Behavioral Obfuscation for IP Protection

Levent Aksoy

Tallinn University of Technology, Tallinn, Estonia

SAFEST – Summer School, University of Montpellier, Montpellier, France, 8-10 of June 2022

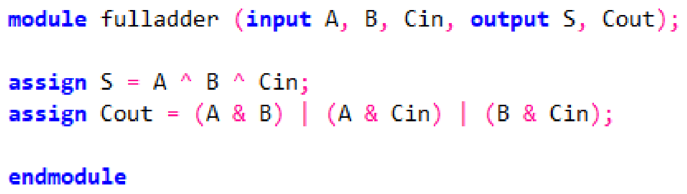
Outline

- Introduction
- Background
- Obfuscation Techniques
- De-obfuscation Methods
- Conclusions

Introduction

- Intellectual property (IP)
  - *is an idea, a design, etc. that somebody has created and that the law prevents other people from copying (source: Oxford dictionary)*
- In hardware, IPs include
  - integrated circuits (ICs) and designs owned by a company or a designer

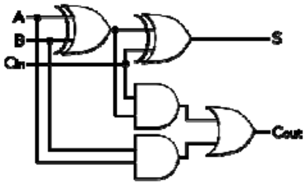
Soft IPs	Firm IPs	Hard IPs
Behavioral level Register transfer level (RTL) design	Structural level Gate-level netlist	Physical level Layout

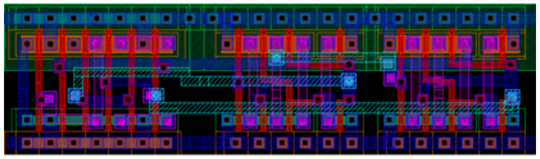


```
module fulladder (input A, B, Cin, output S, Cout);

assign S = A ^ B ^ Cin;
assign Cout = (A & B) | (A & Cin) | (B & Cin);

endmodule
```





Introduction

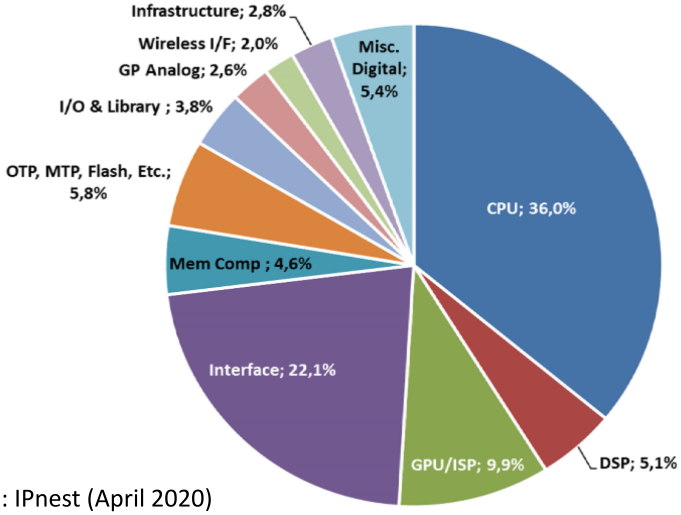
Semiconductor Design IP Revenue, Worldwide (Millions of Dollars)

Segment	2018	2019	Growth
Total Design IP	**3 742,7**	**3 938,0**	**5,2%**

Semiconductor Design IP Revenue by Company, Worldwide, (Millions of Dollars)

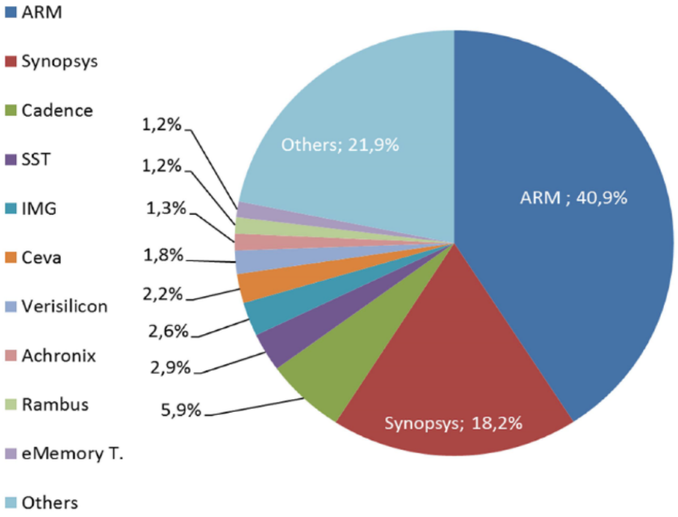
Rank	Company	2018	2019	Growth	2019 Share	Cumulative Share
1	ARM (Softbank)	1 610,0	1 608,0	-0,1%	40,8%	40,8%
2	Synopsys	629,8	716,9	13,8%	18,2%	59,0%
3	Cadence	188,8	232,0	22,9%	5,9%	64,9%
4	SST	104,8	115,0	9,7%	2,9%	67,8%
5	Imagination Technologies	124,6	101,1	-18,9%	2,6%	70,4%
6	Ceva	77,9	87,2	11,9%	2,2%	72,6%
7	Verisilicon	66,3	69,8	5,3%	1,8%	74,4%
8	Achronix	52,5	50,0	-4,8%	1,3%	75,7%
9	Rambus	49,9	48,8	-2,2%	1,2%	76,9%
10	eMemory Technology	47,9	46,8	-2,3%	1,2%	78,1%
	Top 10 Vendors	**2 952,5**	**3 075,6**	**4,2%**	**78,1%**	**78,1%**
	Others	790,2	862,4	9,1%	21,9%	100,0%
	Total	**3 742,7**	**3 938,0**	**5,2%**	**100,0%**	**100,0%**

Design IP by Category 2019

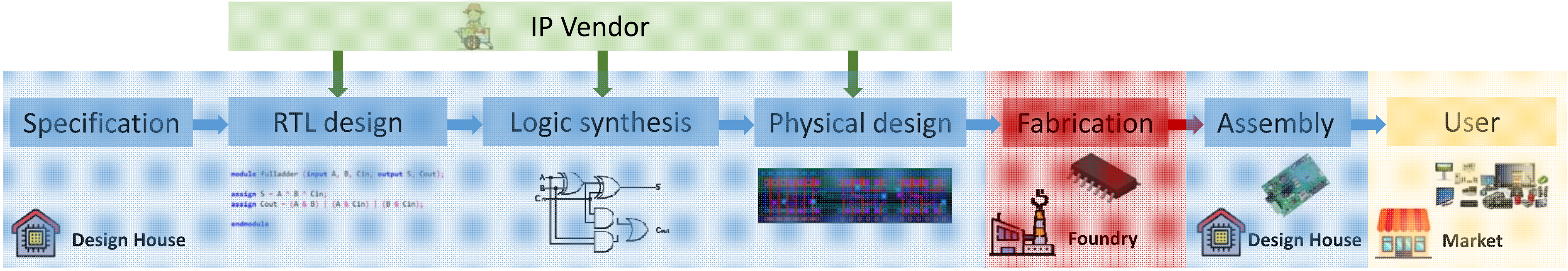


Source: IPnest (April 2020)

Design IP Vendor Market Share Ranking 2019



Background – IC Design Flow



```
module fulladder (input A, B, Cin, output S, Cout);

assign S = A ^ B ^ Cin;
assign Cout = (A & B) | (A & Cin) | (B & Cin);

endmodule
```

🟩 IP Vendor: Trusted

🟦 Design House: Trusted/Untrusted

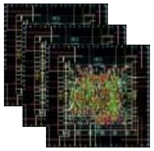
🟥 Foundry: Untrusted

🟨 Market: Untrusted

Background – IP Threats

IP Piracy

An entity, other than IP owner, sells the IP to a third party



IP Overuse

An entity uses the IP in more instances than the agreed one

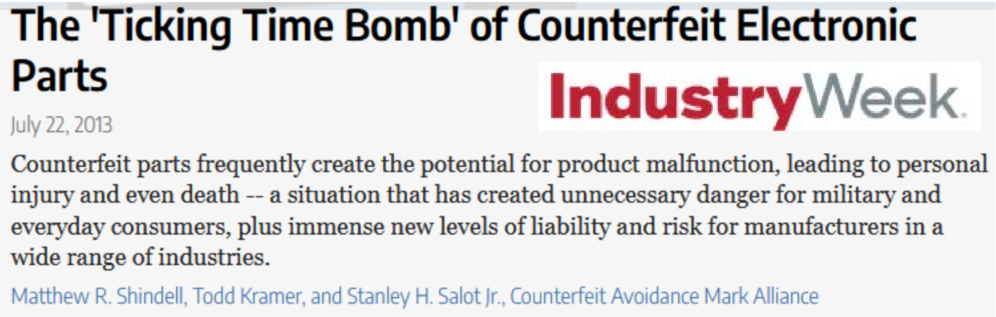
IP Modification

A malicious designer modifies the IP to insert backdoor or hardware Trojans

Reverse Engineering

An adversary extracts the higher level design of an IP and expose it to piracy, overuse, and modification

Background – IP Threats



The 'Ticking Time Bomb' of Counterfeit Electronic Parts

IndustryWeek

July 22, 2013

Counterfeit parts frequently create the potential for product malfunction, leading to personal injury and even death -- a situation that has created unnecessary danger for military and everyday consumers, plus immense new levels of liability and risk for manufacturers in a wide range of industries.

Matthew R. Shindell, Todd Kramer, and Stanley H. Salot Jr., Counterfeit Avoidance Mark Alliance

Recent reports show consumer and industrial businesses are losing approximately $250 billion each year because of counterfeit components.

INNOVATION AND INTELLECTUAL PROPERTY   APRIL 11, 2019 / 8:22 AM / UPDATED 3 YEARS AGO

ASML says it suffered intellectual property theft, rejects 'Chinese' label

By Toby Sterling, Anthony Deutsch

REUTERS

ASML shares slipped 1.5 percent by 12:10 GMT to the bottom of a flat European technology index.

Counterfeit Components Continue to Slam Electronics Industry

Rob Spiegel | Mar 02, 2015

DesignNews

In 2013, the US Customs and Border Protection reported more than 24,300 counterfeit shipment seizures, representing more than $1.7 billion in goods. Over the last five years, counterfeit seizures have increased nearly 50%.

Taiwan's UMC Pays to Settle Tech Theft Litigation With Micron

By Debby Wu
November 26, 2021, 1:34 AM GMT+2

Bloomberg
Europe Edition

United Microelectronics Corp. and Micron Technology Inc. have settled a civil lawsuit in which the U.S. memory chipmaker accused the Taiwanese company of stealing and leaking its intellectual property to a Chinese partner.

Background – Passive Defense Methods

Digital Watermarking

Embeds a designer’s signature in design

Fingerprinting

Embeds the buyer’s signature along with the designer’s watermark

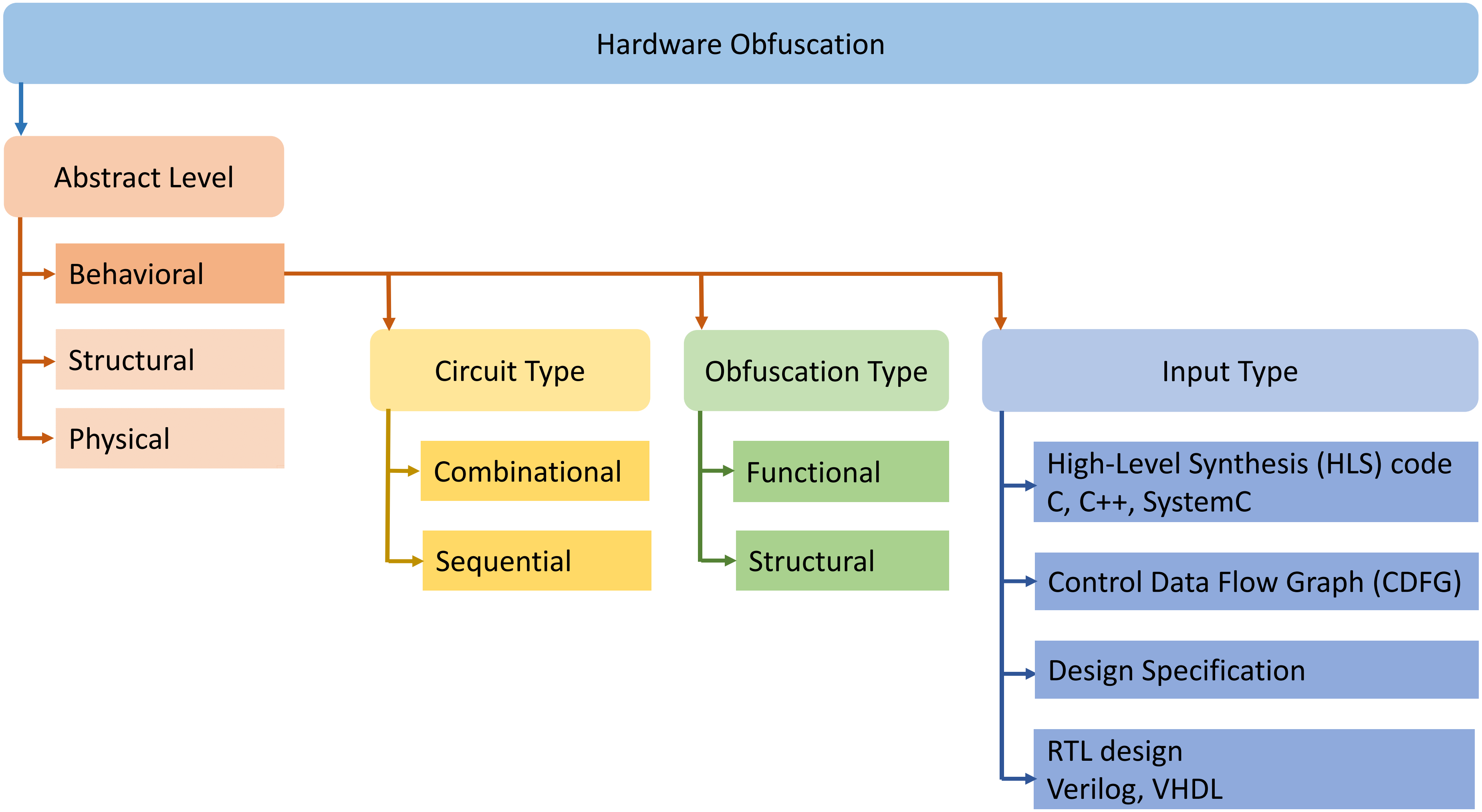
Hardware Metering

Involves a set of protocols which enable to gain post-fabrication control

Background – Obfuscation

- Obfuscation
  - *the act of making something less clear and more difficult to understand, usually deliberately (source: Oxford dictionary)*
- Software obfuscation
  - source and machine code
    - layout obfuscation
    - control obfuscation
    - data obfuscation
- Hardware obfuscation
  - functionality is hidden such that it cannot be retrieved
    - logic locking
    - camouflaging
    - high-level transformations

Background – Behavioral Obfuscation



Hardware Obfuscation

Abstract Level

Behavioral

Structural

Physical

Circuit Type

Combinational

Sequential

Obfuscation Type

Functional

Structural

Input Type

High-Level Synthesis (HLS) code
C, C++, SystemC

Control Data Flow Graph (CDFG)

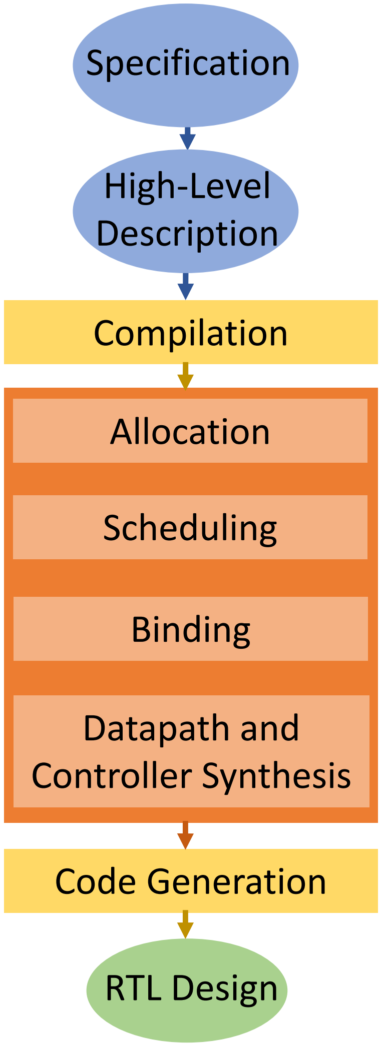
Design Specification

RTL design
Verilog, VHDL

Background – Behavioral Obfuscation

- Advantages of behavioral obfuscation
  - increases the protection level by selecting the critical operations, branches, and functions to obfuscate
  - efficiently explores tradeoffs between overhead, resiliency, and output corruption
  - applies HLS and logic optimizations which are unknown to the attacker
  - increases flexibility in functional verification

Background – HLS Flow

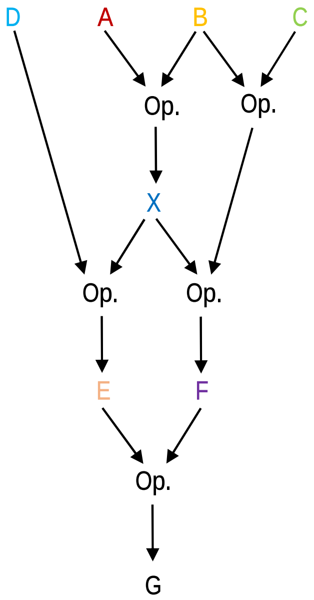


Specification	
High-Level Description	C, C++, SystemC
Compilation	Parse the code, apply compiler optimizations, extract intermediate representation (IR)
Allocation	Allocate operations to available hardware resources
Scheduling	Map operations into different clock cycles
Binding	Bind operations to functional units, variables to storage units, transfers to buses
Datapath and Controller Synthesis	Generate the datapath, identify the control signals, and realize the controller, i.e., finite state machine (FSM)
Code Generation	Generate the RTL description
RTL Design	Verilog, VHDL

Background – HLS Example

```
int example (int A, int B, int C, int D){
  int X = A+B;
  int E = X*D;
  int F = (B+C)*X;
  int G = E+F;
  return G;
}
```

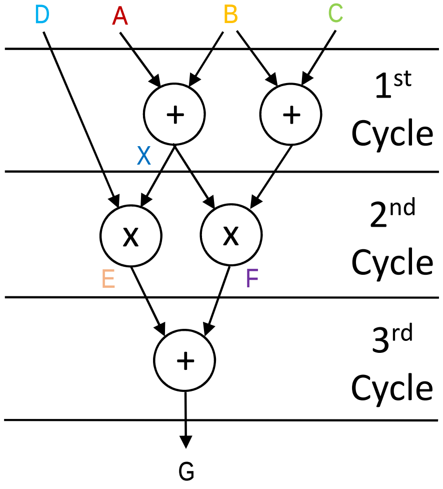
Compilation



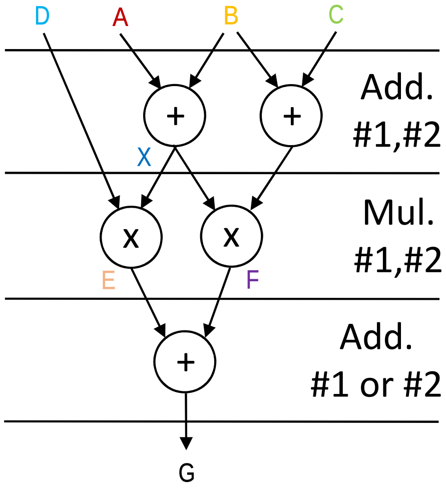
Allocation

#Adders: 2
#Multipliers: 2

Scheduling

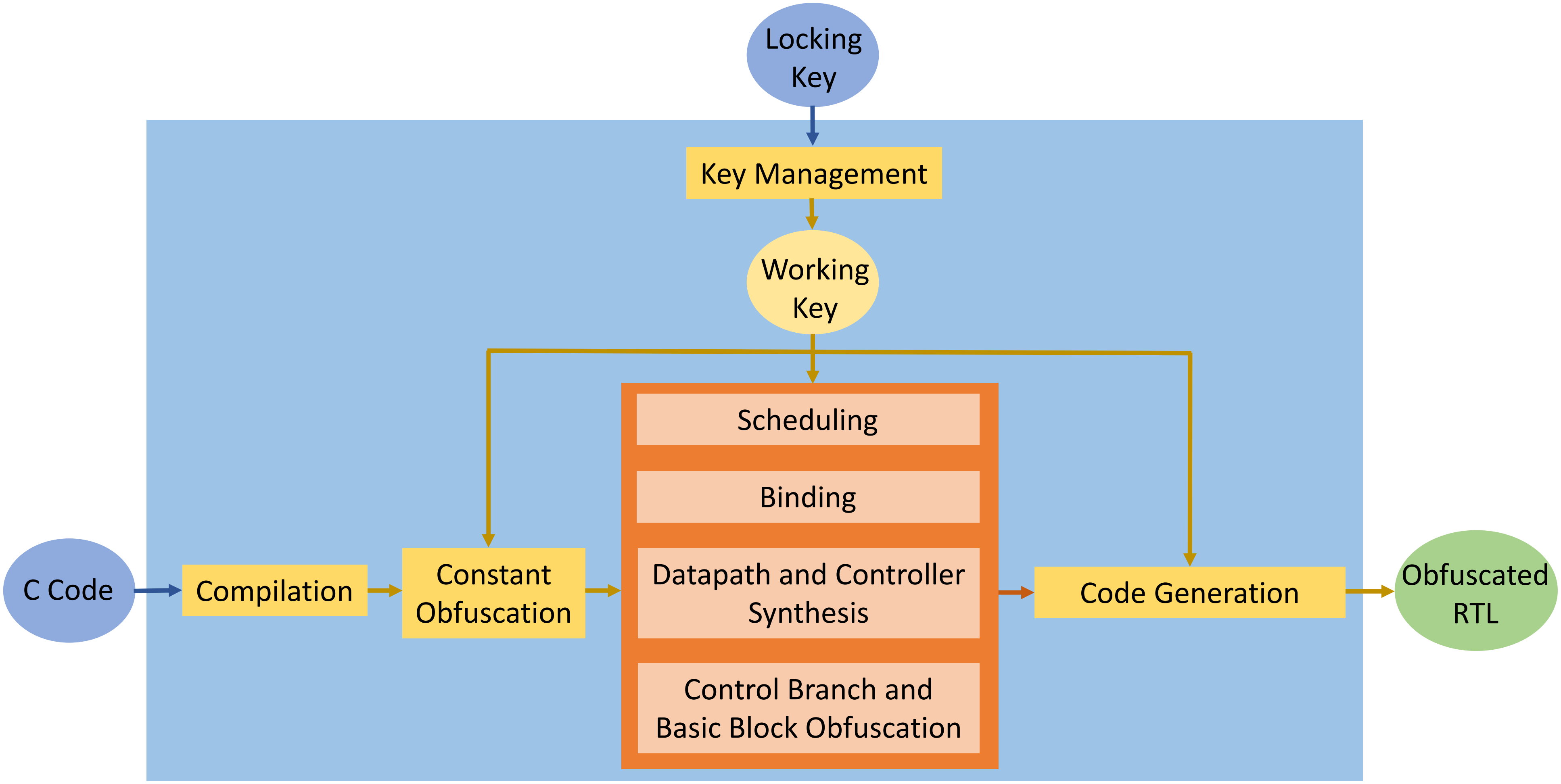


Binding



Source: M. R. Muttaki, R. Mohammadivojdan, M. Tehranipoor and F. Farahmandi, "HLock: Locking IPs at the High-Level Language," *DAC*, 79-84, 2021.

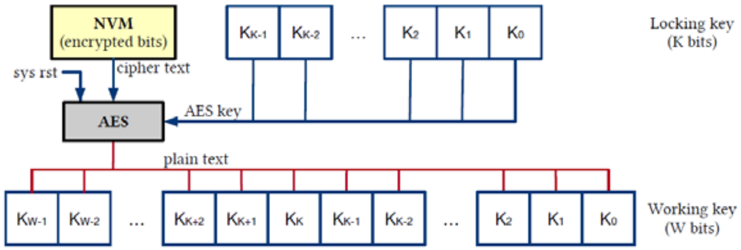
Obfuscation Techniques – DAC’18



Source: C. Pilato, F. Regazzoni, R. Karri, and S. Garg, “TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis,” *DAC*, 1-6, 2018.

Obfuscation Techniques – DAC’18

Key Management (W > K)

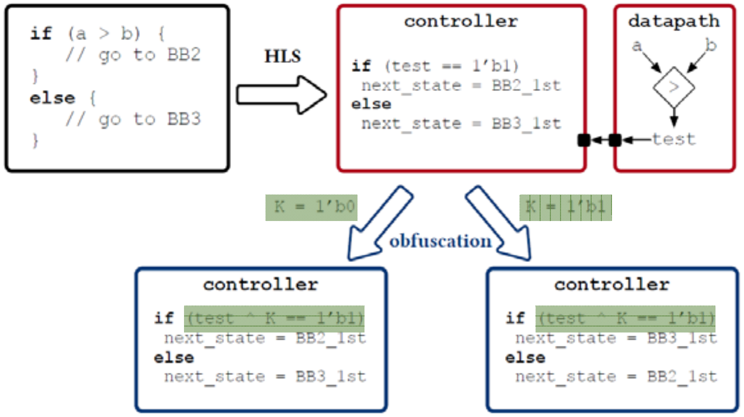


Constant Obfuscation

$$c_i^{obf} = c_i^{org} \oplus k_i$$

$$c_i^{org} = 7 = 4'b0111$$
$$k_i = 4'b1010$$
$$c_i^{obf} = 4'b1101$$

Control Branch Obfuscation



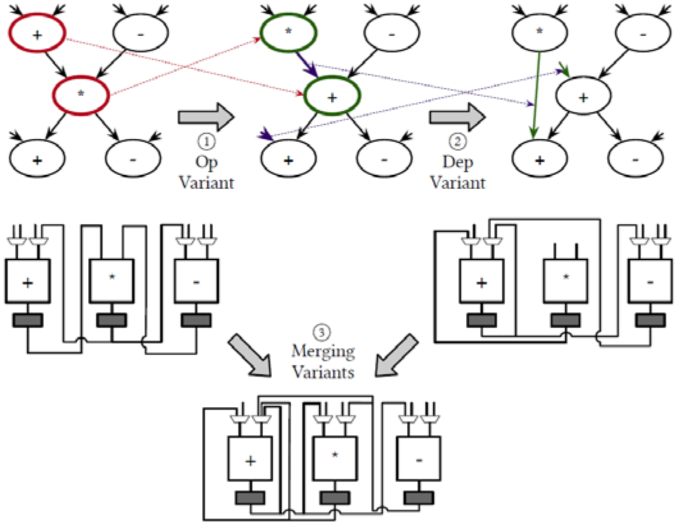
```
if (a > b) {
   // go to BB2
}
else {
   // go to BB3
}
```

```
if (test == 1’b1)
 next_state = BB2_1st
else
 next_state = BB3_1st
```

```
if (test ^ K == 1’b1)
 next_state = BB2_1st
else
 next_state = BB3_1st
```

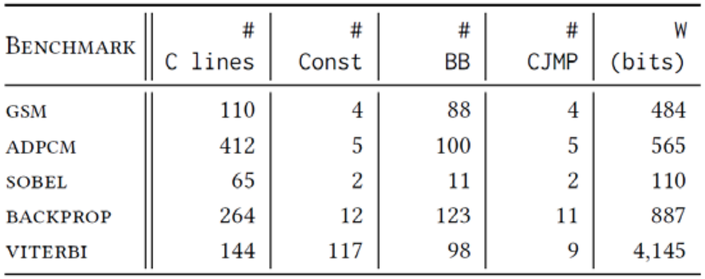
```
if (test ^ K == 1’b1)
 next_state = BB3_1st
else
 next_state = BB2_1st
```

Basic Block Obfuscation

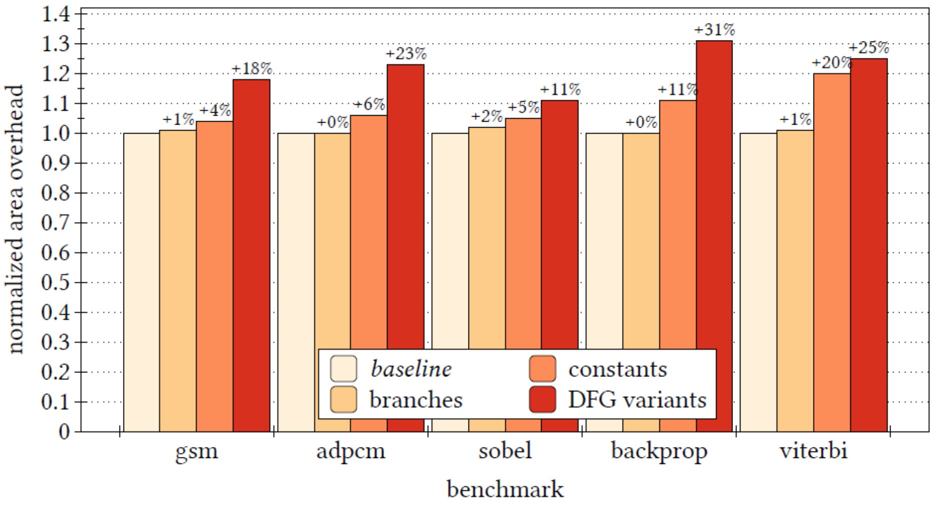


Source: C. Pilato, F. Regazzoni, R. Karri, and S. Garg, “TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis,” *DAC*, 1-6, 2018.

Obfuscation Techniques – DAC’18

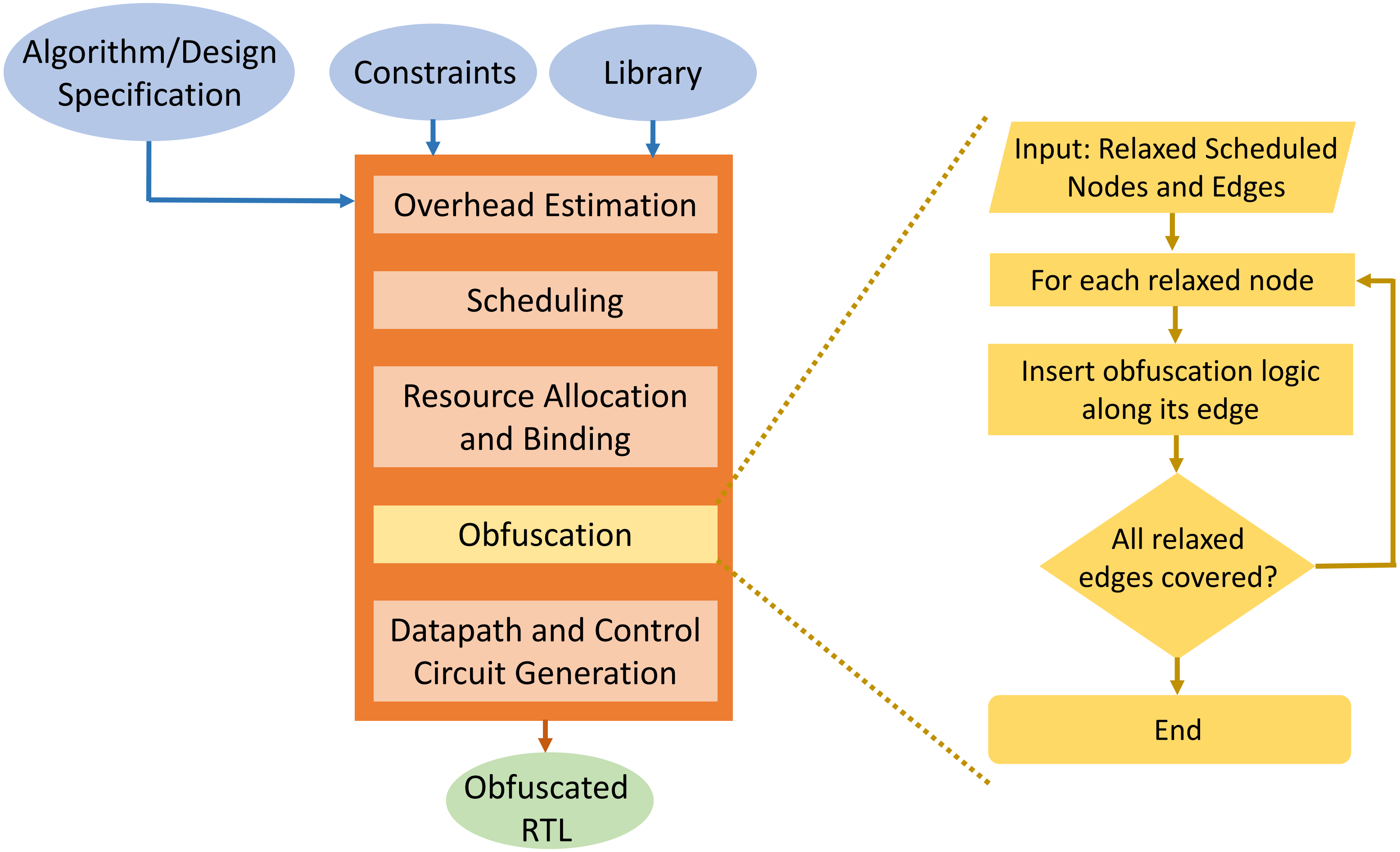


Benchmark	# C lines	# Const	# BB	# CJMP	W (bits)
GSM	110	4	88	4	484
ADPCM	412	5	100	5	565
SOBEL	65	2	11	2	110
BACKPROP	264	12	123	11	887
VITERBI	144	117	98	9	4,145



Source: C. Pilato, F. Regazzoni, R. Karri, and S. Garg, “TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis,” *DAC*, 1-6, 2018.

Obfuscation Techniques – TODAES’20

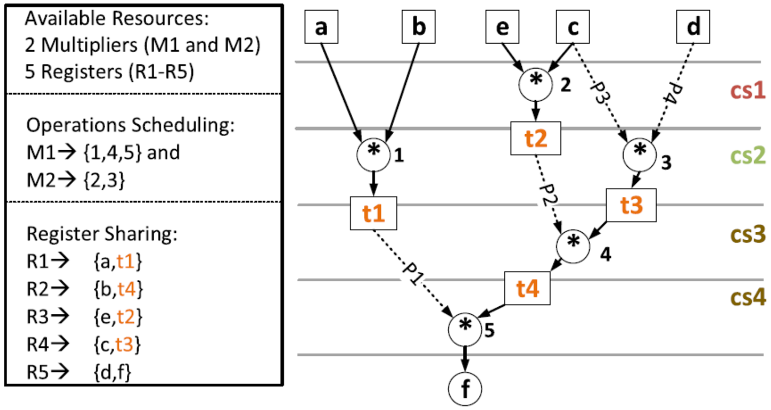


Algorithm/Design Specification
Constraints
Library
Overhead Estimation
Scheduling
Resource Allocation and Binding
Obfuscation
Datapath and Control Circuit Generation
Obfuscated RTL
Input: Relaxed Scheduled Nodes and Edges
For each relaxed node
Insert obfuscation logic along its edge
All relaxed edges covered?
End

Source: S. A. Islam, L. K. Sah, and S. Katkoori, “High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging,” *ACM TODAES*, 26, 1, article 6, 2020.

Obfuscation Techniques – TODAES’20

$$f = (a * b) * (e * c) * (c * d)$$



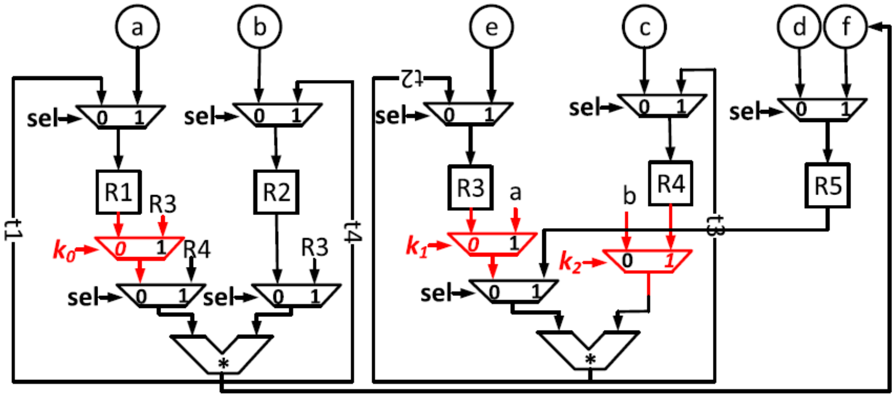
Available Resources:
2 Multipliers (M1 and M2)
5 Registers (R1-R5)

Operations Scheduling:
M1→ {1,4,5} and
M2→ {2,3}

Register Sharing:
R1→ {a,t1}
R2→ {b,t4}
R3→ {e,t2}
R4→ {c,t3}
R5→ {d,f}

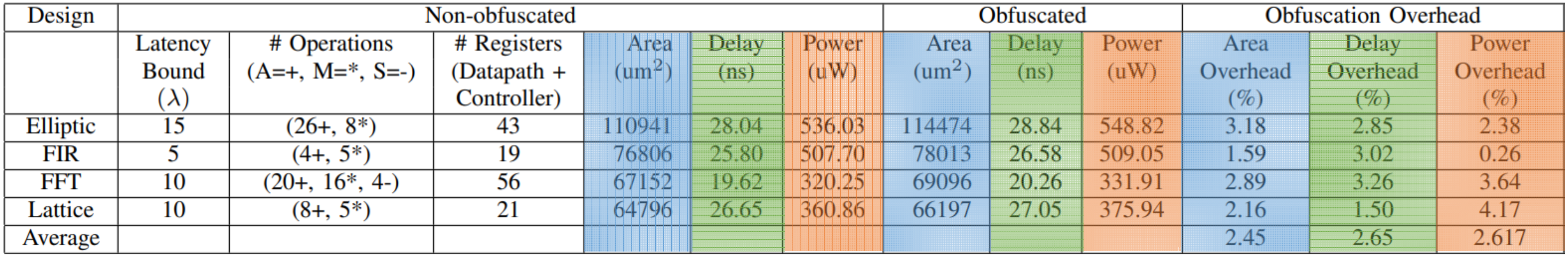
$$f = \left(\overline{k_o}(a * b) + k_0 e\right) * \left(\overline{k_1}(c * e) + k_1 a\right) * \left(\overline{k_2} b + k_2(c * d)\right)$$

$$k_0 k_1 k_2 = 001$$



Source: S. A. Islam, L. K. Sah, and S. Katkoori, “High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging,” *ACM TODAES*, 26, 1, article 6, 2020.

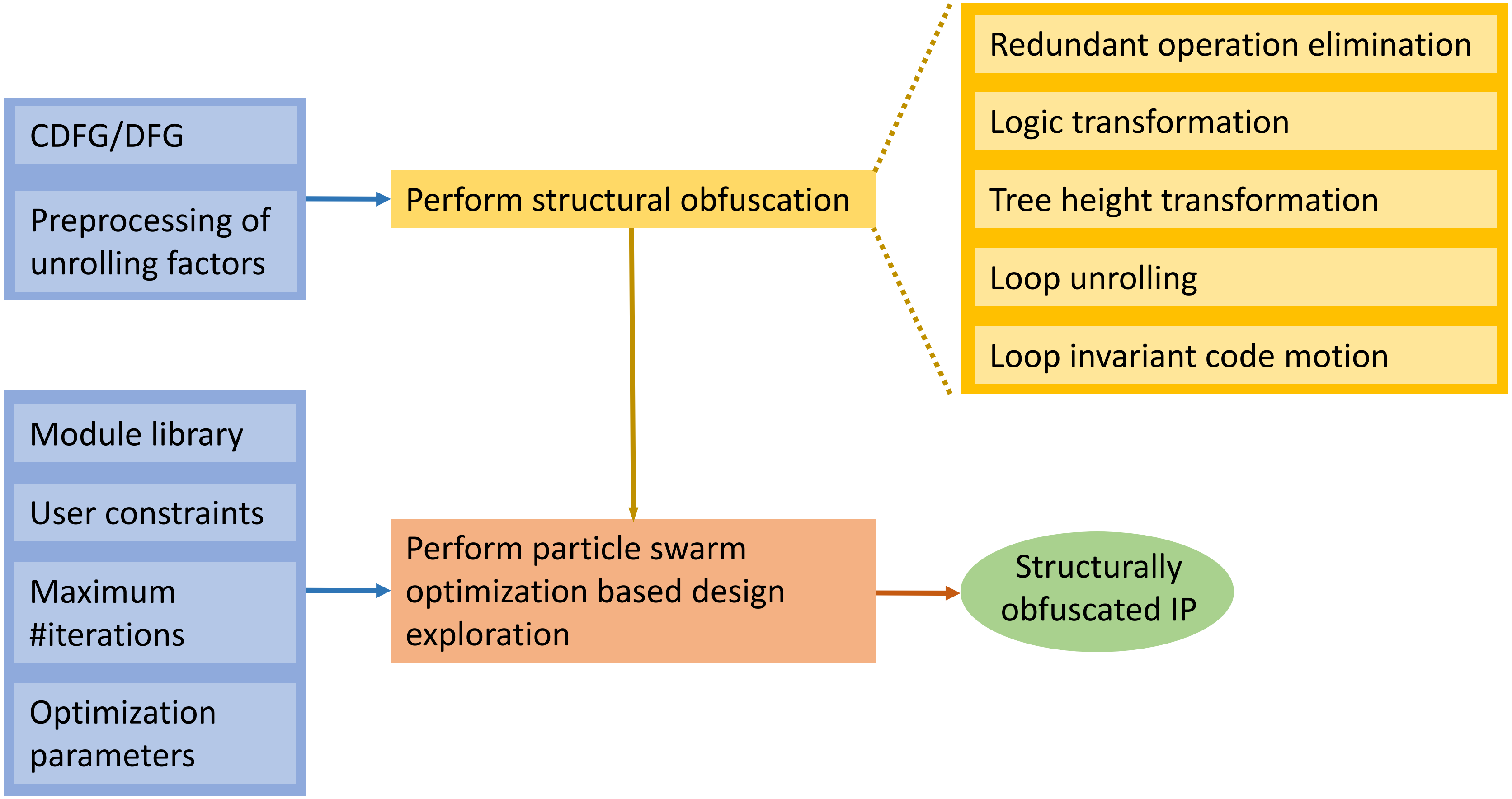
Obfuscation Techniques – TODAES’20



Design	Non-obfuscated						Obfuscated			Obfuscation Overhead		
	Latency Bound (λ)	# Operations (A=+, M=*, S=-)	# Registers (Datapath + Controller)	Area (um^2)	Delay (ns)	Power (uW)	Area (um^2)	Delay (ns)	Power (uW)	Area Overhead (%)	Delay Overhead (%)	Power Overhead (%)
Elliptic	15	(26+, 8*)	43	110941	28.04	536.03	114474	28.84	548.82	3.18	2.85	2.38
FIR	5	(4+, 5*)	19	76806	25.80	507.70	78013	26.58	509.05	1.59	3.02	0.26
FFT	10	(20+, 16*, 4-)	56	67152	19.62	320.25	69096	20.26	331.91	2.89	3.26	3.64
Lattice	10	(8+, 5*)	21	64796	26.65	360.86	66197	27.05	375.94	2.16	1.50	4.17
Average										2.45	2.65	2.617

Source: S. A. Islam and S. Katkoori, "High-Level Synthesis of Key Based Obfuscated RTL Datapaths," *ISQED*, 407-412, 2018.

Obfuscation Techniques – TCE’17



CDFG/DFG

Preprocessing of unrolling factors

Perform structural obfuscation

Redundant operation elimination

Logic transformation

Tree height transformation

Loop unrolling

Loop invariant code motion

Module library

User constraints

Maximum #iterations

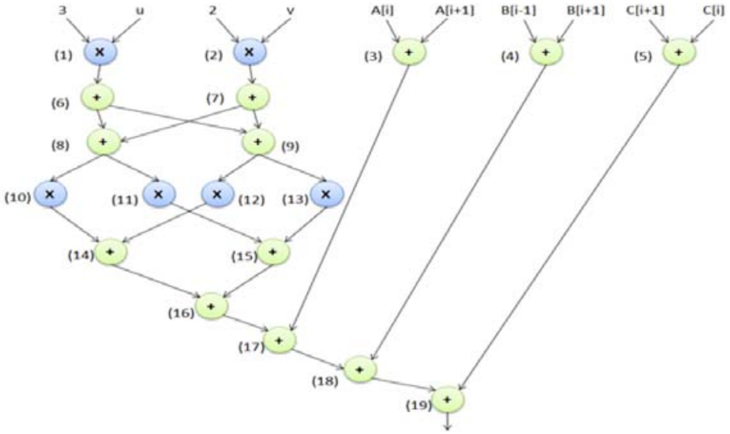
Optimization parameters

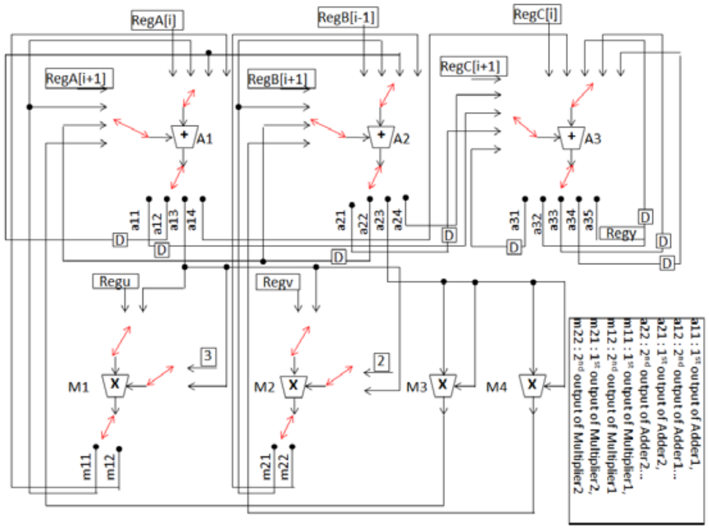
Perform particle swarm optimization based design exploration

Structurally obfuscated IP

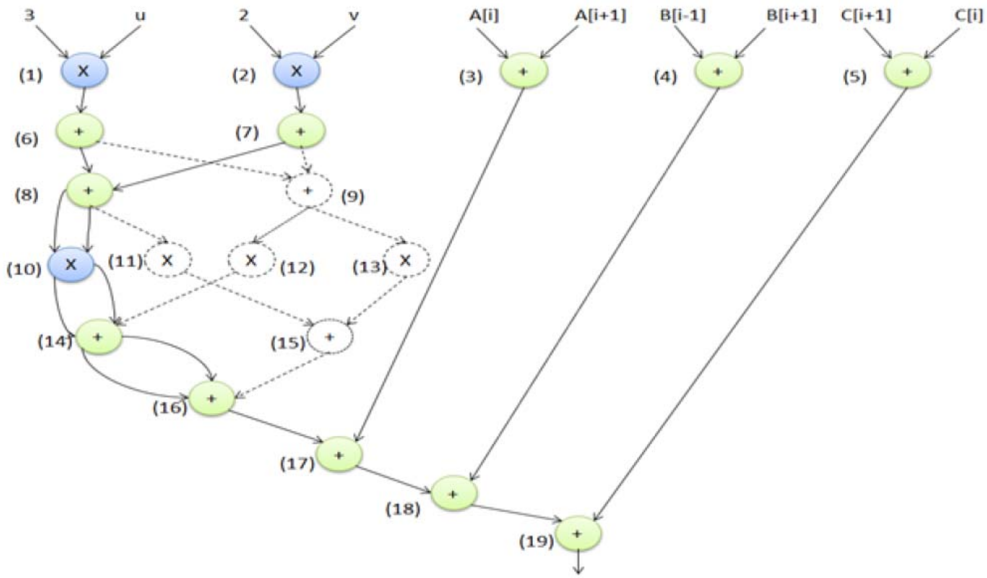
Source: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE TCE*, 63, 4, 467-476, 2017.

Obfuscation Techniques – TCE’17



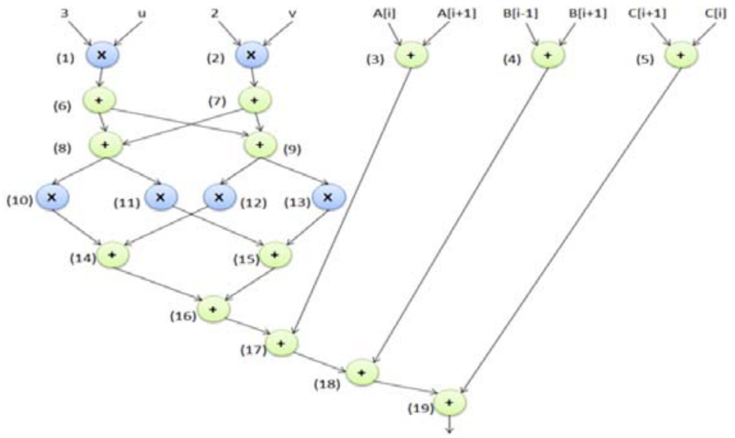

RegA[i]
RegB[i-1]
RegC[i]
RegA[i+1]
RegB[i+1]
RegC[i+1]
A1
A2
A3
a11
a12
a13
a14
a21
a22
a23
a24
a31
a32
a33
a34
a35
D
Regy
Regu
Regv
3
2
M1
M2
M3
M4
X
m11
m12
m21
m22
a11 : 1st output of Adder1,
a12 : 2nd output of Adder1...
a21 : 1st output of Adder2,
a22 : 2nd output of Adder2...
m11 : 1st output of Multiplier1,
m12 : 2nd output of Multiplier1
m21 : 1st output of Multiplier2,
m22 : 2nd output of Multiplier2

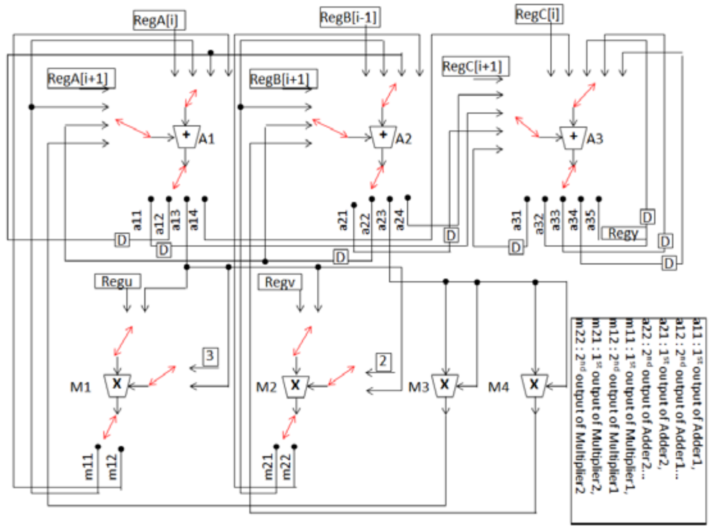
Redundant operation elimination



Source: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE TCE*, 63, 4, 467-476, 2017.

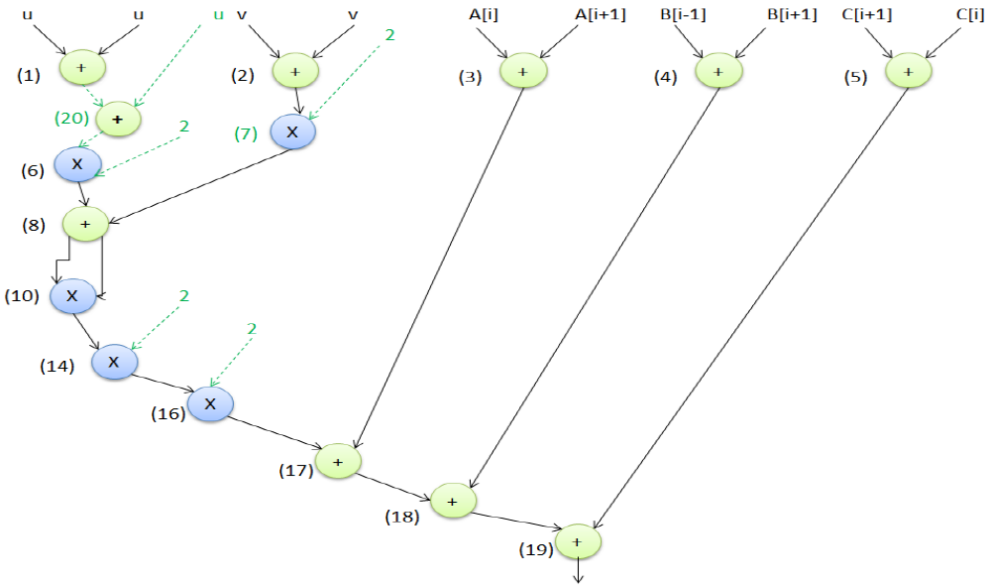
Obfuscation Techniques – TCE’17





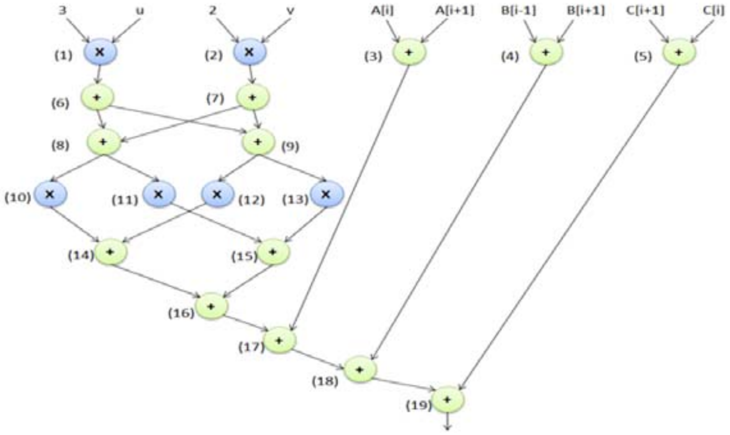
RegA[i]
RegA[i+1]
RegB[i-1]
RegB[i+1]
RegC[i]
RegC[i+1]
A1
A2
A3
a11
a12
a13
a14
a21
a22
a23
a24
a31
a32
a33
a34
a35
D
Regy
Regu
Regv
M1
M2
M3
M4
3
2
m11
m12
m21
m22
a11 : 1st output of Adder1,
a12 : 2nd output of Adder1…
a21 : 1st output of Adder2,
a22 : 2nd output of Adder2…
m11 : 1st output of Multiplier1,
m12 : 2nd output of Multiplier1
m21 : 1st output of Multiplier2,
m22 : 2nd output of Multiplier2

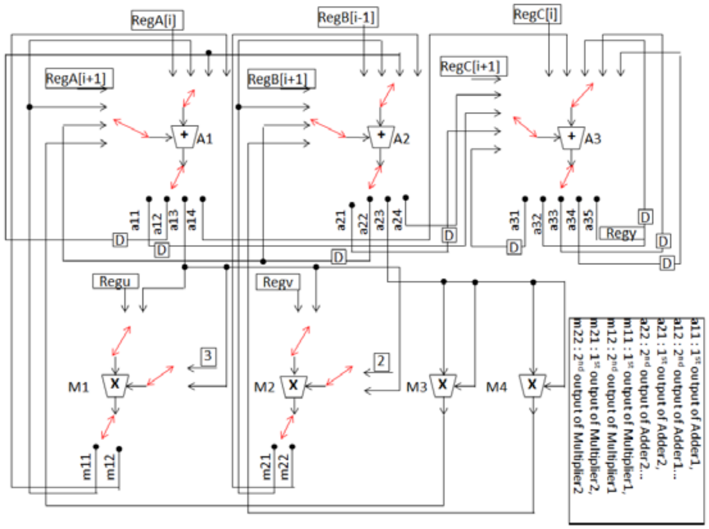
Logic transformation



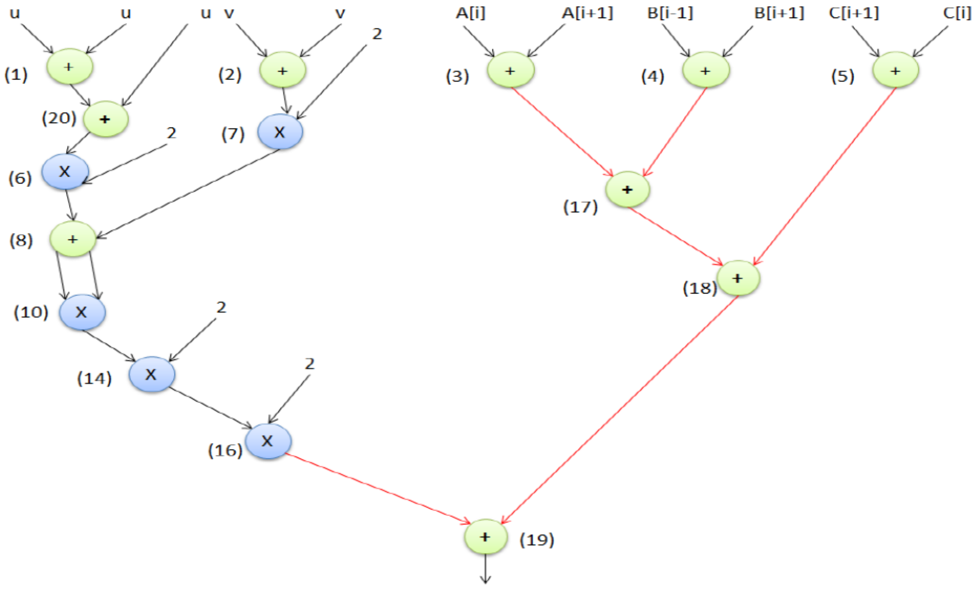
Source: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE TCE*, 63, 4, 467-476, 2017.

Obfuscation Techniques – TCE’17



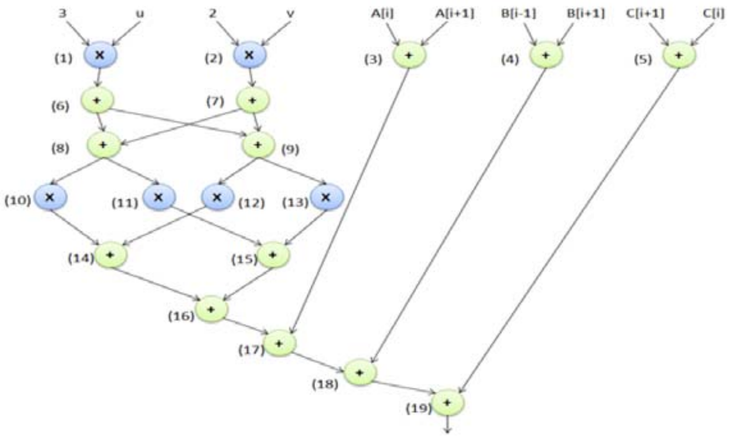

RegA[i]
RegB[i-1]
RegC[i]
RegA[i+1]
RegB[i+1]
RegC[i+1]
A1
A2
A3
a11
a12
a13
a14
a21
a22
a23
a24
a31
a32
a33
a34
a35
Regy
D
Regu
Regv
3
2
M1
M2
M3
M4
m11
m12
m21
m22
a11 : 1st output of Adder1,
a12 : 2nd output of Adder1...
a21 : 1st output of Adder2,
a22 : 2nd output of Adder2...
m11 : 1st output of Multiplier1,
m12 : 2nd output of Multiplier1
m21 : 1st output of Multiplier2,
m22 : 2nd output of Multiplier2

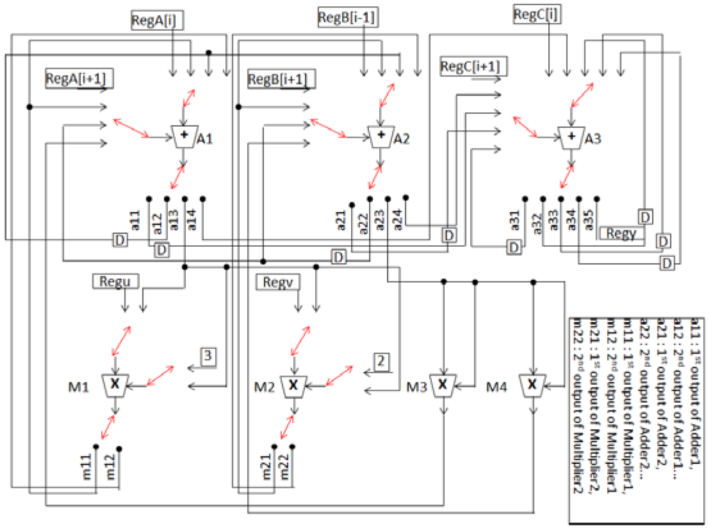
Tree height transformation



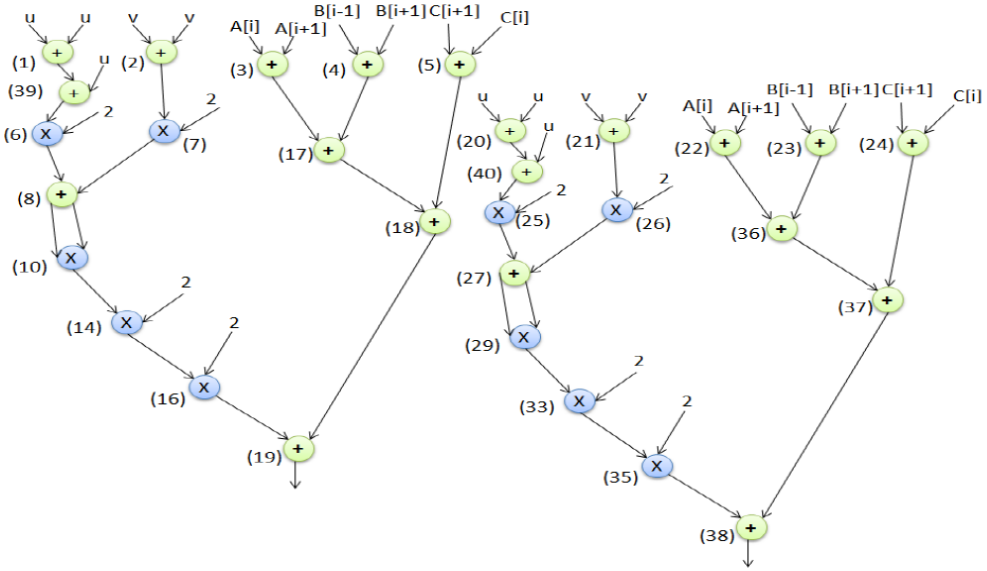
Source: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE TCE*, 63, 4, 467-476, 2017.

Obfuscation Techniques – TCE’17



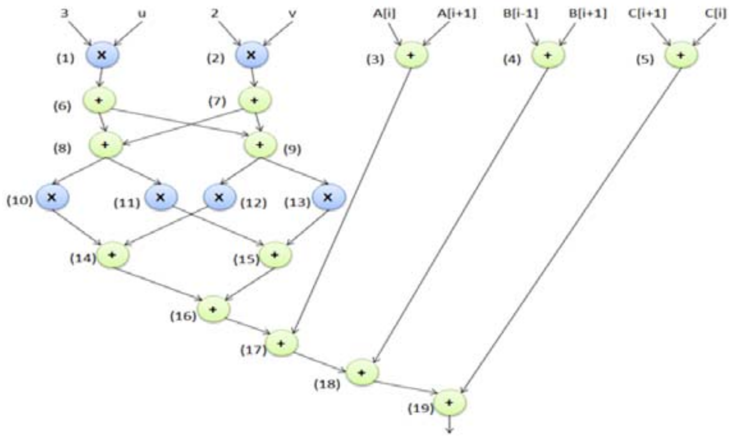

RegA[i]
RegB[i-1]
RegC[i]
RegA[i+1]
RegB[i+1]
RegC[i+1]
A1
A2
A3
a11
a12
a13
a14
a21
a22
a23
a24
a31
a32
a33
a34
a35
D
Regv
Regu
Regv
M1
M2
M3
M4
3
2
m11
m12
m21
m22
a11 : 1st output of Adder1,
a12 : 2nd output of Adder1...
a21 : 1st output of Adder2,
a22 : 2nd output of Adder2...
m11 : 1st output of Multiplier1,
m12 : 2nd output of Multiplier1
m21 : 1st output of Multiplier2,
m22 : 2nd output of Multiplier2

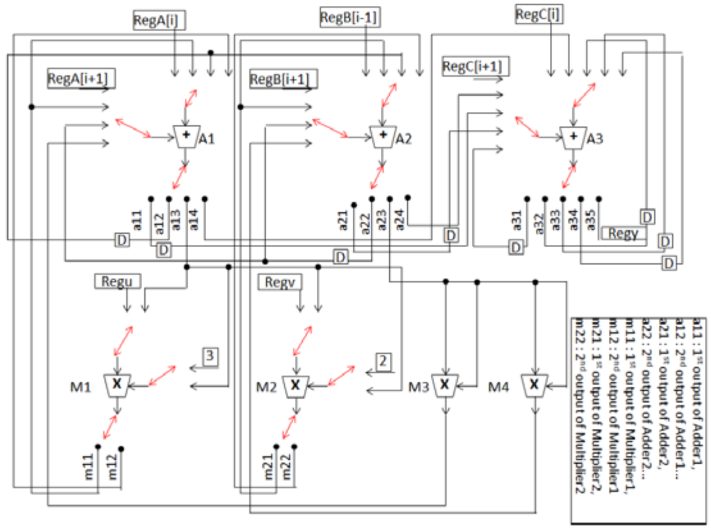
Loop unrolling



Source: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE TCE*, 63, 4, 467-476, 2017.

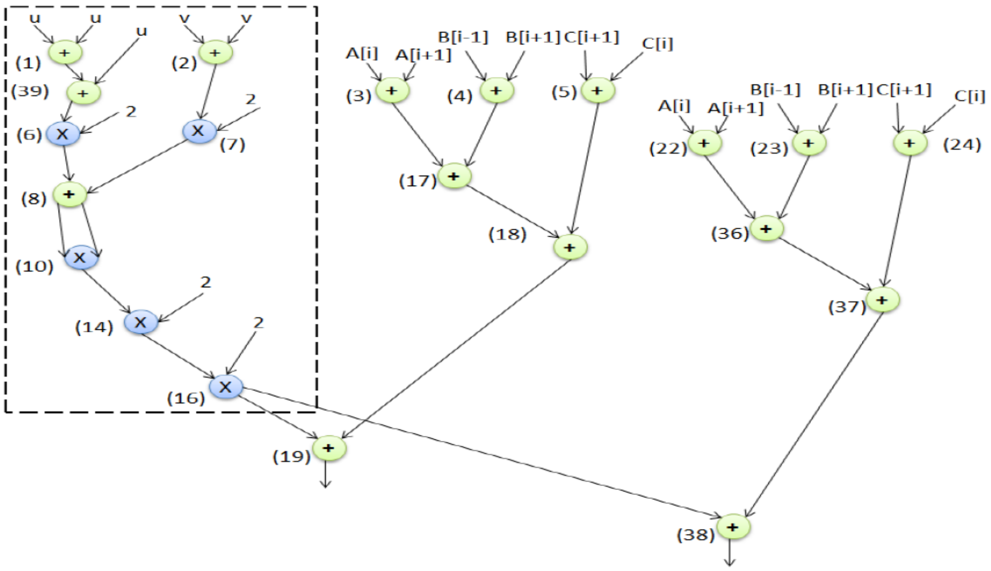
Obfuscation Techniques – TCE’17





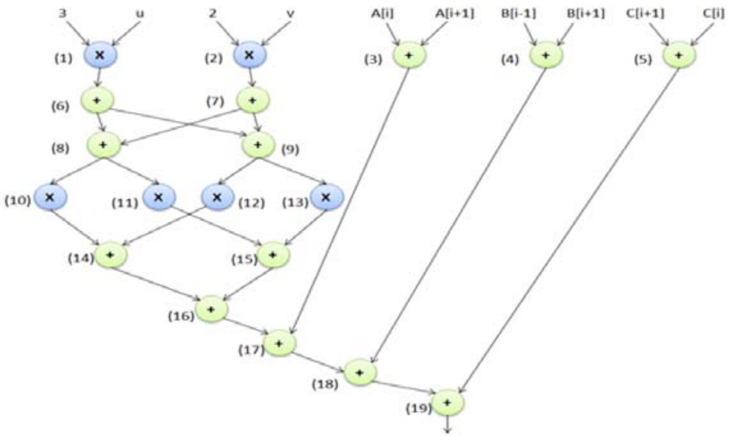
a11 : 1st output of Adder1,
a12 : 2nd output of Adder1…
a21 : 1st output of Adder2,
a22 : 2nd output of Adder2…
m11 : 1st output of Multiplier1,
m12 : 2nd output of Multiplier1
m21 : 1st output of Multiplier2,
m22 : 2nd output of Multiplier2

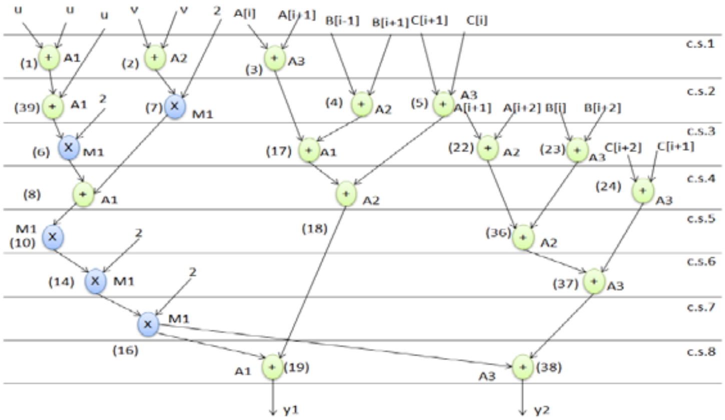
Loop invariant code motion

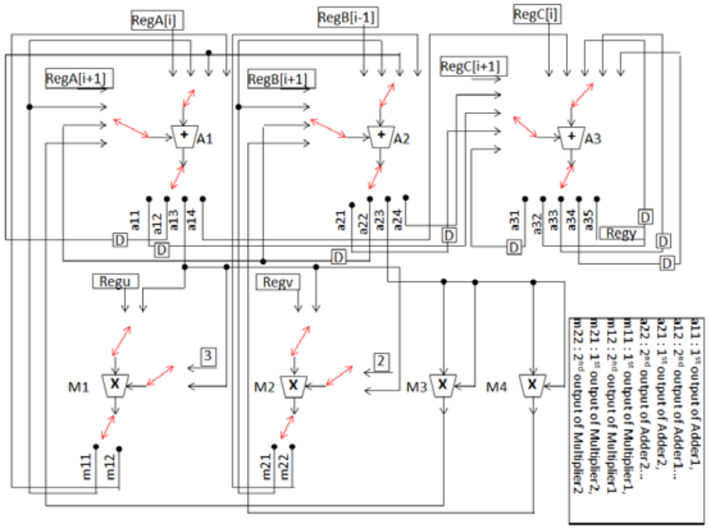


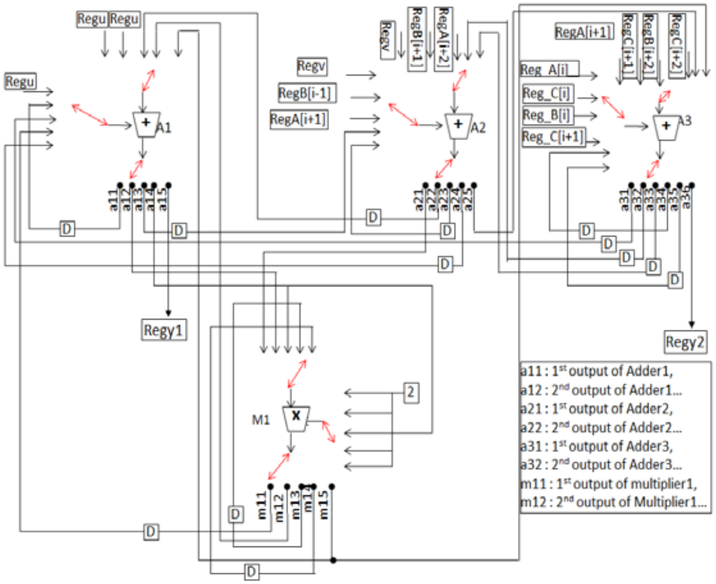
Source: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE TCE*, 63, 4, 467-476, 2017.

Obfuscation Techniques – TCE’17






RegA[i]
RegB[i-1]
RegC[i]
RegA[i+1]
RegB[i+1]
RegC[i+1]
A1
A2
A3
a11
a12
a13
a14
a21
a22
a23
a24
a31
a32
a33
a34
a35
Regy
D
Regu
Regv
M1
M2
M3
M4
3
2
m11
m12
m21
m22
a11 : 1st output of Adder1,
a12 : 2nd output of Adder1…
a21 : 1st output of Adder2,
a22 : 2nd output of Adder2…
m11 : 1st output of Multiplier1,
m12 : 2nd output of Multiplier1
m21 : 1st output of Multiplier2,
m22 : 2nd output of Multiplier2

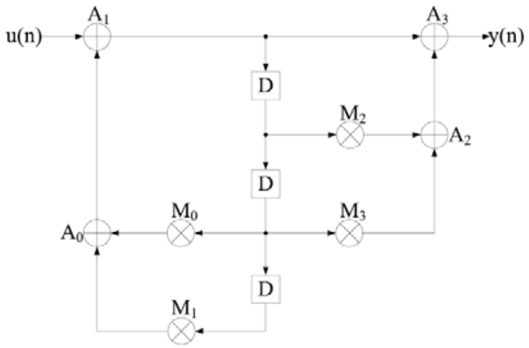

Regu
Regv
RegB[i+1]
RegA[i+2]
RegA[i+1]
RegC[i+1]
RegB[i+2]
RegC[i+2]
RegB[i-1]
Reg_A[i]
Reg_C[i]
Reg_B[i]
Reg_C[i+1]
A1
A2
A3
a11
a12
a13
a14
a15
a21
a22
a23
a24
a25
a31
a32
a33
a34
a35
a36
D
Regy1
Regy2
M1
2
m11
m12
m13
m14
m15
a11 : 1st output of Adder1,
a12 : 2nd output of Adder1…
a21 : 1st output of Adder2,
a22 : 2nd output of Adder2…
a31 : 1st output of Adder3,
a32 : 2nd output of Adder3…
m11 : 1st output of multiplier1,
m12 : 2nd output of Multiplier1…

Source: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," *IEEE TCE*, 63, 4, 467-476, 2017.

Obfuscation Techniques – TVLSI’15

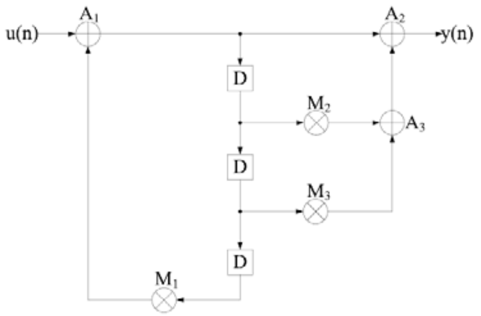
A third-order infinite impulse filter

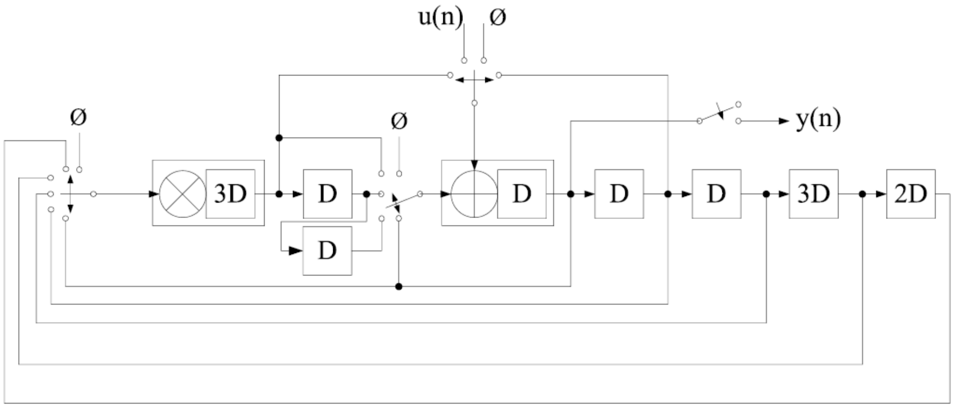
$H(z) = (1 + m_2 z^{-1} + m_3 z^{-2})/(1 - m_0 z^{-2} + m_1 z^{-3})$



Another third-order infinite impulse filter

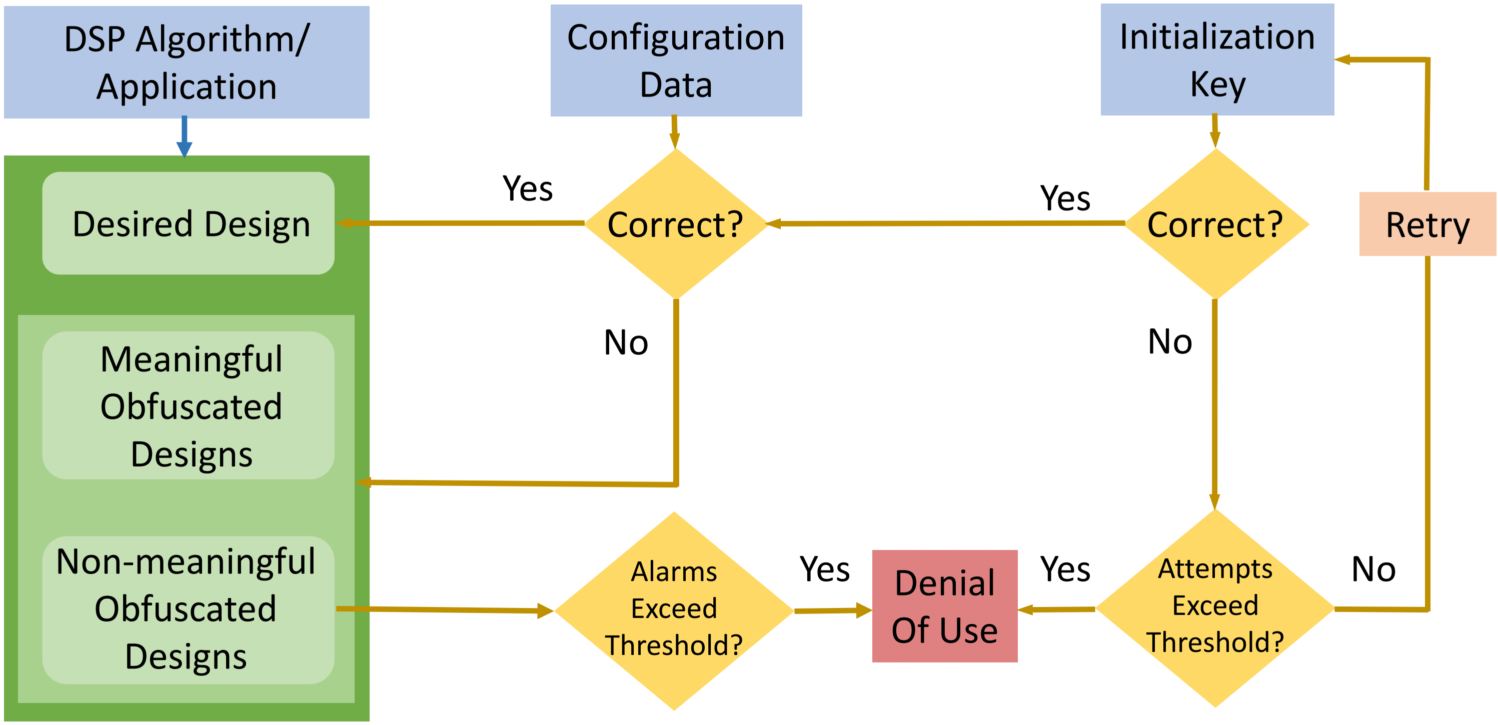
$H(z) = (1 + m_2 z^{-1} + m_3 z^{-2})/(1 - m_1 z^{-3})$





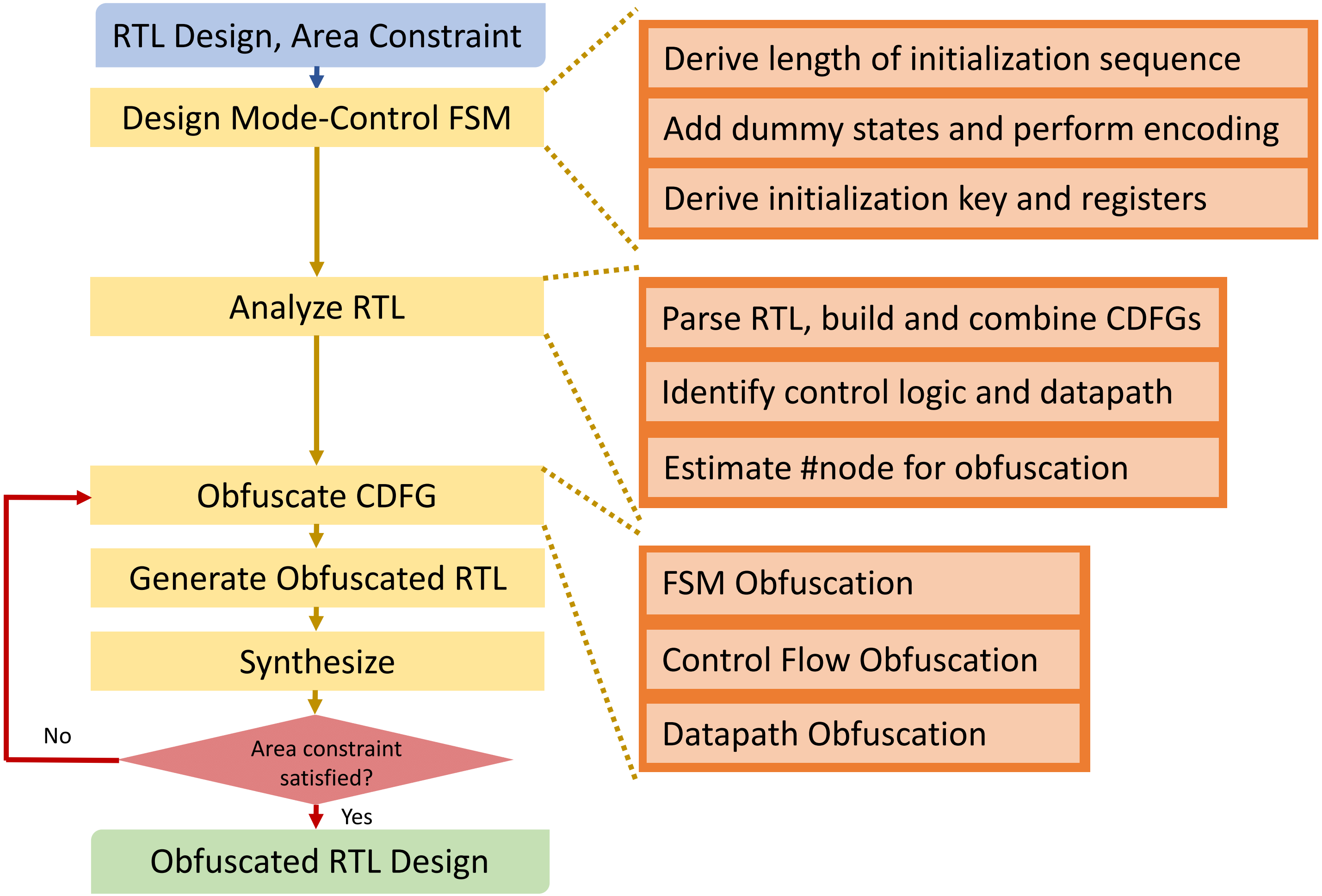
Source: Y. Lao and K. K. Parhi, "Obfuscating DSP Circuits via High-Level Transformations," *IEEE TVLSI*, 23, 5, 819-830, 2015.

Obfuscation Techniques – TVLSI’15



Source: Y. Lao and K. K. Parhi, "Obfuscating DSP Circuits via High-Level Transformations," *IEEE TVLSI*, 23, 5, 819-830, 2015.

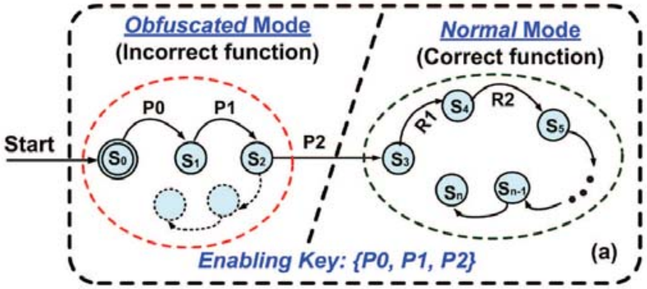
Obfuscation Techniques – VLSID'10


RTL Design, Area Constraint
Design Mode-Control FSM
Analyze RTL
Obfuscate CDFG
Generate Obfuscated RTL
Synthesize
Area constraint satisfied?
No
Yes
Obfuscated RTL Design
Derive length of initialization sequence
Add dummy states and perform encoding
Derive initialization key and registers
Parse RTL, build and combine CDFGs
Identify control logic and datapath
Estimate #node for obfuscation
FSM Obfuscation
Control Flow Obfuscation
Datapath Obfuscation

R. S. Chakraborty and S. Bhunia, "RTL Hardware IP Protection Using Key-Based Control and Data Flow Obfuscation," *International Conference on VLSI Design*, 2010, 405-410.

Obfuscation Techniques – VLSID’10

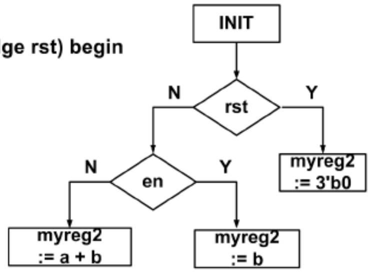
FSM Obfuscation



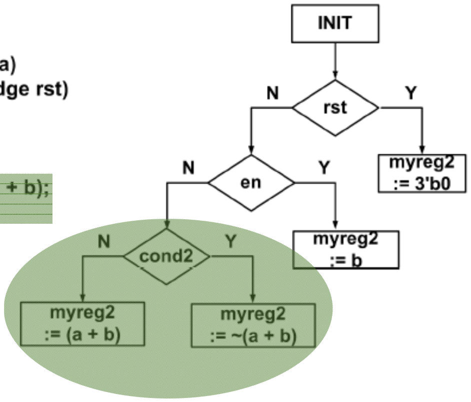
Modified State Transition Function

Control Flow Obfuscation

```
always @(posedge clk or posedge rst) begin
     if (rst) myreg2 <= 3'b0;
     else if (en) myreg2 <= a + b;
     else myreg2 <= b;
end
```

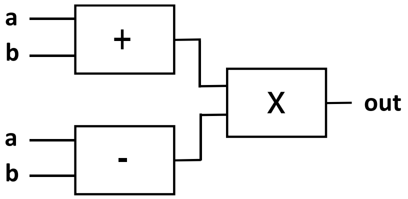


```
assign cond2 = (reg2 == 12'haaa)
always @(posedge clk or posedge rst)
begin
     if (rst) myreg2 <= 3'b0;
     else if (en) begin
          if (cond2) myreg2 <= ~(a + b);
          else myreg2 <= (a + b);
     end
     else myreg2 <= b;
end
```

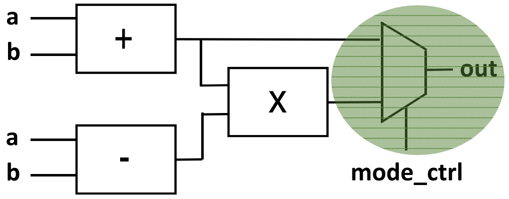


Datapath Obfuscation

assign out = (a+b) * (a-b)

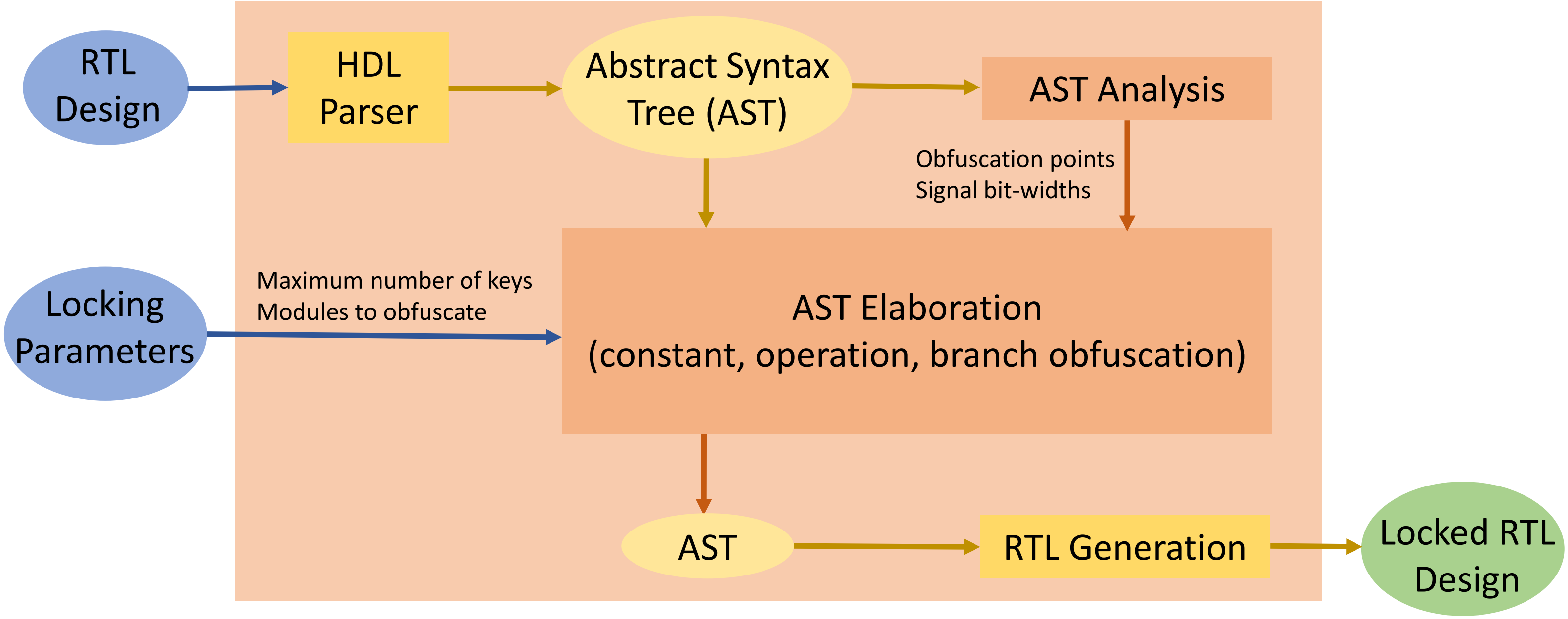


assign out = (mode_ctrl) ? (a+b) : (a+b) * (a-b)



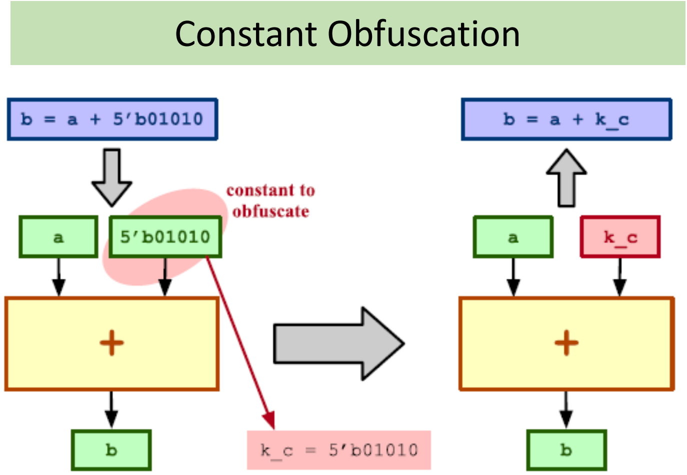
R. S. Chakraborty and S. Bhunia, "RTL Hardware IP Protection Using Key-Based Control and Data Flow Obfuscation," *International Conference on VLSI Design*, 2010, 405-410.

Obfuscation Techniques – TVLSI'21



Source: C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, 29, 7, 1306-1318, 2021.

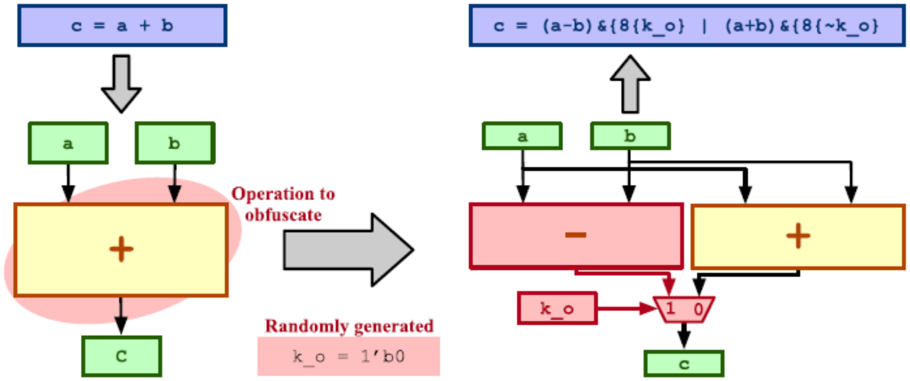
Obfuscation Techniques – TVLSI’21



Source: C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, 29, 7, 1306-1318, 2021.

Obfuscation Techniques – TVLSI’21

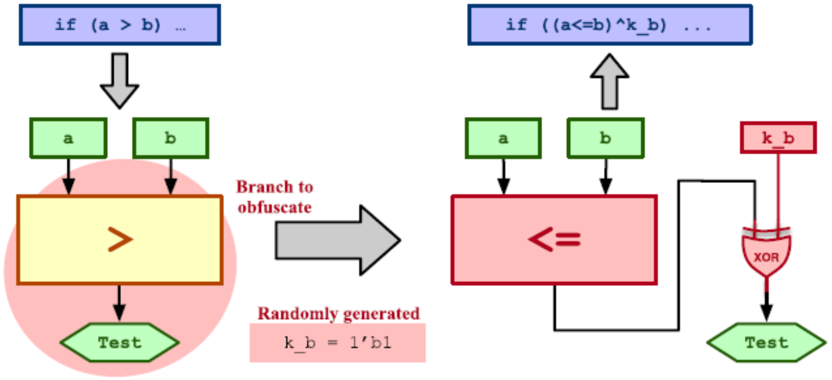
Operation Obfuscation



Source: C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, 29, 7, 1306-1318, 2021.

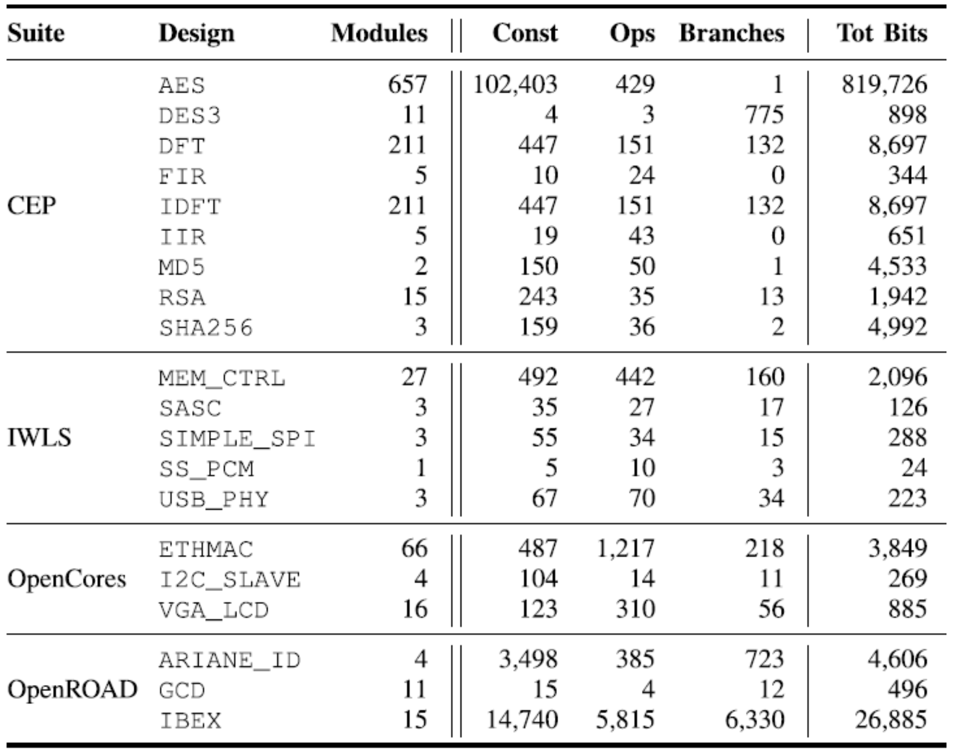
Obfuscation Techniques – TVLSI’21

Branch Obfuscation



Source: C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, 29, 7, 1306-1318, 2021.

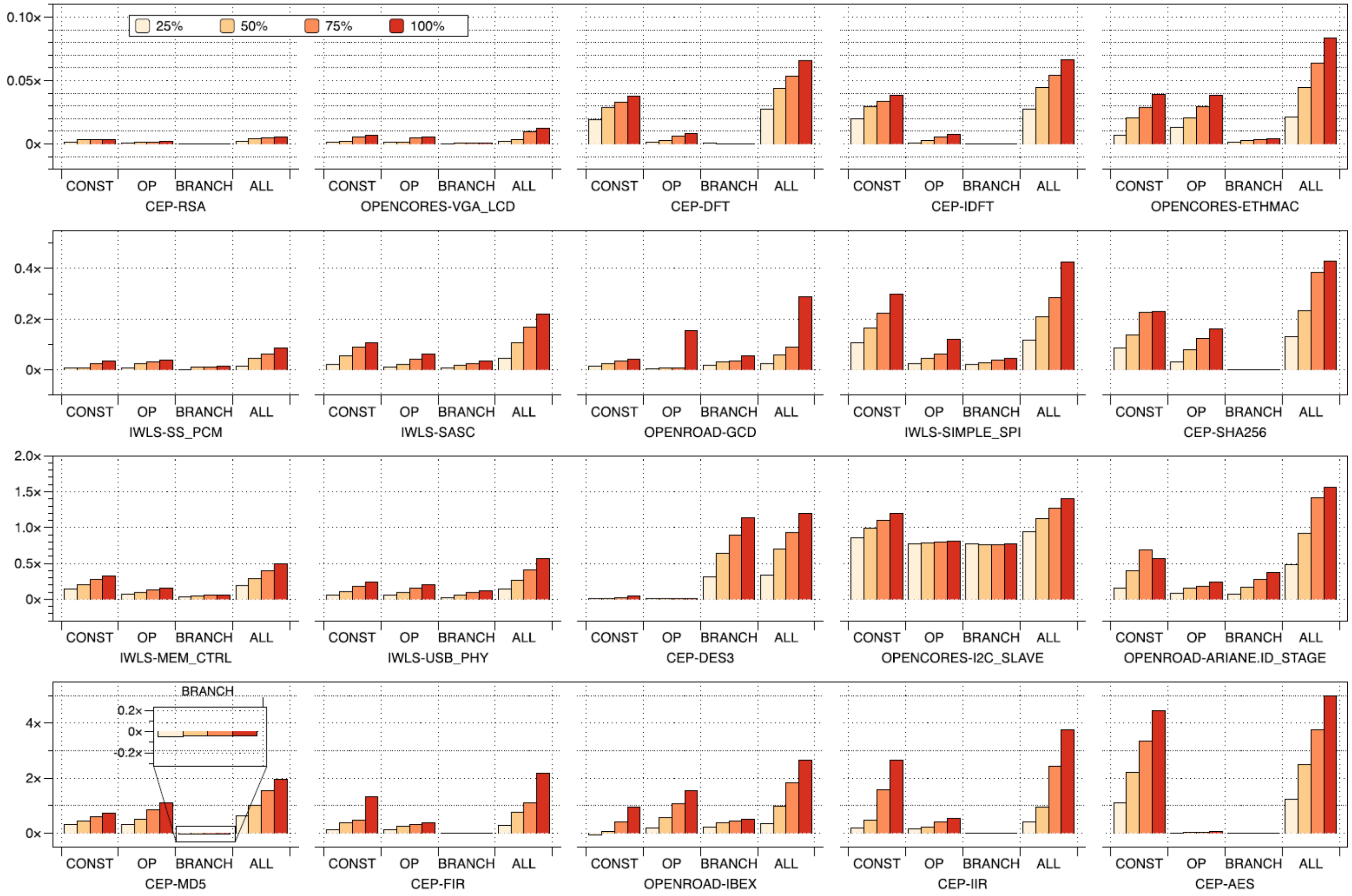
Obfuscation Techniques – TVLSI’21



Suite	Design	Modules	Const	Ops	Branches	Tot Bits
CEP	AES	657	102,403	429	1	819,726
	DES3	11	4	3	775	898
	DFT	211	447	151	132	8,697
	FIR	5	10	24	0	344
	IDFT	211	447	151	132	8,697
	IIR	5	19	43	0	651
	MD5	2	150	50	1	4,533
	RSA	15	243	35	13	1,942
	SHA256	3	159	36	2	4,992
IWLS	MEM_CTRL	27	492	442	160	2,096
	SASC	3	35	27	17	126
	SIMPLE_SPI	3	55	34	15	288
	SS_PCM	1	5	10	3	24
	USB_PHY	3	67	70	34	223
OpenCores	ETHMAC	66	487	1,217	218	3,849
	I2C_SLAVE	4	104	14	11	269
	VGA_LCD	16	123	310	56	885
OpenROAD	ARIANE_ID	4	3,498	385	723	4,606
	GCD	11	15	4	12	496
	IBEX	15	14,740	5,815	6,330	26,885

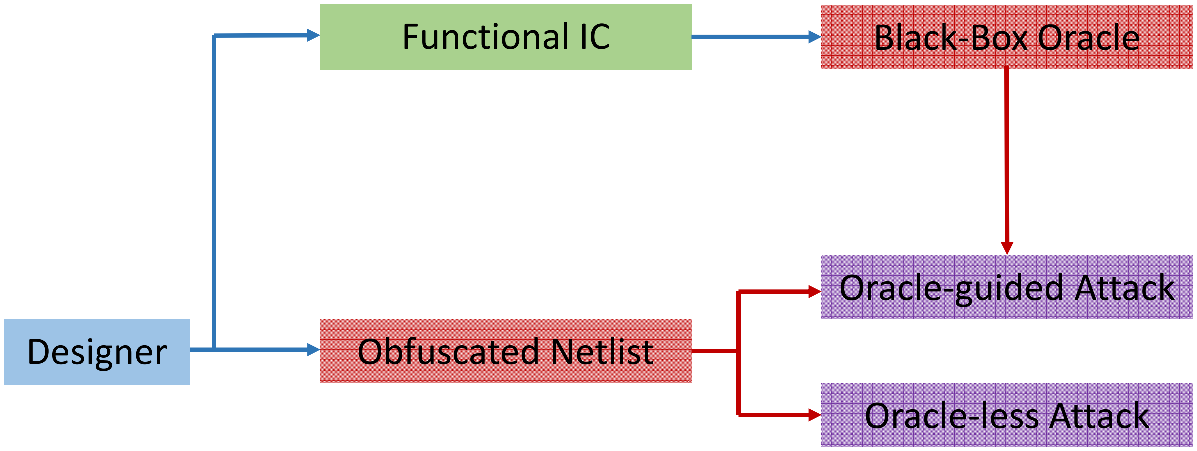
Source: C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, 29, 7, 1306-1318, 2021.

Obfuscation Techniques – TVLSI’21

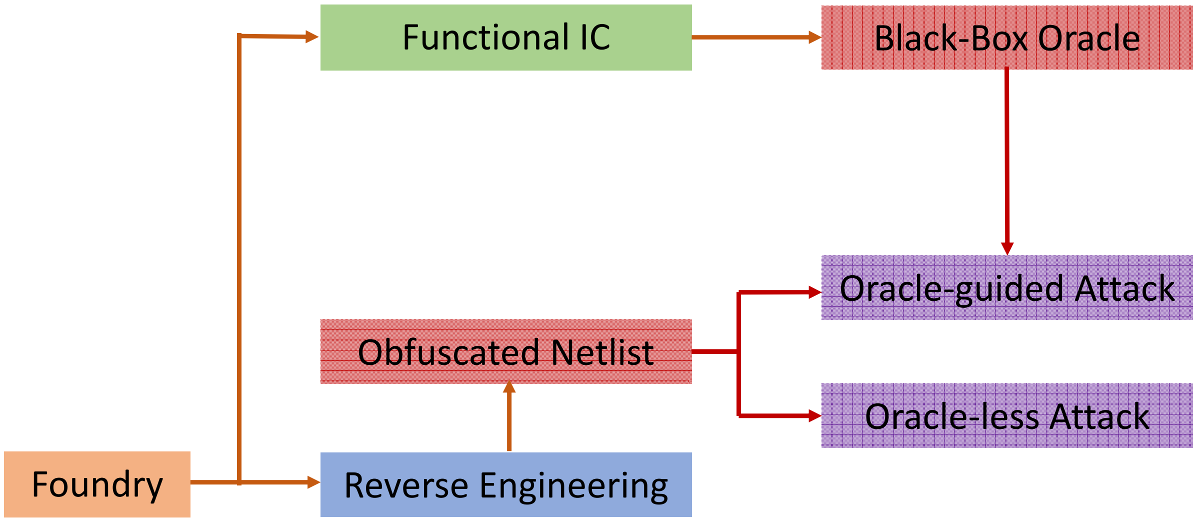


Source: C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, 29, 7, 1306-1318, 2021.

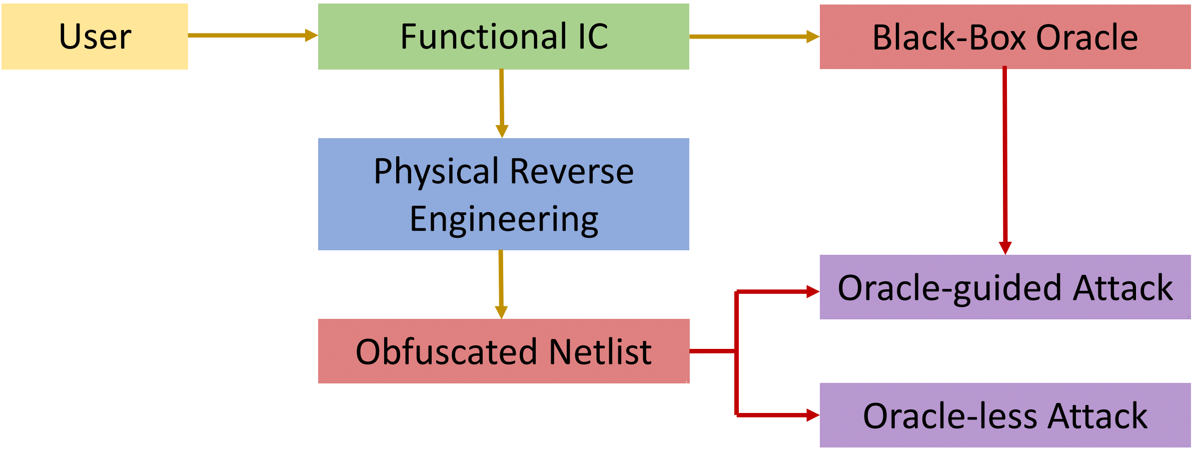
De-Obfuscation Methods – Threats Model



De-Obfuscation Methods – Threats Model



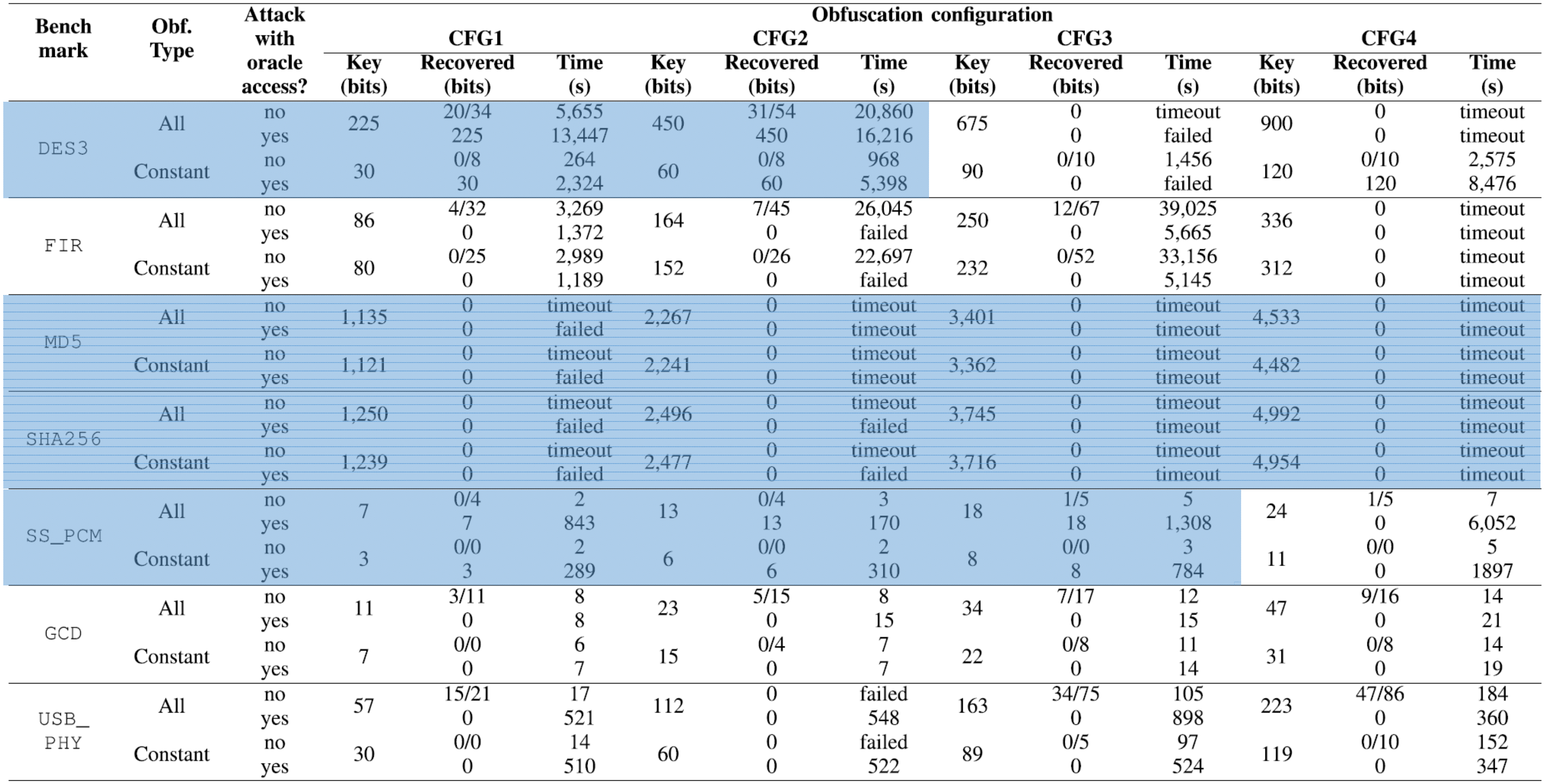
De-Obfuscation Methods – Threats Model



De-Obfuscation Methods – Gate-Level Attacks

ATPG-based Attacks	J. Rajendran and Y. Pino and O. Sinanoglu and R. Karri, “Security Analysis of Logic Obfuscation,” *DAC*, 83-89, 2012. L. Li and A. Orailoglu, “Piercing Logic Locking Keys through Redundancy Identification,” *DATE*, 540–545, 2019.
SAT-based Attacks	P. Subramanyan, S. Ray and S. Malik, “Evaluating the Security of Logic Encryption Algorithms,” *HOST*, 137-143, 2015. K. Shamsi, M. Li, D.Z. Pan and Y. Jin, “KC2: Key-Condition Crunching for Fast Sequential Circuit Deobfuscation,” *DATE*, 534-539, 2019.
SMT-based Attacks	K. Azar, H. Kamali, H. Homayoun and A. Sasan, “SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks,” *CHES,* 97–122, 2019. S. Roshanisefat, H. Kamali, H. Homayoun and A. Sasan, “RANE: An Open-Source Formal De-obfuscation Attack for Reverse Engineering of Logic Encrypted Circuits,” *GLSVLSI*, 221–228, 2021.
Approximate Attacks	K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, “AppSAT: Approximately Deobfuscating Integrated Circuits,” *HOST*, 95-100, 2017. Y. Shen, H. Zhou, “Double DIP: Re-Evaluating Security of Logic Encryption Algorithms,” *GLSVLSI*, 179-184, 2017.
ML-based Attacks	P. Chakraborty, J. Cruz, S. Bhunia, "SAIL: Machine Learning Guided Structural Analysis Attack on Hardware Obfuscation," *AsianHOST*, 56-61, 2018. L. Alrahis et al., “GNNUnlock: Graph Neural Networks-based Oracle-less Unlocking Scheme for Provably Secure Logic Locking,” *DATE*, 780-785, 2021
Structural Attacks	M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, “Removal Attacks on Logic Locking and Camouflaging Techniques,” *IEEE TETC*, 8, 517-532, 2017. Z. Han, M. Yasin, J. Rajendran, “Does Logic Locking Work with EDA Tools?,” *USENIX Security Symposium*, 1055–1072, 2021.

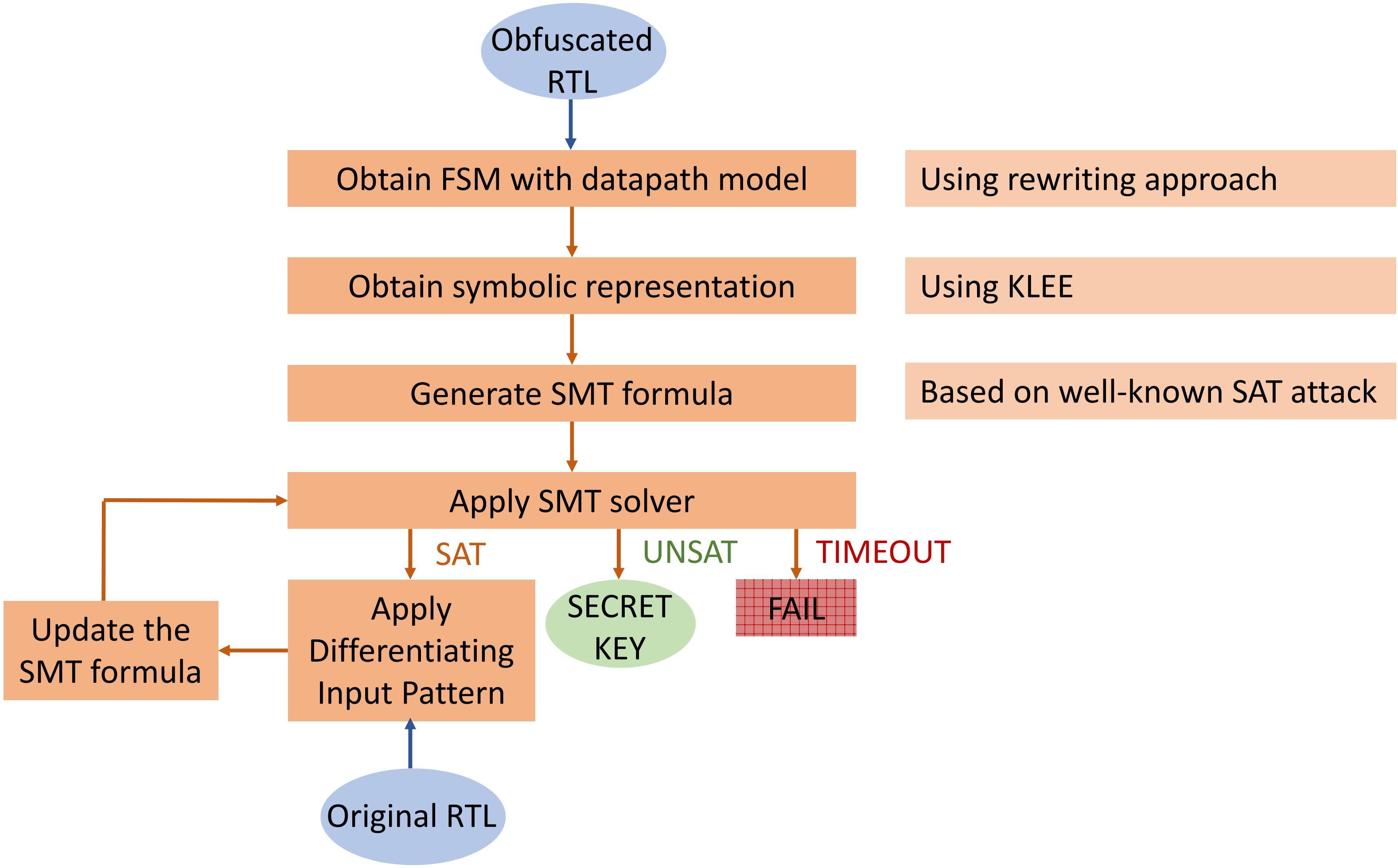
De-Obfuscation Methods – Gate-Level Attacks on TVLSI’21



Bench mark	Obf. Type	Attack with oracle access?	Obfuscation configuration											
			CFG1			CFG2			CFG3			CFG4		
			Key (bits)	Recovered (bits)	Time (s)	Key (bits)	Recovered (bits)	Time (s)	Key (bits)	Recovered (bits)	Time (s)	Key (bits)	Recovered (bits)	Time (s)
DES3	All	no	225	20/34	5,655	450	31/54	20,860	675	0	timeout	900	0	timeout
		yes		225	13,447		450	16,216		0	failed		0	timeout
	Constant	no	30	0/8	264	60	0/8	968	90	0/10	1,456	120	0/10	2,575
		yes		30	2,324		60	5,398		0	failed		120	8,476
FIR	All	no	86	4/32	3,269	164	7/45	26,045	250	12/67	39,025	336	0	timeout
		yes		0	1,372		0	failed		0	5,665		0	timeout
	Constant	no	80	0/25	2,989	152	0/26	22,697	232	0/52	33,156	312	0	timeout
		yes		0	1,189		0	failed		0	5,145		0	timeout
MD5	All	no	1,135	0	timeout	2,267	0	timeout	3,401	0	timeout	4,533	0	timeout
		yes		0	failed		0	timeout		0	timeout		0	timeout
	Constant	no	1,121	0	timeout	2,241	0	timeout	3,362	0	timeout	4,482	0	timeout
		yes		0	failed		0	timeout		0	timeout		0	timeout
SHA256	All	no	1,250	0	timeout	2,496	0	timeout	3,745	0	timeout	4,992	0	timeout
		yes		0	failed		0	failed		0	timeout		0	timeout
	Constant	no	1,239	0	timeout	2,477	0	timeout	3,716	0	timeout	4,954	0	timeout
		yes		0	failed		0	failed		0	timeout		0	timeout
SS_PCM	All	no	7	0/4	2	13	0/4	3	18	1/5	5	24	1/5	7
		yes		7	843		13	170		18	1,308		0	6,052
	Constant	no	3	0/0	2	6	0/0	2	8	0/0	3	11	0/0	5
		yes		3	289		6	310		8	784		0	1897
GCD	All	no	11	3/11	8	23	5/15	8	34	7/17	12	47	9/16	14
		yes		0	8		0	15		0	15		0	21
	Constant	no	7	0/0	6	15	0/4	7	22	0/8	11	31	0/8	14
		yes		0	7		0	7		0	14		0	19
USB_ PHY	All	no	57	15/21	17	112	0	failed	163	34/75	105	223	47/86	184
		yes		0	521		0	548		0	898		0	360
	Constant	no	30	0/0	14	60	0	failed	89	0/5	97	119	0/10	152
		yes		0	510		0	522		0	524		0	347

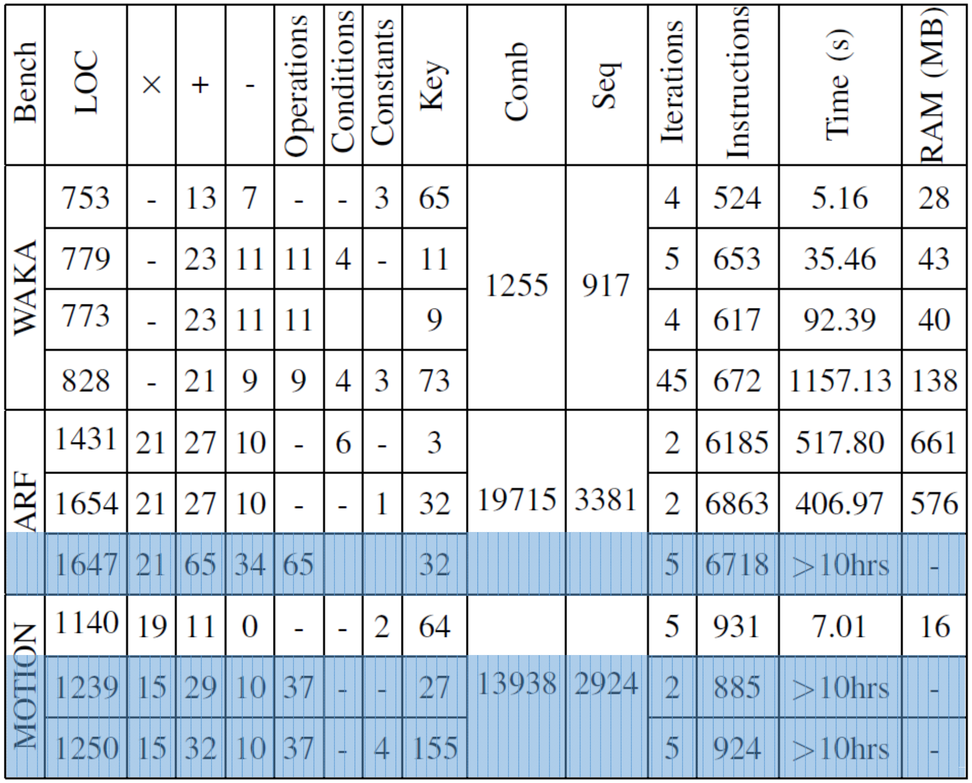
Source: C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE TVLSI*, 29, 7, 1306-1318, 2021.

De-Obfuscation Methods – DATE'20



Source: C. Karfa, R. Chouksey, C. Pilato, S. Garg and R. Karri, "Is Register Transfer Level Locking Secure?," DATE, 550-555, 2020.

De-Obfuscation Methods – DATE’20



Bench	LOC	×	+	-	Operations	Conditions	Constants	Key	Comb	Seq	Iterations	Instructions	Time (s)	RAM (MB)
WAKA	753	-	13	7	-	-	3	65	1255	917	4	524	5.16	28
	779	-	23	11	11	4	-	11			5	653	35.46	43
	773	-	23	11	11			9			4	617	92.39	40
	828	-	21	9	9	4	3	73			45	672	1157.13	138
ARF	1431	21	27	10	-	6	-	3	19715	3381	2	6185	517.80	661
	1654	21	27	10	-	-	1	32			2	6863	406.97	576
	1647	21	65	34	65			32			5	6718	>10hrs	-
MOTION	1140	19	11	0	-	-	2	64	13938	2924	5	931	7.01	16
	1239	15	29	10	37	-	-	27			2	885	>10hrs	-
	1250	15	32	10	37	-	4	155			5	924	>10hrs	-

Source: C. Karfa, R. Chouksey, C. Pilato, S. Garg and R. Karri, "Is Register Transfer Level Locking Secure?," DATE, 550-555, 2020.

Conclusions

- Many hardware-efficient obfuscation techniques have been introduced at behavioral level
  - different input parameters
  - different obfuscation styles
  - different obfuscation parameters
- No provably-secure behavioral obfuscation techniques have been proposed
- Only a single behavioral de-obfuscation method has been introduced
  - avoiding the increase in the problem complexity observed at gate-level
- No de-obfuscation methods, that can handle all the designs obfuscated at the behavioral level, have been proposed

Questions

THANKS for YOUR ATTENTION

Contact: Levent Aksoy
E-mail: levent.aksoy@taltech.ee