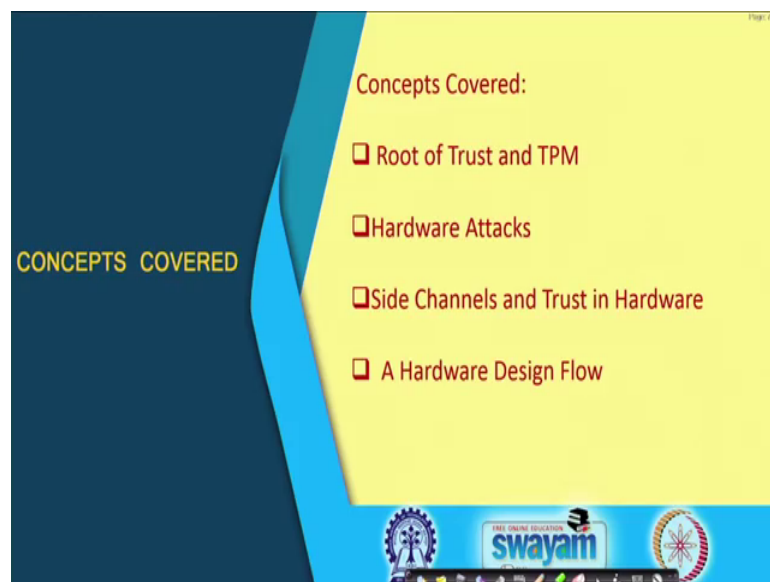


Hardware Security
Prof. Debdeep Mukhopadhyay.
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 02
Introduction to Hardware Security Part – 2

Start. Welcome, to the second lecture on Hardware Security. So, as in the last class we were introducing ourselves to the topic of hardware security. So, I will continue with that and try to see some more implications on hardware security in the present day context. So, I will start with the elaborations on the concepts that we will be covering in today's lecture.

(Refer Slide Time: 00:45)

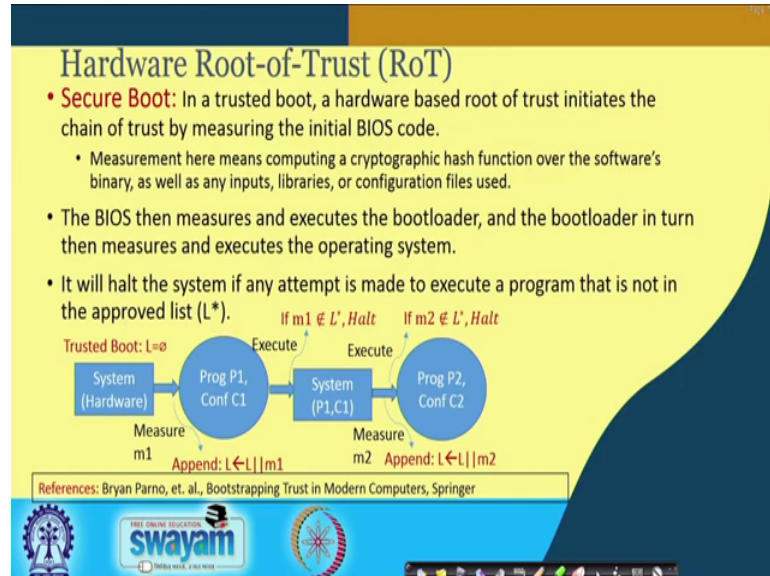


So, we will be trying to see about root of trust which is one of the fundamental objectives of hardware security like we try to develop a root of trust using the technology of hardware. And in this context we will be discussing about something which is called as trusted platform modules or TPMs.

So, we will also discussing about hardware attacks in this context, like some of the attacks which we are mounted on TPMs and essentially have challenged the opportunities which hardware in general brings in. We will be also touching upon side channel analysis and also talk about in general about trust in hardware. And finally, we

will be discussing about a hardware design flow like trying to understand how we can eventually design hardware using our basic CAD tools.

(Refer Slide Time: 01:31)



So, to start with let us take a look at the developing a root of trust which is a very fundamental I would say fundamental objective of bringing in hardware is that we try to we develop a you know like a very minimal component which is called as the root of trust or RoT. So, in this context there is a very important objective like you would like for example; suppose, imagine that you have a computing system and you would like to boot your computing system, how do you know that the operating system that you are loading right essentially is trusted.

Because many of the attacks we know essentially which happens on software happens because of the fact is that the operating system is itself compromised. So, even if you know like run your software which essentially has for example, cryptography in it right will not be enough because the OS on which the software is executing is itself compromised. So, in order to address this there is a concept of secure boot which I will try to introduce here.

So, in trusted boot there is a basic component which is which is essentially hardware. So, if the basic root of trust is hardware and it tries to initiate a chain of trust, ok. So, the idea is that there is a small component which is essentially your system hardware as shown here. It tries to essentially you know; like basically what it tries to bring in is; it tries to

bring in or it tries to develop or basically like this is the starting point of your root of trust, ok. So, it tries to measure or bring in the initial BIOS code, but before it executes the BIOS code it does a measurement, ok. So, this measurement is trying to understand that whether the BIOS is legitimate or not ok. So, what essentially is measurement can be discussed, but let us try to get what all idea, ok.

So, what we try to do here is or the root of trust or the hardware root of trust has to do is it tries to initially measure the initial BIOS code and then. So, the measurement could typically imply very simply take the program take it is configuration and compute a cryptographic hash on it, ok. So therefore, what it tries to do is it tries to develop this measurement or calculate this hash output and let us call it as a measurement of the BIOS, and once the measurement is done it is appended to a chain of trust.

So, we append this and if this measurement does not belong to an approved list; this approved list is denoted as L star then the process itself halts at that point, ok. So, the processor itself stops at that point and it does not allow it to go further. On the other hand if the measurement is fine. That means, if the measured output belongs to an approved list then we continue. So, then we bring then we go into the next layer where we basically. So now, the BIOS comes into the picture the BIOS takes control and the BIOS starts to measure and execute a boot loader again the same process continues. That means, this measurement is done the measurement is appended. If the appendment is fine and if the appending is fine then it is allowed otherwise it is it halts, ok.

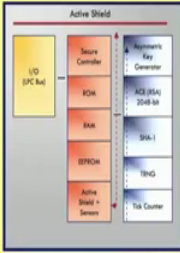
So, finally, the operating system comes into the control. And, the idea is now, therefore, that if you go behind right, if you consider this chain of trust then the entire trust is built upon this assumption that the hardware which is your root of trust is pure, is essentially trusted, ok. So, that is essentially the reason why you know like hardware is very promising because it can give you a very small component which you can trust, which you can believe and it way and if this works fine, right then virtually you have a very low cost an efficient way of building trust to your systems, ok.

So, here is a reference which you can look into it gives a very nice description about several other things related to secured boot and also TPMs in general, ok.

(Refer Slide Time: 05:47)

Trusted Platform Modules (TPM)

- **Definition:** "A Trusted Platform is a computing platform that has a trusted component, probably in the form of built-in hardware, which it uses to create a foundation of trust for software processes." -S. Pearson.
- I/O: Low Pin Count Bus (LPC)
- Secure Controller: Controls internal TPM execution flow and verifies commands
- EEPROM: Stores TPM Keys [Endorsement Keys (EK), Storage Root Key (SRK)], Other data like Owner Authorization Data, EK Certificate
 - The EK and SK never leaves the IC boundary.



The diagram shows the internal components of a TPM chip. On the left, a yellow box labeled 'LPC Bus' is connected to a central vertical stack of components: 'Secure Controller', 'SRK', 'EK', 'EEPROM', and 'Active Shield - Sensor'. To the right of this stack is another vertical stack of components: 'Random Bit Generator', 'PCR (SHA) Seed In', 'SRK-I', 'TRNG', and 'Tick Counter'. The entire diagram is titled 'Active Shield' at the top.

References: http://www.cs.unh.edu/~it666/reading_list/Hardware/tpm_fundamentals.pdf

swayam

So therefore, this essentially brings into this concept of trusted platform module or TPM. So, TPM is basically a technology which essentially right is tries to bring in trust into your system and it is a standard. It is a standard of trying to build in trust in your system using this concept of minimalistic root of trust, which is built in hardware.

So, the idea is that trust you get there are many definitions of their many definitions of TPM, like here I have in listed one of the popular definitions which says that a trusted platform module is a computing platform that has you know like a trusted component probably in the form of built in hardware. Just like what we saw in the previous slide which it uses to create a foundation of trust for your software processes, ok.

So, there are different components of a TPM chip. For example, here is sort of a diagram which tries to capture how a TPM chip looks like. So, I will just try to show you the most important components. So, one of them is of course there is a LPC bus. So, it is a low pin count bus and which essentially does the input output. So, it does the I O with the external world.

Then there is a secured controller. So, this controller controls the internal TPM execution flow and verifies the command. Like suppose you give a command to that TPM chip the commands are often encrypted and they need to be verified to know that they are legitimate commands, ok.

So, then there is an EEPROM which basically stores several keys. So, in TPM there is a management of several keys and I have just enlisted two of the most important ones. So, one of the one is called as an endorsement key and the other one is called as a storage key or SRK, ok. And, there are also other important keys like owner authorization data and EK certificate which are also maintained and stored inside the TPM, ok.

So, you may note here that the EK and the SK or the EK and the SRK never actually leaves the IC boundary. So, this essentially remains embedded and in particular the private component of this. So, these are typically based on public key cryptography. So, they have a public key and a private key component. The private key components of these keys like of the endorsement key and so on the SRK never leaves the IC boundary. So, it leaves it basically remains inside the TPM chip and is never externally exposed, ok.

So, this is a feature of the TPM chip, and there are some criticism against that also because of the fact that even the user who is using the TPM chip is not aware of these keys, ok. So, this can lead to some privacy allegations and things like that. But, at least from security point of view this seems to be a very you know like a same solution where the idea is that the chip essentially is kind of storing the secret key, and this secret key component is never externally re-built, ok. So, even the user of the TPM chip is not aware of this these keys, ok.

(Refer Slide Time: 08:51)

TPM Cryptographic Hardware

- **Asymmetric Key Generation:** RSA Key Generator, Supports 1024-2048 bits
- **RSA Encryption Engine:** RSA operations
- **SHA-1 engine:** Hashing to measure integrity stored in Platform Configuration Registers (PCR).
 $PCR \leftarrow \text{hash}(PCR || \text{hash}(\text{newcode}))$.
- **Random Noise Generator:** True random generator to be used for cryptographic operations
- **Tick Counter:** Provides an audit trail of TPM commands
- **Security Features:** Active Shield, Voltage fluctuation detectors, high frequency sensors, Reset filters, etc.

References: http://www.cs.unh.edu/~it666/reading_list/Hardware/tpm_fundamentals.pdf

The diagram shows the internal components of a TPM chip, including the Active Shield, Secure Controller, ROM, SRK, EEPROM, Active Shield Sensor, Random Noise Generator, RSA Key Generator, PCR (SHA-1) Storage, SRK-1, TRNG, and Tick Counter. A video inset shows a man speaking.

So, that is essentially the basis of this you know like the of the trust which is built using TPMs, and the TPM right as you understand does lot of cryptographic operations. And because of these there are some cryptographic components which are already being in built in to the TPM chip, ok. So, for example, it has got a support of a 1024 bit or 2048 bit RSA key generation as well as RSA encryption. So, it can do both key generation as well as encryption, ok.

Then, there is a SHA – 1 engine. So, this is a hash engine, ok. So, there is a hash function as I said that when you are doing the measurements you are doing hash computations, right. So, therefore, what you are typically doing is that; so, there is a platform configuration registered and these stores the stores the current calculations like the current measurements, ok. So, the measurements are usually done in this way like you take an initial PCR and then you do an hash of a new code. So, you kind of do something like an eternity of hash function. So, we apply the hash function one after the other and you try to compute these digest and you try to keep it inside your inside your PCR, ok.

So, then one of the fundamental requirements of cryptography is that it requires or is based on random number generators, ok. So therefore, a TPM also has got a small hardware or a hardware for doing or calculating what are called as TR engines or true random number generations. So, there is a true random number generator which is inbuilt into this. Then there is a tick counter which is essentially provides on audit trail of the all the TPM commands which are arriving at the TPM. And, the idea is that this is a you know like a built-in where in with lot of security features so that you cannot do you know like active attacks like. So, it is got an active shield. It has got voltage fluctuation detections and the high frequency sensors reset filters and so on.

So, these are basically I would say more of like safety measurements which are being kept in the TPM chips, ok.

(Refer Slide Time: 10:41)

TPM 2.0

- **TPM 2.0 is not backward compatible.**: RSA Key Generator, Supports 1024-2048 bits
- Specification requires SHA-1 and SHA-256
- Elliptic Curve Cryptosystems (NIST P-256, Barreto-Naehrig 256-bit curves)
- 128-bit AES encryption
- Many other algorithms are also defined.

References: https://en.wikipedia.org/wiki/Trusted_Platform_Module#TPM_1.2_vs_TPM_2.0

swayam

So, you know like having said that TPM is also upgrading. So, the one which I told before was more of like TPM what is called as TPM 1.2 which is getting operated into TPM 2.0 and note that TPM 2.0 is not backward compatible, but TPM 2.0 comes up with many other algorithms, ok.

For example it has got support of not only RSA, but it has also got support of elliptic curve cryptosystems which supports like NIST P-256, Barreto-Naehrig curves. These are different forms of curves which we need not you know like which we will probably see at least some of them in our future classes. But, as I mention you in the previous class also like elliptic curve crypto is a very popular and efficient way of developing public key cryptography and TPM 2.0 apprehend the also supports the ECC then the SHA-1 is of course there and also it has got extended to SHA 256 which is an improved watched improved hash function, they are a modern hash function.

Then there is a support of a 128-bit AES encryption and along with this there are several other algorithms which are also defined in that specifications of TPM 2.0. So, here is a reference if you are interested from Wikipedia about how TPM 1.2 compares to TPM 2.0.

(Refer Slide Time: 11:51)

The slide is titled "Hardware Attacks on TPMs" and features a yellow background with a dark blue curved shape on the right side. It contains two bullet points: one from 2010 about an attack at Black Hat Conference involving an Infineon SLE 66 CL PC, and another from 2015 about Differential Power Analysis (DPA). A reference URL is provided at the bottom left. The Swamyam logo is at the bottom center, and a small video feed of a man is in the bottom right corner.

Hardware Attacks on TPMs

- In 2010, researchers presented an attack against TPM in Black Hat Conference
 - Extract secrets from a single TPM by inserting a probe and spying on the internal bus of an Infineon SLE 66 CL PC.
- In 2015, reports on Differential Power Analysis (DPA) were reported to extract the secret keys.

References: https://en.wikipedia.org/wiki/Trusted_Platform_Module#TPM_1.2_vs_TPM_2.0

swamyam

But, having said that I mean the point which I want to kind of you know like hit upon here is that in spite of this you know like there have been reports of several hardware attacks on TPMs, ok. So, I have just enlisted two of the very important attacks. One of them was told in 2010 where researchers at black hat showed how to extract secrets from a single TPM by inserting a probe and spying on the internal bus of an infinite chip, which has got TPM. So, basically they probe the bus and found out the secrets and as I said that if those secrets like EK and so on and SRK are revealed then the entire trust collapses, ok.

Likewise in 2015, there was an attack on a power attack. So, this is a side channel attack, and we will be studying how or what differential power attack sees and how it works. So, if the so, the idea is that a DPA essentially was mounted to extract the secret keys, ok.

So, this shows that even if you know like you have as a nice architecture like TPM the idea of storing secrets is not a very I would say safe and secure means, because there can be potential attacks. At the same time it also tells us that we should build our crypto systems or crypto hardware with these kinds of physical attacks in; like with this kind of side channel attacks in perspective.

(Refer Slide Time: 13:05)

Hardware Attacks

- **Side Channel Attacks:** Monitoring of analog signals, like time, power, Electromagnetics, sound, etc.
- **Fault Attacks:** Physical perturbation of Vcc, clock, temperature, UV light, X-Rays, Laser
- **Invasive Attacks:** Probe data, modify circuit!
 - Modern **Focused Ion Beam (FIB)** machines can create test points and modify the chip structure from the rear side thus overcoming sophisticated top metal mesh protections and sensors
- Expensive but feasible!

References: http://www.cs.unh.edu/~it666/reading_list/Hardware/tpm_fundamentals.pdf

The slide includes three images: a server rack, a computer workstation, and a Focused Ion Beam (FIB) machine. At the bottom, there are logos for Swamyam and other educational institutions, and a small video feed of a man speaking.

So, this brings us to like topic on hardware attacks. So, basically what I try to I am trying to say is that you know like although hardware comes up with lot of you know like lot of advantages and it basically rules out several attacks on the software. But, then there are some challenges on it is own which were which are essentially challenges because of it is nature of being hardware, ok.

So, for example, right you know like when you talk about hardware you can actually monitor several analog signals. For example, you can monitor the time taken. Of course, like it is therein even in software implementations, but you know like if there are time variations. For example, if you have say a sequential circuit which you have implemented and the output comes up depending upon the input, ok. So then you know like you can you immediately do fewer do few attacks, ok. Likewise is the power consumptions, electromagnetic, sound, etcetera all of them can be potentially used to attack, ok.

So, for example, you can probably appreciate this point that when you try to do traditional hardware design you try to minimize power, ok. But when you are trying to develop secure hardware your objective may be you know like maybe not that, maybe rather than you know of course, you have to minimize power. But what is more important is that you have to make or ensure that the power consumption does not vary

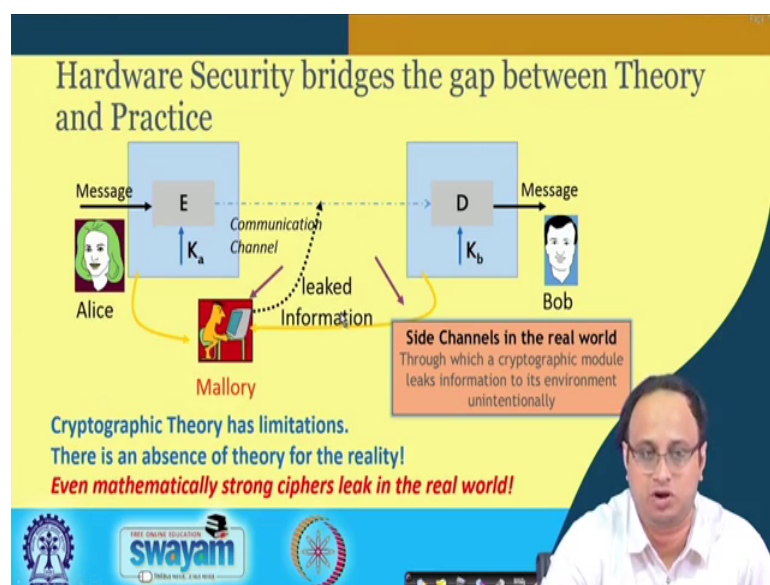
with your inputs, ok. Because if it varies then the attacker can catch those variations and from there can try to obtain this obtain the secrets, ok.

Likewise you know like you can do physical attacks, like fault attacks. For example, where you can part of the environment, you can part of the voltage, you can part of the voltage, you can create temperature fluctuations, you can take UV light, you can use x-rays, lasers, and so on to create bit flips in your circuit. And we will see later on you know in our discussions how you can use faults effectively to get the secret keys in with very less effort, ok.

Likewise you can also do invasive attacks you can probe data and you can modify circuit. So, in this context there is a technology which is called as focused ion beam or FIB which can actually create test points and even modify the chief structure, ok. From the rear side of the chip and thus it can overcome sophisticated top metal mesh protections and sensors. Like you may have in the top layer light you may have mesh protections you may have protections against attacks, but the attack essentially takes pair from the rear side from the rear structure.

And, these are you know like extremely sensitive and powerful equipments which are expensive no doubt, but they are definitely feasible. And therefore, we need to take care of these kinds of hardware attacks when we are designing our secure systems.

(Refer Slide Time: 15:41)



So, hardware security bridges the gap between theory and practice for sure you know like as I said that you know like cryptographic theory has limitations, because cryptographic theory often does not address the real world. And there is an absence of theory for the reality, because of which even mathematically strong ciphers can leak in the real world. So, you may take a nice AES algorithm or an RSA algorithm or an ACC algorithm, but when you implement them right because of these leaked information through what are called as side channels which may be intentional or it may be unintentional the secrets can be compromised, ok. So therefore, what we need to do is that we need to understand this gap between theory and practice.

(Refer Slide Time: 16:23)

The slide, titled "Hardware Security Design Goals", is presented on a yellow background with a blue header and footer. It features three main categories of goals, each in a yellow box with a red title:

- Performance:** Speed, Clock Frequency, Latency, Fast Arithmetic
- Security:** Side Channel Attacks, Fault Attacks, Lightweight Countermeasures
- Cost:** Area, Power, Energy

In the center of the slide is a photograph of a green printed circuit board (PCB) with a central microchip. The footer of the slide contains logos for "swayam" (Free Online Education) and "INDIA WISE, FUTURE WISE". A small inset video of a man in a white shirt and glasses is visible in the bottom right corner of the slide.

So, therefore, right there are several goals which will be trying to address in our course in due time is that we of course need to take care of performance which is like one of the most important reasons why we have hardware, because we want to make our designs more efficient. So, we want to of course take care of speed clock frequency latency do fast arithmetic, bring in parallelism in our computations, ok, but at the same time of course, like we have we have to also take care of some other things, ok.

The another very important thing which is of importance in mode probably in today's world when we talk about internet of things and cipher physical systems is that we need to make our designs lightweight. So, we need to make our designs consume less area take low power take low energy and so on, but even with this kind of constraints right we

cannot compromise on security, because if security is compromised then in the entire technology will collapse, ok. So, we need to take care of side channel attacks, fault attacks and also countermeasures; like when we build in countermeasures as we will be seeing in the class there extremely they have they have an high overhead, ok.

So, we need to try to ensure that the countermeasures are light and they are can they can be utilized for IOT kind of subsystems.

(Refer Slide Time: 17:37)



The slide is titled "Hardware Design Flow on FPGAs" and has a yellow background. It contains the following text:

What is an FPGA?

- Field Programmable Gate Arrays.
- Array of logic cells connected via routing channels.
- Special I/O cells.
- Logic cells are mainly lookup tables (LUT) with associated registers.

At the bottom of the slide, there are logos for "swayam" (Free Online Education) and "Utkal University". A small inset video of a man speaking is visible in the bottom right corner of the slide area.

So, I will you know in the remaining part in today's class I will be trying to tell you about how we can actually develop hardware, ok. So of course, there are different ways of designing hardware; like there are application specific integrated circuits ASICs which may be costly, ok.

There is another computing technology which is called as a FPGA which will be trying to kind of you know like focus in our class, ok. For example, an FPGA stands for what is called as an field programmable gate array. So, it is a very interesting technology which will basically comprised of an or an array of logic cells which are connected by routing channels, ok. So, the idea is that as I if you go into the name of FPGA you will see that there are certain important parts, ok.

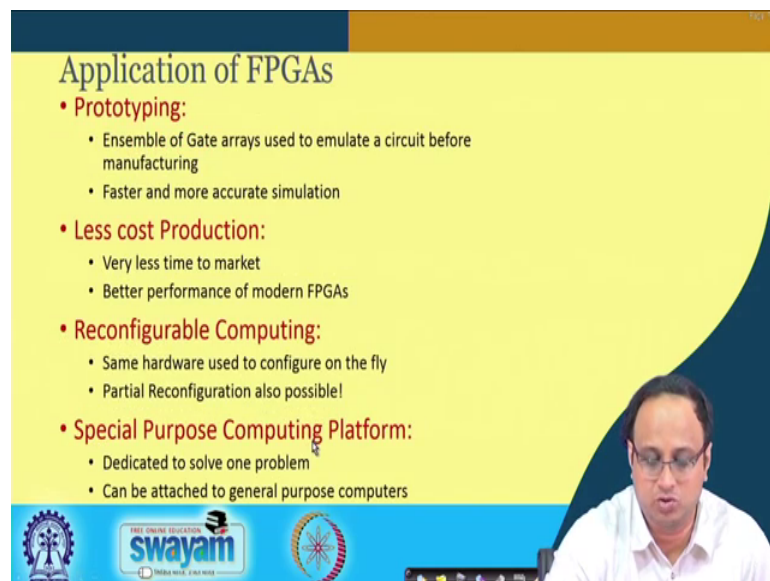
One of the interesting part is this programmability ok. So, it says that it is programmable. So, it is a hardware which is programmable. We know that there are general purpose

computers and we have general purpose computers, when we write programs on them then it is programmable, right. So, we can write any program, you can write today bubble sort, tomorrow quick sort on the same architecture, ok.

Likewise in and when you talk about FPGA, so, you can actually reconfigure it. So, you can reconfigure an adder and to maybe work tomorrow as a multiplier, ok. That is the biggest advantage of an FPGA. So, you can actually do an in-house design, you can do a design inside your house, inside your lab environment and you can go from your basic conception of the design to the final execution, in situ in your lab.

And, you know like because of this the technology is interesting, because the technology you will see has got an array of logic cells which are connected via routing channels there are special IO cells, and the logic cells are typically what are called as LUT. So, these LUTs are essentially you know like the fundamental blocks of FPGA.

(Refer Slide Time: 19:17)



The slide is titled "Application of FPGAs" and lists four main categories of applications:

- Prototyping:**
 - Ensemble of Gate arrays used to emulate a circuit before manufacturing
 - Faster and more accurate simulation
- Less cost Production:**
 - Very less time to market
 - Better performance of modern FPGAs
- Reconfigurable Computing:**
 - Same hardware used to configure on the fly
 - Partial Reconfiguration also possible!
- Special Purpose Computing Platform:**
 - Dedicated to solve one problem
 - Can be attached to general purpose computers

The slide also features logos for "SWAYAM" (Free Online Education) and "MOE, GOVT OF INDIA" at the bottom. A video inset in the bottom right corner shows a man with glasses speaking.

So, let us take a look about how and or how an FPGA you know like looks like. So, here before that you know here I have kind of any state some applications or FPGA. For example, I think I think the one of the biggest advantages of FPGA is to prototype. Like before you are going into commercial deployment of an ASIC as I said that ASIC is costly you would like to prototype your design. You would like to see that whether your design at all works of works fine or not and there is an huge application of FPGAs, ok.

And, the less then and the production cost is less because of it there is a very less time to market you can get better performance and modern FPGAs are improving in leaps and bounds. So, the bridge or the gap between ASICs and FPGAs is also you know like getting lesser and lesser. So, our FPGAs nowadays are pretty advanced. So, you can really develop I would say competing technologies on FPGAs as well ok, because of various dedicated blocks that they have like DSP blocks, multipliers and so on.

And, other thing whether I already said is that it is reconfigurable. So, you can also do you know like you can make the same hardware to reconfigure and also you can reconfigure on the fly. So, with techniques like partial reconfiguration you can actually do dynamic reconfiguration, and that is a huge advantage with FPGAs brings it, ok. Particularly in the context of things like IOT and those kinds of things very probably need to change things very rapidly.

Then also you can develops special purpose computing platform. So, for example, delegated to solve one problem and this is exactly where you can bring in crypto, because in crypto right you need to solve dedicated problems. And thus in this context you can try to bring in or develop FPGA accelerators, ok.

(Refer Slide Time: 20:57)

A Classic Example of DES Cracker

- In 1998, a custom hardware attack was mounted against the Data Encryption Standard :
 - \$250,000 to build and decrypted DES cipher in 56 hours
- In 2006, **COPACOBANA** (Cost-Optimized PArallel COde Breaker) was built:
 - Consists of commercially available, reconfigurable integrated circuits.
 - \$10,000 and decrypts DES cipher in around 6.4 days
 - Cost decrease by roughly a factor of 25
 - Adjusting for inflation over 8 years yields an even higher improvement of about 30x.
- Since 2007, SciEngines GmbH, a spin-off company of the two project partners of COPACOBANA has enhanced and developed successors of COPACOBANA.
 - In 2008 their COPACOBANA RIVYERA reduced the time to break DES to the current record of less than one day, using 128 Spartan-3 5000's

[Source: http://en.wikipedia.org/wiki/Custom_hardware_attack]

The slide features a yellow background with a blue and orange header. At the bottom, there are logos for 'swayam' and 'SCIENTIA' along with a small inset image of a man speaking.

So, let us take a look about ok. So, before that here is a classic example of a des cracker. So, this design is called as COPACABANA. So, all many of you know and probably know about the des or the data encryption standard. So, in 1998, a custom hardware was

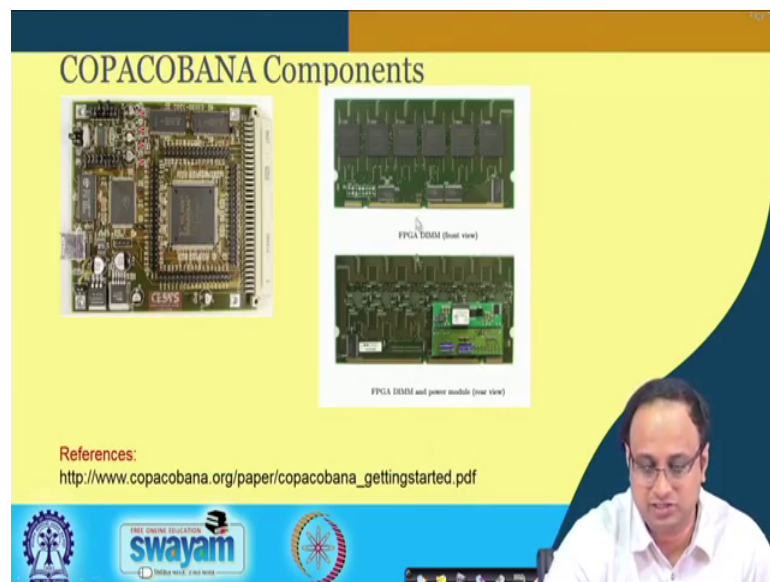
developed to develop an attack on this and you can see the cost. It was something like dollar 25000 to build and decrypt DES cipher in 56 hours.

Later on, in 2006, there was an effort we just called as COPACOBANA which stands for Cost-Optimized PARallel COde Breaker which was built and it was built on commercially available reconfigurable integrated circuits. And, the cost reduced to something like dollar 10000 and decreased des cipher in around 6.4 days, ok. So, the cost decreases by roughly a factor of 25, ok. So, if you just adjust for inflation over 8 years. So, you can see that even you can see the improvement is probably in around 30 x, ok.

Now, since 2007, there have been several efforts. So, for example, this is an example of a spin-off company which tries to develop successors of COPACOBANA, and one of the success of COPACOBANA is called as RIVYERA. And it reduces the time to break DES to the current record of less than one day using 128 Spartan-3 FPGAs, ok.

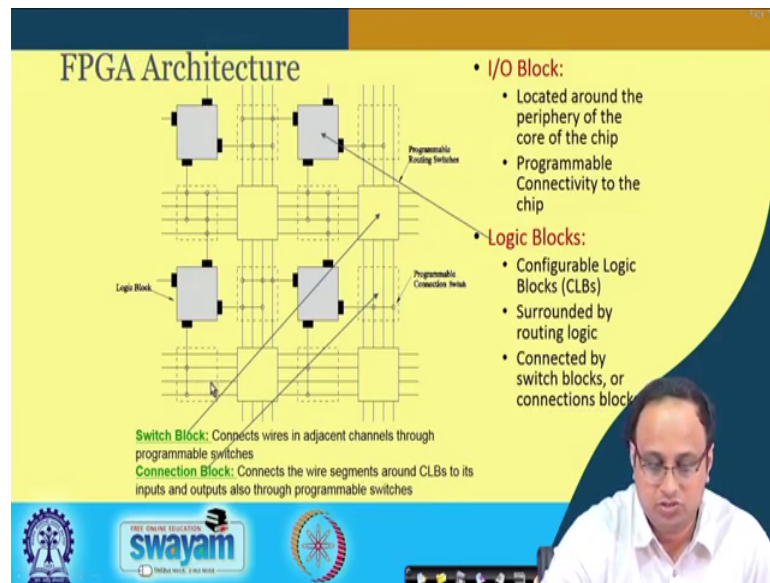
So, this is an interesting example and this is a Wikipedia link where you can go and see how if FPGAs can be used to develop custom attack hardwares.

(Refer Slide Time: 22:23)



So, that you know like. So, this here is a photograph of how the COPACOBANA looks like. And you can see it is like made of nice structures we are built around FPGAs.

(Refer Slide Time: 22:33)



So, here is a diagram about how an FPGA architecture looks like. So, this architecture is what is called as the island architecture. So, in this architecture you will see that there are some IO blocks. So, these input-output blocks are located around the periphery of the core of the chip and the objective of an IO block is just to develop connectivity of the chip. That means, the chip tries to communicate with the external world through these IO blocks.

The other important block is the logic block. So, this logic block is where your logic resides ok. So, this logic block is compromised or what are called as CLBs or Configurable Logic Blocks which are in turn made about what are called as LUTs or Look Up Tables, ok.

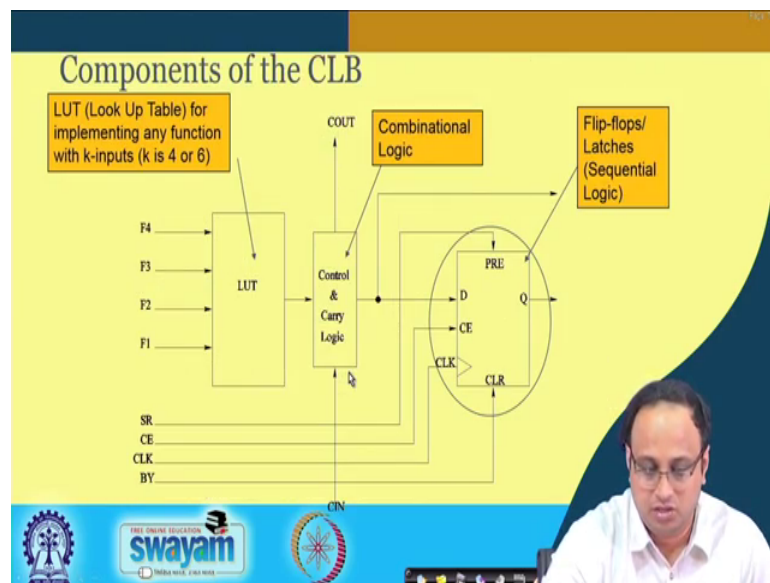
So, in general this architecture may vary from slide to slide from vendor to vendor, but more or less this is how it looks like, ok. So, you have got the logic blocks and this logic blocks essentially are interconnected with; I mean they are interconnected with each other, they are interconnected with the power with the lines around and for that you also have got special blocks. So, these blocks are what are called as the switch blocks and the connection blocks.

So, if you see this diagram you will see that there are two important routing points, and they are all switches, ok. So, for example, this is this logic block is connected to these rails, and these are essentially what are called as a connection blocks. For example; if

you see this logic block it is connected with these rails or the wire segments around the CLBs they are connected by programmable switches, ok. So, you have got programmable switches which you can configure to get connected, ok. Likewise right if you take the wires in adjacent channels like this channel and this channel they are connected here by what are called a switch blocks, ok. So, these are nothing, but interconnection logic or interconnection switches which you can program.

So, for example, if I want to turn on a switch I need to say you know like program one over there, whether you know if I want to turn off that switch I will program it by 0. So, it is basically using switches or maybe digital switches to be more precise, ok. So, at the end of the day what you are trying to do is that you have a design, and your tool right will give you a 0 1 pattern which is often called as the bit file and you take that bit file and the bit file configures these switches, ok. And therefore, right the FPGA gets reconfigured and not only the switches, but also your lookup table, and your entire structure gets reconfigured to work as you wish, ok, as you intend.

(Refer Slide Time: 24:57)



So, here is an how the internal of the CLB. So, this is a slightly and on an older FPGA so, but I think the idea remains the same. So, you will see that in the CLB that is a block which is called as the LUT or the Look Up Table, ok. So, these lookup table typically always have a fixed number of inputs. So, the number of inputs here a designated as 4, ok. In modern FPGAs we may probably see it go up to 6, ok.

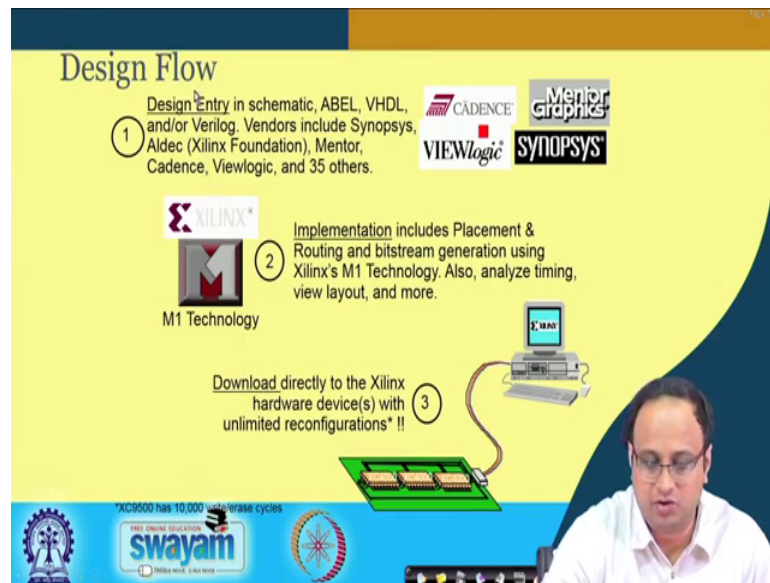
So, there was an older work where people try to do lot of you know like explorations trying to find out that what should be the size of this input of the LUT, ok, whether it should be 4, 5, 6, 8 or so on. So, it was found out that if the value is around 4 or 6, right the around this point it has found out that the utilization of the look up tables is maximum, and therefore, right you will see that if you whenever realizing Boolean functions then most of the FPGA is typically have got inputs which are within 4 to 6, ok.

So, therefore, I mean you have the LUT block. So, this LUT block can realize any Boolean function of 4 input in this diagram, if it is 6 then it can realize any Boolean function of 6 input. Sometimes in modern LUTs we do not have only one output, but you have got two outputs also, ok. So, but anyway without loss of generality and without losing too much information we can you know like I think about this architecture right. Now we have one output in the LUT and there are four inputs to this LUT.

Apart from this there is also a control and carry logic which is a combinational block which is often a fast carry chain. And this fast carry chain essentially you know like is used to pass on your input carry to the output, ok. So now, what you can do is, you can pass this input carry and calculate the output and from there you will get fast carry chain. So, for example if you are trying to implement then adder; for example, you know that the biggest hurdle of implementing an adder is a carry chain. So, you can actually implement an adder nicely in this structure where the lookup table does your other computations, but the carry chain is a dedicated path, which is much faster than your other LUT blocks.

You also have you know like a flip-flop which is essentially supposed to provide you support for the sequential logics. So, you can see that you have got support for combinational logics and sequential logics in together you can actually realize our this basically functions as your basic logic block.

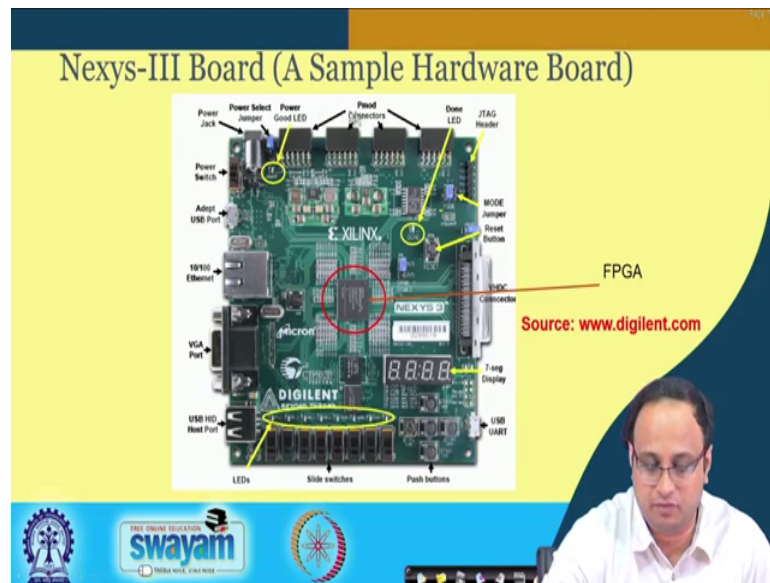
(Refer Slide Time: 27:19)



So, that is how the design flow looks like. So, the design flow you will see has got or it starts with you know like what is called as an RTL description. So, you start with an RTL description you have got a design and you start with an basic RTL. So, you can implement your RTL using say Verilog or VHDL or some other languages, but typically Verilog is a very standard language which is adopted. Then you put like pass that to several CAD tools; again you have got several CAD tools like you have got Xilinx and there are like I have just written more hear about the FPGA tools which are used, but if you go into the ASIC world then the other tools like which are designed by Cadence and Synopsys.

So, if you base the idea is that you basically do an implementation here. So, the implementation with include blocks like placement and routing. And finally, the bitstream generation, and once the bitstream generation is done you essentially can download that bitstream through a port which is often the z tag port and you can configure the FPGA to work as you wish, ok. So, you basically can reconfigure the FPGA.

(Refer Slide Time: 28:29)



So, this is how the overall design flow looks like and what I will try to you know like show you was one of the sample boards which you can probably try to get access to. Is a relatively low cost FPGA device which is a called as an Nexys-III board and you know like we can actually. So, this is the FPGA inside the board and what we will be trying to do is we will be trying to program this FPGA using our design and typically and reconfigure it to work as we want.

(Refer Slide Time: 28:59)

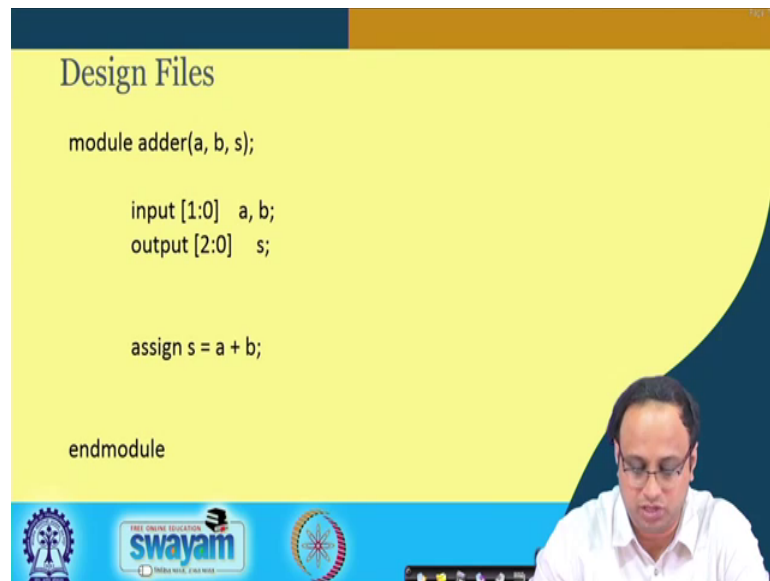
A Simple Design Problem

- Design a 2 bit adder:
- Input : 'a' and 'b'
- Output: $s = a + b$

So, I will take a simple design problem and try to show or illustrate about how an how basically we can do design on these FPGAs.

So, to start with you know like you can take a 2 bit adder. So, your objective is to build this logic of s equal to a plus b.

(Refer Slide Time: 29:17)



```
Design Files

module adder(a, b, s);

    input [1:0] a, b;
    output [2:0] s;

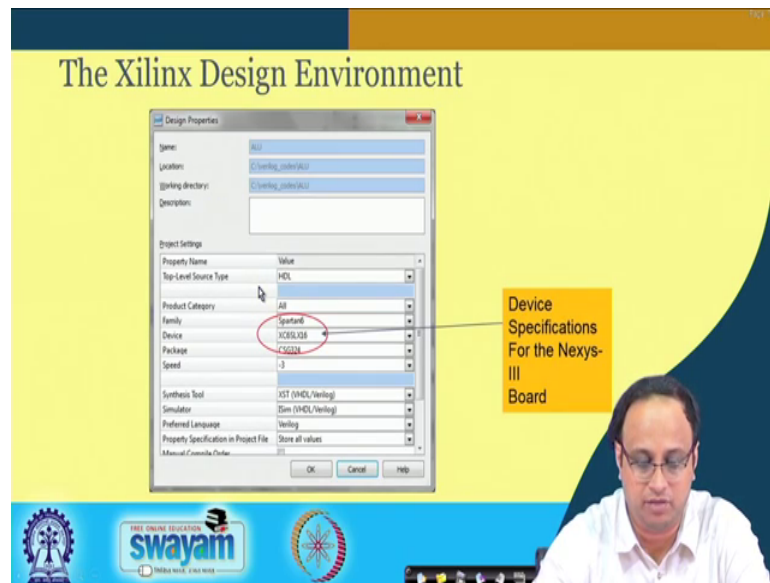
    assign s = a + b;

endmodule
```

The slide features a yellow background with a dark blue header and footer. The Verilog code is displayed in a monospaced font. In the bottom right corner, there is a small video inset of a man with glasses speaking. The footer contains logos for 'swayam' and 'INDIA'S MOOC PLATFORM'.

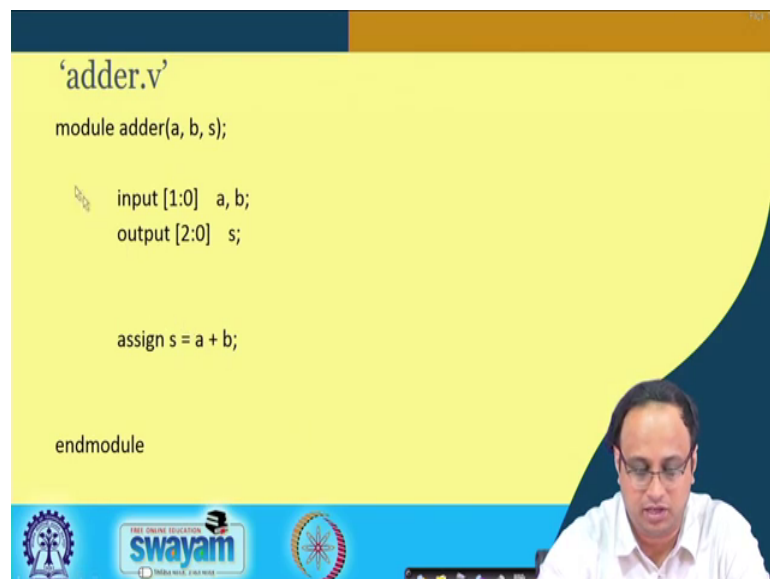
So, if you when you want to do that you are starting thing is writing or developing a design file. So, this design file is often described it Verilog, ok. So, I am not going into the syntax of Verilog, because that may be you can look into yourself. So, there is a module description inside that you develop the logic for the adder, ok.

(Refer Slide Time: 29:35)



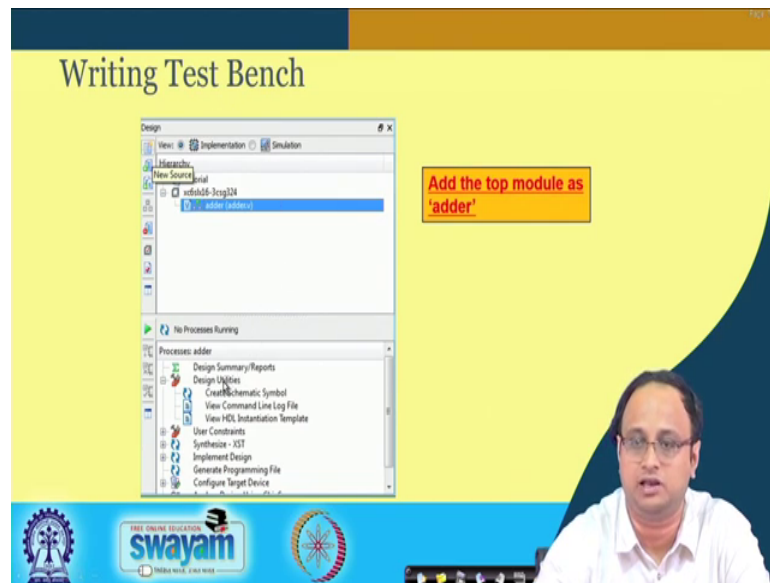
Once you have developed that added you take it through a flow. So, easier is as Xilinx design flow here, you can choose the target FPGA right the corresponding family device package and other stuff, and then you can actually pass it through some sequence of steps. And finally, you will get the bit file, ok.

(Refer Slide Time: 29:53)



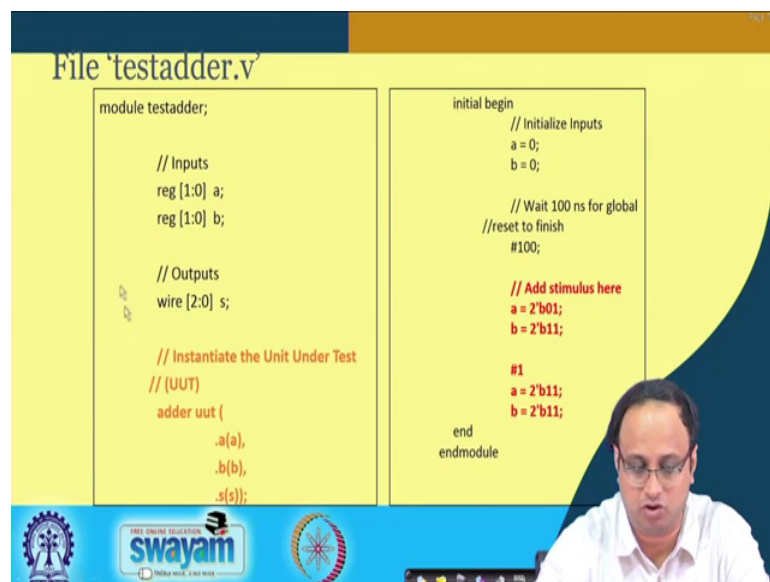
So, the bit file. So, the bit file; so here is are adder logic.

(Refer Slide Time: 29:57)



You also want to test your design. So, you want to know that whether your design is correct or not. So, the usual way of doing that is by writing what is called as a test bench. So, you write a corresponding test bench of your adder.

(Refer Slide Time: 30:05)



So, if you see the test bench right I mean again without going into the syntax, you basically instantiate the adder which means you kind of scan sane inputs to the adder. And the input sequence is denoted here. So, you are basically giving stimuli to your adder. So, you are giving say 0 0 as an input or a 1 1 or 1 3 as input and so on.

Now, this adder will essentially give you the corresponding output and you can actually verify whether your adder is correct or not.

(Refer Slide Time: 30:33)

The slide titled "Simulated Behavioral Model" displays a screenshot of a simulation tool. On the left, a "Design" window shows a hierarchy with "adder" selected. Below it, "Processes" for "adder" are listed, including "Esim Simulator", "Behavioral Check System", and "Synthesize Behavioral Model". On the right, a timing diagram shows signals "a[2:0]", "b[2:0]", and "s[2:0]" over time. A table below the diagram shows the values of these signals at 100 ns intervals:

Name	Value	100 ns	200 ns	300 ns	400 ns
a[2:0]	110	000	000	000	000
b[2:0]	011	000	000	011	011
s[2:0]	111	000	000	000	000

The slide also features a video inset of a man speaking and logos for "swayam" and "Free Online Education".

So, usually what you do initially is you do a simulation. So, you basically do a simulation which is called as a behavioral simulation. And gradually as you go down the design cycle you do more and more accurate simulations.

(Refer Slide Time: 30:45)

The slide titled "Adding User Constraint File (UCF) for Synthesis" shows a "New Source Wizard" dialog box with "Implementation Constraints File" selected. To the right, a list of UCF constraints is provided:

```
NET "a[0]" LOC = T10;  
NET "a[1]" LOC = T9;  
NET "b[0]" LOC = V9;  
NET "b[1]" LOC = M8;  
NET "s[0]" LOC = U16;  
NET "s[1]" LOC = V16;  
NET "s[2]" LOC = U15;
```

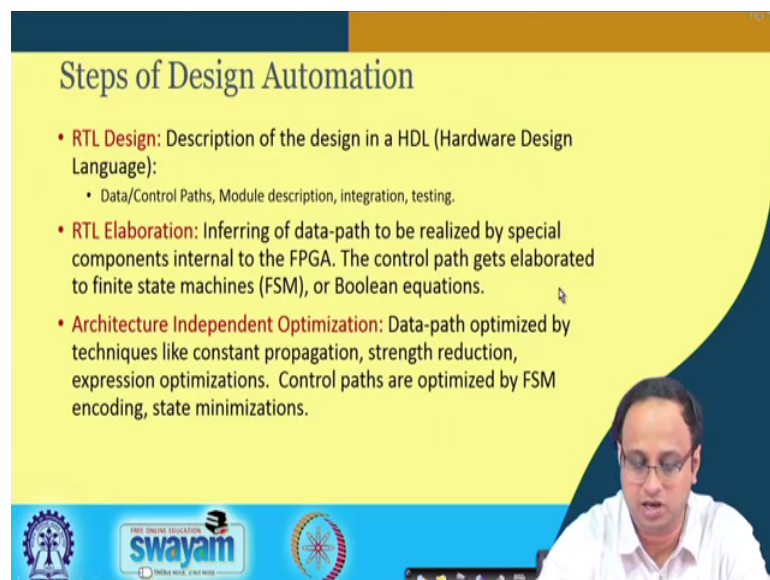
Below the list, there are instructions: "(SEE THE NEXYS-III BOARD FOR DESCRIPTIONS)", "Go to implement and Press Synthesize-XST (Xilinx Synthesis Technology)", and a video inset of a man speaking. The slide also features "swayam" and "Free Online Education" logos.

So, there is another when you are trying to really put this design on to the FPGA you also have to do what is called as a user constant file or UCF where you have to describe the

how the interconnection between the FPGA and your Verilog code. For example, the Verilog that I wrote has inputs like a 0, a 1, b 0, b 1, s 0, s 1 and s 2. So, these are the outputs of the adder. You can actually tie it to the FPGA pins by this LOC configurations, ok.

So, for more descriptions you can actually look into the Nexys-III board manual from there you can actually get this descriptions and accordingly you can write the corresponding UCF file once you have done that you can go into the implement and play synthesis step.

(Refer Slide Time: 31:25)



Steps of Design Automation

- **RTL Design:** Description of the design in a HDL (Hardware Design Language):
 - Data/Control Paths, Module description, integration, testing.
- **RTL Elaboration:** Inferring of data-path to be realized by special components internal to the FPGA. The control path gets elaborated to finite state machines (FSM), or Boolean equations.
- **Architecture Independent Optimization:** Data-path optimized by techniques like constant propagation, strength reduction, expression optimizations. Control paths are optimized by FSM encoding, state minimizations.

The slide includes logos for IIT Bombay and Swamyam at the bottom. A small inset video shows a man speaking.

And, automatically your tool will go through a bunch of steps, ok. And, these steps essentially I have got several important components. So, first important component is that it does and RTL design as a like you already done this RTL design. Once you have done these you know like the next step which is done is what is called as an RTL elaboration.

So, in RTL elaboration what is done is that it basically infers your design into data path components, or control path components. So, data path components are essentially realized by special components which are internal to the FPGA and the control path gets elaborated into either an FSM or some Boolean logic.

So, there are several optimizations which are done some of the optimizations are architecture independent. That means, they are done by usual compiler optimization techniques like constant propagation, strength reduction, expression optimizations whereas, the control paths are optimized by FSM encoding and state minimization.

(Refer Slide Time: 32:21)

Steps of Design Automation

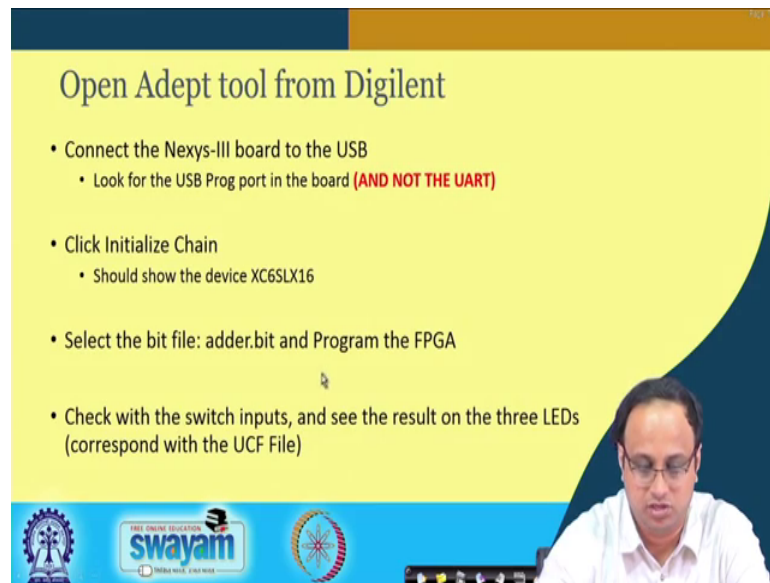
- **Mapping:** Various elements of the design are optimally assigned to FPGA resources. Data-path elements get inferred to adders, multipliers, memory elements. Control-path are realized in the FPGA logic block. The optimizations depend on the FPGA fabric, the LUT structures, etc.
- **Placement:** Decides the physical locations and inter-connections of each logic block.
 - Placement Driven Optimizations: In order to reduce the inter-connects, the initial placement is updated.
- **Routing:** The CLBs and the wires are interconnected using restricted programmable switches.
- **Bit-Stream Generation:** Final Step! It takes the routed design as input, and produces the bit stream to program the logic and inter-connects (every thing is a switch which can be programmed)

The slide includes logos for Swamyam (Free Online Education) and other educational institutions at the bottom.

And on the other hand the next steps which you are done are more specific to the FPGA where you actually do a mapping. So, the mapping is where you basically take various elements of the design and you optimally assign to FPGA resources. So, FPGA resources essentially you know like the they are typically like the data path elements gets infer to adders, multipliers and memory elements, and the control paths are realized in the FPGA blocks and in the FPGA logic block often, ok. So, the optimizations are often depend on the FPGA fabric like the LUT structure like whether it is a 4 (Refer Time: 32:51) LUT and so on.

Then, it is followed by two important steps which are called as placement and routing and finally, you get the bitstream, ok. The once you have got the bit stream that is a final step because the bit stream takes a routed design as input and produces a bit stream which programs the logic and interconnects, ok. As I said that the all of them are programmable, so, you can program and everything is a switch. And therefore, in this you can actually program them, either turn on or turn off the switches.

(Refer Slide Time: 33:17)



Open Adept tool from Digilent

- Connect the Nexys-III board to the USB
 - Look for the USB Prog port in the board **(AND NOT THE UART)**
- Click Initialize Chain
 - Should show the device XC6SLX16
- Select the bit file: adder.bit and Program the FPGA
- Check with the switch inputs, and see the result on the three LEDs (correspond with the UCF File)

The slide features a yellow background with a blue and orange header. At the bottom, there are logos for Swamyam and a small image of a person's face.

So finally, you open the tool from adept. So, there is just an example, but you can do it with other tools as well. So, then you know like you basically download this bit file. And finally, your design works as intended ok.

(Refer Slide Time: 33:27)



References:

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

The slide features a yellow background with a blue and orange header. On the left, the word 'References' is written in a stylized yellow font. In the center, there is a book cover for 'Hardware Security: Design, Threats and Safeguards'. At the bottom, there are logos for Swamyam and a small image of a person's face.

So, reference you can actually take a look at this book. You will find more details about how you know like the design works and how these designs can be made and this is a standard reference you will be trying to follow in most parts of the class; although we will try to use some other materials as well.

(Refer Slide Time: 33:47)

Conclusion

- TPM a hardware security co-processor to work as a root of trust
- Hardware Attacks are still possible in the form of side channel attacks
- FPGAs provide a reconfigurable platform for hardware design
- FPGA design flow takes the RTL description to the final prototype in the form of bit-files

The slide also features a small video inset of a man in a white shirt and glasses, and a logo for 'SWAYAM' (Free Online Education) at the bottom.

So, what we have discussed in today's class is; we have discussed about TPM which is a hardware security core processor to work as a root of trust, ok. And, the hardware attacks and we just say that even when we are using hardware then still attacks are possible in the form of specialized hardware attacks we are often called as side channel attacks.

FPGAs provide a reconfigurable platform for hardware design. So, it is very handy and not only to understand designs, but also to really develop applications using a for hardware. And, the FPGA design flow typically takes the RTL description which is often done using languages like Verilog which are high level description or hardware description languages where high level languages. And finally, you take it to the prototype. And, the prototype is usually done in the form of bit file. So, you create a bit file and then you download that into your design and your design starts working, ok.

So, with this I would like to come conclude this part of the class. And thank you, for your attention and we take the next class, where we will be trying to look more into hardware designs.