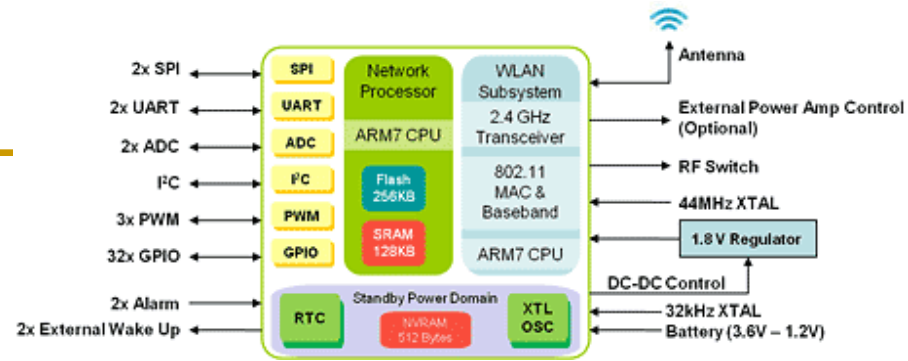
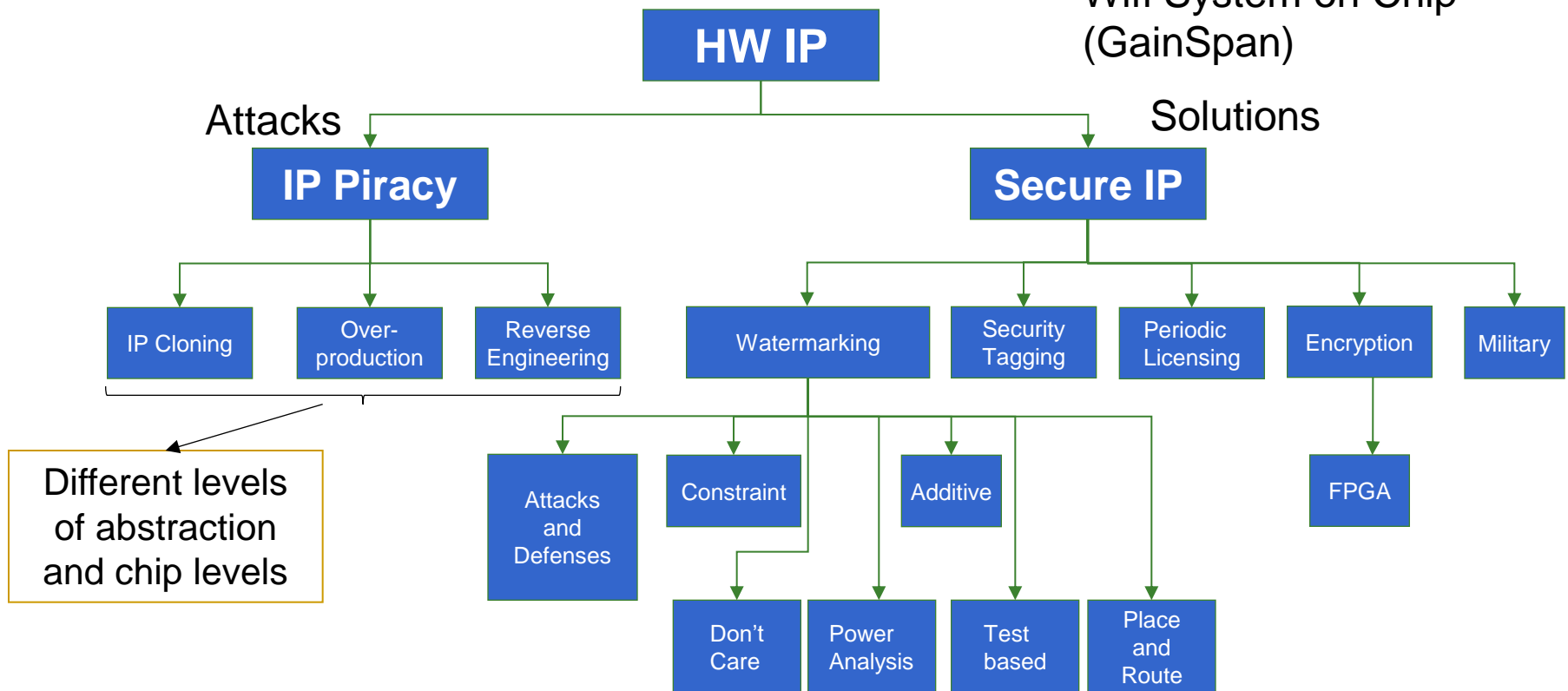

Watermarking of HW IPs/ Secure IP

Contents



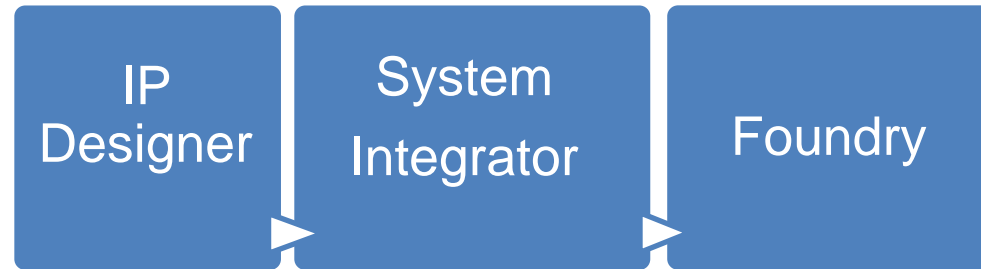
Wifi System on Chip
(GainSpan)



Hardware Intellectual Property (HW IP)

■ Why HW IP?

- ❑ Design reuse
- ❑ System-on-Chip (SoC)
- ❑ FPGA

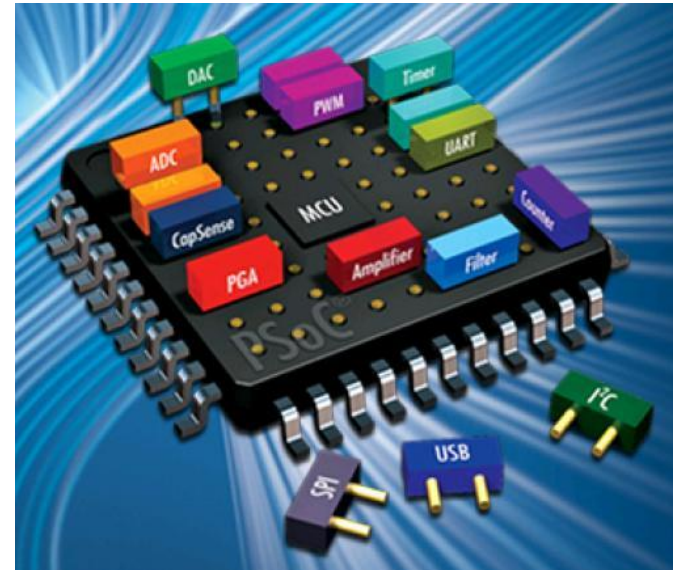


■ Players

- ❑ IP designer
- ❑ System integrator
- ❑ Foundry

■ What qualifies as an HW IP?

- ❑ HDL code, GDSII, netlist, layout, technique, ...



HW IP Lexicon

■ Soft IPs

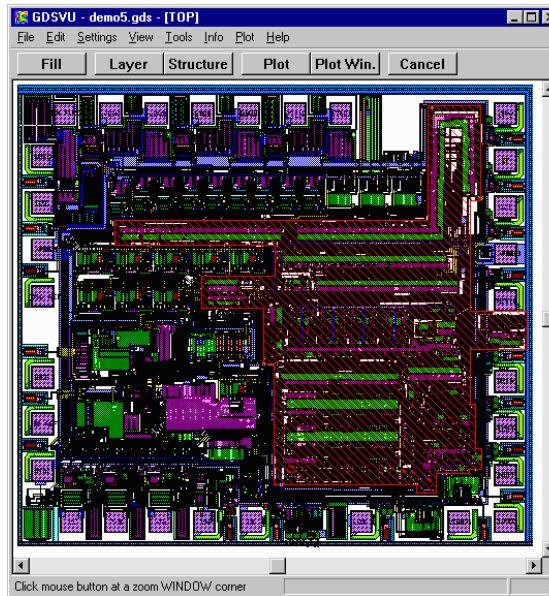
- ❑ HDL codes

■ Firm IPs

- ❑ Placed RTL design
- ❑ Fully placed netlist

■ Hard IPs

- ❑ GDSII file



GDSII file

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity signed_adder is
6 port
7 (
8   aclr : in    std_logic;
9   clk  : in    std_logic;
10  a     : in    std_logic_vector;
11  b     : in    std_logic_vector;
12  q     : out   std_logic_vector
13 );
14 end signed_adder;
15
16 architecture signed_adder_arch of signed_adder is
17   signal q_s : signed(a'high+1 downto 0); -- extra bit wide
18
19 begin -- architecture
20   assert (a'length >= b'length)
21     report "Port A must be the longer vector if different sizes!"
22     severity FAILURE;
23   q <= std_logic_vector(q_s);
24
25   adding_proc:
26   process (aclr, clk)
27   begin
28     if (aclr = '1') then
29       q_s <= (others => '0');
30     elsif rising_edge(clk) then
31       q_s <= ('0' & signed(a)) + ('0' & signed(b));
32     end if; -- clk'd
33   end process;
34
35 end signed_adder_arch;
```

HDL code

IP Piracy: Vulnerabilities

- **ICs:**

- ❑ Reverse engineering

- **Hard IPs**

- ❑ Overproduction
 - ❑ Reverse engineering
 - ❑ Cloning of FPGA bitstream

- **Firm and Soft IPs**

- ❑ Overusage
 - ❑ Cloning
 - ❑ Reverse engineering
-

IP Piracy: Threats

■ IP Overusage

- Use more than contracted to (**Attacker**: SOC Integrator)

■ IP Modification

- Modify an IP from 3PIP vendor, sell as a new IP to other SoC integrators (**Attacker**: SOC Integrator)

■ IP Cloning

- Use of HW IPs without license (**Attacker**: SOC Integrator)
 - Production of compatible IPs are not IP cloning

■ Overproduction

- By foundry
-

IP Piracy: Threats, cont'

■ Reverse Engineering

- ❑ By 3rd party system integrator
- ❑ ***On hardware: is often legal***
 - Often used to determine infringement
 - Often banned in software end-user license agreement (EULA)
- ❑ ***Types***
 - In order to make cloning possible
 - Also is crucial in attacking secure IP designs
- ❑ ***Methods***
 - IO analysis
 - Decap, Delayer, SEM
 - Thermal imaging

Approaches to Secure IP

■ Prevention/Protection

□ Chemical (Military)

- in military applications chemicals and other self-destruct devices are used to protect the military secrets

□ Obfuscation

□ Expiration of Service

- Periodic licensing
- Trojan insertion (Kill Switch?)

□ Encryption

- FPGA bitstream
-

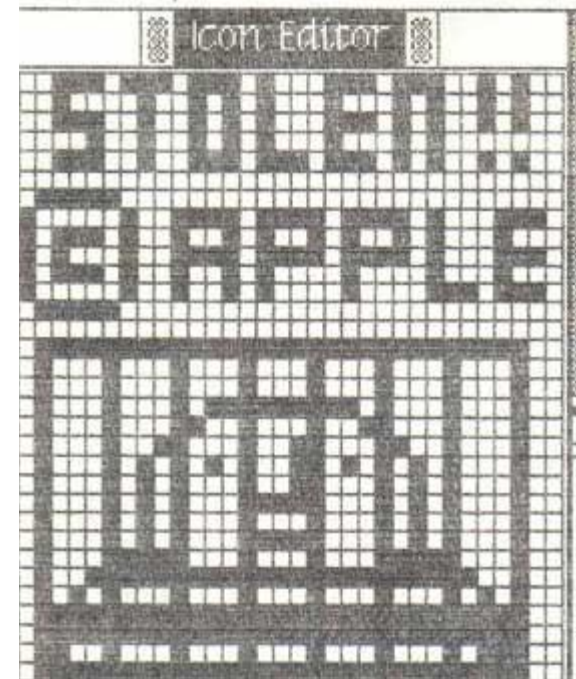
Approaches to Secure IP

■ Detection/Identification

- ❑ "Stolen from Apple"
- ❑ Watermarking
- ❑ Security tagging

In 1980, a company called *Franklin Computer* produced a clone of the Apple II called the Franklin Ace, designed to run the same software. They copied almost every detail of the Apple II, including all of its ROM based software and all the documentation, and sold it at a lower price than Apple. We even found a place in the manual where they forgot to change "Apple" to "Ace". Apple was infuriated, and sued Franklin. They eventually won, and forced Franklin to withdraw the Ace from the market.

After the incident with Franklin, Steve Jobs decided to protect the ROM by embedding a small token into it that could be displayed in court during a trial. The idea was that he would be able to enter a few keystrokes on the offending machine and display the token, proving unequivocally that the ROM was stolen from Apple.



"Stolen from Apple" icon in MacIntosh® ROM, inserted after *Apple v. Franklin Computers* lawsuit, intended as identification method

Secure IP Design Goals

■ Robustness

- Higher predictable performance
- More design flexibility

■ Overhead

- Less risks
- Acceptable constraint

■ Tradeoff

- Pre-processing: more robustness, more overhead
- Post-processing: less overhead, less robustness

Obfuscation

- Denies unlicensed usage
 - Normal functional behavior is enabled only after application of a specific *input sequence* or *key*
 - Inserting finite state machine to obfuscate IO
 - Insert a key to unlock logic block
-

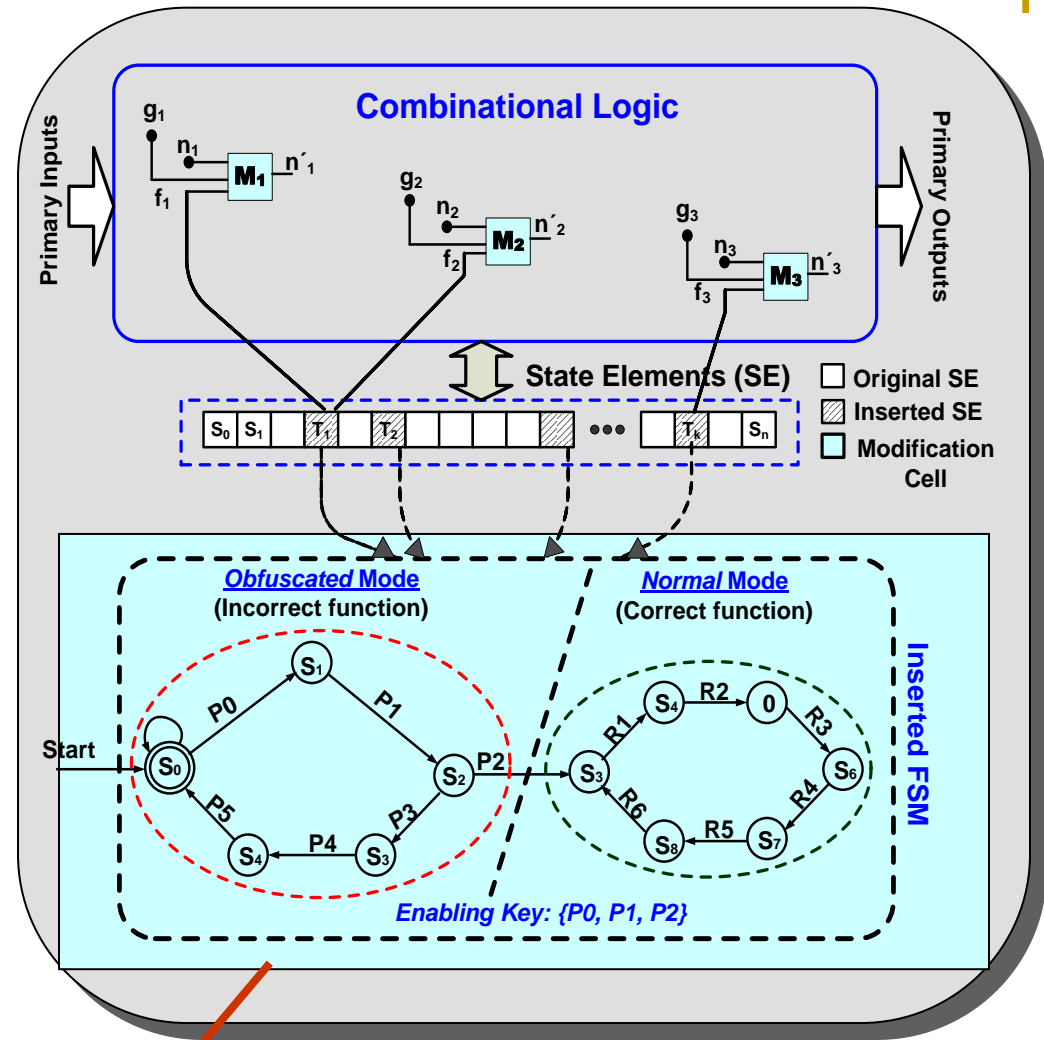
State Space Obfuscation

Basic Idea:

- A locking approach where normal behavior is enabled only upon appn. of a key
- Provable robustness

Key Innovations:

- It obfuscates the state space AND the comb. logic
- Uses rich theory of automata to transform the state space & associated logic



Protects against Piracy, RE & Tampering

Secure IP: Periodic Licensing

■ System composition

- License token
- License controller
- Timer
- Non-volatile memory

■ Goal

- Disable the IP when expire/fail to access

■ Method

- Encrypted token (Encryption)

Encryption: Secure FPGA

■ Why is FPGA vulnerable?

- Configuration bitstream stored in EPROM or Flash
 - When power on -> download
 - When power off -> removed

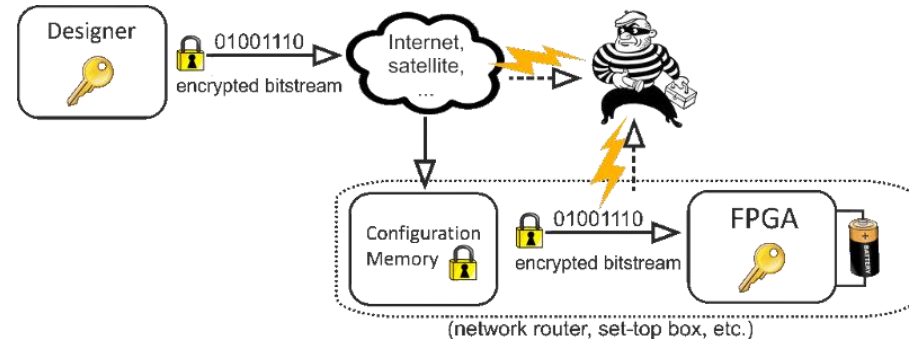
■ Encryption

- Used to prevent interception (Prevention)
- Limitations:
 - Key is exposed to a lot of people.
 - Battery is needed to “power on the Encryption key.
 - Vulnerable to side channel attacks.

Encryption: Secure FPGA, cont'

■ One Time Programmable FPGA

- ❑ No encryption and on-site battery is needed
- ❑ Configuration data is permanently burned in chip.
- ❑ Based on Antifuse technology.
- ❑ Limitations:
 - Expensive
 - Inflexible
 - Difficult to test



■ Non-volatile Reprogrammable FPGA

- ❑ Configuration data stored in on-chip non-volatile memory.
- ❑ Robust security scheme prevents readback of configuration data.
- ❑ Present technology even prevents direct probing of the memory cell.
- ❑ Limitations:
 - Extra non-standard voltages are needed to reprogramming.

Secure IP: Security Tagging

■ System composition

- Tag to transmit label of the IP core
- Detector to identify that label

■ Goal

- Identify the existence of the IP core and determine its version

■ Method

- E.g., Covert transmission channel (obfuscation)

Secure IP: Watermarking

- **Why Watermarking?**
 - Uniquely identify IP Cores
 - Ease of detection of piracy and counterfeiting
 - Trace pirated parts back to the source
 - Deter piracy

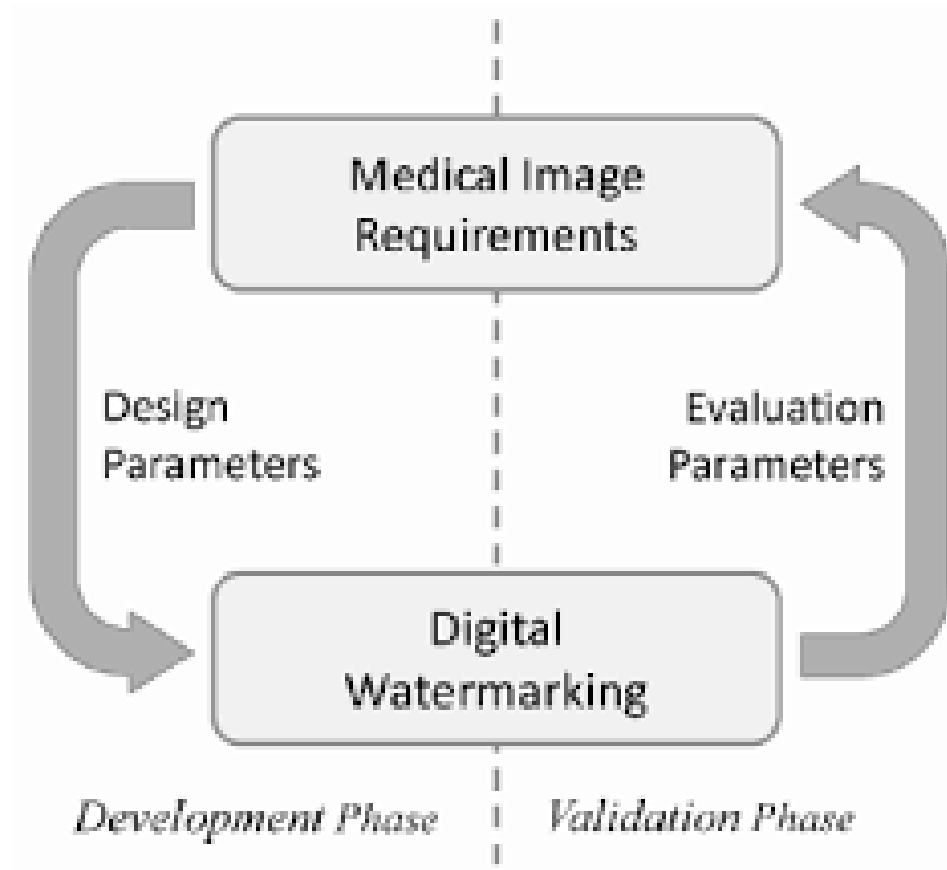
Digital vs. IP Watermarking

■ Digital Watermarking

- ❑ Performed on data: Image, audio, video, ...
- ❑ Insert unique signature into the data
- ❑ Data carries a unique identifier
- ❑ **Invasive**: the data is altered by the watermark



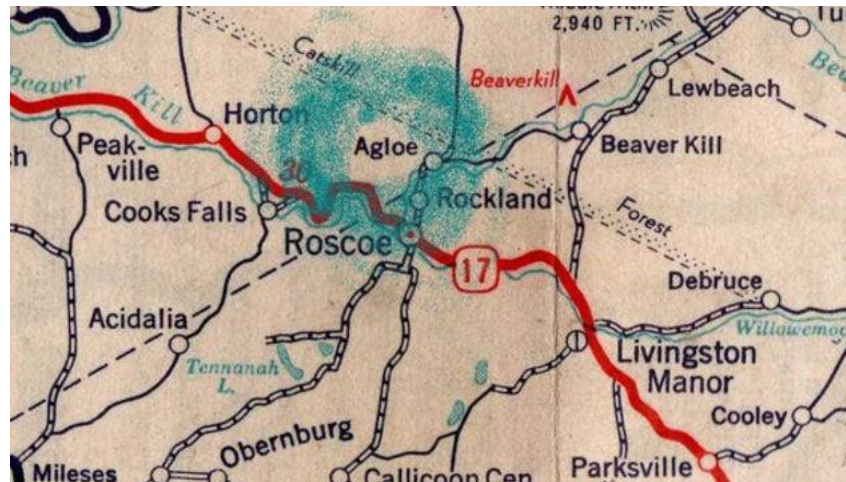
Watermarking Validation



Digital vs. IP Watermarking Example

■ Cartographer's Watermark

- ❑ "Trap streets" can be used on maps
- ❑ However, this cannot enter navigational data (e.g. GPS) lest hurt the functionality



Digital vs. IP Watermarking

■ IP Watermarking

- ❑ Performed on IP (*RTL, Gate level, or GDSII*)
- ❑ **Noninvasive**: data can't be altered otherwise the function will be altered

```
If rising_edge(Clk) then
  SyncA <= (SyncA(1), SyncA(0), SwitchA);
  SyncB <= (SyncB(1), SyncB(0), SwitchB);
  SyncC <= (SyncC(1), SyncC(0), SwitchC);
  StateA <= SyncA(2) and SyncA(1);
  StateB <= SyncB(2) and SyncB(1);
  StateC <= SyncC(2) and SyncC(1);
  If rising_edge ( StateA) then
    If StateB = '0' then
      Direction_Buf <= '1'; -- set dir clockwise
    ElseIf StateB = '1' then
      Direction_Buf <= '0';
    End if;
  elsif Falling_Edge (StateA) then
    If StateB = '1' then
      Direction_Buf <= '1';
    Elsif StateB = '0' then
      Direction_Buf <= '0';
    end if;
  ElseIf Falling_edge (StateB) then
    If StateA = '0' then
      Direction_Buf <= '1';
    ElseIf StateA = '1' then
      Direction_Buf <= '0';
    End If;
  ElseIf Rising_edge (StateB) then
    If StateA = '1' then
      Direction_Buf <= '1';
    Elsif StateA = '0' then
      Direction_Buf <= '0';
    End if; -- set dir anti clockwise
  End If;
End if;
```

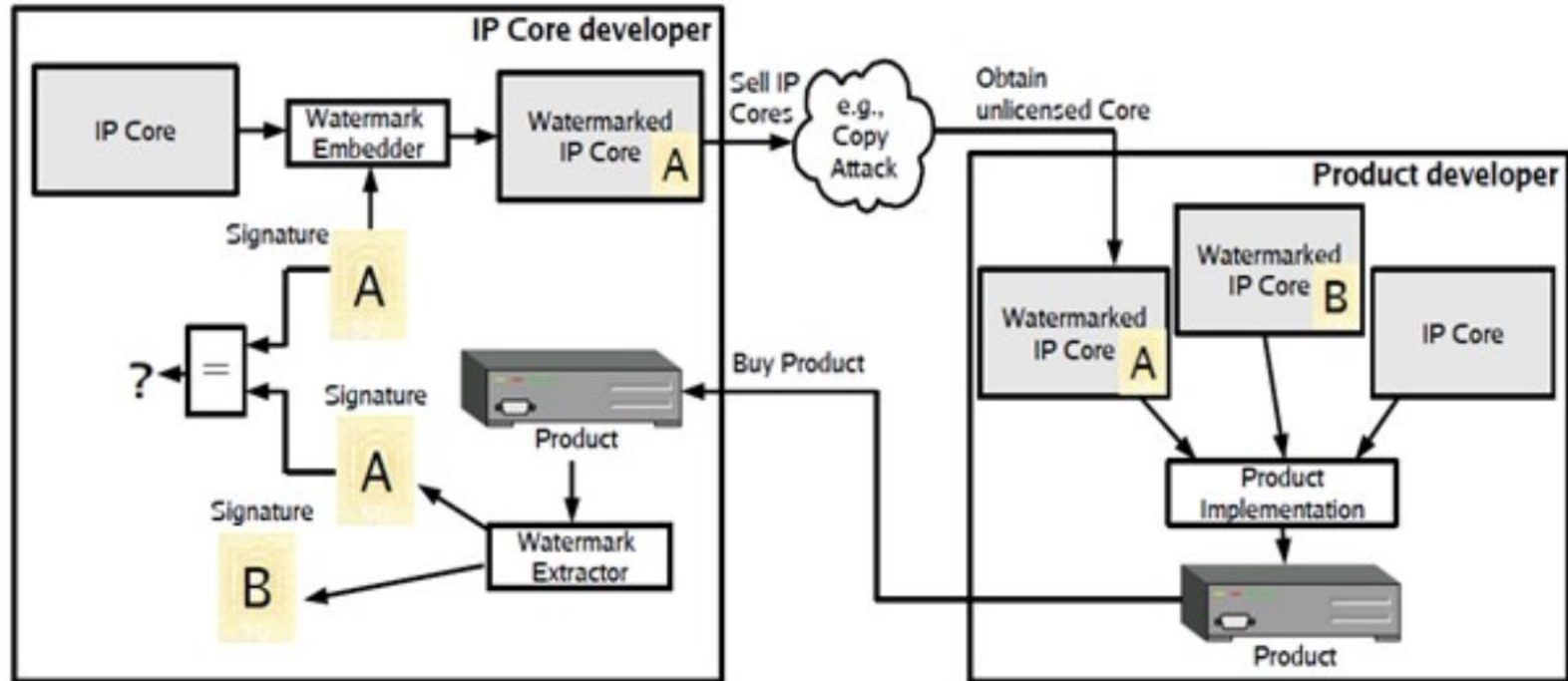
IP Core Watermarking: Principles

- Must not ***alter*** the functionality of the IP Core
- ***Performance degradation*** should be unnoticeable
- Should be ***hard*** to detect or remove
- Overall goal: high probability of authorship

Topics in IP Watermarking

- **Major approaches**
 - Constraint-based watermarking
 - Additive watermarking
- **Evaluation** of watermarking technique
- **Method of application**
 - Test-based watermarking
 - Don't Care Condition watermarking
 - Power Analysis watermarking
 - Placement and Route-based watermarking
- **Attacking and defending watermarks**

IP Watermarking Flow



Additive Watermarking

■ Method

- ❑ Done at source code level, independent from layout constraints
- ❑ Implanted into functional logic so as to prevent removal

■ Advantage

- ❑ Strength easily adjustable
- ❑ Hard to discover or remove
- ❑ Easy to read in stego key
 - Stego key refers to key used to encrypt/decrypt author signatures.

■ Weaknesses

- ❑ Could potentially incur area overhead, depending on the implementation
 - ❑ Could degrade performance
-

Don't-Care Condition Based Watermarking

■ Method formulation

- ❑ Function blocks that have unneeded input combinations
- ❑ Outputs to these input combinations can be forced
- ❑ Don't care: in Logic and FSM

■ Application

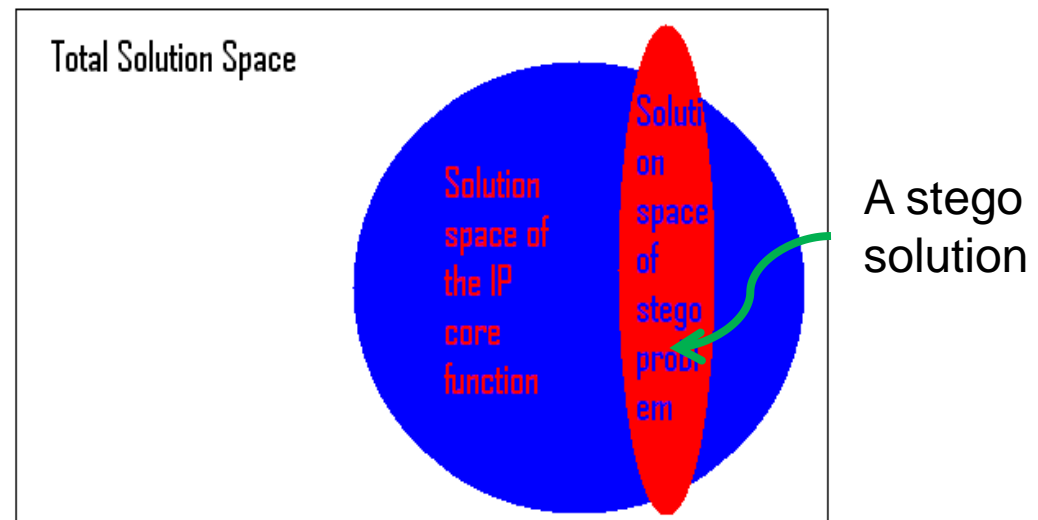
- ❑ To assert a '1', add one such input combination
- ❑ To assert a '0', remove this input combination

■ Example

- ❑ Original function $f(a,b,c,d) = \bar{a}\bar{b}\bar{c} + \bar{a}bd + b\bar{c}d$
- ❑ To assert $\bar{a}\bar{b}\bar{c}\bar{d} = '1'$, change it to $f(a,b,c,d) = \bar{a}\bar{b}\bar{c} + \bar{a}bd + b\bar{c}d + \bar{a}\bar{b}\bar{c}\bar{d}$

Constraint Based Watermarking

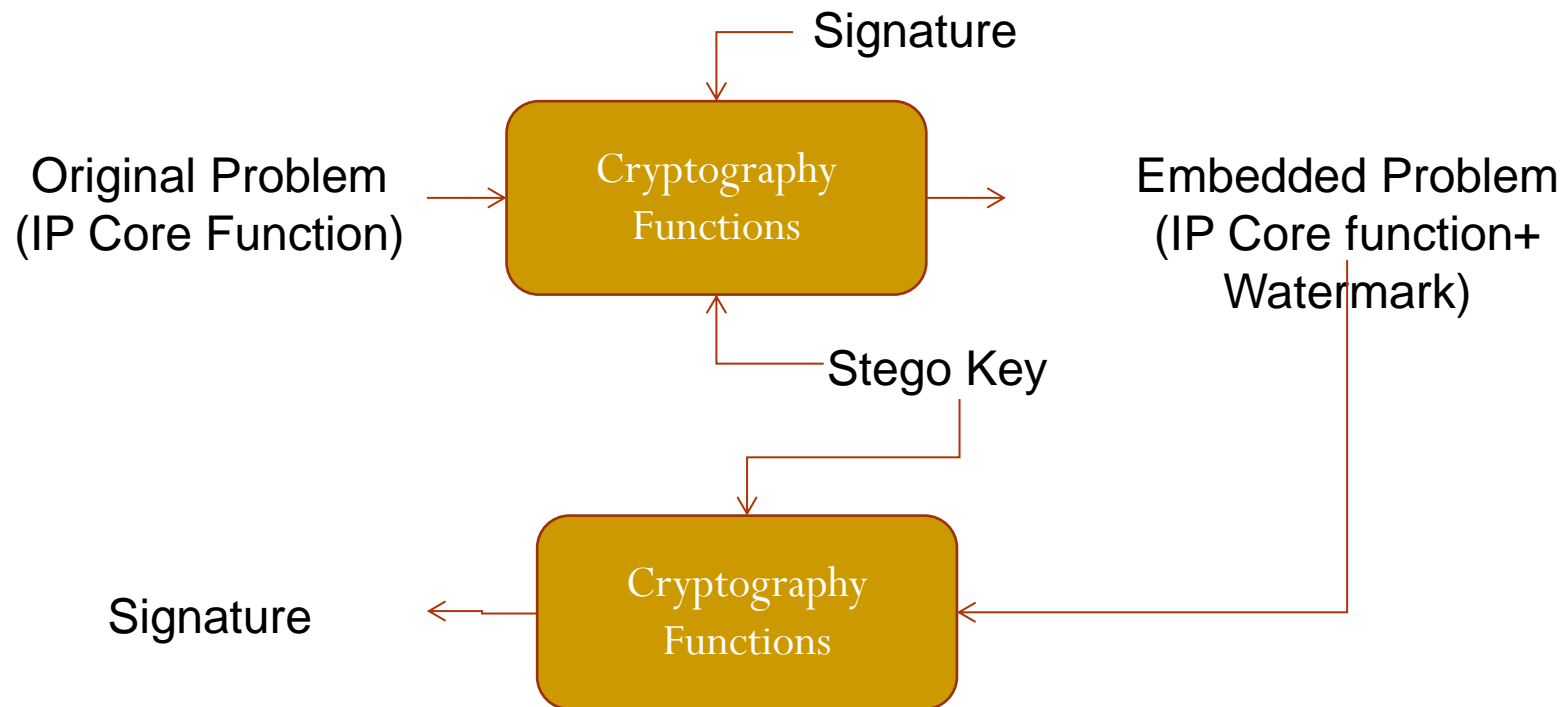
- Author's signature \rightarrow a set of constraints
 - Watermarking constraints in addition to functional constraints
 - More constraints, higher probability of authorship
- Choice of constraint must not impact performance



Constraint Based Watermarking, cont'

■ Example methods

- ❑ Unused CLB (Combinational Logic Block) blocks
- ❑ Path timing constraints



Advantages and Disadvantages

■ **By constraining CLB block**

- ❑ - Easy to discover through reverse engineering
- ❑ - Easy to remove from bitstream
- ❑ + No performance degradation and minimal area overhead

■ **By constraining path timing**

- ❑ + Sub path can also be used
- ❑ + Hard to detect or remove
- ❑ + No performance degradation, minimal area overhead

Testing based Watermarking

■ Method

- ❑ Add watermark to the test circuitry at functional level

■ Extraction

- ❑ Switch to scan more, scan out the signature

■ Advantage

- ❑ Very tough to remove

■ Weaknesses

- ❑ Potentially large area overhead, depending on the implementation
- ❑ Power consumption overhead

Example:

1. Run the circuit in function mode for some number of clock cycles
2. Switch to test mode
3. Scan out the content

Other Testing based Watermarking

■ **Methods**

- ❑ Apply watermark as a header
 - Easy to implement, but easy to remove
- ❑ Pseudorandom bit insertion
 - Harder to implement, added obscurity
- ❑ XOR the watermark and test bits
 - Easy to implement, secure, but prone to errors

Power Analysis Watermarking

■ Method

- Add components that draw power at certain frequencies (not multiples of clock)

■ Extraction

- Monitor the power supply pin on the IC
- Dynamic power consumption magnitude will be higher at certain frequencies

■ Advantage

- Very tough to remove

■ Weaknesses

- Large area overhead
- Power consumption overhead
 - Depending on the implementation

Place-and-Route based Watermarking

■ Method

- Region and grouping constraints
 - Constrain the physical placement of standard cells
 - Can cause performance degradation when not used properly
- Row placement constraints
 - Deliberately place standard cells in even or odd rows of the layout with a grid abstraction

Place-and-Route based Watermarking, cont'

■ Advantage

- ❑ + Negligible area overhead
- ❑ + Strong proof of authorship easily attainable
 - Coincidence chance = 1 over 2 to the power of number of blocks placed
- ❑ Removal or circumvention will most likely render the IP Core useless
 - IP core thus protected are hard/firm IP, therefore tampering requires reverse engineering its soft IP

■ Weakness

- ❑ Not applicable to soft IP