Robust IP Watermarking Methodologies for Physical Design

Andrew B. Kahng, Stefanus Mantik, Igor L. Markov,
Miodrag Potkonjak, Paul Tucker*, Huijuan Wang
and Gregory Wolfe

UCLA Computer Science Dept., Los Angeles, CA
*UCSD Computer Science & Engineering Dept., La Jolla, CA

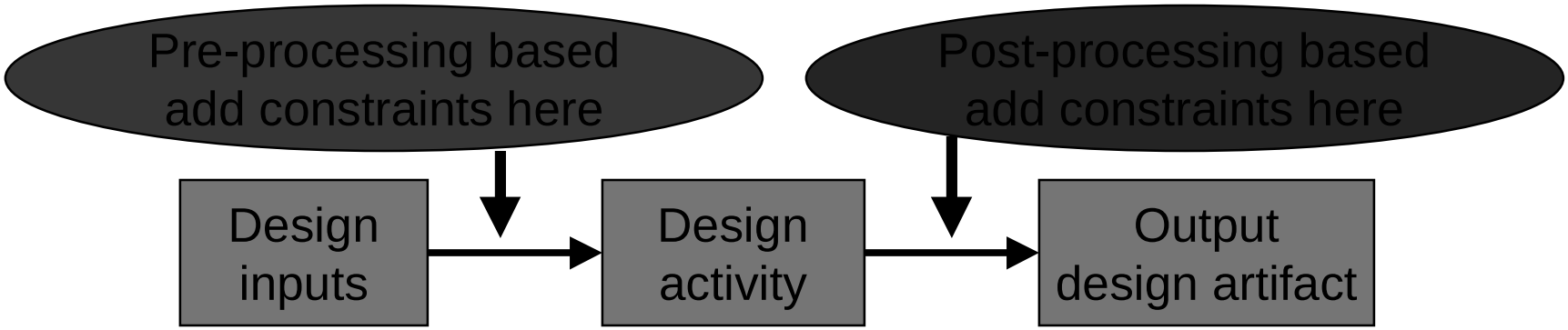
SM980618

Need for IP Protection

- Consumerization of silicon technology
  - available gate count ↑, functionality ↑
  - shorter design cycle
  - reuse, integration of existing HW and SW IP
- Improved technology for reverse engineering
- ⇒ Need Intellectual Property Protection

Watermarking Based IPP

- *Design watermark*: an **invisible** identification code that is **permanently** embedded, as an integral part, in the design.
  - Achieved by adding *constraints*



Practical Implementation

Given a digital signature *X*, embed *X* within the output design artifact subject to:

- maintain correct functionality
- transparent to design flow
- low overhead cost
- strong proof of authorship
- invisible
- resistant to attack

Physical Design

- Higher-level constraints not satisfied until physical design
- “Difficult” domain
  - small percentage variation in quality can make or break a design
- Rich opportunities for watermarking
  - large number of gates and nets compared to several lines of code

Outline

- Introduction
- New watermarking approaches in PD
  - placement
  - routing
- Evaluation of watermark strength
- Resistance to tampering attacks
- Conclusions

Available Constraint Types in Placement

- Context
  - row-based, standard-cell designs
  - PDEF/GCF/SDF/DEF formats

Available Constraint Types in Placement

- Context: row-based, standard-cell designs
- Region constraints
  - ☺ easily supported:
    - KL/FM partitioners use “fixed” preassignments
    - analytic placers use variable bounds
  - ☹ no a priori knowledge of good placement
    $\rightarrow$ degradation of solution quality

Available Constraint Types in Placement

- Context: row-based, standard-cell designs
- Region constraints
- Grouping constraints
  - ☺ easily enforced by inducing netlists over clustered representations of the design
  - ☹ no a priori structural knowledge
    $\rightarrow$ bad quality

Available Constraint Types in Placement

- Context: row-based, standard-cell designs
- Region constraints
- Grouping constraints
- Path delay budgeting

  $t\ (C_1, C_2, \ldots, C_{10}) \leq 50\text{ns}$

  $\Rightarrow\ t_1\ (C_1, C_2, \ldots, C_5) \leq 20\text{ns}$ and $t_2\ (C_5, C_6, \ldots, C_{10}) \leq 30\text{ns}$

  ☹ arbitrary timing constraints can be harmful to convergence

  ☹ watermark unrealized until after routing

Our Approach: Row Parity Constraint

- Example:
  - cell `DP_1` placed in cell row with EVEN index
  - cell `MUX_3a` placed in cell row with ODD index
- Post-processing approach

☺ Very few constraints needed for strong signature

☺ Compatible with existing placement methodology

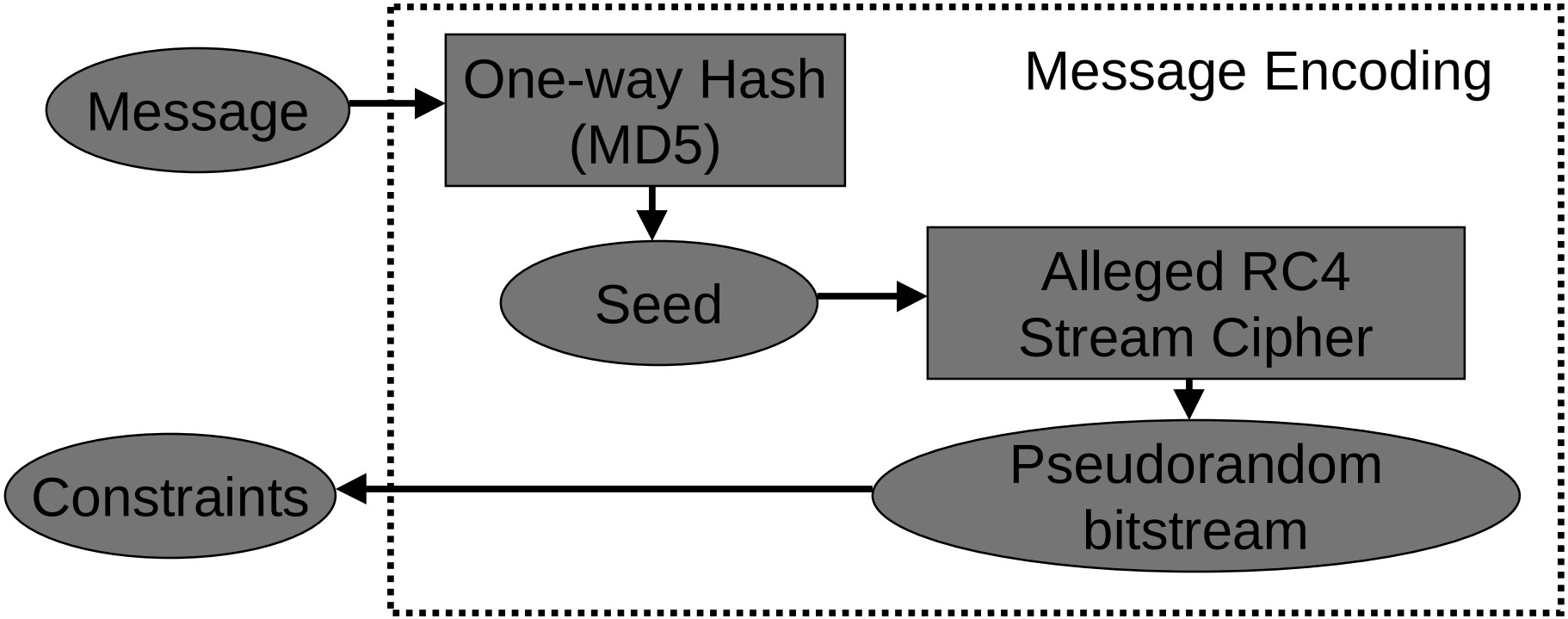
☺ Local perturbations won't easily disturb signature

☺ Not affected by downstream stages of design flow

☺ Watermark completely realized during placement

Example

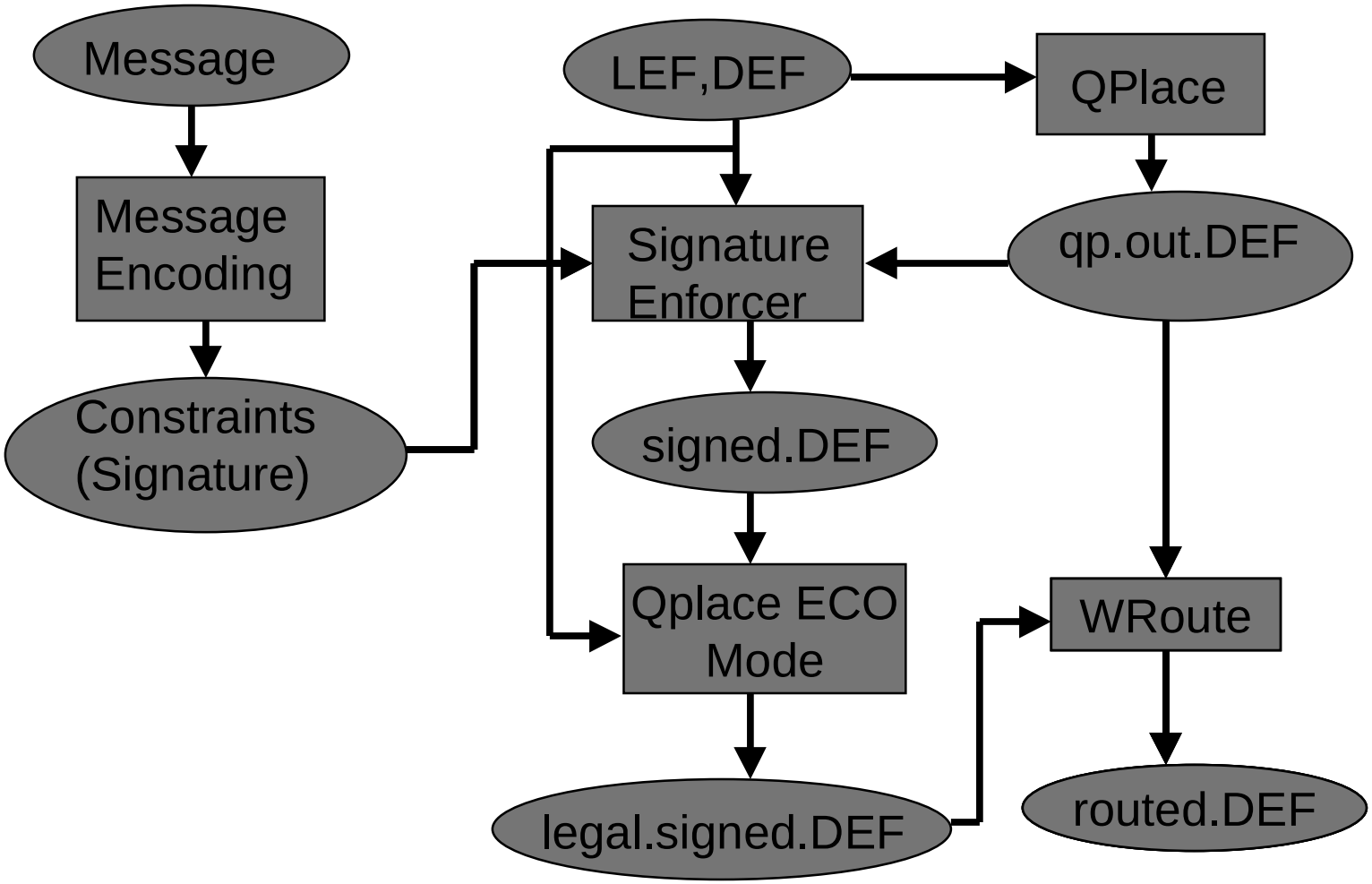
- 100-cell design (C_1, C_2, … , C_{100})
- Message: “DAC98, San Francisco”
- Pass to *Message Encoding*



Example

- 100-cell design (C_1, C_2, … , C_{100})
- Message: “DAC98, San Francisco”
- Pass to *Message Encoding*
  - *EVEN parity*: C_3, C_{27}, C_{40}, C_{66}, C_{88}, C_{95}
  - *ODD parity*: C_{15}, C_{36}, C_{38}, C_{72}, C_{89}, C_{98}
- Pass to *Signature Enforcer*

Placement Watermarking Flow



Available Constraint Types in Routing

- Context
  - standard-cell design

Available Constraint Types in Routing

- Context: standard-cell design
- Segment width, spacing constraints
  - ☺ readily available
  - ☹ potential harmful interaction with performance-driven routing constraints

Available Constraint Types in Routing

- Context: standard-cell design
- Segment width, spacing constraints
- Layer assignment constraints
  - ☺ easy to implement
  - ☹ no a priori knowledge of timing
    $\rightarrow$ unacceptable solution

Available Constraint Types in Routing

- Context: standard-cell design
- Segment width, spacing constraints
- Layer assignment constraints
- Limit via constraints
  - ☺ supported by constraint-driven shape-based routers
  - ☹ routing time can increase substantially

Our Approach: Limit Wrong-Way Constraint

- Limit wrong-way = N $\rightarrow$ given net has at most N microns of wrong-way routing
- Pre-processing approach
- ☺ Very few constraints needed for strong signature
- ☺ Compatible with routing constraints in existing routers
- ☺ Local perturbations won't easily disturb signature

Example

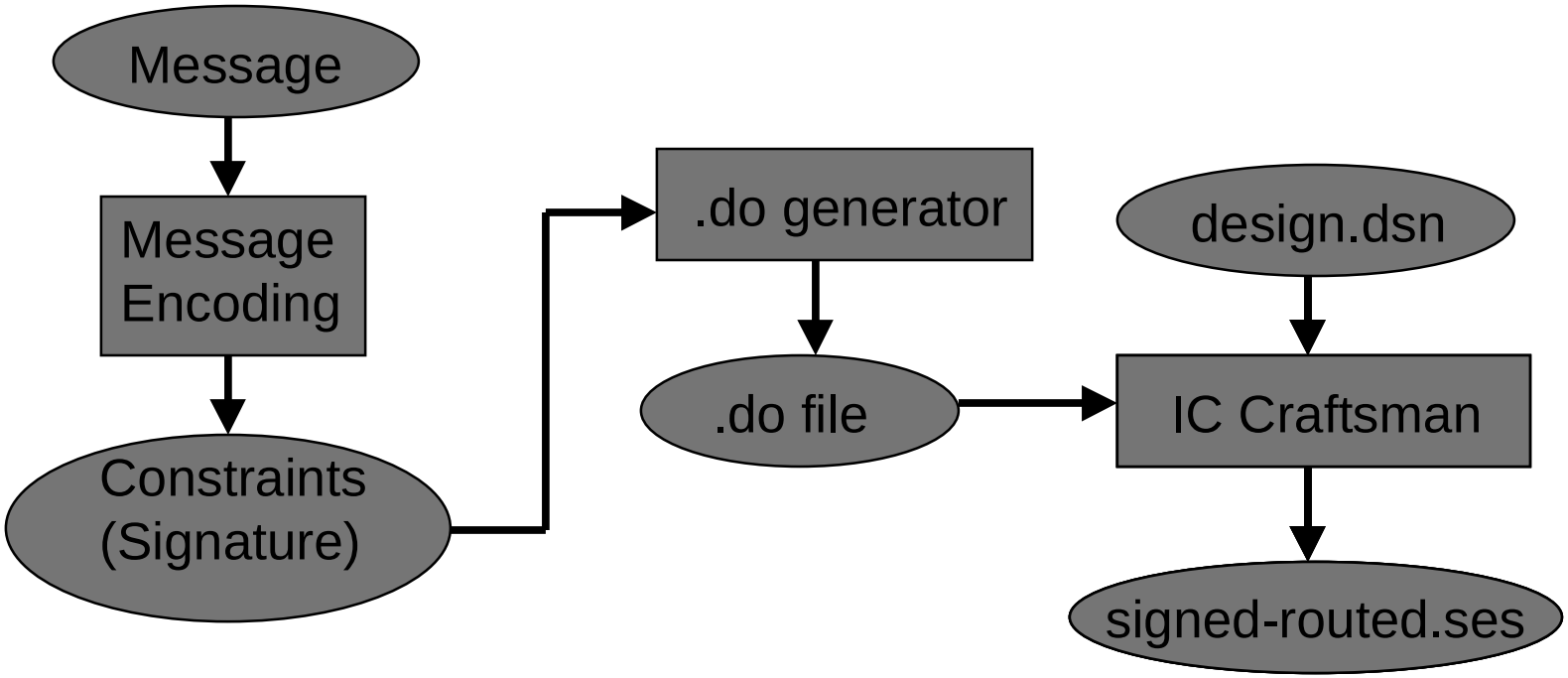
- 100-net design (N_1, N_2, … , N_{100})
- Message: “DAC98, San Francisco”
- Pass to *Message Encoding*

  signature: (N_3, N_{15}, N_{27}, N_{36}, N_{38}, N_{40}, N_{66}, N_{72}, N_{88}, N_{89}, N_{95}, N_{98})

- Pass to *.do Generator* (IC Craftsman)

  (net N_3 (rule (limit_way 1))), (net N_{15} (rule (limit_way 1))),
  (net N_{27} (rule (limit_way 1))), …

Routing Watermarking Flow



Outline

- Introduction
- New watermarking approaches in PD
  - placement
  - routing
- Evaluation of watermark strength
- Resistance to tampering attacks
- Conclusions

Evaluation of Signature Strength

- Probability of a solution carrying an author's watermark purely by coincidence:

$$P_c = \sum_{i=0}^{x} \left(\binom{X}{i} \cdot p^{X-i} \cdot (1-p)^{i} \right)$$

  - X : number of constraints imposed
  - x : number of constraints *not* satisfied
  - p : probability that any given constraint is satisfied purely by coincidence

Experimental Results - Placement

- Placement watermarking
  - number of cells: 12133
  - number of nets: 11828

CASE T2	WL	# Vias	Congest	CPU(Norm)	P_C
Original Result	3.32	95601	0.86	1.00	N/A
WM-56,53	3.33	95811	0.78	1.03	4.1e-13
WM-112,110	3.33	95978	0.80	1.01	1.2e-30
WM-224,208	3.34	95913	0.77	1.02	4.5e-44
WM-448,389	3.35	96554	0.91	1.02	3.4e-79
WM-896,837	3.38	97902	0.94	1.05	3.2e-177
WM-1792,1740	3.42	99467	1.13	1.10	2.9e-357

Experimental Results - Routing

- Routing watermarking
  - number of cells: 4250
  - number of nets: 1597

CASE SC2	WL	WW	# Vias	CPU (Norm)	P_C
Original	2.29	3.54	5590	1.00	N/A
WM-20,20	2.29	3.60	5547	0.90	1.1e-8
WM-40,40	2.30	3.77	5716	0.79	1.2e-16
WM-80,78	2.30	3.47	5713	0.71	1.8e-30
WM-160,159	2.29	3.23	5650	0.87	1.3e-48
WM-320,315	2.30	2.93	5706	0.87	2.2e-85

Outline

- Introduction
- New watermarking approaches in PD
  - placement
  - routing
- Evaluation of watermark strength
- Resistance to tampering attacks
- Conclusions

Tampering Attack

- Placement tampering:
  - select *N* random pairs of cells
  - swap the locations of each cell pair
  - run legalizer
- Routing tampering:
  - select *N* random nets
  - reroute them with the original constraints

Resistance to Tampering - Placement

- Placement experiment
  - perturb 1%, 2%, 5%, and 10% of total cells
  - total wirelength, and P_c values for initial WM and after 10% perturbation

WM Size	Initial WL	Initial *Pc*	1 %	2 %	5 %	10 %	10% P_C
56	3.15	7.9e-16	3.24	3.34	3.61	4.00	4.1e-13
112	3.14	1.2e-30	3.25	3.33	3.60	4.03	4.9e-25
224	3.16	1.7e-58	3.26	3.34	3.61	4.02	4.5e-44
448	3.17	2.1e-124	3.27	3.36	3.62	4.04	9.1e-95
896	3.21	7.2e-235	3.30	3.39	3.65	4.08	1.1e-180
1792	3.25	3.2e-439	3.35	3.44	3.69	4.12	1.1e-334

- Solution quality degrades faster

Resistance to Tampering - Routing

- Routing experiment
  - perturb 1%, 2%, 5%, and 10% of total nets
  - total wirelength, original and incremental CPU time, and (original, 10%) P_c values

WM Size	Initial WL	Initial *Pc*	1 %	2 %	5 %	10 %	10% P_c
20	2.29	1.1e-8	2.29	2.29	2.29	2.30	7.7e-8
40	2.30	1.2e-16	2.30	2.30	2.31	2.31	8.6e-13
80	2.30	1.8e-30	2.29	2.30	2.30	2.30	1.1e-23
160	2.29	1.3e-48	2.29	2.29	2.30	2.30	2.3e-36
320	2.30	2.2e-85	2.30	2.30	2.31	2.31	1.7e-67
CPU(Norm)	1.00		0.07	0.12	0.20	0.44	

- Perturbation times approach initial time

Conclusions

- Protocols for embedding design watermark at physical design level
- Strong proof of authorship with low overhead cost
- Transparent to existing tool flows
- Robust with respect to tampering
- Current research: alternative watermarking strategies for timing, layout and higher-level design artifacts