopadhyay
 Rajat Subhra Chakraborty

You will find more details about those tradeoffs in the in the book. And you can read and try to gather them in the more importantly I will tell you about the performance that we

can say that the inversions will have on say as Xilinx Virtex E. Because it is important as an engineer to know the figures right like for example, if I ask you like if I take a quad Itoh-Tsujii algorithm. And if the dimension is say 193, what would be the order of delay which a hardware will look on say on Xilinx Virtex ok.

So, here the delay is something like 0.55 microsecond ok. So, you can compare it with probably with the software design. And you will see that there is a significant amount of advantage which you get when you resort to hardware designs. In this case, we also compare with some other competing designs and show that this design is optimal, but what is more important to understand is that for a properly chosen design and for a properly architected design, you essentially get and get a significant amount of improvement when you are going into a parallel hardware architecture which is properly done ok.

(Refer Slide Time: 20:47)



Conclusion:

We studied a very important finite field operations in characteristic 2:

- Finite Field Inverse
- Itoh-Tsujii Inversion Algorithm
- Generalization
- Hardware Design of finite Field Inverse

swayam
MOOC SWAYAM

So, with this I will stop here. So, what we studied essentially is a very important finite field operation in characteristic 2, which is essentially called as a finite field inverse which is often used in several other computations. We will see them being used largely in the elliptic curve hardware that we will studying subsequent to this. We discussed about an efficient algorithm which is called as Itoh-Tsujii inversion algorithm which is essentially relies upon Fermat's little theorem.

We also generalized it to a version where we can actually do where you can speed up your computation. You can also try to use the resources in the FPGA much more efficiently. And finally, we also discussed about a hardware design of the final field inversion algorithm and discussed about the performance issues like how do you choose the parameters. So, you so that you get an optimal performance of your hardware ok.

Thank you for your for your time. And we will continue our discussions in the next class.