**Mini Project Presentation**

* Welcome to our presentation.
* Our presentation subject is: IMPLEMENTATION OF ABC ON HARDWARE TROJANS and presentation will be presented by SUPRITI DAS, SHANTONU DEBNATH and TARUN CHAUHAN
* A Hardware Trojan is a malicious modification of the circuitry of an integrated circuit. A hardware Trojan is completely characterized by its physical representation and its behavior. The payload of an HT is the entire activity that the Trojan executes when it is triggered.
* Once activated results in

– data leakage

– harm to the normal functionality of the circuit

* After activation, a HT can deliver its payload either through

– standard I/O channels

– side channels

* Hardware Trojan is generally split into two categories; combinational and sequential. A combinational trojan will activate upon detection of a speciﬁc set of circumstances within the internal signals of the IC. Sequential trojans will also monitor the internal signals of the IC. However, instead of looking for a speciﬁc condition, they activate when a speciﬁc sequence of events occurs. Sequential Trojans also 2 types 1) Synchronous counterTrojans 2) Asynchronous counterTrojans. Synchronous counter trojan triggers a malfunction on reaching a particular count and asynchronous counter Trojans count is increased not by the clock, but by a rising transition at the output of an AND gate.
* HT PROPERTIES BASED ON HATCH ALGORITHM: 1.**Trigger Signal Dimension d(T )** represents the number of wires used by HT trigger circuitry to activate the payload circuitry in order to exhibit malicious behavior. A large d shows a complicated trigger signal, hence it is harder to detect.

2.**Payload Propagation Delay t(T )** is the number of cycles required to propagate malicious behavior to the output port after the HT is triggered. A large t means it takes a long time after triggering until the malicious behavior is seen, hence less likely to be detected during testing.

**3.Implicit Behavior Factor α(T )** represents the probability that given a HT gets triggered, it will not (explicitly) manifest malicious behavior; this behavior is termed as implicit malicious behavior. Higher probability of implicit malicious behavior means higher stealthiness during testing phase.

4.**Trigger Signal Locality l(T )** shows the spread of trigger signal wires of the HT across the IP core. Small l shows that these wires are in the close vicinity of each other. Large l means that these wires are spread out in the circuit and therefore it is harder to figure out exactly which wires form the trigger signal, hence the HT becomes harder to detect.

* Now we are designing FSM based HT. For this reason we require the tool ABC for sequential synthesis. Lets see it.
* BERKELEY ABC : A public-domain system for logic synthesis and formal verification of binary logic circuits appearing in synchronous hardware designs. It’s very Fast, Scalable, find high quality results, exploits synergy between synthesis and verification. The programming environment is Open-source and evolving and improving over time.
* Now we see some ABC commands : For Inputs (*aig, baf, bench, blif* some kind of file) 1.**read** – Parses an input file using one of the available file readers. The file extension is used to determine what file parser to invoke. The recognized file extensions are: *aig, baf, bench, blif, eqn, pla, verilog.*

2. **read\_bench** – Parses the input file in BENCH (ISCAS) format.

3. **read\_blif** – Parses the input file in [BLIF](http://www1.cs.columbia.edu/~cs4861/s07-sis/blif/index.html). This command can also read hierarchical BLIF.

For another file we write read\_baf, read\_aig like that

* For output:

1. **write** – Writes the output file using one of the available file writers. The file extension is used to determine what file writer to invoke. The recognized file extensions are: *aig, baf, bench, blif, cnf, dot, eqn, gml, pla, verilog.*

2. **write\_bench** – Outputs the current network into a BENCH file.

3. **write\_blif** – Outputs the current network into a [BLIF](http://www1.cs.columbia.edu/~cs4861/s07-sis/blif/index.html) file. If the current network is mapped using a standard cell library, outputs the current network into a BLIF file.The current mapper does not map the registers. As a result, the mapped BLIF files generated for sequential circuits contain unmapped latches. Additionally, command *write\_blif* with command-line switch –*l* writes out a part of the current network containing a combinational logic without latches.

* For print:

1. **print\_factor** – Prints the factored forms of the nodes in the current network.

2. **print\_fanio** – Prints the distribution of nodes by the number of fanins and fanouts.

3. **print\_gates** – Prints statistics about the gates used after technology mapping. For a technology-independent networks, prints how many nodes have a given type of logic function.

4. **print\_io** – Prints the lists of primary inputs (PI), primary outputs (POs), and latches of the network. When called for a node (given by a name on the command line), prints its fanouts and fanouts.

5. **print\_kmap** – Prints Karnaugh map of the logic function of a node.

6. **print\_latch** – Prints the information about latches of the current networks.

7. **print\_level** – Prints the distribution of the COs by the number of levels in their logic cones. If the network is mapped, prints a delay profile of the COs..

8. **print\_sharing** – Prints the number of nodes shared by each pair of the COs in the current network.

9. **print\_stats** – Prints the vital stats of the current networks. The statistics printed depend on the current network representation.

* Now we see the some screenshot.
* FSM: A finite-state machine or finite-state automaton, finite automaton, or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time.

Here we see the fsm state table and diagram.

We see that in the even state it not change and the odd state it changed.

Here we are implement the fsm use c program. Here we use only one int function and the switch case function.

* Here we see the ppt reference
* This is the end of the presentation. Thank you.