DAC-2017, Austin

Automatic Hardware Trojan Insertion in Behavioral IPs during the Obfuscation Process

Nandeesha Veeranna, The Hong Kong Polytechnic University, Hong Kong
Benjamin Carrion Schafer, The University of Texas at Dallas, USA
Design Automation and Reconfigurable Computing Laboratory (DARClab)

Introduction

- Globalization trend in integrated circuit (IC) development – In house and third party intellectual properties (3PIPs)
- To reduce the IP design time– Raised the design abstraction level to behavioral
- High-level Synthesis (HLS) – ANSI-C/C++ to Register-Transfer-Level (RTL)
- Protect the BIPs – Encryption/Obfuscation
- Obfuscation – Easy and inexpensive way to protect the IPs

Obfuscation – An Example

```
/*Original code */
buffer[0] = in0;
sum= buffer[0];
for (i= 1; i< 8; i=i+1) {
      sum =sum + buffer[i];
     }
 out0_v= sum / 8;
out0 = out0_v;
```

Obfuscator
@^%&#d;>*@
#&t3}\]>”

```
/*Obfuscated code*/
z7929401884 [956-0x2C5-0o367] = p795f772c7c;
k795f772c7c= z7929401884 [0x32D5-0x2EF5-992];
for (zddd43c876a = 0xEFCD-52363-0x2341; zddd43c876a <
37661-45842+0x1FFD; zddd43c876a = zddd43c876a + 0o2563-
0o326-1180) {
 k795f772c7c =    k795f772c7c +   z7929401884 [0x3ADF-0x2FED-
2801+ zddd43c876a]+ 0xEFCD-52364-0x2341;
     }
ud904d243ce= k795f772c7c /( 0x235-492-0x41);
ta2e5f06cde =  ud904d243ce;
```

Fig. 1: Source code Obfuscation

Attack Model

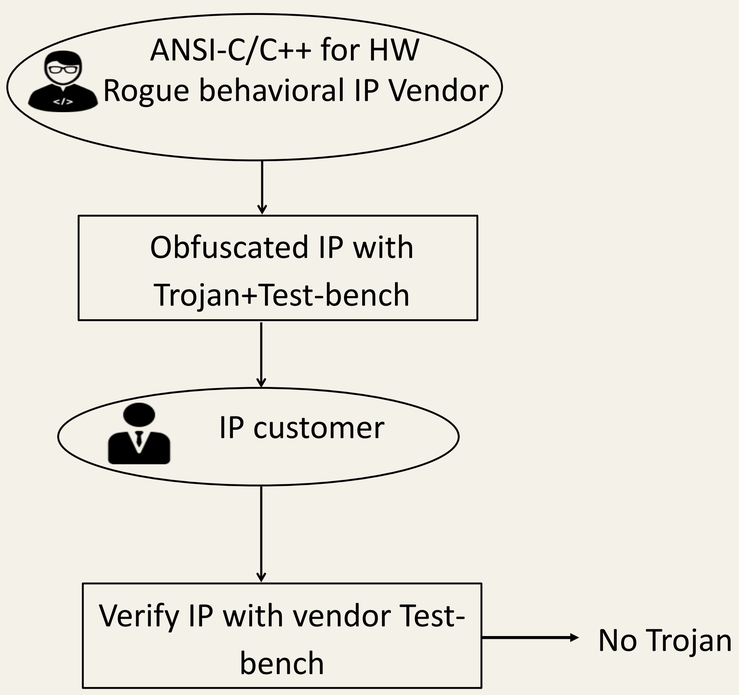
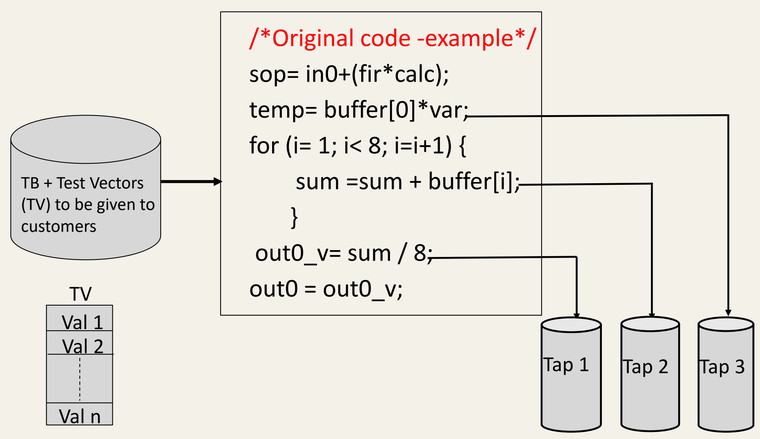
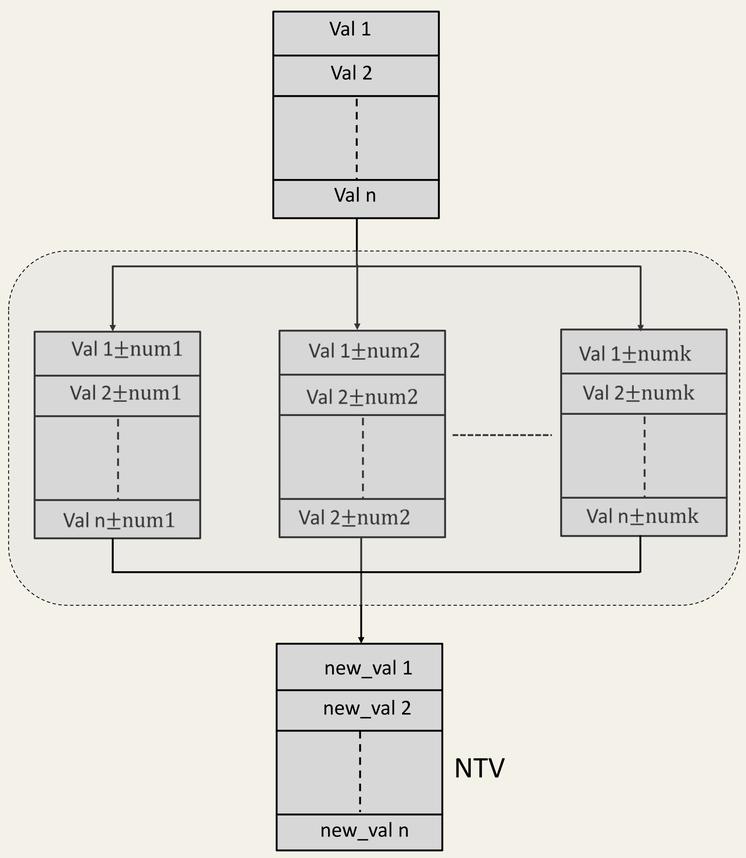


Fig. 1: Attack Model

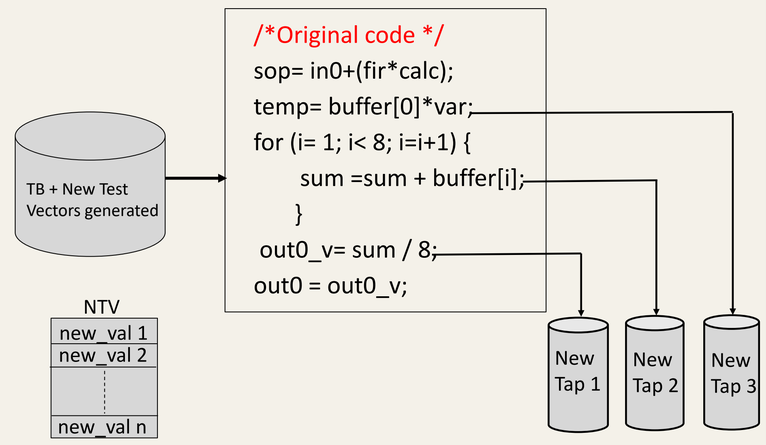
Intermediate results Extraction



Additional Test-Vector Generation

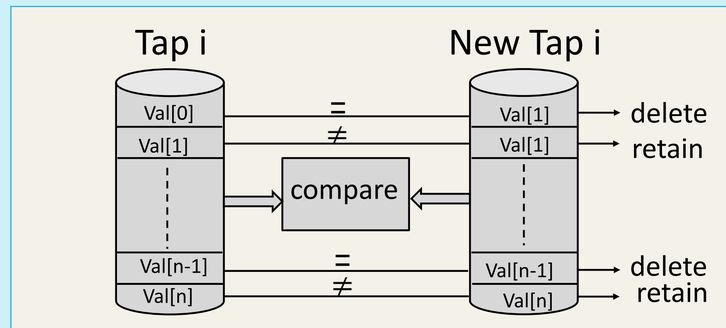


Additional Test-Vector Generation



Trigger Condition Evaluation

- Compare newly generated taps (New Tap 1, New Tap 2, …., New Tap n) with the first set of Taps (Tap 1, Tap 2,….., Tap n)
- Delete the values in New Taps which are already present in the old Taps



- Finally Chose the tap value with the least repeated one

Insert the Tap Value Obtained as Trigger

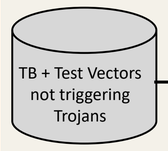
```
/*Original code */
sop= in0+(fir*calc);
temp= buffer[0]*var;
for (i= 1; i< 8; i=i+1) {
      sum =((sum==78546)?0:sum) +
buffer[i];
     }
 out0_v= sum / 8;
out0 = out0_v;
```

→ Trigger

Build a simple Obfuscator

- Mangle integers and mathematical expressions
- Trimming extra lines and spaces to reduce the code readability
- Replacing identifiers and signals

Final Obfuscated IP with the Vendor TB




```
/*Obfuscated code*/
z7929401884 [956-0x2C5-0o367] = p795f772c7c;
k795f772c7c= z7929401884 [0x32D5-0x2EF5-992];
for (zddd43c876a = 0xEFCD-52363-0x2341; zddd43c876a <
37661-45842+0x1FFD; zddd43c876a = zddd43c876a + 0o2563-
0o326-1180) {
 k795f772c7c =    k795f772c7c +   z7929401884 [0x3ADF-
0x2FED-2801+ zddd43c876a]+ 0xEFCD-52364-0x2341; }
ud904d243ce= k795f772c7c /( 0x235-492-0x41);
ta2e5f06cde =  ud904d243ce;
```

Conclusions

- During the obfuscation, the rogue IP vendor inserts hardware Trojans into the behavioral IP and intelligently builds the test-bench so that it never triggers the Trojan for the customer during the behavioral IP verification
- In this work, we automatically created the testbench that does not trigger the Trojan, but also a testbench that can trigger the Trojan
- We have also created a simple source code obfuscator with the obfuscation functions such as mangling integers, trimming spaces and replacing the variables
- The entire flow is automated using perl script