Team Name: *Downloading Often Is Terrible (D.O.I.T)*

Team member names: Kai Zhao

CS759 Final Report
IC Piracy Prevention via Design Withholding and Split Manufacturing

University of Wisconsin-Madison

2015 Fall

Table of Contents

Abstract: **Globalization of the semiconductor industry has raised serious concerns about hardware security. In particular, untrusted foundries could engage in reverse engineering to steal design information, or engage in IC piracy by overproducing chips to sell in the black market. Design withholding and split manufacturing are both candidates to address these attacks. Design withholding replaces a part of the design with a reconfigurable block on chip. The chips will not work until the designer activates the reconfigurable block. Split manufacturing withholds even larger portions of the design to be manufactured in another foundry. The dies can then be packaged together into one chip using 3D VLSI technology. This paper proposes combining the two ideas of design withholding and split manufacturing for IC piracy prevention. This paper analyzes IC piracy from a powerful attacker's point of view. Design withholding and split manufacturing has double exponential attacking cost at a linear hardware cost.**

1. Introduction

In the past, semiconductor companies designed, manufactured, and tested integrated circuits (IC) in-house. However, continuously shrinking transistor sizes due to Moore's Law imposes an expensive overhead cost of upgrading fabrication facilities to remain competitive in the semiconductor industry. Therefore, companies carrying out design and fabrication alone are no longer feasible due to the increased time-to-market pressure for many high-speed and low-power chips [9]. In fact, more and more semiconductor companies like Qualcomm, Broadcom, Texas Instruments, and even AMD is choosing to outsource fabrication as shown in figure 1.1 [2].

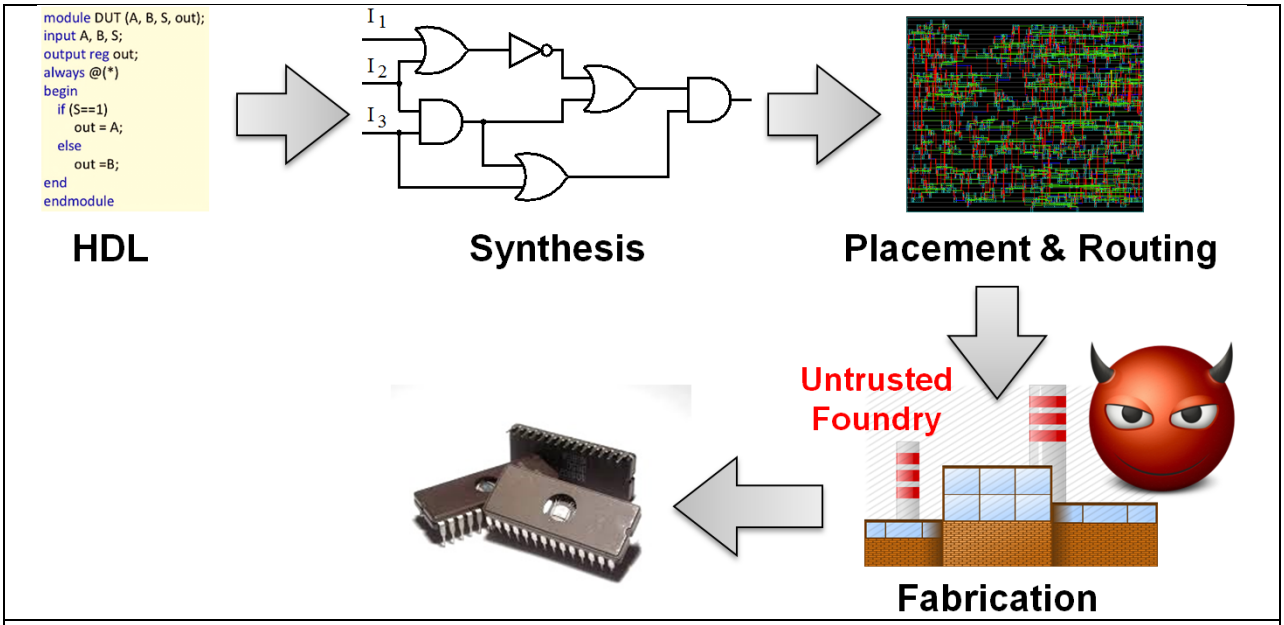


Figure 1.1: Integrated circuit design flow with outsourced fabrication.

Outsourcing the fabrication and testing process of chips to presumably untrusted foundries raises concerns in intellectual property (IP) theft, counterfeit, and hardware Trojans [3]. The foundry could engage in reverse engineering to steal design information, or engage in IC piracy by overproducing chips to sell in the black market. This occurred in the "fake NEC Corporation" [4]. NEC is a company that

manufactures electronics like speakers and routers. Pirates manufactured and distributed cheap, inferior, and copied NEC products carrying the NEC brand. The pirates kept all the proceeds while the real NEC company is left with complaints about their products. Hardware security is momentous because masks can be stolen for industrial and military purposes, and global hardware piracy costs $4 billion annually [5].

This paper proposes combining the two ideas of design withholding and split manufacturing for IC piracy prevention. This paper also looks into IC piracy prevention for sequential circuits. Lastly, this paper accounts for the parallel computing capabilities of a powerful attacker.

2. Previous Works

IC piracy prevention via design withholding and entanglement withholds a small portion of the design to be replaced with lookup tables (LUTs). The design is then sent to the foundry to be fabricated. Afterwards, the chips are returned to the designers to activate by programming the LUTs. This approach was successful in exponentially increasing the cost of the attack while only imposing a relatively small hardware overhead cost. The time to deduce the key for the LUTs is over 10 years by using a hardware overhead of 15% on 8-bit ALU, 16x16 multiplier, and 32-bit adder/comparator (c3540, c5315, c6288, and c7552 of ISCAS '85 circuits) [1]. Design withholding and entanglement has a very high computational complexity ($O(2^{n\times k})$) at relatively low hardware cost ($O(k \times 2^n + 2^k)$), where *k* is the number of one-output LUT, each with *n* address bits. The basic idea of design withholding is illustrated in figure 2.1.

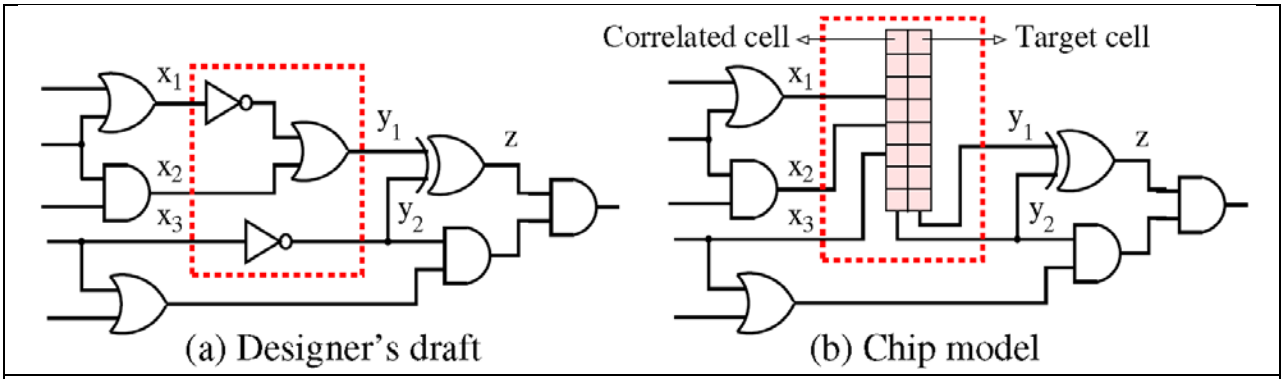


Figure 2.1: Withholding a multi-output function elevates each attack from NP-Complete to NP-Hard complexity. $2^{number\ of\ inputs}$ cells are required for every output to model all possible functions. For this example, 8 cells (3 inputs) are required for each of the 2 outputs, which totals to 16 cells.

Automatic Test Pattern Generation (ATPG) is a design automation method of propagating circuit faults generated by defects. ATPG can be used to propagate the cell key values of the LUT to the primary output. A fast-fault simulator (FSIM) simulates the circuit in the forward order to prune off unnecessary gates in the early stages [6]. Figure 2.2 is a demonstration of using ATPG to solve 1 key cell.

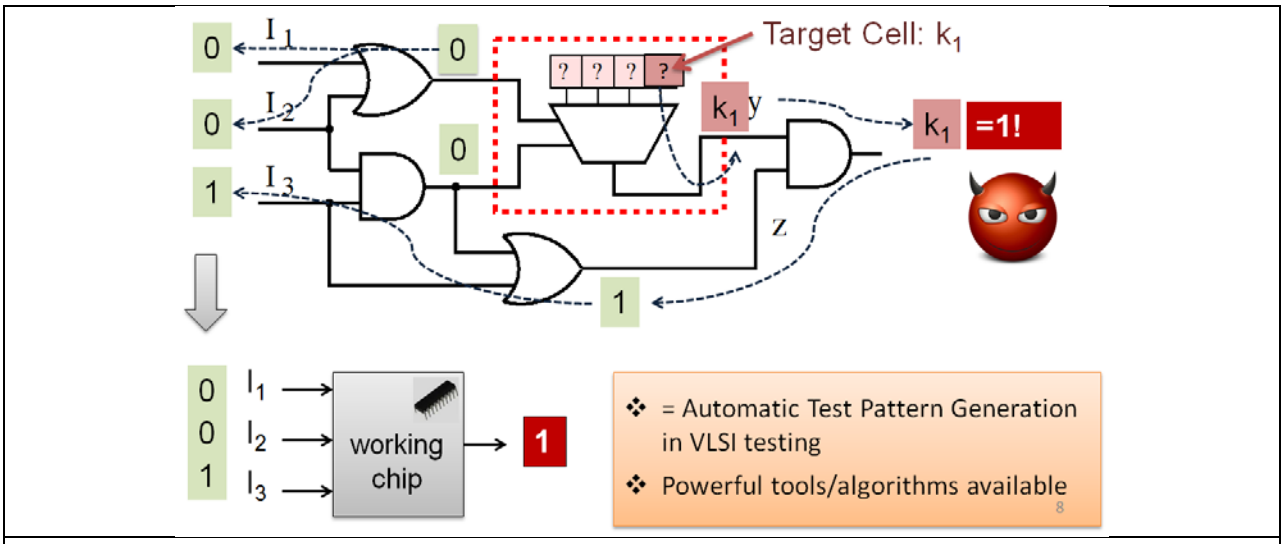


Figure 2.2: Example of using ATPG to solve 1 key cell. In order to solve the first key cell, the attacker must force the input of the LUT to be 00. This in turn forces $I_1 I_2$ to be 00. The key will then propagate to the primary output in the working chip via y only if z is 1, which can be set if I_3 is 1. Therefore, an input pattern of $I_1 I_2 I_3 = 001$ will solve the first key cell.

Split manufacturing thwarts IC piracy by splitting the design into multiple dies, which are fabricated separately in different foundries. The dies can then be packaged together into one chip using 3D IC technology [8]. Each die includes a portion of the design, which includes gates and interconnections. The basic idea of this model of split manufacturing is illustrated in figure 2.3.

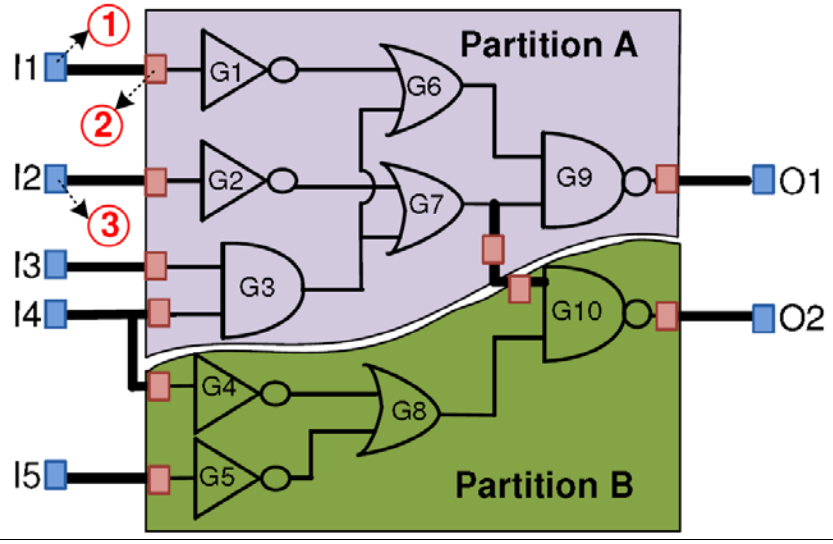


Figure 2.3: In split manufacturing, the two partitions (A and B) are manufactured in different foundries [2].

3. Problem Formulation

On a high level, the designer wants to manufacture an IC chip design with constraints of low hardware overhead and high security.

First, the chip designer can prevent giving the entire design to untrusted foundries by replacing a small portion with a look-up table (LUT). The attacker then has to solve each key cell to obtain the LUT data and activate the chips.

Second, the chip designer can also use split manufacturing either with a trusted foundry or with foundries that do not collaborate with each other. The goal of split manufacturing is to inhibit the attacker's ability of putting the design back together by exponentially increasing the number of key cells to solve.

A strong IC piracy prevention scheme assumes a powerful attacker in terms of knowledge and utilities. Like modern cryptography, the scheme should rely on the secrecy of a certain key that takes an unacceptably long time to deduce. Therefore, the attacker will have 1) access to the gate-level net list design sent to a fabrication foundry, 2) fast computation tools and algorithms with a cluster of computers, and 3) access to functional chips purchased from the open market, which are activated by the designer. The goal of the attacker is to deduce the withheld portions and complete the design. After the design is complete, attackers can then make modifications or overproduce the chip to sell for profit.

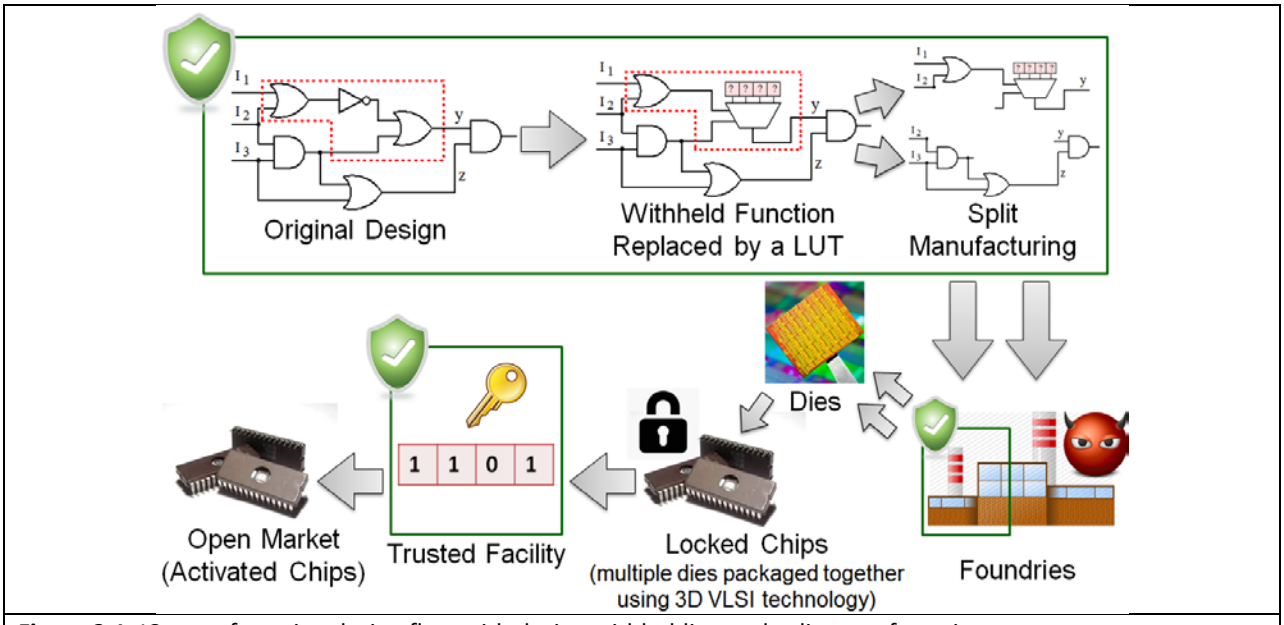


Figure 3.1: IC manufacturing design flow with design withholding and split manufacturing.

The entire design flow with withholding and split manufacturing is illustrated in figure 3.1. The goal of IC piracy prevention via split manufacturing is to cause an exponentially expensive attacking cost at a low hardware cost. The problem is formulated to play the role of the attacker and show that deducing the withheld portions is computationally unacceptable.

4. Methods and Procedure

This paper explores 1) sequential circuits, 2) design withholding with split manufacturing, and 3) parallel computing for IP piracy prevention. The entire procedure of all 3 ideas combined is shown in figure 4.1.

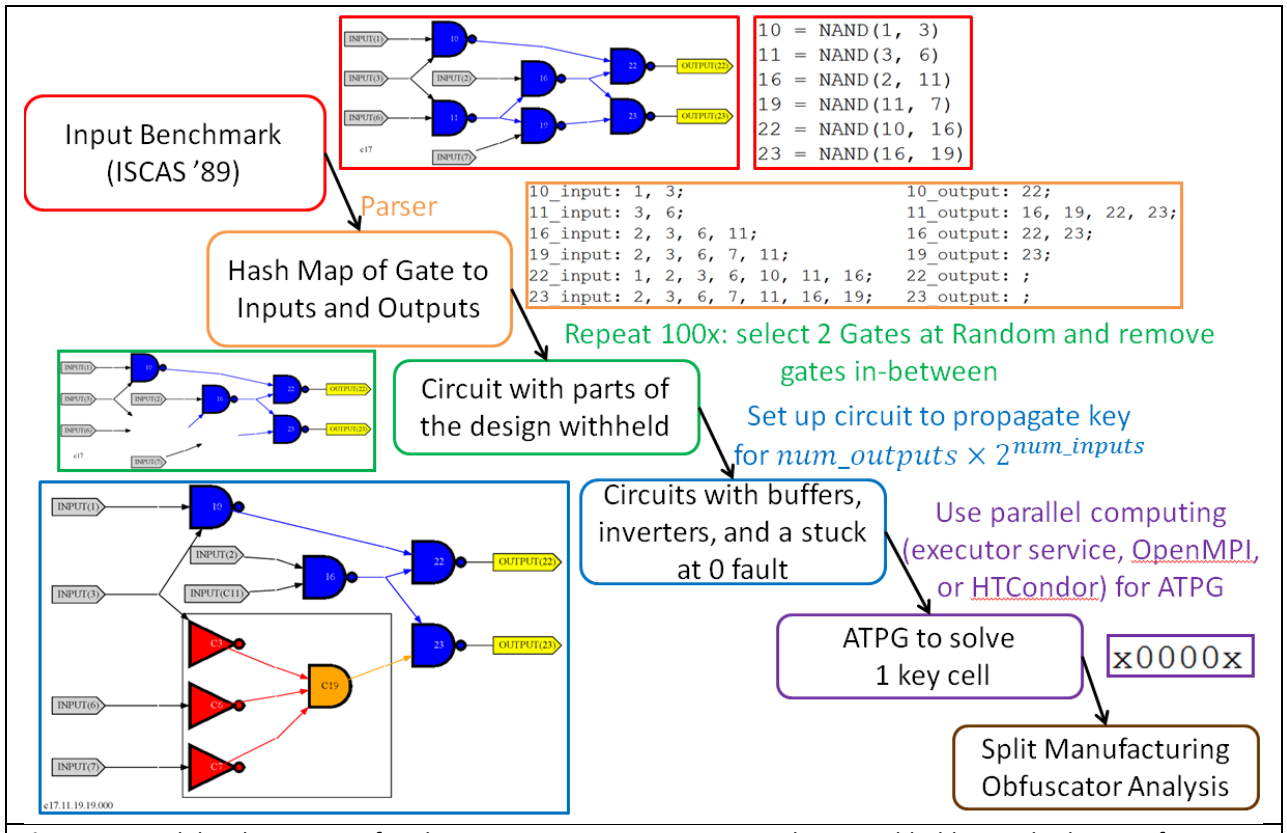


Figure 4.1: High level overview of each step in IC piracy prevention via design withholding and split manufacturing.
Benchmarks: ISCAS '89 sequential benchmark circuits are used as a basis for comparison
Parser: takes the list of each gate and stores entire input and output path into a hash map
Hash map of gate to inputs and outputs: shows all other gates that propagate in and out of key gate
Circuit with parts withheld: part of the design sent to a foundry for fabrication
Removing gates in-between: remove any overlapping gates that are outputs of one gate and inputs of the other gate
Circuit with buffers, inverters, an AND gate, and a fault: forces the attack to target a specific entry in the LUT
Parallel computing: solving multiple keys can be done in parallel. OpenMPI and HTCondor supports using multiple cores in a cluster of computers
ATPG: automatic test pattern generator finds the input test vector to solve 1 key cell
Split Manufacturing Obfuscator Analysis: analyze the complexity of adding split manufacturing

Benchmarks: ISCAS '89 benchmarks circuits are sequential circuits distributed at the 1989 International Symposium on Circuits and Systems for comparing results in test generation. The ISCAS '89 circuits have been widely used in the research community as a basis for comparison [7]. Table 4.1 provides the list of ISCAS '89 benchmarks circuits.

Table 4.1: ISCAS'89 Benchmarks

Circuit Name	Circuit Function	Number of Transistors
s349	4-bit multiplier	690
s298, s400, s444, s526	Traffic light controllers	582, 736, 758, 1058
s9234, s13207, s15850, s38417, s38584	32-bit SEC circuit	18714, 25694, 31960, 73128, 82394
s386, s510, s953, s1494	Controllers synthesized from high level description	930, 974, 1656, 3902
s1238	Combinational circuit with inserted flip-flops	2574
s208, s420, s838	Digital fractional multipliers	374, 746, 1488
s298, s208, s713, s641, s832	Programmable logic devices (PLDs)	582, 374, 1404, 1284, 1822
s344, s382, s526n, s641, s820, s1196, s1488	Re-synthesized from other benchmarks after removing redundancies in full-scan mode	680, 712, 1056, 1284, 1786, 2456, 3874

There are computational benefits of using sequential circuits over combinational. Sequential circuit requires drivability analysis with nine-valued logic instead of five-valued logic as shown in figure 4.2. Also, drivability analysis for sequential circuits has to be repeated for every primary output until justification is achieved while combinational circuits can use any fan-out. The higher complexity of doing ATPG on sequential circuits increases the runtime and helps deter the attack.

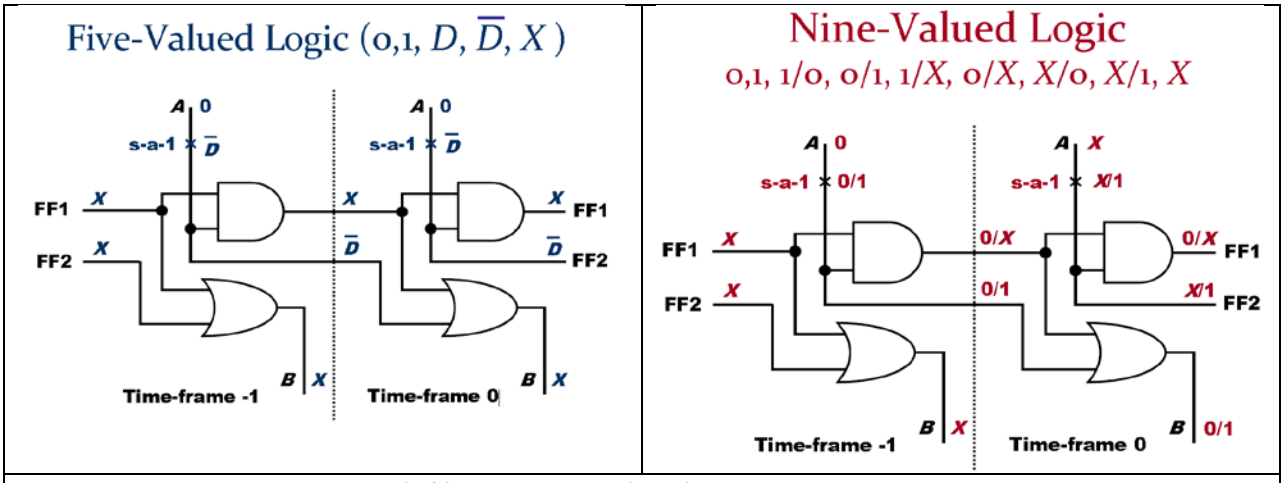


Figure 4.2: ATPG on combinational (left) and sequential (right) circuits.

Removing gates in-between: Removing a path of connected gates makes the protection scheme stronger with an exponentially larger LUT. Figure 4.3 shows an example of the gates to remove for two pairs of selected gates.

```
10_input: 1, 3;                    10_output: 22;
11_input: 3, 6;                    11_output: 16, 19, 22, 23;
16_input: 2, 3, 6, 11;             16_output: 22, 23;
19_input: 2, 3, 6, 7, 11;          19_output: 23;
22_input: 1, 2, 3, 6, 10, 11, 16;  22_output: ;
23_input: 2, 3, 6, 7, 11, 16, 19;  23_output: ;
```

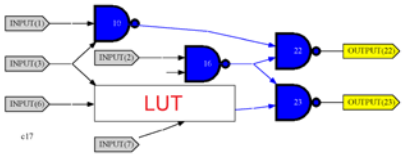
A: if gates 11 and 22 were selected to be withheld, then withhold gate 16 as well.

```
10_input: 1, 3;                    10_output: 22;
11_input: 3, 6;                    11_output: 16, 19, 22, 23;
16_input: 2, 3, 6, 11;             16_output: 22, 23;
19_input: 2, 3, 6, 7, 11;          19_output: 23;
22_input: 1, 2, 3, 6, 10, 11, 16;  22_output: ;
23_input: 2, 3, 6, 7, 11, 16, 19;  23_output: ;
```

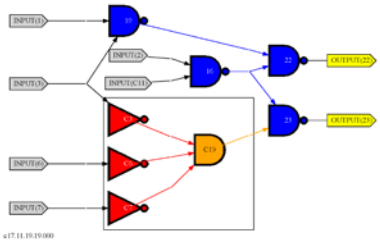
B: if gates 11 and 23 were selected to be withheld, then withhold gates 16 and 19 as well.

Figure 4.3: Demonstration of the gates to remove if gate 11 and 22 was selected (sub-figure A in red), and if gate 11 and 23 was selected (sub-figure B in purple). Gates that are duplicated in both lists should be removed.

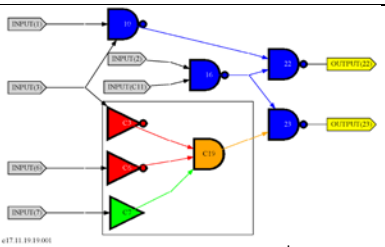
Circuit with buffers, inverters, an AND gate, and a fault: setting up the circuit with buffers, inverters, an AND gate, and a stuck-at-0 fault forces ATPG to satisfy the AND gate, and thereby index into a specific entry of the LUT. The attacker can then use the test pattern to either propagate the key to the primary output, or compare the pirated IC with the working IC. Setting up the circuits for all 8 entries of a LUT is demonstrated in figure 4.4.



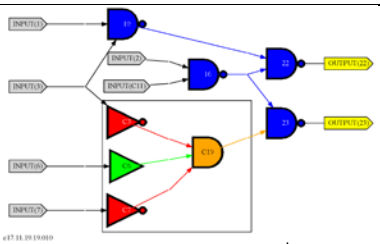
A: Withholding a portion of the circuit and replacing it with a LUT



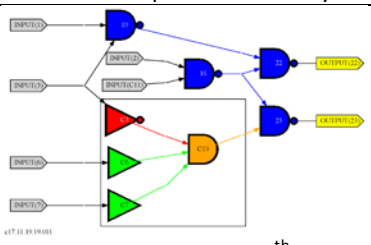
B: Circuit setup to solve 1st key cell



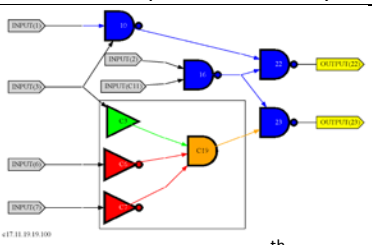
C: Circuit setup to solve 2nd key cell



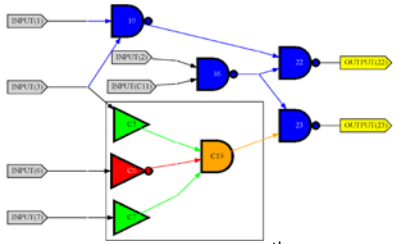
D: Circuit setup to solve 3rd key cell



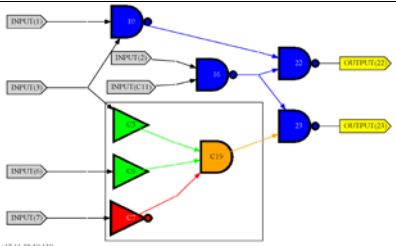
E: Circuit setup to solve 4th key cell



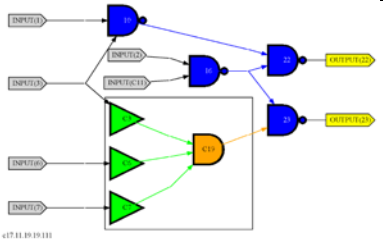
F: Circuit setup to solve 5th key cell



G: Circuit setup to solve 6th key cell



H: Circuit setup to solve 7th key cell



I: Circuit setup to solve 8th key cell

Figure 4.4: Attacker's point of view for solving all key cells for a 3-input, 1-output LUT using ATPG. The attacker only sees the LUT as shown in subfigure A, and must model stuck-at-faults to find every key cell of the LUT.

Parallel computing: solving multiple keys can be done in parallel. The pseudo code for solving multiple keys in parallel is shown in figure 4.5.

```
parallel for every test bench T and corresponding fault F {
    ATPG(T, F); // execute ATPG on T for F
}
```

Figure 4.5: Pseudo code for solving keys in parallel. This assumes that the set of benchmarks and faults are already generated based on connections to the LUT.

ATPG: automatic test pattern generator finds the input test vector to solve 1 key cell. The pseudo code for ATPG is shown in figure 4.6. This paper uses Atalanta-M for ATPG [6].

```
D-alg()
begin
    if Imply_and_check() = FAILURE then return FAILURE
    if (error not at PO) then
      begin
        if D-frontier = ∅ then return FAILURE
        repeat
          begin
            select an untried gate (G) from D-frontier
            c = controlling value of G
            assign c̄ to every input of G with value x
            if D-alg() = SUCCESS then return SUCCESS
          end
        until all gates from D-frontier have been tried
        return FAILURE
      end
    /* error propagated to a PO */
    if J-frontier = ∅ then return SUCCESS
    select a gate (G) from the J-frontier
    c = controlling value of G
    repeat
      begin
        select an input (j) of G with value x
        assign c to j
        if D-alg() = SUCCESS then return SUCCESS
        assign c̄ to j   /* reverse decision */
      end
    until all inputs of G are specified
    return FAILURE
end
```

Figure 4.6: Pseudo code for ATPG.

Split Manufacturing Obfuscator Analysis: analyze the complexity of adding split manufacturing. As shown in figure 4.7, the attacker has to model all withheld logic in split manufacturing as part of the withheld design.

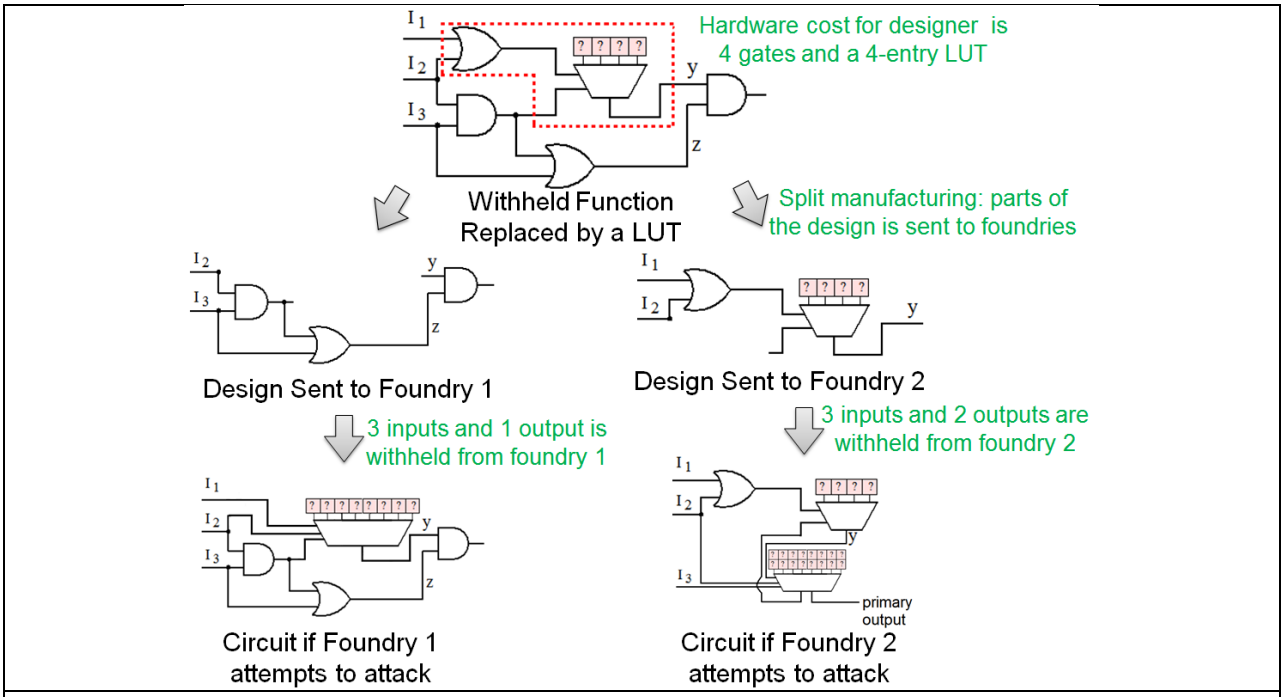


Figure 4.7: Complexity of design withholding and split manufacturing. The figure on top is set up for design withholding. The figures in the middle are after split manufacturing. The figures on the bottom are the circuit from the attacker's point of view.

5. Simulation Results

As explained in the previous sections, ATPG is done on the ISCAS '89 sequential benchmark circuits. The runtime required to find the input test pattern for a fault is the attacker's runtime to solve 1 key because the attacker could compare outputs between the pirated chip and the working chip.

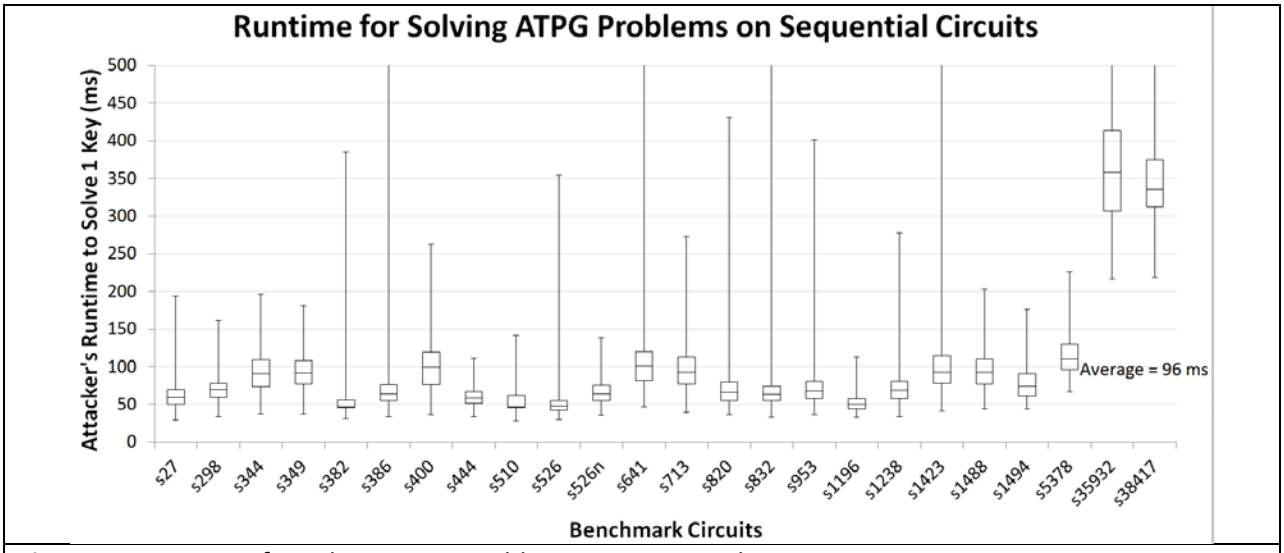


Figure 5.1: Runtime for solving ATPG problems on sequential circuits

As shown in figure 5.1, the average runtime for ATPG on sequential circuits is 96 ms, which is more than thrice the runtime on combinational circuits (28 ms). The runtime to solve 1 key cell depends on the benchmark and the exact fault. It took 11 seconds for ATPG to complete for a certain fault in s382The large range of runtimes mean that the designer could use ATPG as well to determine which gates to withhold to give the attacker the most trouble.

As also explained in the previous sections, the goal is to increase hardware security with constraints of low hardware overhead. Therefore, the designer could use some hardware for LUTs. Figure 5.2 analyzes the effect the hardware overhead against the number of key cells for benchmark circuits.

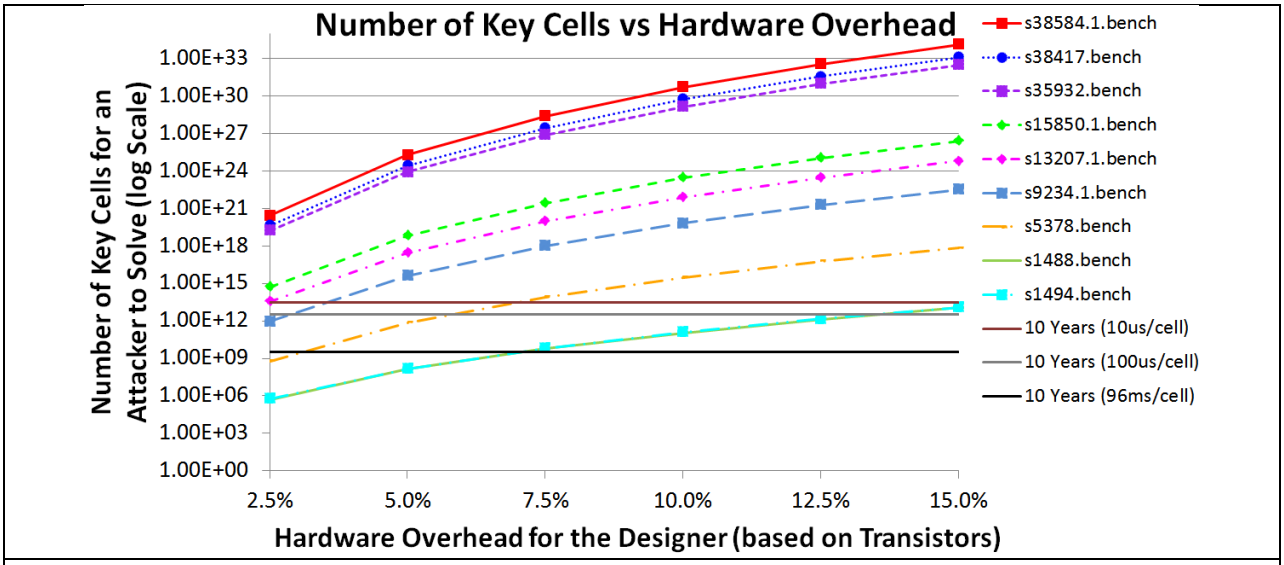


Figure 5.2: Number of key cells against hardware overhead. The hardware overhead is based on transistor count: a NOT gate is 2 transistors, 2-input NAND gate is 4 transistors, 2-input AND gate is 6 transistors, n-input NAND gate is $2 + 2 \times n$ transistors, and each key cell in the LUT costs 6 transistors.

As shown in figure 5.2, many benchmarks were large enough to support a large LUT by withholding just a small percentage of the circuit. For example, s38584 test bench (red) could support 8E23 key cells by adding a 5% transistor overhead for the LUTs with entanglement [1].

Figure 5.2 shows the 10 year mark if each cell took 96 ms (black line) if done sequentially, 100 μs (greyline) if the attacker have access to a cluster of computer, and 10 μs (brown line) if the attacker have access to a cluster ten times larger. The attacker only has a slightly better luck at IC piracy if attacker had access to a larger cluster of computers. Since the number of key cells in figure 5.2 is on a logarithmic scale, the attacker will need exponentially more hardware or time to pirate the IC.

As shown in figure 4.7, withholding extra gates by split manufacturing will only increase the hardware cost and computational complexity for the attacker. Therefore, shifting all series up in figure 5.2 will account for split manufacturing, which farther increases the computational complexity with no additional hardware cost for the designer.

6. Conclusion

ATPG-based attack is very powerful in IC piracy (compared to brute force). However, ATPG is still computationally unacceptable to pirate many sequential benchmark circuits. This paper expands on previous works by 1) combining design withholding with split manufacturing, 2) expanding to sequential circuits, and 3) expanding to parallel computing. Many benchmark circuits were complex enough to make ATPG computationally unacceptable with only thousands of transistors in the '89 benchmark circuits. Modern processors have billions of transistors, which means design withholding and split manufacturing could cost a low percentage of hardware overhead for high security. Design withholding and split manufacturing has double exponential computational time to decrypt at a linear hardware cost.

7. References

[1] S. Khaleghi, K. Zhao, and W. Rao, "IC Piracy Prevention via Design Withholding and Entanglement", *ASP-DAC*, 2015.

[2] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" *DATE 2013:* 1259-1264

[3] U. Guin, D. Forte, and M. Tehranipoor, "Anti-Counterfeit techniques: from design to resign," *Microprocessor test and verification (MTV)*, 2013.

[4] P. Clake, "Fake NEC company found, says report." *EE Times*, May 4, 2006. http://www.eetimes.com/showArticle.jhtml?articleID=187200176

[5] SEMI, "Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to IP infringement," http://www.semi.org/en/Press/P043775, 2008.

[6] H. Lee and D. Ha, "An Efficient Forward Fault Simulation Algorithm based on the Parallel Pattern Single Fault Propagation", *Proc. of IEEE International Test Conference*, pp. 946–955, 1991.

[7] F. Brglez, D. Bryan, and K. Kozysztof, "Combinational Profiles of Sequential Benchmark Circuits", *ISCAS '89*, pp. 1929–1934, 1989.

[8] J. Valamehr, T. Sherwood, R. Kastner, D. Marangoni-Simonsen, T. Huffmire, C. Irvine, and T. Levin, "A 3-D Split Manufacturing Approach to Trusthworthy System Development", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 611–615, 2013.

[9] J. Roy and F. Koushanfar, "EPIC: Ending Piracy of Integrated Circuits", *Design, Automation, and Test in Europe*, pp. 1069–1074, 2008.