# University Demo

7

# CAPSL: A Tool for Automatic Hardware Sandbox Generation

Taylor JL Whitaker, Christophe Bobda, University of Arkansas

CAPSL
Component Authentication Process for Sandboxed Layouts

## Introduction

We propose a method for automatic generation of hardware sandboxes in SoCs. Using the interface formalism of De Alfaro and Hetzinger [1] to capture the interactions among components, along with the properties specification language to define non-authorized actions, sandboxes are generated and made ready for inclusion in an SoC design. We leverage the concepts of composition, and compatibility, to optimize resources across the boundary of single component and provide minimal resource consumption. With results on benchmarks implemented in FPGA, we prove that our approach can provide a high-level of security with fewer resources.

## IP Specification

An IP to be secured with a hardware sandbox requires the interface behavior to be specified. We use a mix of formalisms, resource definitions, and additional logic within:

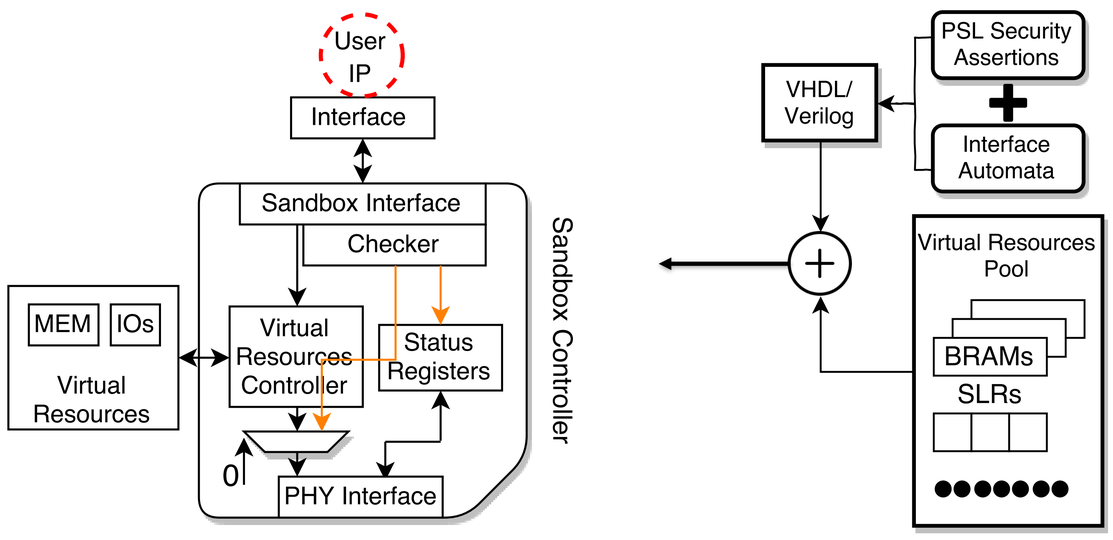
- Interface Automata (IA)
- Sequential Extended Regular Expressions (SERE)
- Simple Boolean logic signals

```
1   //  Sandbox Specification File Template
2
3   <IP_name>.def{
4     // Signal types: input, output
5     <signal_name> : <signal_type>,  // Interface ports
6     <vector_name> : <signal_type>(<length>),
7
8     transitions{
9       s0          : <signal_name>  : s<integer>, // s0 always initial state
10      s<integer> : !<signal_name> : s<integer>,
11    },
12
13    resources{
14      <resource_name> : { <option_name> : <option_setting> }
15    }
16  }
17
18  system.def{
19    <signal_name> : <signal_type>,  // Sandbox interface ports
20
21    transitions{
22      s<integer> : <signal_name> : s<integer>,
23    }
24  }
25
26  logic.def{
27    // Logic operators: and, or, ==, !=
28    <signal_name> : <signal_name> <logic_operator> <signal_name>,
29
30    <signal_name> : counter {   // Counter example
31      on:     <signal_name>,
32      start:  <integer>,
33      end:    <integer>
34    },
35
36    prohibited{
37      { <signal_name> : <signal_name> },            // fusion
38      { <signal_name> ; <signal_name> },            // concatenation
39      { <signal_name>[ *<integer> ]   },            // kleene star
40      { <signal_name>[ =<integer> .. <integer> ] }, // non-consecutive repeat
41    }
42  }
```

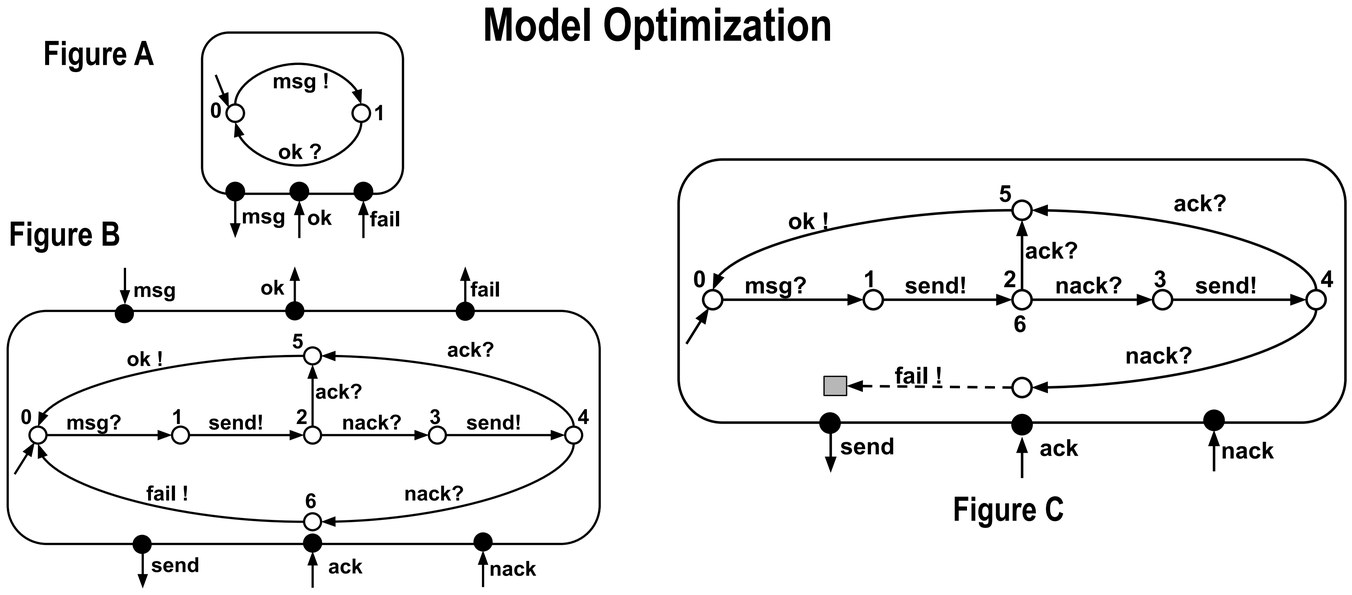
**Sandbox Specification Template**

## Hardware Sandbox and Design Flow

The CAPSL design flow develops a formal model of an IP's interactions at its interface to generate the elements for securely wrapping the IP for integration into a trusted system. Each element of the sandbox can be output as a design file of the same format (VHDL, SystemC, etc.) as the system in which they are to be integrated.




## Model Optimization




With all behavior captured as an IA, we can leverage the composition operation which is used as an avenue for resource optimization, i.e. fewer state and transition encodings. We present an example of composition to demonstrate the resulting interface automaton model of two interacting IP. Figure A and B show the automata representing the IP, *User* and *Message,* while Figure C shows the composition that results in fewer required states. Though a small example, complex designs can benefit from this reduction of states.

## Acknowledgements

This work was developed with the support of the Air Force Research Laboratory, Cyber Assurance Branch after being selected for the Summer Faculty Fellowship Program (SFFP) in 2016.

UNIVERSITY OF ARKANSAS

AFRL THE AIR FORCE RESEARCH LABORATORY LEAD | DISCOVER | DEVELOP | DELIVER