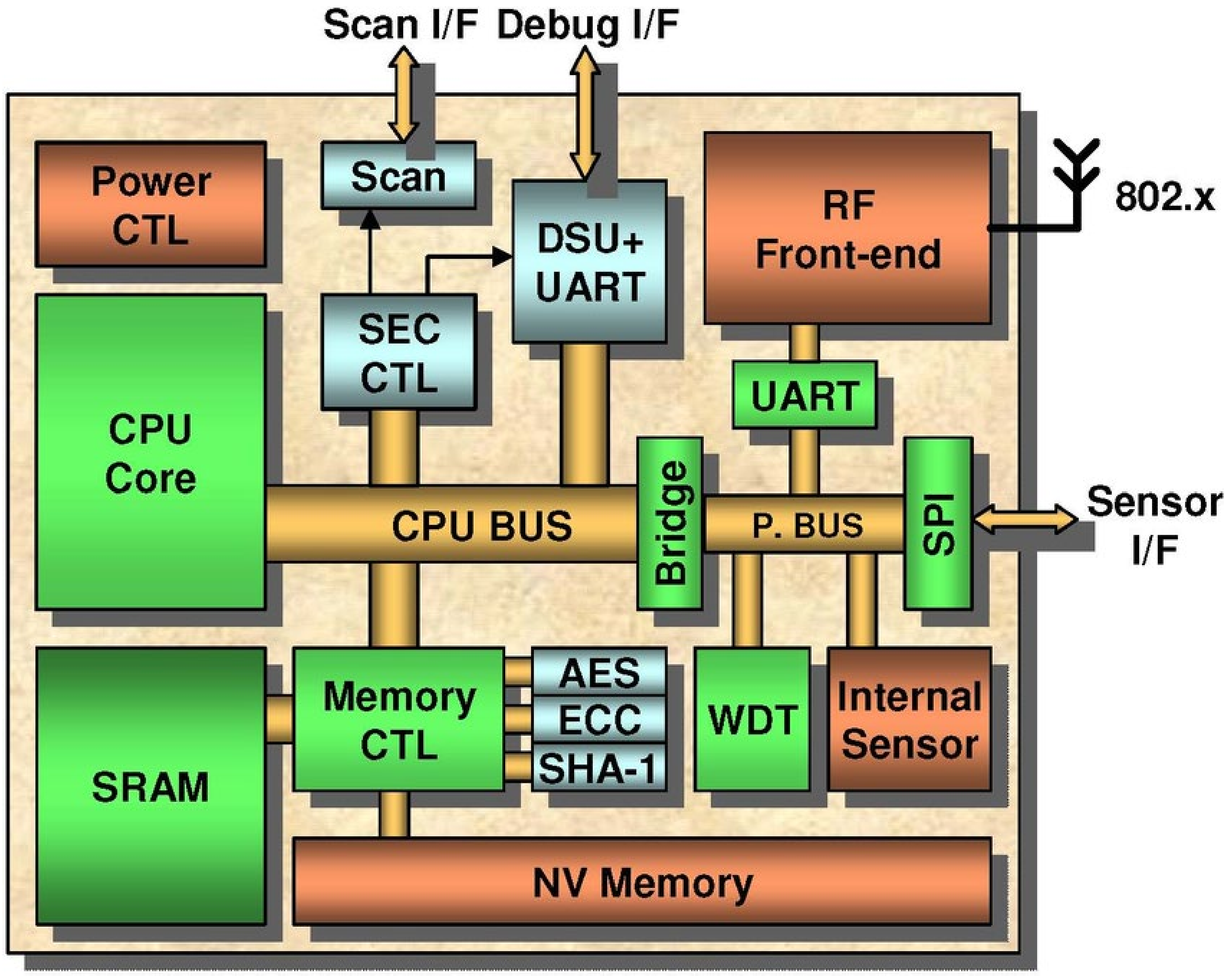
HW watermarking

Prof. Maurizio MARTINA

IP based design

- Complex Systems on Chip
- IPs from different companies
  - SOFT IP
    - HDL code
  - HARD IP
    - layout (gdsII file)
- Design flow:
  - IP designers
  - System integrator
  - Foundry



Taken from “Resistive switching behavior in TiN/HfO2/Ti/TiN devices” by D. Walczyk et al.

Vulnerabilities

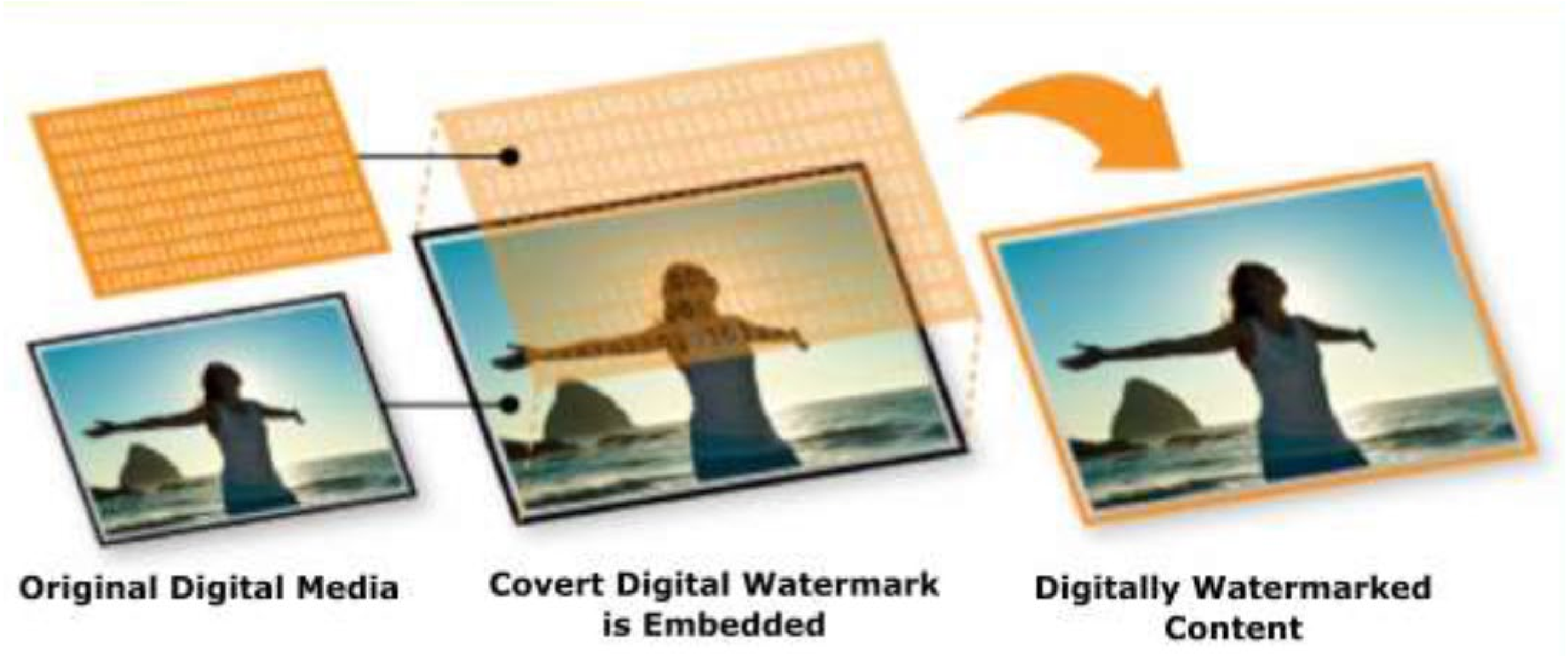
- Each of the actors (IP designer, system integrator, foundry) can make reverse engineering on part/full system.
- Different approaches:
  - Physical Attacks (e.g. side channel attacks; microarchitectural vuln.)
  - Trojan Horses (implemented at different design levels)
  - IP Piracy (cloning of IP)
  - IC Piracy & Counterfeiting (cloning, overproduction)
  - Backdoors (modifications leaking secret)
  - Tampering (e.g. FPGA bitstream modifications)
  - …

Watermarking

- Working principle: hide a signature in an IP
  - Cannot prevent IP infringement, piracy, or over-production of ICs
  - Suitable only for proving intellectual property use
- Requirements:
  - not alter the functionality of the IP Core
  - unnoticeable performance degradation
  - hard to detect or remove
  - verifiable, the owner must be able show that the IP has his/her signature

Note

- Digital watermarking techniques do not work
  - Data is altered !


Original Digital Media
Covert Digital Watermark is Embedded
Digitally Watermarked Content

Techniques

- Static watermarking
  - Requires reverse engineering
  - Intrusive and expensive
  - Constraint based watermarking
- Dyanmic watermarking
  - Run the design with specific input pattern
  - FSM based watermarking
  - Testing based watermarking
  - Side channel based watermarking

Techniques

- Static watermarking
  - Requires reverse engineering
  - Intrusive and expensive
  - Constraint based watermarking
- Dyanmic watermarking
  - Run the design with specific input pattern
  - FSM based watermarking
  - Testing based watermarking
  - Side channel based watermarking

Constraint based watermarking - idea

- Add constraints in the design flow to obtain a signature
- It is a constraints satisfaction problem
  - Usually an NP-hard problem
- Assuming to impose C constraints each with probability p of being satisfied
  - Let X be a random variable that represents how many of the constraints are not satisfied
  - The probability to have b (or less) constraints not satisfied is

$$P(X \leq b) = \sum_{i=0}^{b} \binom{C}{i} p^{C-i} \cdot (1-p)^{i}$$

Constraint based watermarking - list

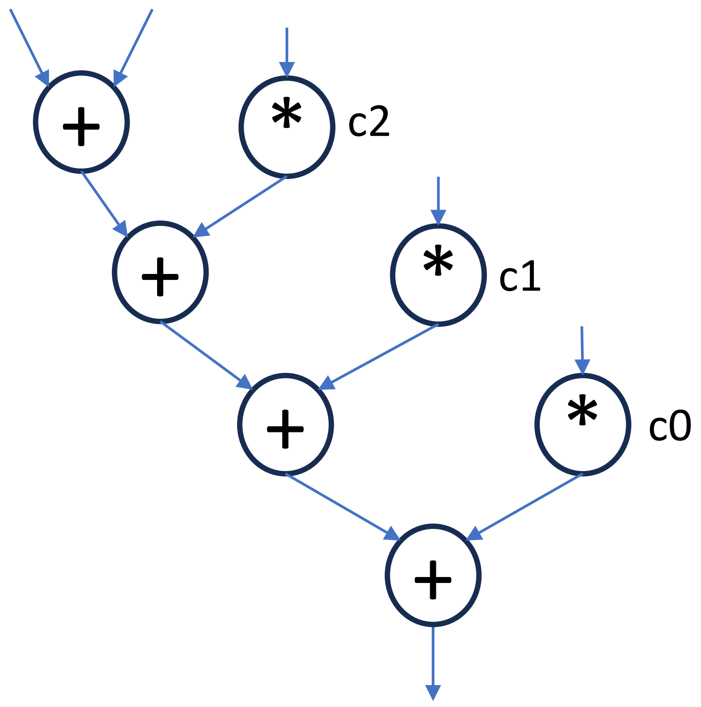
- Add constraints at:
  - System level
  - Register allocation
  - Synthesis level
  - Physical synthesis (P&R) level

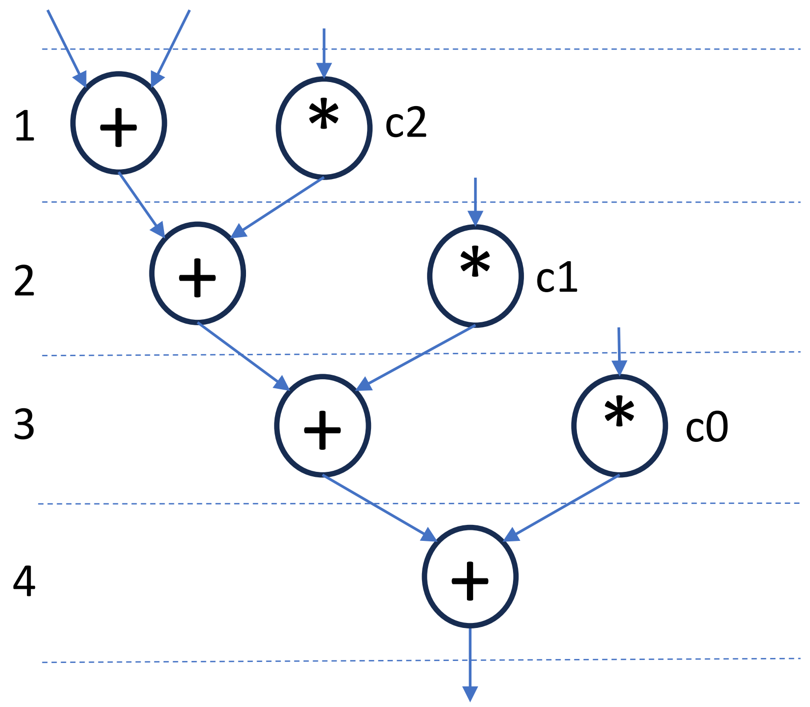
Constraint based watermarking

- Add constraints at:
  - System level
  - Register allocation
  - Synthesis level
  - Physical synthesis (P&R) level

System level

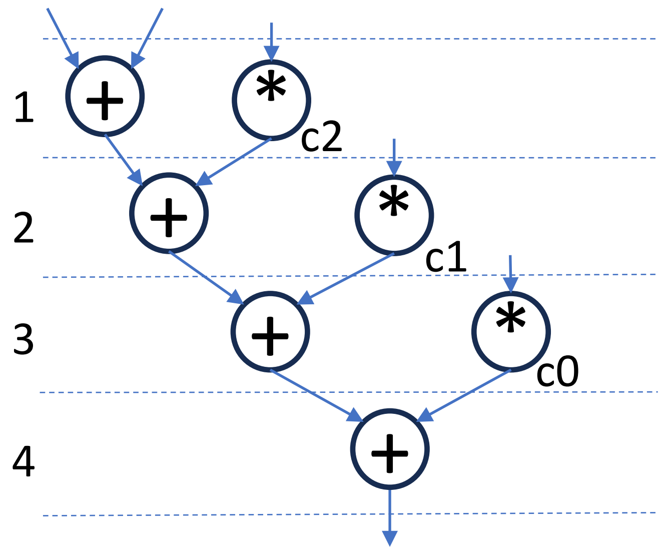
- Example: add constraints to operations schedule

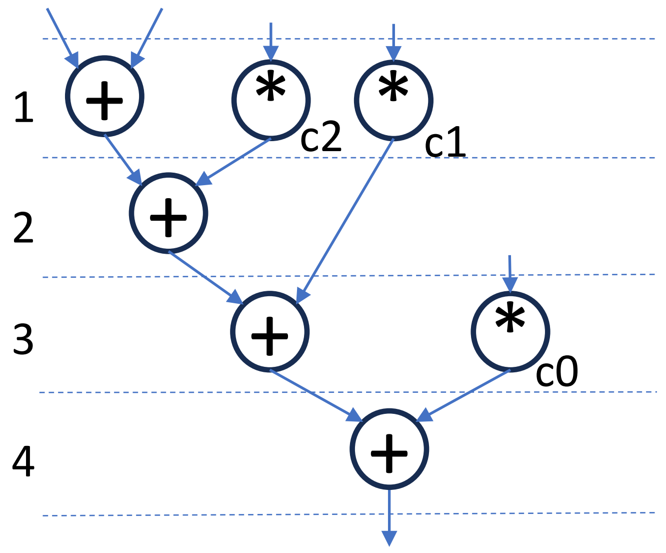


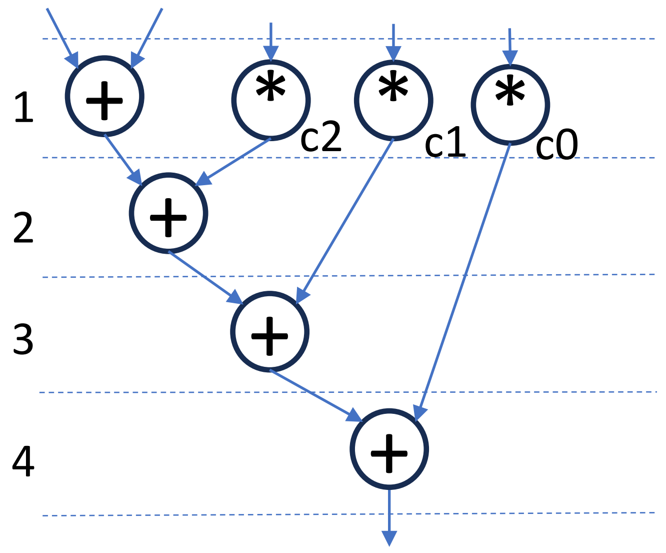


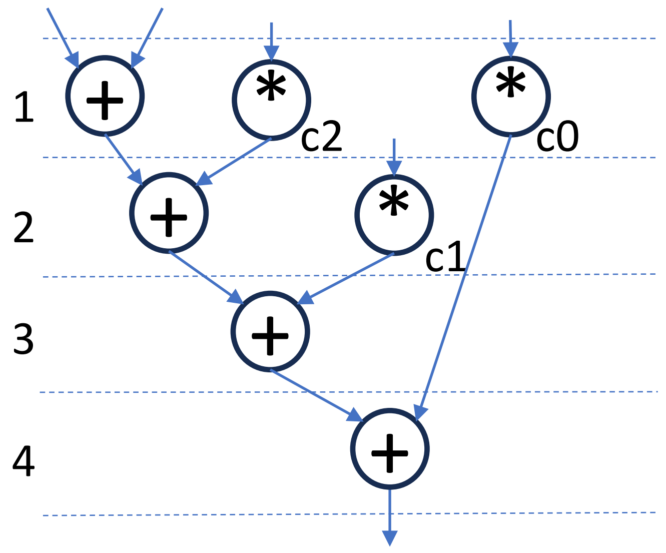
I. Hong et al. "Techniques for intellectual property protection of DSP designs"

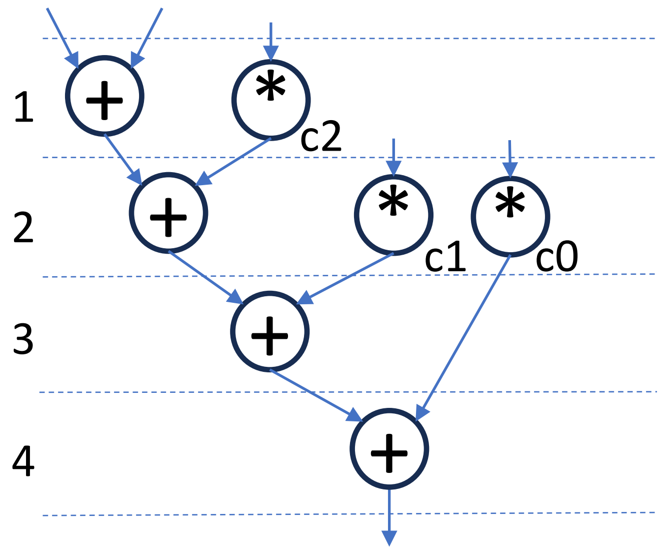
Code a scheduling sequence as signature

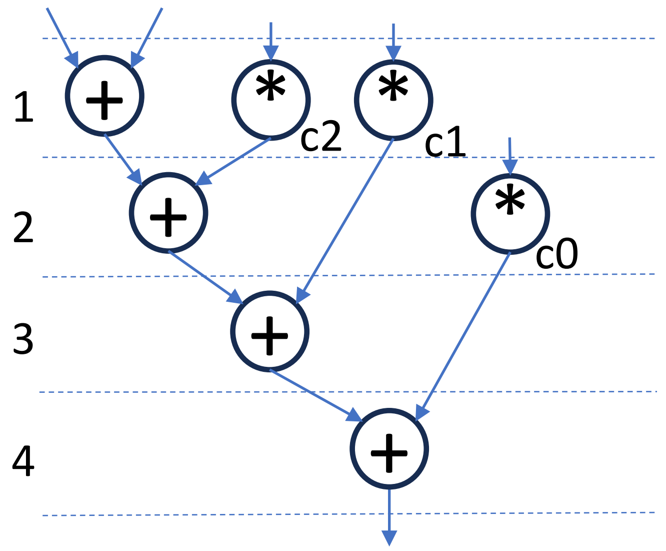












Constraint based watermarking

- Add constraints at:
  - System level
  - Register allocation
  - Synthesis level
  - Physical synthesis (P&R) level

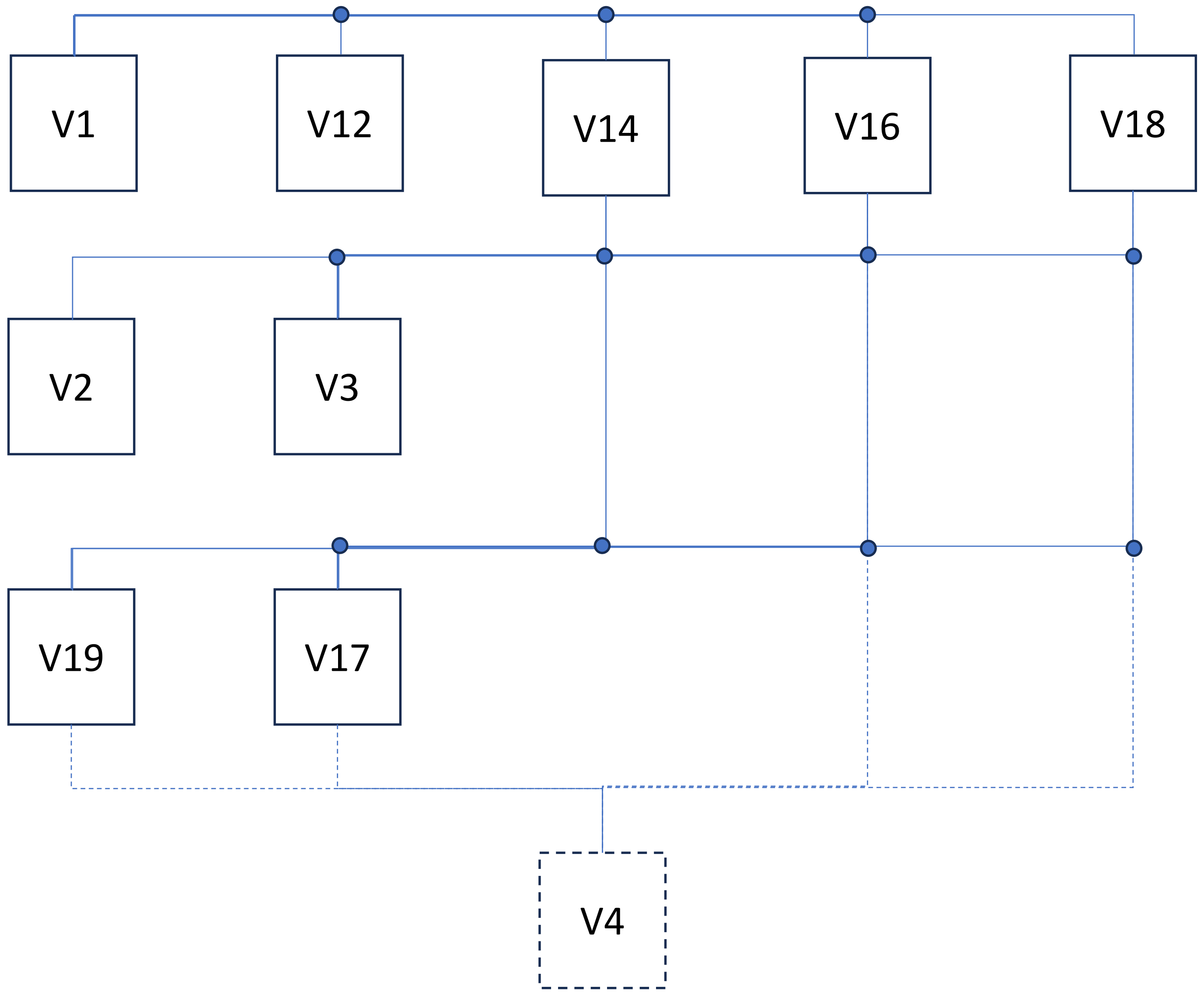
Register allocation

- Given an algorithm which produces intermediate results: how can registers be reused ?
  - Define temporary variables duration
  - If two temporary variables do not overlap in duration, they can share a register
  - The problem can be solved as a graph colouring problem
  - Add constraints in the coulouring to embed a signature

F. Koushanfar et al. “Behavioral Synthesis Techniques for Intellectual Property Protection”

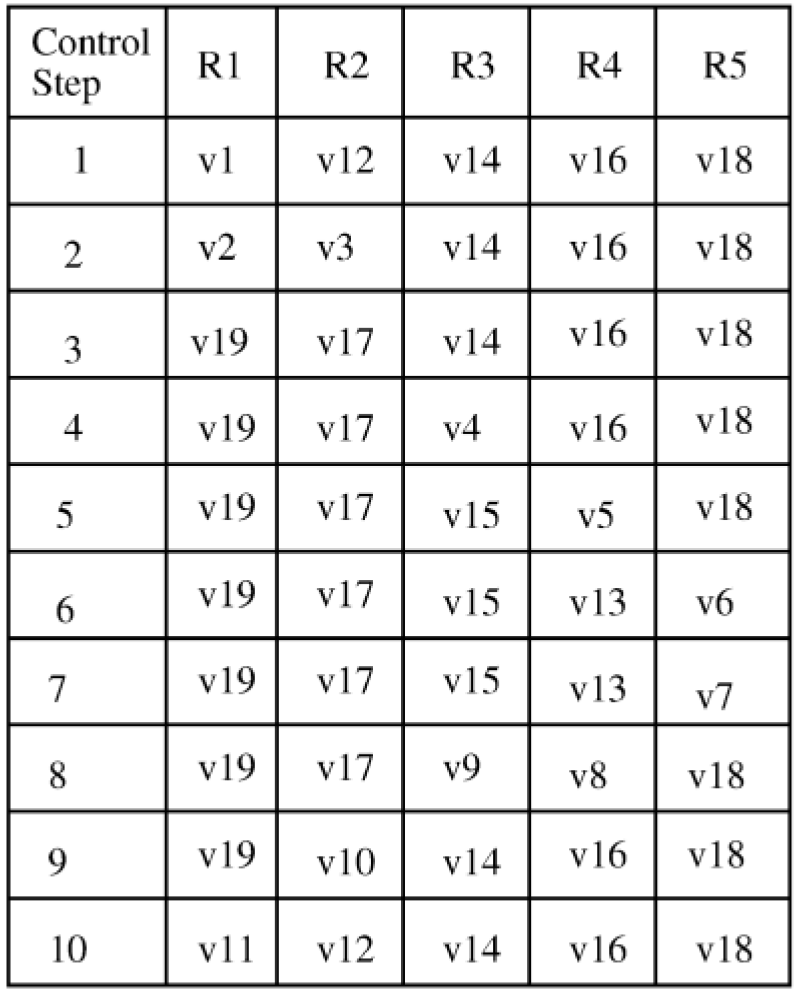
Example

Control Step	R1	R2	R3	R4	R5
1	v1	v12	v14	v16	v18
2	v2	v3	v14	v16	v18
3	v19	v17	v14	v16	v18
4	v19	v17	v4	v16	v18
5	v19	v17	v15	v5	v18
6	v19	v17	v15	v13	v6
7	v19	v17	v15	v13	v7
8	v19	v17	v9	v8	v18
9	v19	v10	v14	v16	v18
10	v11	v12	v14	v16	v18



Graph colouring problem

- The problem has a 5-register constraint
- The graph contains several cliques of size 5
  - It can be coloured with 5 colours
- Add constraints to fix the colouring strategy
- Sort nodes in increasing duration order.
  1. v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12
  2. v13
  - …



Control Step	R1	R2	R3	R4	R5
1	v1	v12	v14	v16	v18
2	v2	v3	v14	v16	v18
3	v19	v17	v14	v16	v18
4	v19	v17	v4	v16	v18
5	v19	v17	v15	v5	v18
6	v19	v17	v15	v13	v6
7	v19	v17	v15	v13	v7
8	v19	v17	v9	v8	v18
9	v19	v10	v14	v16	v18
10	v11	v12	v14	v16	v18

Embedding a signature

- Add a rule to connect nodes, to embedd a signature in the couloring of the graph.
- E.g.
  - Embed 0 if the new connection connects to an even node
  - Embed 1 if the new connection connects to an odd node

Example - I

- Embed the signature 10000010110111
- Sorted nodes in increasing duration order.
    1. v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12
    2. v13
    - …

Example - II

- Embed the signature 10000010110111
- Sorted nodes in incresing duration order.
  1. v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12
  2. v13
  - …
- New connection to the next odd node
  - v1 → v3

Example - III

- Embed the signature 10000010110111
- Sorted nodes in incresing duration order.
    1. v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12
    2. v13
    - …
- New connection to the next even node
    - v1 → v3
    - v2 → v4

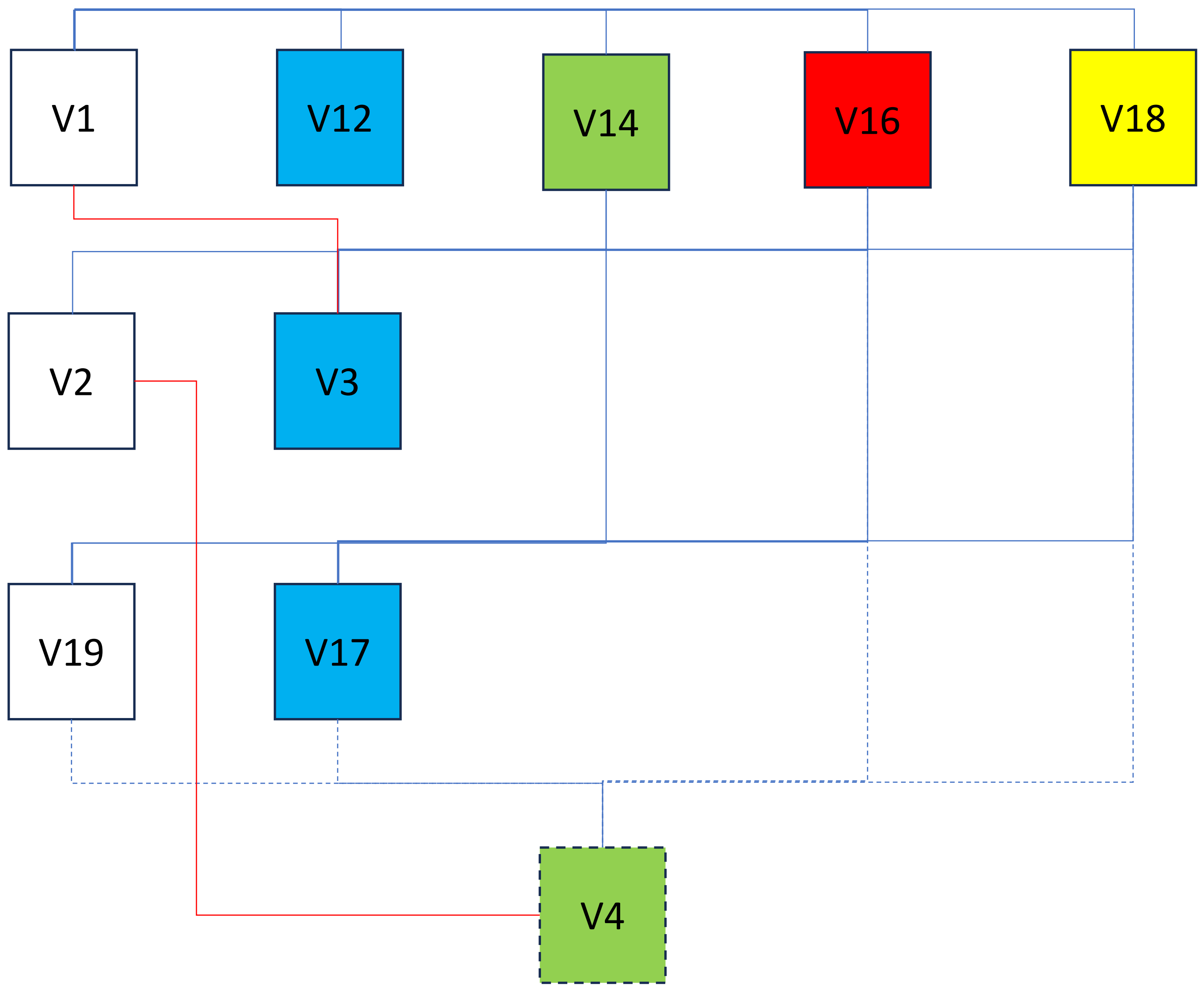
Example - III

- Embed the signature 10000010110111
- Sorted nodes in incresing duration order.
  1. v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12
  2. v13
  - …
- New connection to the next even node
  - v1 $\rightarrow$ v3
  - v2 $\rightarrow$ v4
  - v3 $\rightarrow$ v6

Example - IV

- Embed the signature 10000010110111
- Sorted nodes in incresing duration order.
    1. v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12
    2. v13
    - …
- New connection to the next even node
    - v1 → v3
    - v2 → v4
    - v3 → v6
    - v4 → v8

Result



Constraint based watermarking

- Add constraints at:
    - System level
    - Register allocation
    - Synthesis level
    - Physical synthesis (P&R) level

Synthesis level

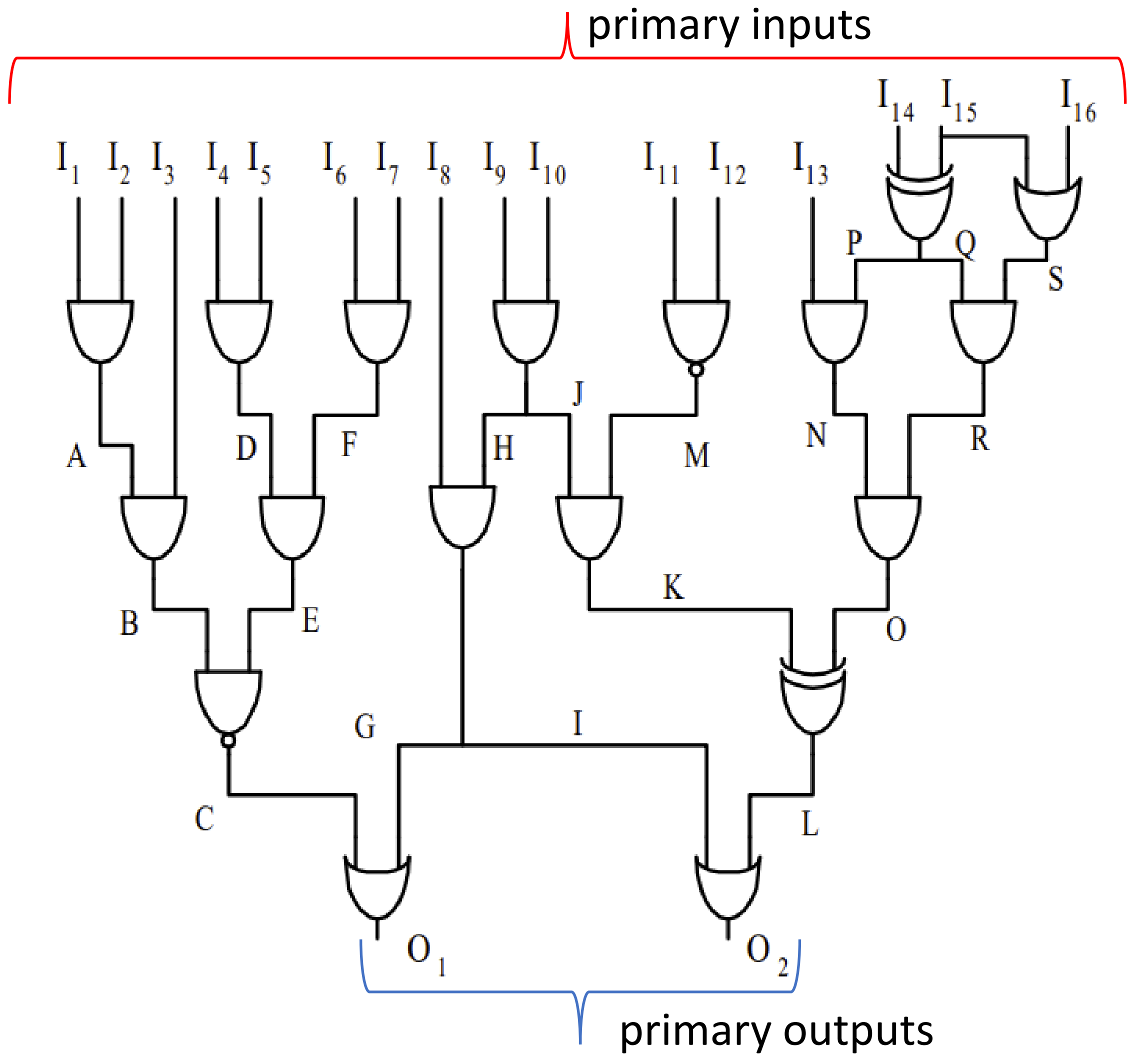
- Combinational logic synthesis is made of two main optimization steps:
  - Multilevel logic minimization
  - Technology mapping
- The longest path in a combinational circuit is the critical-path
- Assuming registers before and after the combinational circuit
  - The critical-path determines the maximum working frequency
- Idea: add watermarking in non-critical-paths

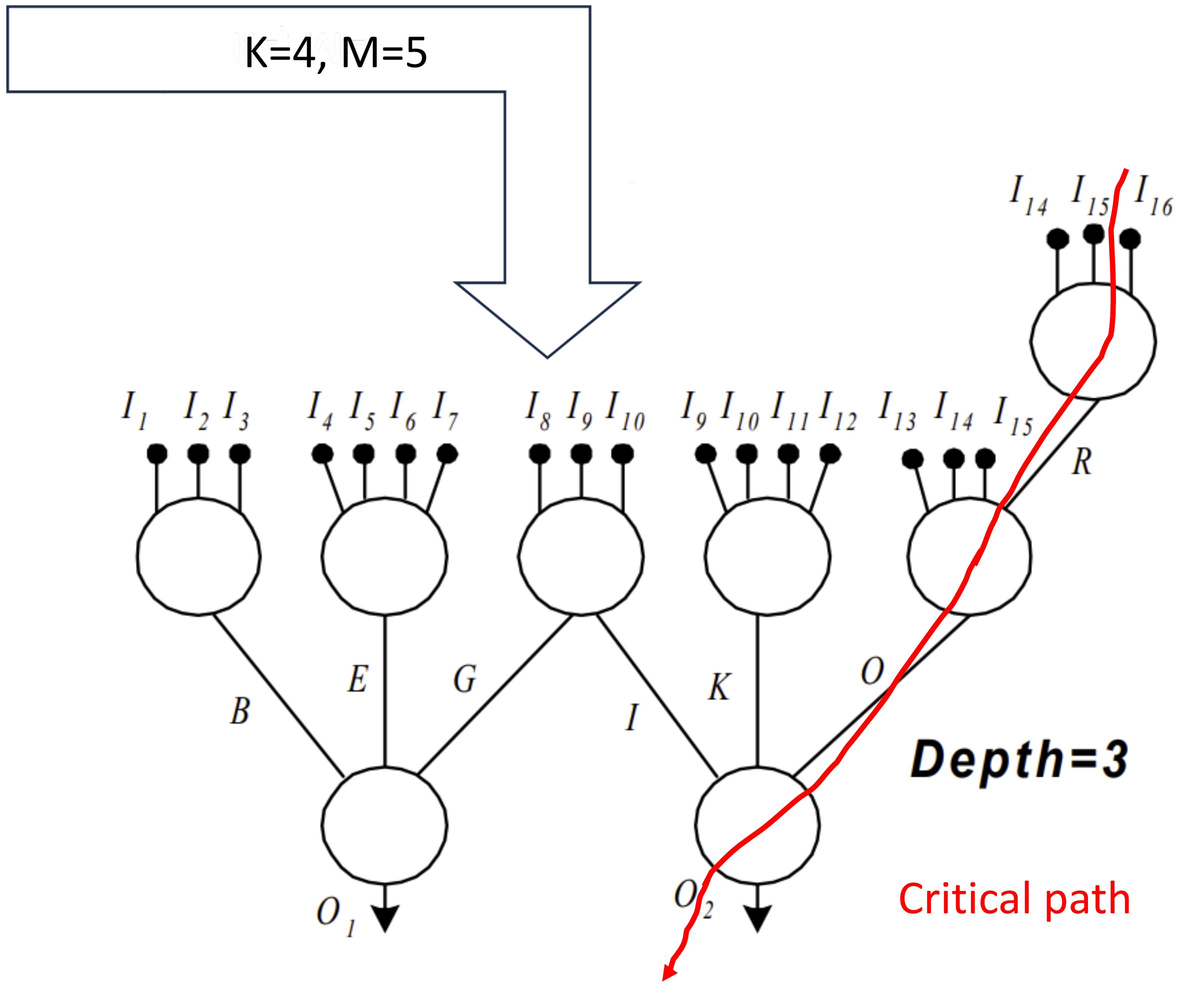
S. Meguerdichian et al. “Watermarking While Preserving The Critical Path”

K-M-macrocell mapping approach

- Assume:
  - logic minimization already done.
  - technology mapping constrained to
    - Physical block with at most K inputs
    - Boolean function with at most M product terms
      - E.g. $y = a \cdot b + a \cdot b + b \cdot c \rightarrow$ 2 sums and 3 products
  - Inputs of the block are primary inputs (I_x)
  - Outputs of the block are primary outputs (O_y)

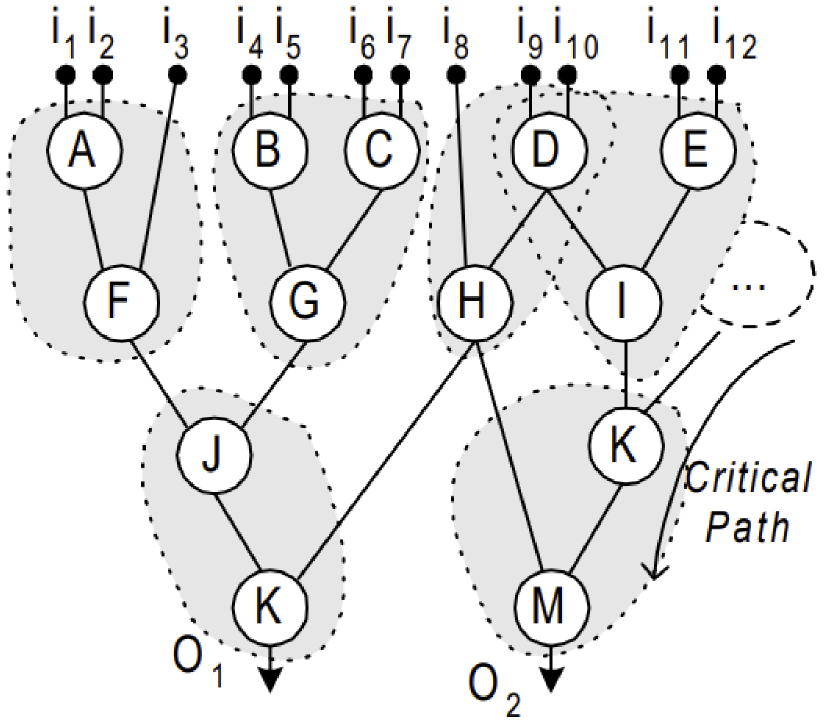
Example: logic circuit to map - I


primary inputs
I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9 I_{10} I_{11} I_{12} I_{13} I_{14} I_{15} I_{16}
P
Q
S
J
A
D
F
H
M
N
R
K
B
E
O
G
I
C
L
O_1
O_2
primary outputs



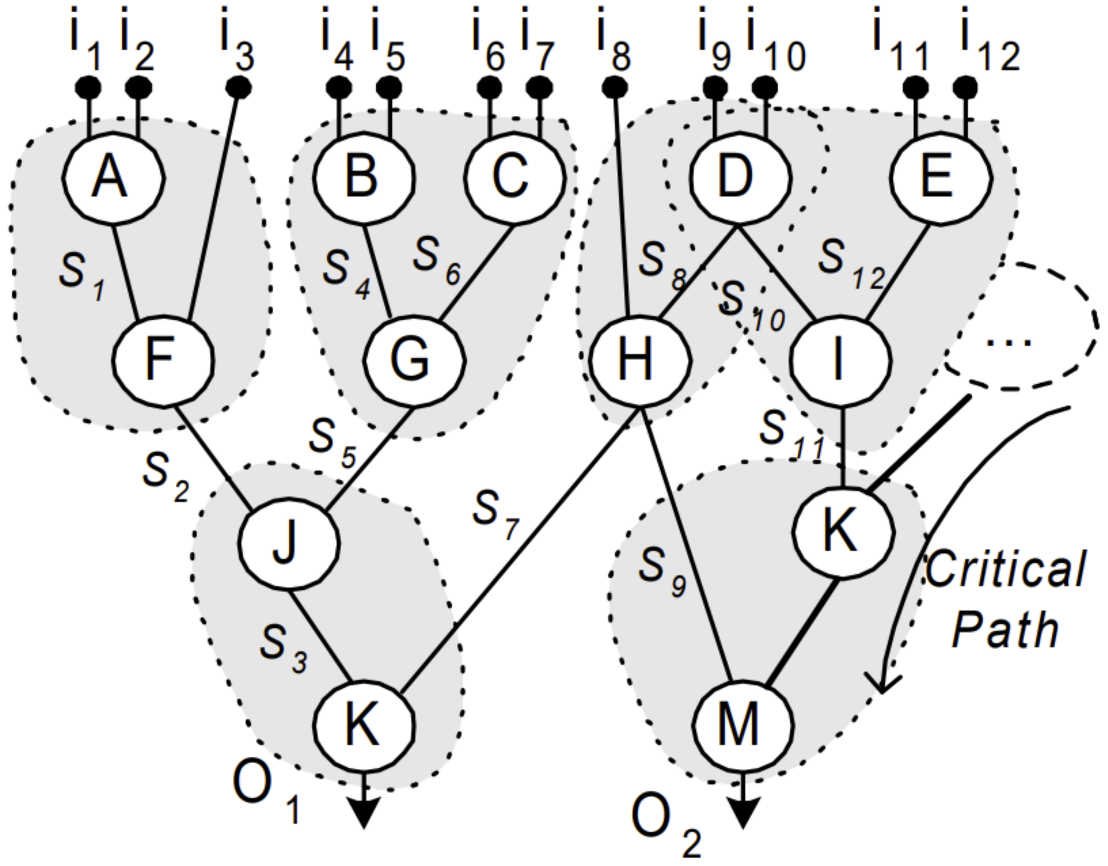
Example: logic circuit to map - II

- Assume that the critical path is in the (…) node
- The remaining nodes are mapped with K-M-macrocells



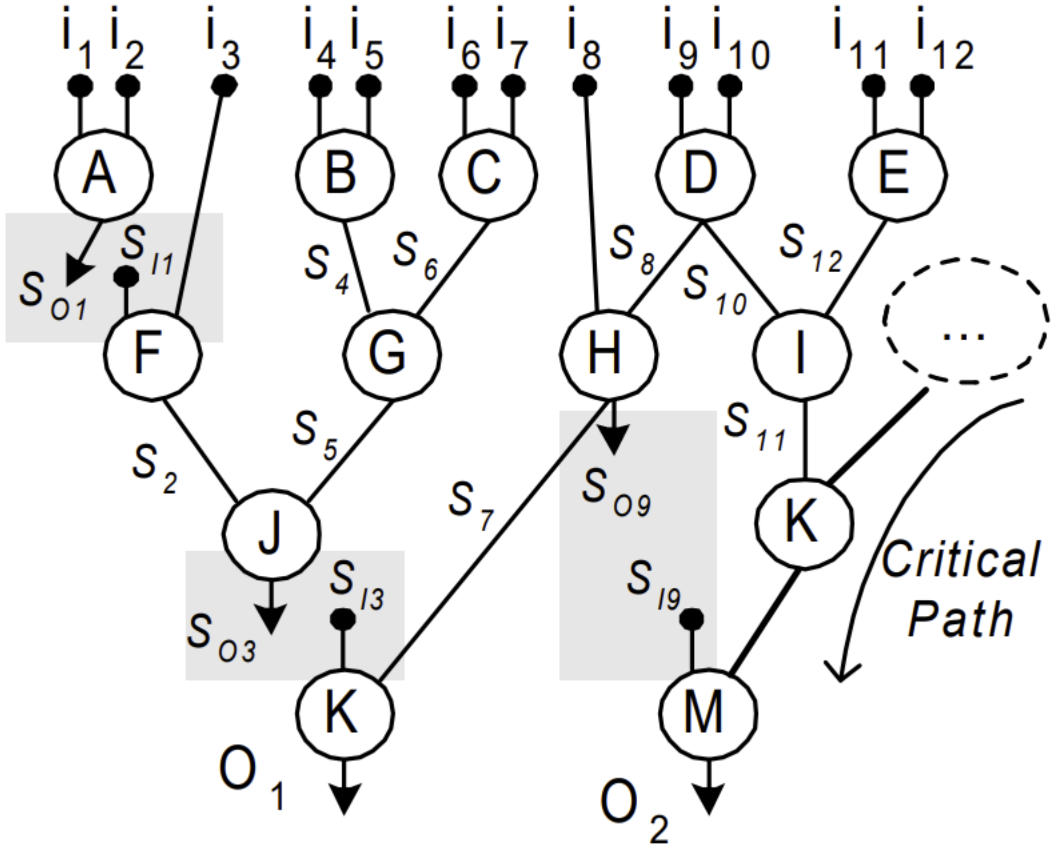
Example: logic circuit to map - III

- Assign labels to each non critical signal.



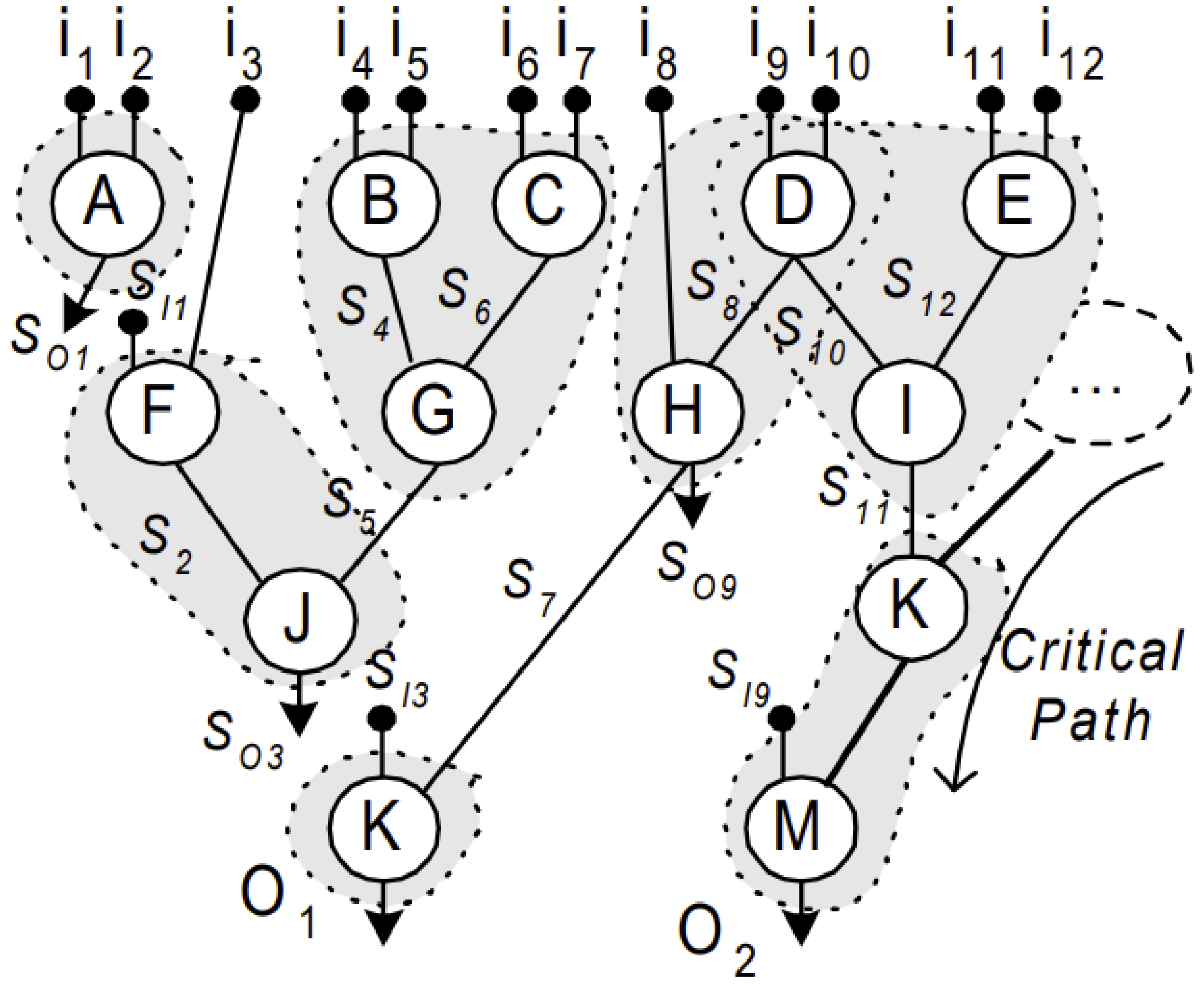
Example: logic circuit to map - IV

1. Identify some non critical signals to encode the watermark
   - E.g. with an hash function you map the signature to S1, S3 and S9
2. Remove current K-M-macrocell mapping
3. Break selected signals in pairs of primary inputs and primary outputs



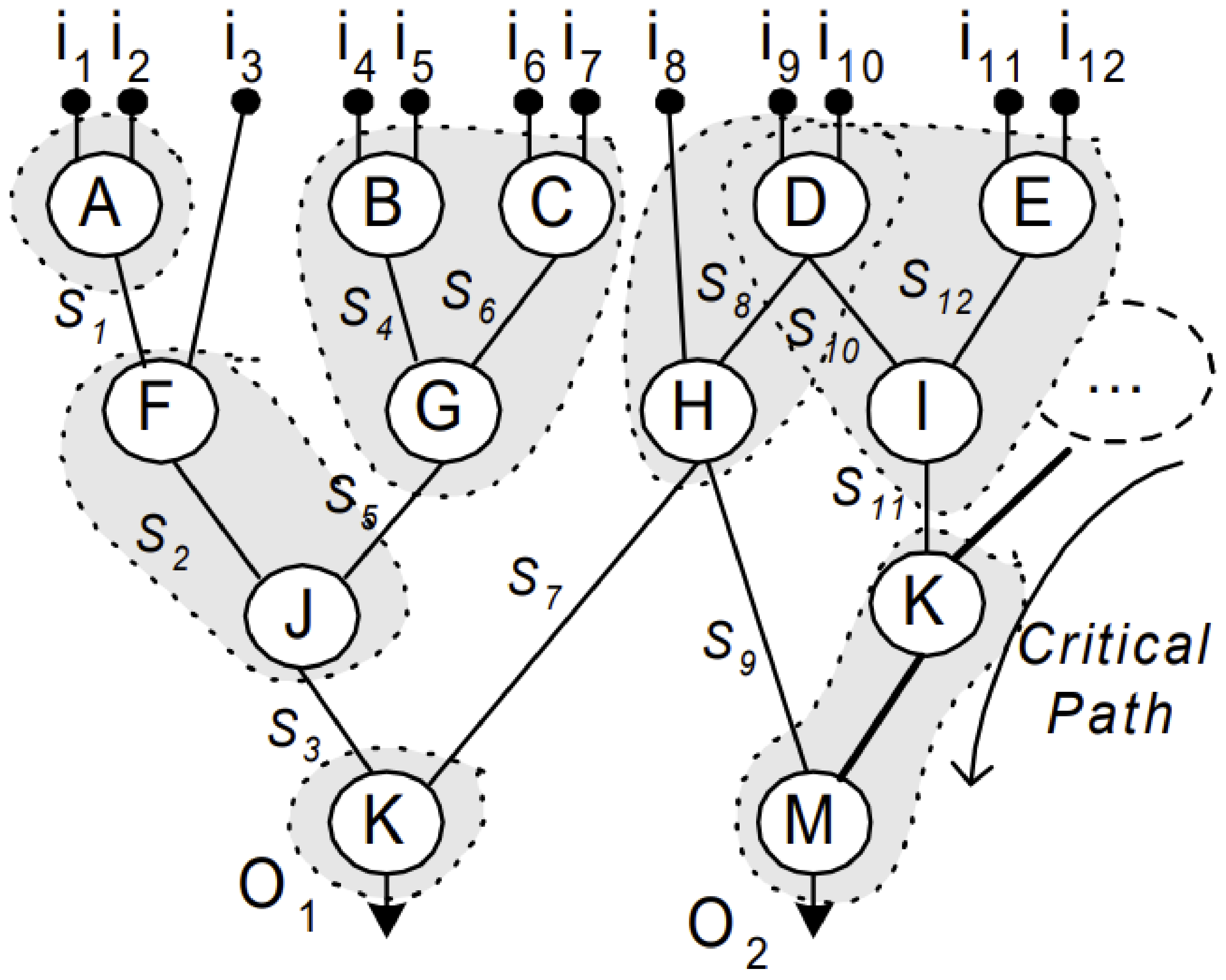
Example: logic circuit to map - V

4. Re-run the K-M-macrocell mapping



Example: logic circuit to map - VI

5. Patch the signals
   - E.g. S1, S3 and S9

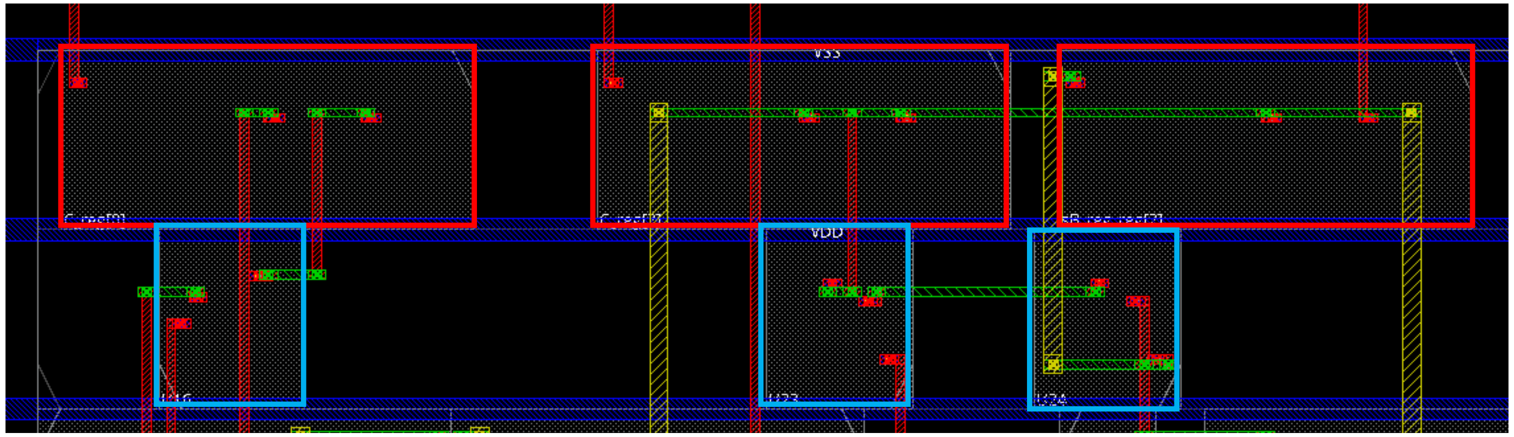

i_1 i_2 i_3 i_4 i_5 i_6 i_7 i_8 i_9 i_{10} i_{11} i_{12}
A
B
C
D
E
F
G
H
I
J
K
K
M
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12}
...
Critical Path
O_1
O_2

Constraint based watermarking

- Add constraints at:
  - System level
  - Register allocation
  - Synthesis level
  - Physical synthesis (P&R) level

Physical synthesis level

- FPGA: manipulate unused configurable logic blocks
- ASIC - placement: watermark the position of certain cells for a subset of the core
  - E.g.
    - Sequential cells only in first, third, fifth, … row
    - Combinational cells only in second, fourth, sixth, … row



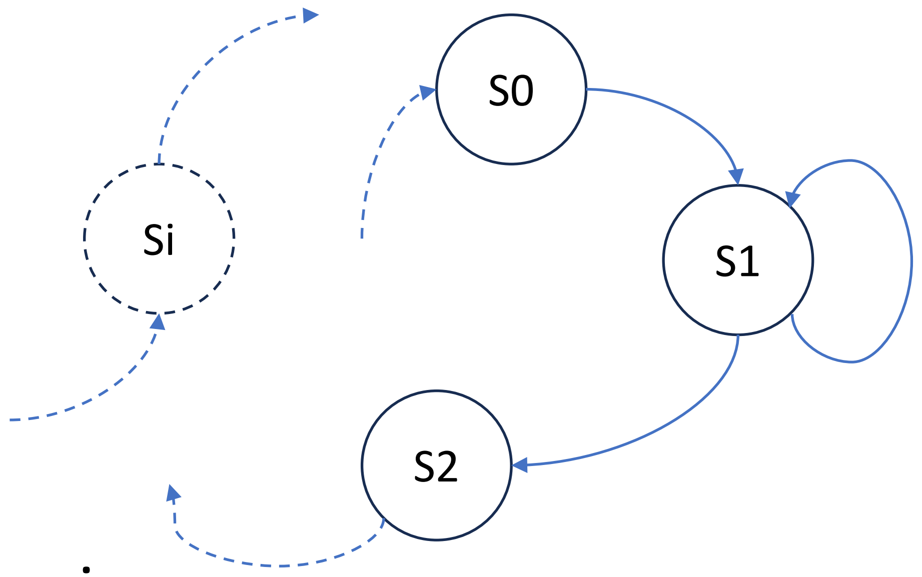
A. B. Kahng et al. “Constraint-Based Watermarking Techniques for Design IP Protection”

Techniques

- Static watermarking
  - Requires reverse engineering
  - Intrusive and expensive
  - Constraint based watermarking
- Dyanmic watermarking
  - Run the design with specific input pattern
  - FSM based watermarking
  - Testing based watermarking
  - Side channel based watermarking

FSM based watermarking - I

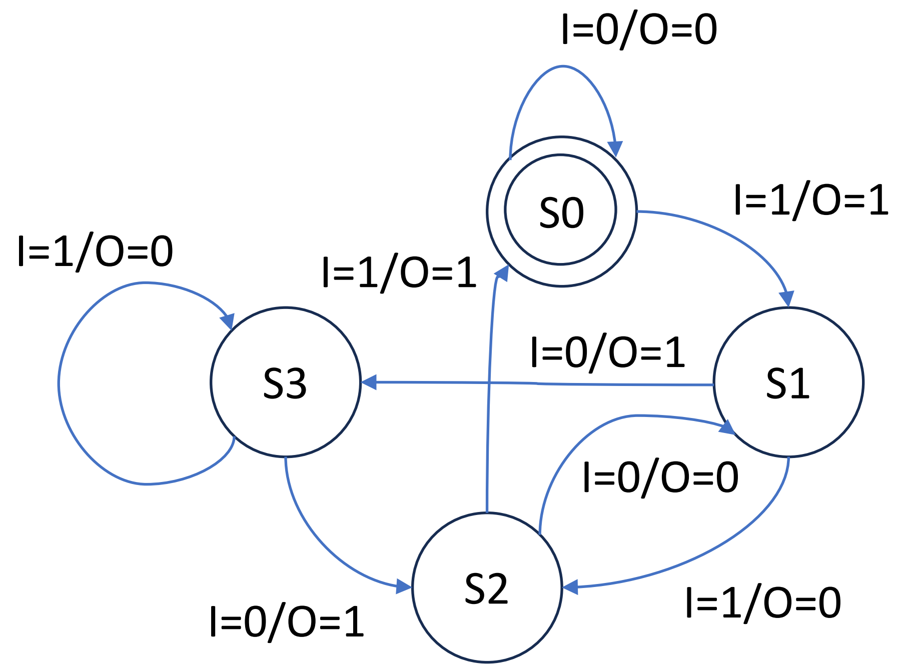
- Starting from STG representation of a FSM



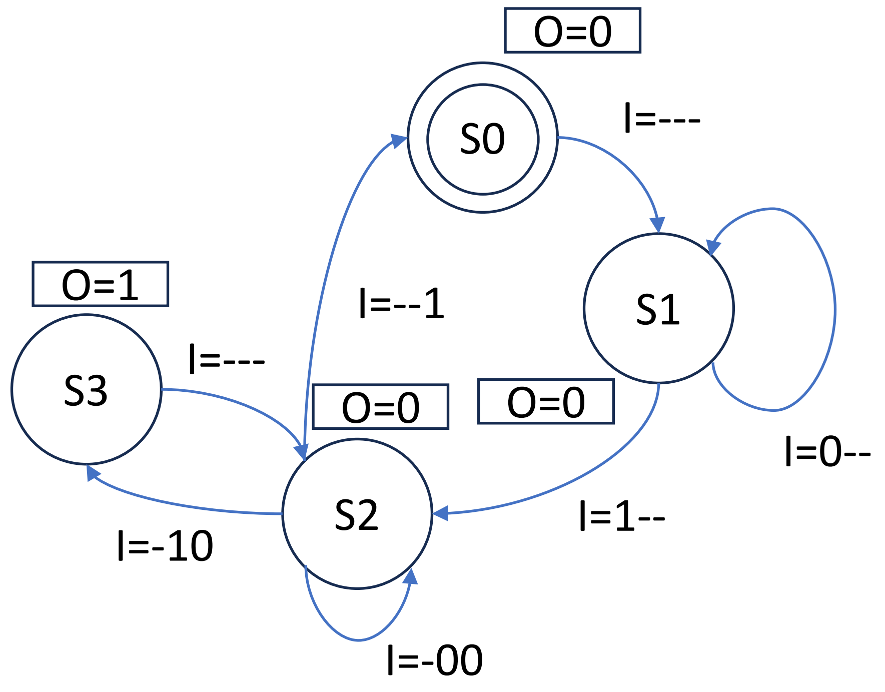
- Embed watermarking in
  - States
  - Transitions

Mealy Vs Moore FSM - I

- Formal difference



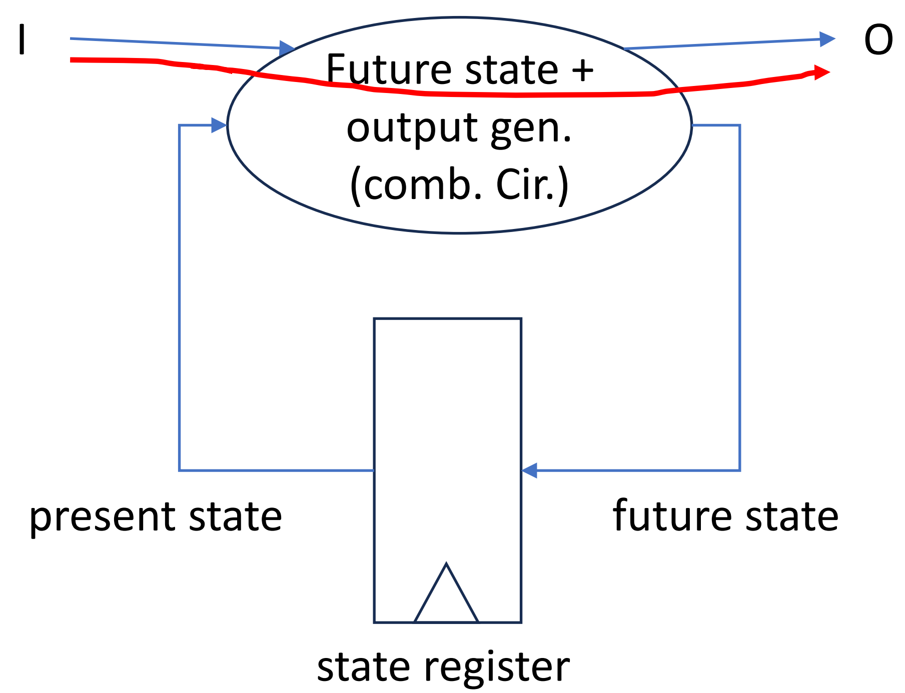
Mealy



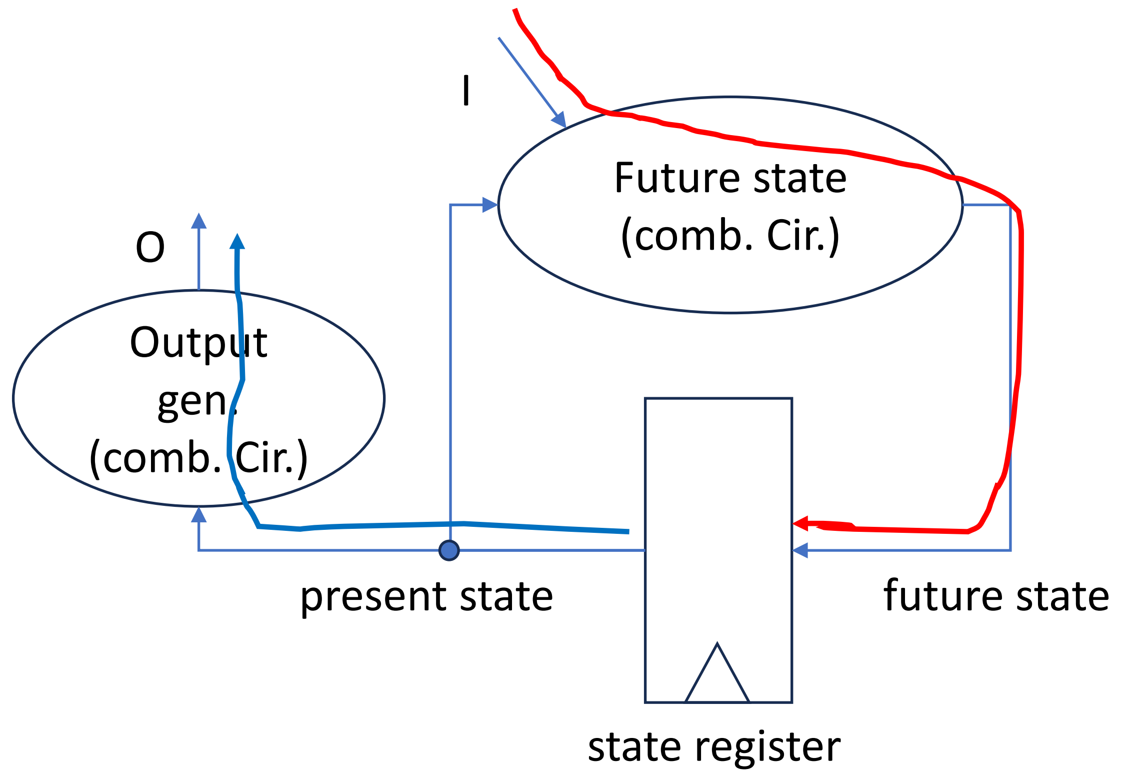
Moore

Mealy Vs Moore FSM - II

- Circuit difference



Mealy



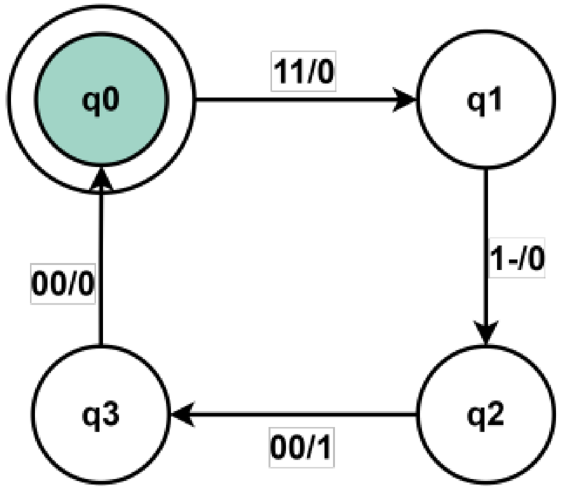
Moore

FSM watermarking - II

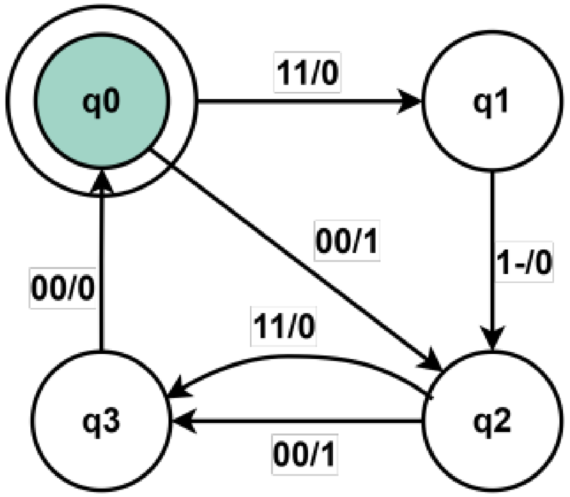
- State-based FSM watermarking
  - Add states to sign the FSM
    - Vulnerable to the state minimization/encoding
- Transition-based FSM watermarking
  - Exploit free inputs from the states in STG to generate extra transitions
  - The output carries the watermark
    - High overhead, need for adding pseudo inputs
    - Applicable to Mealy FSMs
- Difficult to observe

Transition-based FSM watermarking

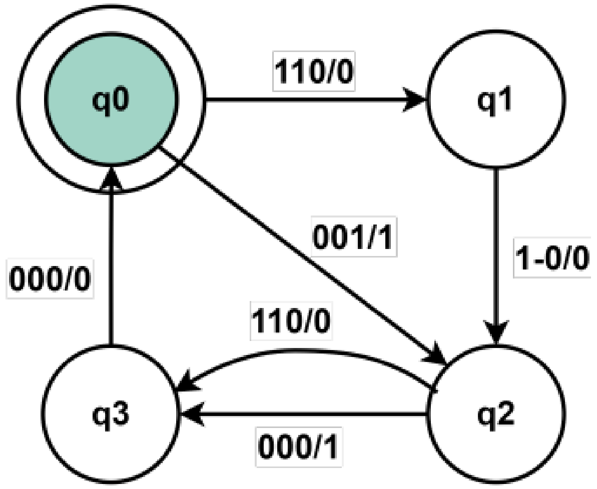
- Example



Original FSM



FSM – with new transitions



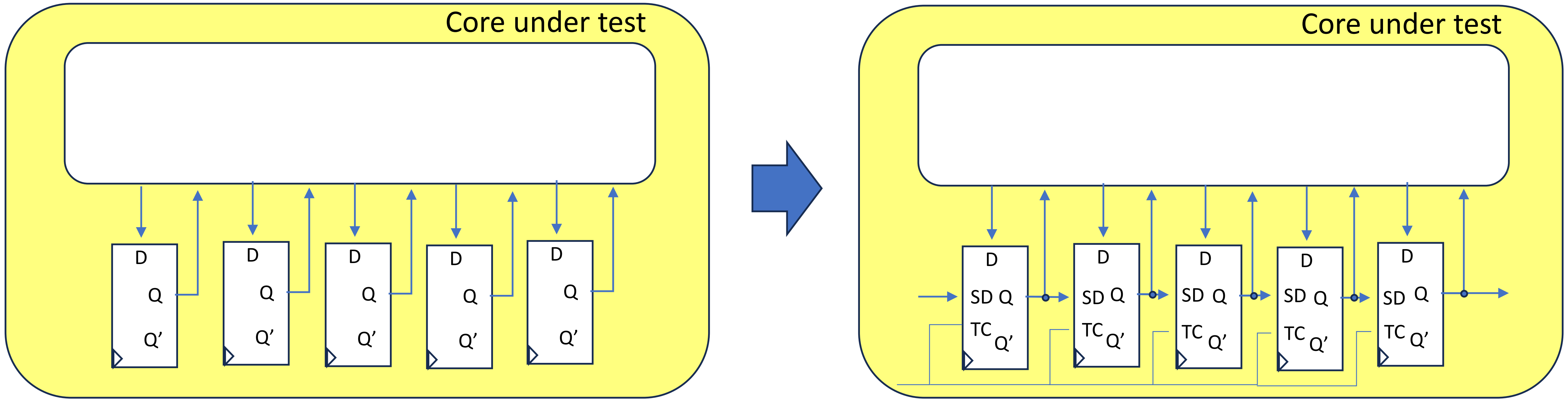
FSM – with input extension and new transitions

Techniques

- Static watermarking
  - Requires reverse engineering
  - Intrusive and expensive
  - Constraint based watermarking
- Dyanmic watermarking
  - Run the design with specific input pattern
  - FSM based watermarking
  - Testing based watermarking
  - Side channel based watermarking

Testing based watermarking - I

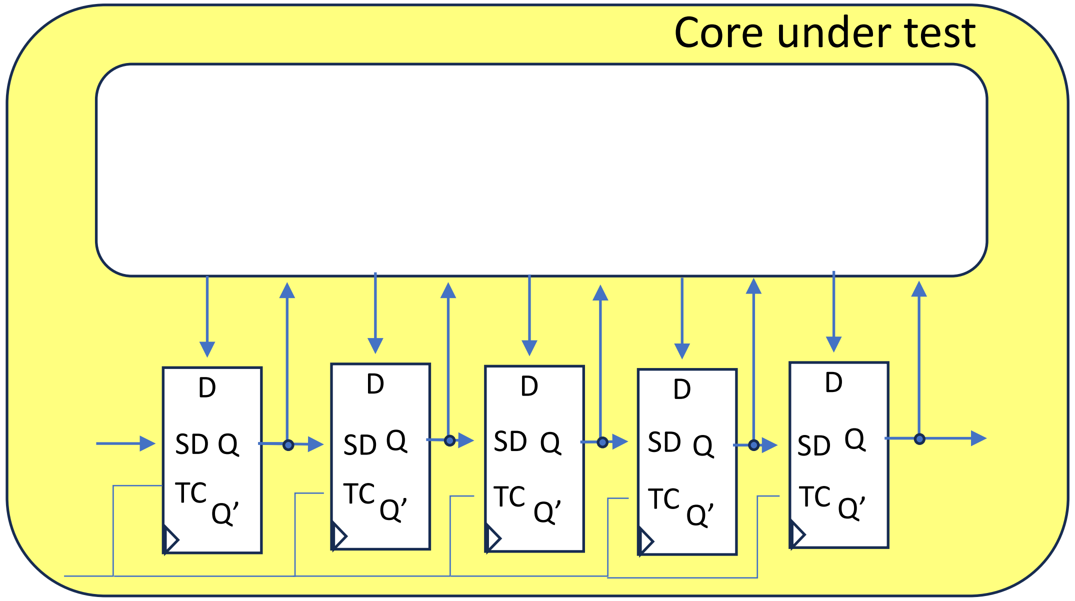
- Boundary scan-chain based techniques
- Design for testability approach
  - Replace D-type Flip-Flops with scan Flip-Flops



A. Cui et al. “Ultra-Low Overhead Dynamic Watermarking on Scan Design for Hard IP Protection”

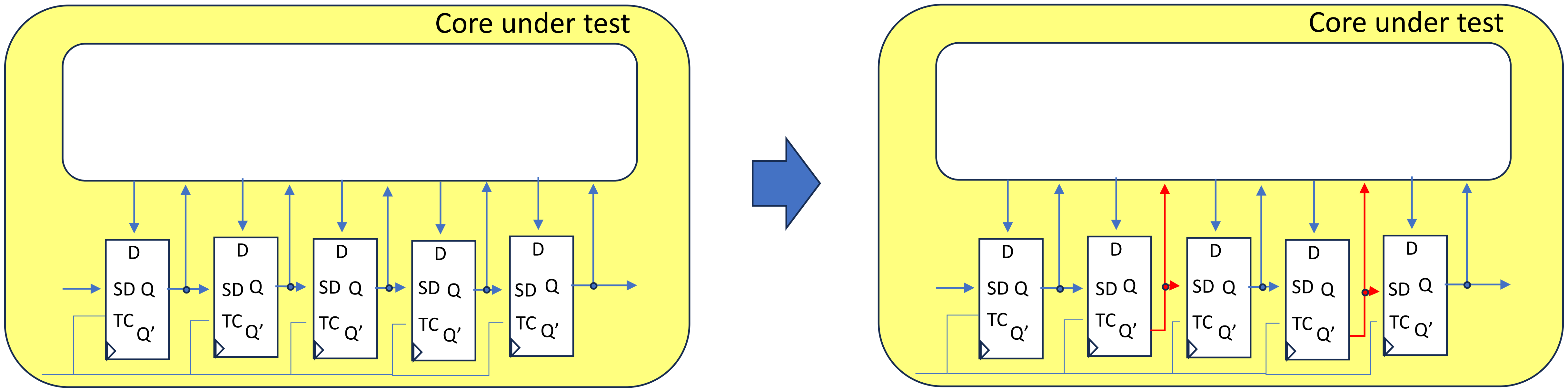
Testing based watermarking - II

- TC = ‘0’ → normal mode
- TC = ‘1’ → test mode
    - Write/Read a test pattern



Testing based watermarking - III

- Modify the boundary scan chain by adding a signature
  - Interaction with the synthesis tool
  - Computation of an input sequence to extract the signature



Techniques

- Static watermarking
  - Requires reverse engineering
  - Intrusive and expensive
  - Constraint based watermarking
- Dyanmic watermarking
  - Run the design with specific input pattern
  - FSM based watermarking
  - Testing based watermarking
  - Side channel based watermarking

Side channel based watermarking

- Often based on adding a power consumption signature
- The signature should be below the noise floor
- Examples:
  - Ring oscillators triggered by an input sequence
  - Leakage circuit driven by proper sequences:
    - Spread spectrum based watermark →Linear Feedback Shift Register (LFSRs)
    - Input-modulated watermark → xoring properly masked input bits

G. T. Becker et al. “Side-Channel Based Watermarks for Integrated Circuit”

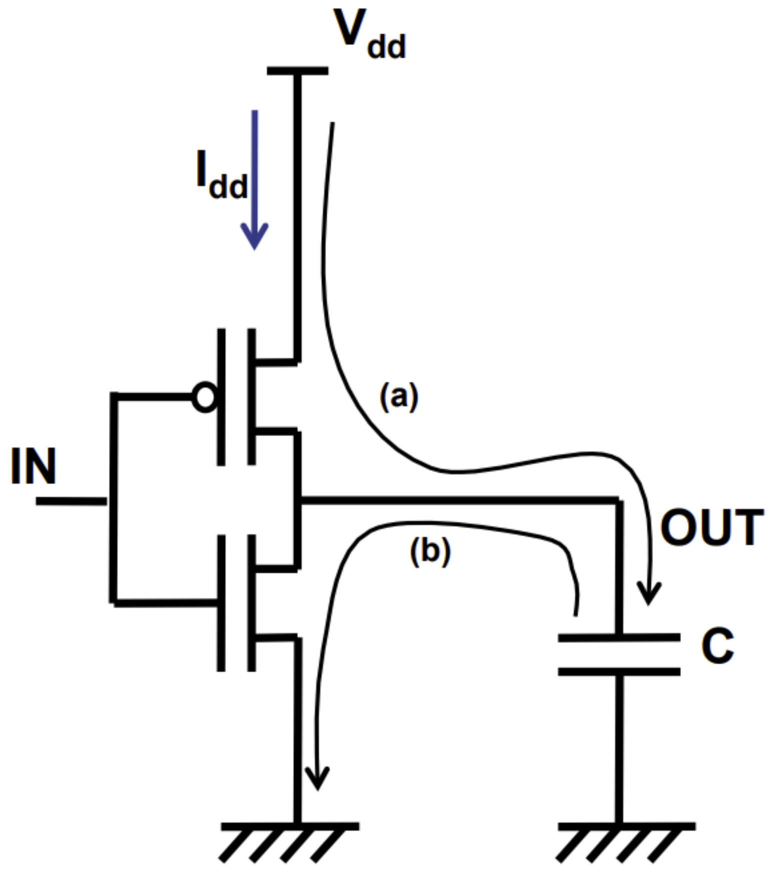
Side channel based watermarking

- Often based on adding a power consumption signature
- The signature should be below the noise floor
- Examples:
  - Ring oscillators triggered by an input sequence
  - Leakage circuit driven by proper sequences:
    - Spread spectrum based watermark →Linear Feedback Shift Register (LFSRs)
    - Input-modulated watermark → xoring properly masked input bits

G. T. Becker et al. “Side-Channel Based Watermarks for Integrated Circuit”

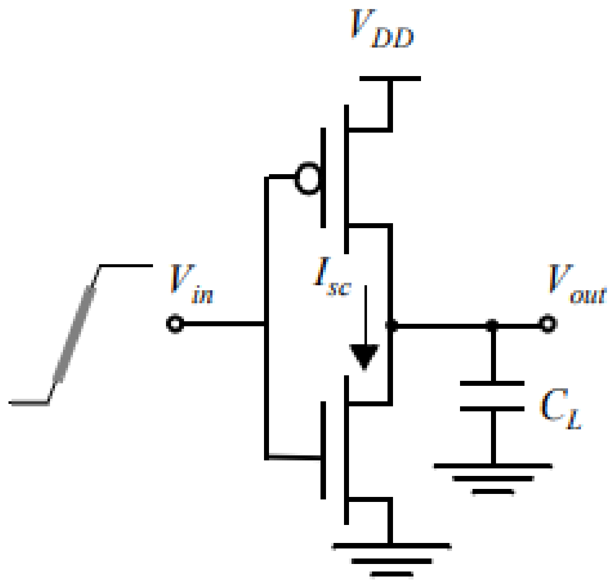
CMOS power consumption - I

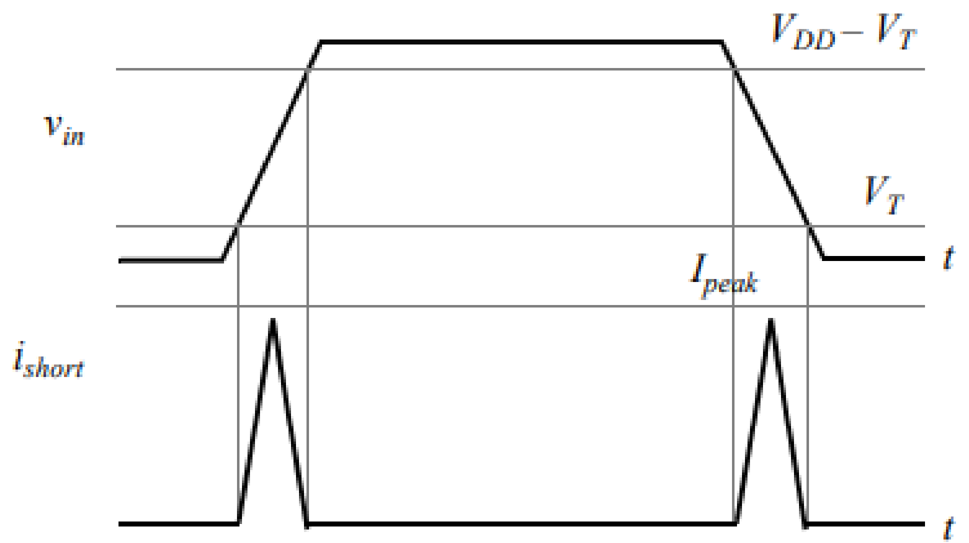
1. Switching power: charging and discharging of output capacitances



CMOS power consumption - II

2. Short circuit power: rise/fall times





3. Leakage power: sub-threshold and device leakage currents

CMOS dynamic power consumption

- Each charge/discharge cycle dissipates $E = C \cdot V_{dd}^2$
- Assuming a switching frequency f
- And a transition probability α

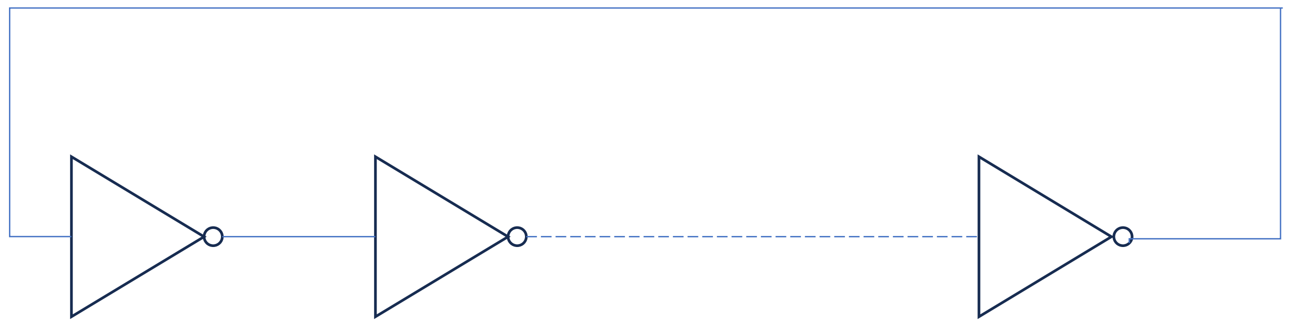
$$P_{dyn} = \alpha \cdot f \cdot C \cdot V_{dd}^2$$

Side channel based watermarking

- Often based on adding a power consumption signature
- The signature should be below the noise floor
- Examples:
  - Ring oscillators triggered by an input sequence
  - Leakage circuit driven by proper sequences:
    - Spread spectrum based watermark →Linear Feedback Shift Register (LFSRs)
    - Input-modulated watermark → xoring properly masked input bits

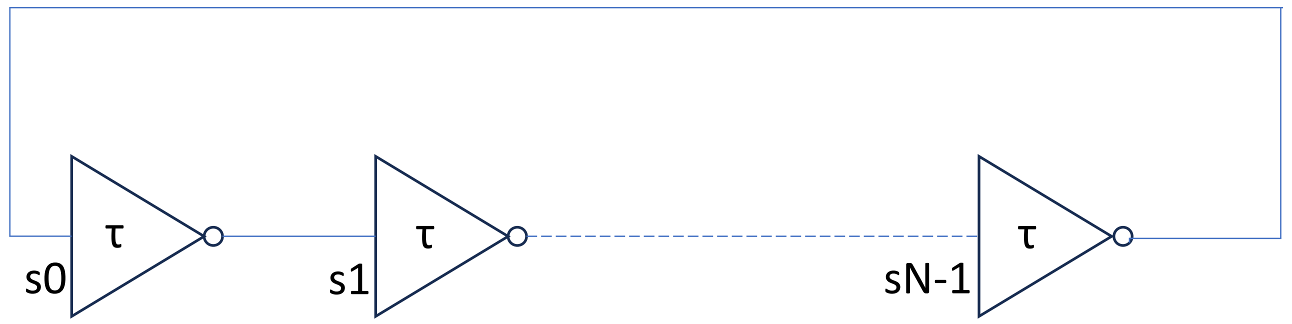
Ring oscillator - I

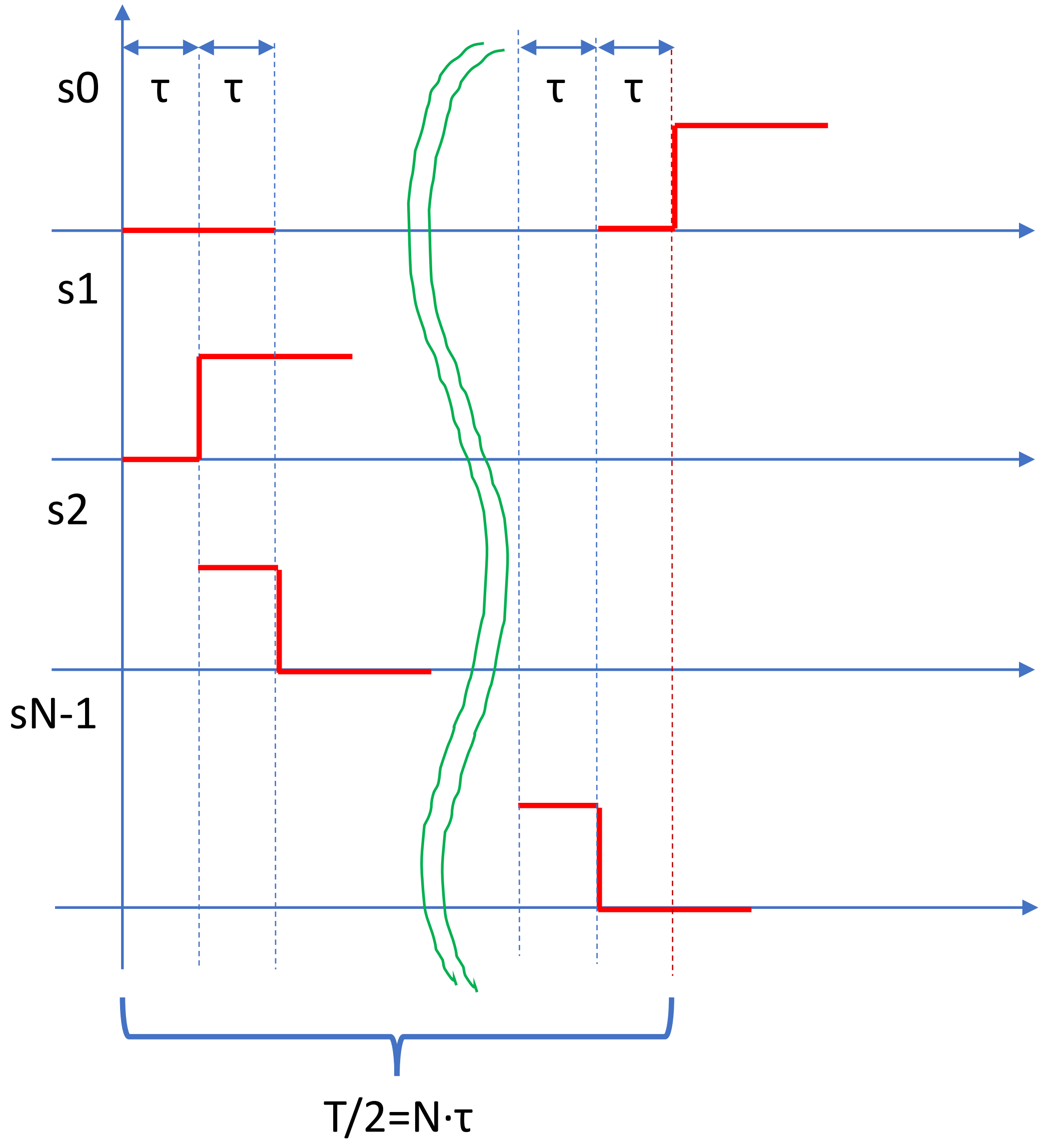
- High switching activity component
- Made of an odd number of inverters connected as a ring



- Oscillating frequency depends on the delay of the inverters

Ring oscillator - II





Side channel based watermarking

- Often based on adding a power consumption signature
- The signature should be below the noise floor
- Examples:
  - Ring oscillators triggered by an input sequence
  - Leakage circuit driven by proper sequences:
    - Spread spectrum based watermark →Linear Feedback Shift Register (LFSRs)
    - Input-modulated watermark → xoring properly masked input bits

Leakage circuit

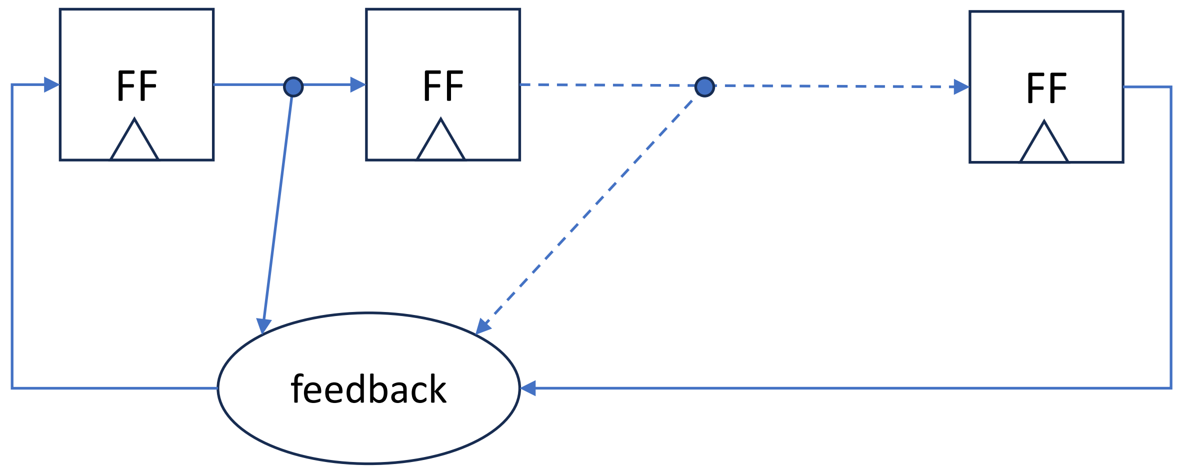
- Generating additional power consumption when receaving a `1’
- Not generating additional power consumption when receavin a `0’
- Examples:
  - Large capacitors
  - Toggling logic
  - …

Side channel based watermarking

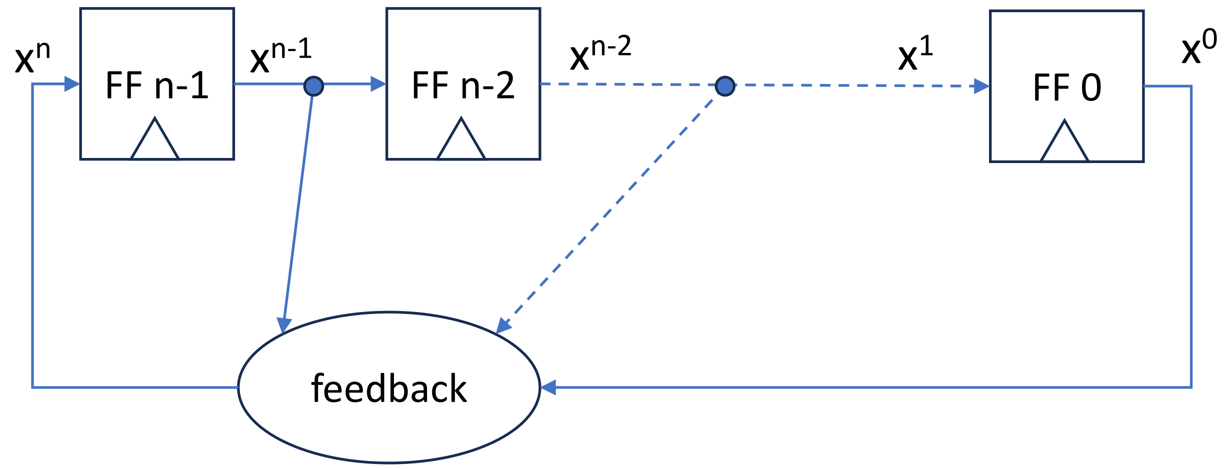
- Often based on adding a power consumption signature
- The signature should be below the noise floor
- Examples:
  - Ring oscillators triggered by an input sequence
  - Leakage circuit driven by proper sequences:
    - Spread spectrum based watermark →Linear Feedback Shift Register (LFSRs)
    - Input-modulated watermark → xoring properly masked input bits

Linear Feedback Shift Registers - LFSRs

- A set of flip-flops (FFs) connected as a shift regiter
- XOR gates combining the output of some FFs to create a feedback



LFSR polynomial description



- Connections represented as ‘1’ (present) ‘0’ (absent)
- The feedback
  - XORs the output of conncted FFs
  - becomes the input of FF n-1
- Polynomial representation $x^n+a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\ldots+a_1x+a_0$

LFSR states

- The possible configuration of an n-bit LFSR are 2^n
- If the LFSR is initialized with all zeros it never changes its state
- Otherwise it can reach up to 2^n-1 different states

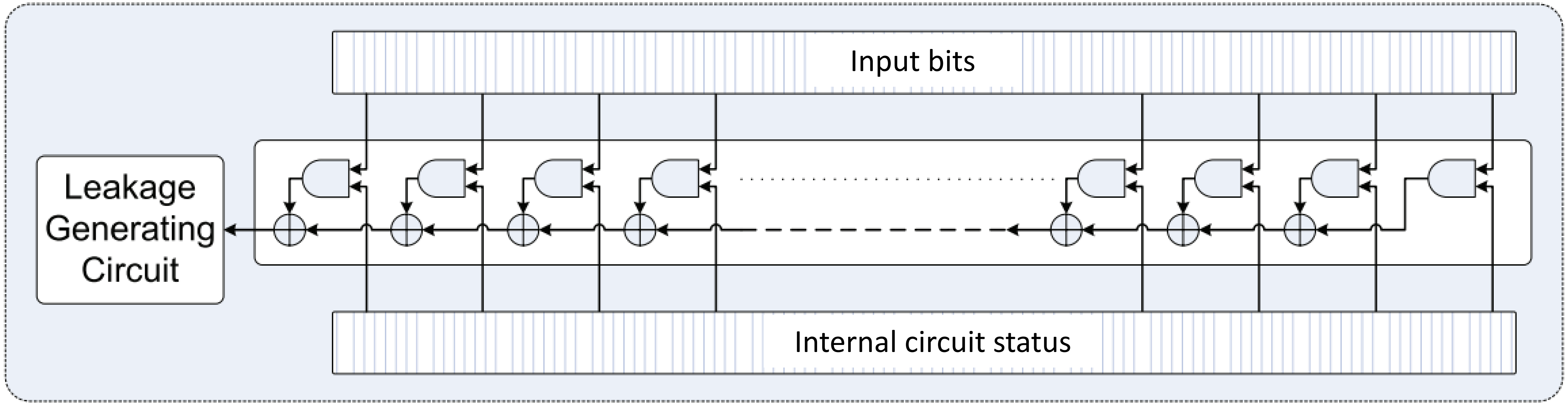
- Can any LFSR reach 2^n-1 different states ?
  - No, only if the corresponding polynomial is primitive(*)

(*) https://mathworld.wolfram.com/PrimitivePolynomial.html

Side channel based watermarking

- Often based on adding a power consumption signature
- The signature should be below the noise floor
- Examples:
  - Ring oscillators triggered by an input sequence
  - Leakage circuit driven by proper sequences:
    - Spread spectrum based watermark →Linear Feedback Shift Register (LFSRs)
    - Input-modulated watermark → xoring properly masked input bits

Input-modulated watermark


Input bits
Leakage Generating Circuit
Internal circuit status