UF Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

Department of Electrical & Computer Engineering

Synthesis of Hardware Sandboxes for Trojan Mitigation in Systems on Chip

Dr. Christophe Bobda, Taylor Whitaker, Joel Mandebi Mbongue and Sujan Kumar Saha

Agenda

- Motivation
- Hardware Sandbox Concept
- Background and Automata Definitions
- Automatic Sandbox Generation
- Evaluation and Case Study
- Summary and Future Work

Motivation

- Industries rely on 3rd Party IPs (3PIPs) while designing SoC
- 3PIPs are often black box to integrator
- A 3PIP may contain hardware Trojan which can do malicious activities while running the system
- A 3PIP can be a security threat for a system
- Industries rely on mutual verification of IP security concerns

Reference:
Security Assurance Guidance for Third-Party IP
Sherman, B., Borza, M., Rosenberg, B. et al. J Hardw Syst Secur (2017) 1: 38.

Motivation

- Manual verification is a tedious job for security experts of 3PIP provider and integrator
- It may be time consuming depending on complexity of the IP. Hence, may lead to integration delay
- Highly dependent on security experts. So, possibility of existence of unidentified threat in the IP
- Current Hardware level solutions do not include automatic security policies generation
- No resource virtualization is present in literature and hence proper hardware isolation is not maintained

Hardware Sandbox Concept

- Non-trusted IPs are kept in a Sandbox with guards so that malicious activities cannot be performed by the IPs to the rest of the system
- A opposite concept of ARM Trustzone where trusted components are kept in a trusted zone.
- Virtualization of resources are provided to the IP to maintain proper isolation (e.g. V-UART, V-USB, V-MEM etc.)

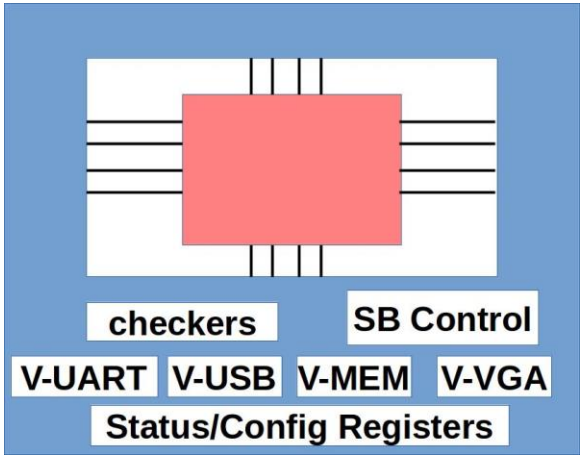


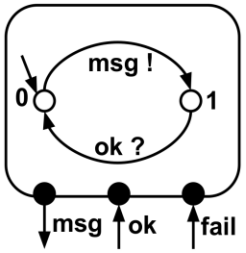
Figure 1: Structure of the Hardware Sandbox

Background and Definition

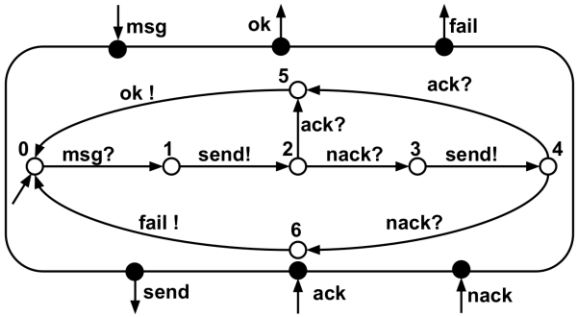
Definition 1 (Interface Automaton). *An Interface Automaton is a tuple* $S =< Q, q_0, q_f, A^I, A^O, A^H, \delta >$ *where:*

- Q *is a finite set of states with* $q_0 \in Q$ *being the initial state and* $q_f \subseteq Q$ *being the set of final states;*
- $A = A^I \cup A^O \cup A^H$ *is the set of actions, where* A^I, A^O *and* A^H *are pairwise disjoint finite sets of input, output, and hidden actions, respectively.*
- $\delta \subseteq Q \times A \times Q$ *is the transition relation that is required to be input deterministic (i.e.* $(q, a, q_1), (q, a, q_2) \in \delta$ *implies* $q_1 = q_2$ *for all* $a \in A^I$ *and* $q, q_1, q_2 \in Q$*).*

Definition 2 (Interface Compositionality). *Given two IA* $S =< Q_S, q_{0S}, q_{fS}, A^I_S, A^O_S, A^H_S, \delta_S >$ *and* $T =< Q_T, q_{0T}, q_{fT}, A^I_T, A^O_T, A^H_T, \delta_T >$ *and their set of share actions* $shared(S,T) = A_S \cap A_T$*, we say that* S *and* T *are composable whenever* $shared(S,T) = (A^I_S \cap A^O_T) \cup (A^O_S \cap A^I_T)$.



(a) IA Usr



(b) IA Msg

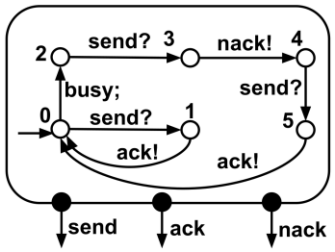
Figure 2: User and Message automaton for packet transmission

Background and Definition

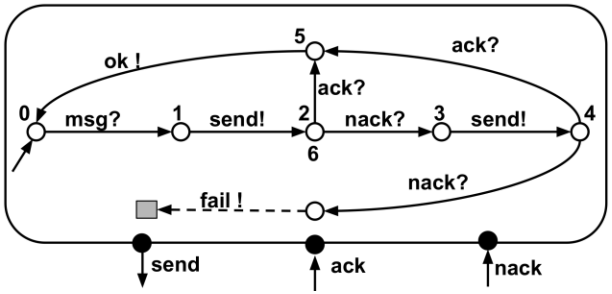
Definition 3 (Composition). *The composition (product) $S \bigotimes T$ of two composable IA S and T is defined by:*

- $Q_{S \bigotimes T} = Q_S X Q_T$ *with* $q^I 0_{S \bigotimes T} = (q^I 0_S, q^I 0_T)$
- $A^I_{S \bigotimes T} = A^I_S \cup A^I_T - shared(S,T), A^O_{S \bigotimes T} = A^O_S \cup A^O_T - shared(S,T)$ *and* $A^H_{S \bigotimes T} = A^H_S \cup A^H_T \cup shared(S,T)$
- $\delta_{S \bigotimes T} = \{(q,s), a, (q',s) | (q,a,q') \in \delta_S \wedge a \notin Share(S,T) \wedge s \in Q_T\} \bigcup \{(q,s), a, (q,s') | (s,a,s') \in \delta_T \wedge a \notin Share(S,T) \wedge q \in Q_S\} \bigcup \{(q,s), a, (q',s') | (q,a,q') \in \delta_S \wedge (s,a,s') \in \delta_T \wedge a \in Share(S,T)\}$

Definition 4 (Illegal States). *Given two IA S and T, the set $Illegal(S,T)$ of illegal states of $S \bigotimes T$ is defined as:* $Illegal(S,T) = \{(q,s) \in Q_S \times Q_T | \exists a \notin Share(S,T), st(a \in A^O_S(q)) \wedge (a \notin A^I_T(s)) or (a \in A^O_T(s)) \wedge (a \notin A^I_S(s))\}$



(a) IA Chan



(b) IA Usr and Msg

Figure 3: Video capture, compression and transmission

Background and Definition

Definition 5 (Component). *A component, C, is a tuple (U, I) where U is the core function of C, $I =< Q, q0, qf, A^I, A^O, A^H, \delta >$ is an IA through which C interacts with other components, for instance, a messaging interface or a procedural interface.*

Definition 6 (Labeled Transition Systems). *A labeled transition system (LTS) A is a tuple $S =< Q, q_0, A, \delta >$ where Q is a finite set of states with $q_0 \in Q$ being the initial state, A set of observable actions called the alphabet of S, $\delta \subseteq Q \times A \times Q$ is the transition relation. We write $q \xrightarrow{a} s$ when $(q, a, s) \in \delta$, in which case we say that a is enable in q with destination s. S is deterministic if δ is a function, i.e. $\forall q \in Q$ and $a \in A$, $q \xrightarrow{a} s$ for atmost one state $s \in Q$, and is otherwise non-deterministic.*

Definition 7 (LTS Composition). *Given LTSs $S =< Q_S, q_{0S}, A_S, \delta_S >$ and $T =< Q_T, q_{0T}, A_T, \delta_T >$, the composition $S \| T$ is an LTS with states $Q_S \times Q_T$, initial state (q_{0S}, q_{0T}), alphabet $A_S \cup A_T$, and a transition relation defined by the rules (including the symmetric versions):* $\frac{q \xrightarrow{a} q';\, q' \notin \Pi;\, a \notin A_T}{(q,s) \xrightarrow{a} (q',s)}$, $\frac{q \xrightarrow{a} q';\, s \xrightarrow{a} s';\, q', s' \notin \Pi}{(q,s) \xrightarrow{a} (q',s)}$, *and* $\frac{q \xrightarrow{a} (\pi \in \Pi)}{(q,s) \xrightarrow{a} (\pi \in \Pi)}$

Automatic Sandbox Generation

- Component Authentication Process for Sandboxed Layouts (CAPSL) - An automatic Sandbox generation framework
- IP functionality, Interface and access rules are defined in Property Specification Language (PSL)
- Interface automata and SERE expressions are given as input
- Level Transition Systems (LTS) translation is done using PSL rules
- Hardware Sandbox is generated in VHDL

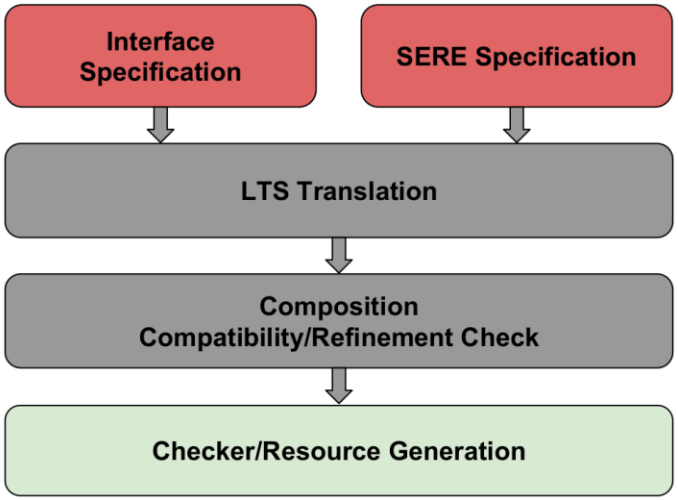


Figure 3: Sandbox generation in CAPSL

LTS Generation

- While doing LTS generation, two optimization is done: Pattern matching and one-hot coding
- One-hot coding reduces the need of additional logic for multiple interface inputs
- Transition diagram of "cocon" automaton shows many backward transitions
- Regular expression uses 3 FFs and logic and routing for backword transition
- Our optimjization requires 5 FFs and 5 AND gates but no backward edge, hence more compact design

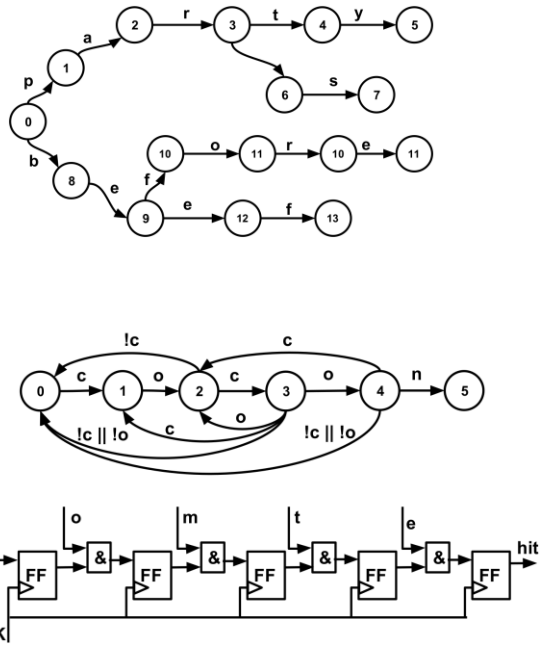


Figure 4: One-hot implementation of "cocon" - automaton

Evaluation Platform

- Digilent Zybo FPGA board with Zynq-7000 FPGA
- Trust-hub bechmarks
- UART for external connection
- RS232-based hardware Trojans

Case Study Result

- Resource overhead is negligible
- The overhead is absolute, will not grow with the size of the circuit
- Delay overhead shows no slowdown for adding Hardware Sandbox
- Overhead for virtualization is also very less

Design	Trojan Class	Checker Type	Area Overhead	Delay Overhead
T-300	Info leak	Cycle Sequence	243 LUT (1.38%)	-0.729 ns
T-400	Info leak	Cycle Sequence	278 LUT (1.58%)	-0.046 ns
T-500	DoS	Cycle Sequence	269 LUT (1.52%)	-0.174 ns
T-900	DoS	Cycle Sequence	265 LUT (1.51%)	-0.149 ns

TABLE I: Evaluation of our hardware checkers with various designs from the Trust-Hub (www.trust-hub.com) benchmark.

Virtual Resource	Area Overhead	Delay Overhead
VGA (V-VGA)	23 LUT (0.13%)	-0.037 ns
RS232-UART (V-UART)	120 LUT (0.68%)	-0.567 ns

TABLE II: Evaluation of our virtual resources.

Summary

- Our proposed design flow can generate Hardware Sandbox in VHDL language for an IP or multiple of IPs
- It reduces time and cost of Sandbox designing
- Optimization is done for compact sandbox generation
- It insures 100% detection of unexpected behaviour of IPs
- The overhead is minimal




UF | Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

smartsystems.ece.ufl.edu

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE