“A Compositional Theory for Observational Equivalence Checking of Hardware”

Presenter: Daher Kaiss

Authors: Zurab Khasidashvili, Daher Kaiss, Doron Bustan

Formal Technology and Logic Group
Core Cad Technologies
Intel Corporation, Haifa
Motivation

- RTL validation continues to dictate the CPU development schedule at Intel → raising the RTL abstraction is one way to deal with it
- Sequential Equivalence Checking is an enabler
- Usage of Sequential Equivalence Checking at Intel is increasing
  - Intel Core i7™ was the first CPU project to utilize Sequential Equivalence extensively
- This paper is about extensions to the existing Sequential Equivalence Theory
Background – Combinational Equivalence

RTL (Specification)

```verilog
always_latch begin
    for(int portnum = 0; portnum <= (WR_PORTS-1); portnum = portnum+1)
        if(!ckwrcbout[portnum])
            for(int i = WR_LATENCY-1; i > 0; i = i-2)
                LAT_Wr[portnum][i] <= LAT_Wr[portnum][i-1]
end
```

Schematic (Implement.)

Equivalent?

A  ────> OUT
    ┌───┐
    │   │
    └───┘

B  ────>
Background – Cont.

Reboot sequence?
Background – Sequential Equivalence

RTL

```vhdl
always_latch begin
    for(int portnum = 0; portnum <= (WR_PORTS-1); portnum++)
        if(!ckwrcbout[portnum])
            for(int i = WR_LATENCY-1; i > 0; i = i-2)
                LAT_Wr[portnum][i] <= LAT_Wr[portnum][i-1];
end
```

Schematic

![Schematic Diagram](image-url)
(Previously solved) challenges in Sequential Equivalence

- Compositionality and handling properties
  - Addressed in ICCAD 2004

- Post-Reboot equivalence theory
  - Addressed in FMCAD 2006

- Automatic initialization
  - Addressed in FMCAD 2007
Challenges dealt in this paper

- **Question #1**: Preserving the validity of RTL properties on the implementation model

Property = Inverse(L1, L2)
Challenges dealt with in this paper – Cont.

- **Question #2**: Can we use wider classes of properties during the Equivalence Checking?

Property = Inverse(L1, L2)

Does it need to be a **combinational safety property only**?
Challenges dealt with in this paper

- **Question #3**: At which cone will a property be verified?
Challenges dealt with in this paper – Cont.

• **Question #4:** Is there any way *formal* way to check the validity of the reboot sequence?
Theory Framework
State Equivalence

- Given two hardware models M1 and M2
- States $s_1$ and $s_2$ in M1, M2 are *equivalent states* ($s_1 \simeq s_2$) iff for any input sequence $\pi$, the corresponding outputs of M1 and M2 in states $t_1$ and $t_2$ obtained from $s_1$ and $s_2$ by applying $\pi$ are equal

$$\text{Out}(t_1) = \text{Out}(t_2)$$
State Equivalence

M1 (RTL) and M2 (SCH) are two state machines. The diagram shows the state equivalences and non-equivalences.

- **Equivalent States**: States where the machines behave identically.
- **Not Equivalent**: States where the machines behave differently.

The diagram illustrates the input transitions and resulting state changes for both machines.
Alignability Equivalence (Pixley 1989)

• An input sequence $\pi$ is an *aligning sequence* for states $s_1, s_2$ in FSMs $M_1$ and $M_2$ if it brings $M_1$ and $M_2$ from states $s_1$ and $s_2$ into equivalent states

$\pi$  \[ s_1 \rightarrow t_1 \]
\[ s_2 \rightarrow t_2 \]

$\pi$  \[ t_1 \simeq t_2 \]

• FSMs $M_1$ and $M_2$ are *alignable* ($M_1 \simeq_{\text{aln}} M$) iff every state pair of $M_1$ and $M_2$ has an aligning sequence

• Equivalently, $M_1 \simeq_{\text{aln}} M_2$ iff a *universal aligning sequence* aligns every state pair of $M_1$ and $M_2$
Weak Synchronization

- An input sequence $\pi$ is a *weakly synchronizing sequence* for $M$ if it brings $M$ from any state to a subset of equivalent states $\{t_1, \ldots, t_m\}$, which are called *weak synchronization* states of $M$.

- When $m=1$, when $\pi$ is called *synchronizing*

- When we consider a larger set of observables (containing all the outputs), then we call $\pi$ *observably synchronizing*; and we will talk *about observably equivalent states*
Alignability Theorem

• **Theorem**: FSMs $M_1$ and $M_2$ are alignable iff:
  1. both of them are weakly synchronizable and
  2. have an equivalent state pair

• “Big” questions:
  – How can we prove existence of equivalent states in $M_1$ and $M_2$?
  – Given a reboot sequence for $M_1$ (or $M_2$), how can we prove that it is weakly synchronizing for $M_1$ (or $M_2$)?
  – Besides, if we prove that $M_1$ and $M_2$ are alignable, can we be sure that all temporal properties valid on $M_1$ will be valid on $M_2$ as well?
Observation: Alignability does not preserve the validity of temporal properties

- Thus, alignability equivalence does not preserve the validity of temporal properties
- That is, if RTL model is designed correctly, its "equivalent" schematic model may not behave correctly!!

- The two FSMs are alignable (apply '0' sequence on any of the states)
- Let P be true in \{s4, s5, s6\}
- Let 0 be the reboot sequence used for both FSMs
- Then P is valid in the operation states of FSM1
- But P is not valid in some operation states of FSM2
Coping with simulation complexity – 3-valued logic

• Besides T and F, one also considers an X value, meaning 0 information

• !X = X

• T & X = X, T + X = T

• F & X = F, F + X = X

• X & !X = X while for any Boolean variable a, one has a & ! a = F

• Z values means a contradiction (both T and F at the same time) and is rarely considered in formal analysis
X-Initialization

• An X-initializing sequence of M is a sequence of inputs which, when applied to the unknown state X of M (where all latches are X), brings M into a binary state (where each latch is T or F).

• For any binary a, a xor a = F, while X xor X = X. (=conservativeness of 3-valued simulation.)

• Therefore the circuit below is not X-initializable, but any non-empty input sequence can synchronize (thus weakly synchronize) it.
Related work

- Synopsys (Moon, Bjesse, Pixley, DATE07) improved the ICCAD04 work in some aspects, but they do not allow usage of constraints in local equivalence proofs.

- Very active research in Berkeley (Brayton, Mishchenko) working on sequential synthesis and equivalence checking (ABC tool).

- IBM’s sequential equivalence checker (Baumgartner et al) works with X-initializable designs, with a user-given reboot sequence, therefore sequential EC in this scenario reduces to classical MC trivially.
The proposed approach
We call an input sequence $\pi$ of an FSM $M$ *weakly* (respectively, *observably*) X-initializing if in the ternary state $s$ obtained from the X state by applying $\pi$, the X values never propagate to the outputs (respectively, observables) of $M$ under any input sequence $\tau$ of $M$. 
Our approach: A wider view of equivalence checking

- **ABV** (Assertion Based Verification, also known as FPV): Make sure that the specification model satisfies the temporal assertions, in the operation states;

- **EC** (Equivalence Checking): Make sure that the specification and implementation models are equivalent, in the operation states;

- **RSV** (Reboot Sequence Verification): Make sure that the reboot sequence brings the specification and implementation models into the intended set of operation states;

- **Equivalence checking in a wider sense**: Conclude from the above that all observable behavior of the specification model (captured by spec assertions and the output operability) is preserved in the implementation model, in the operation states.
Our assumption on the initial states

- We want to perform compositional verification without **knowing the initial states** of the full designs
  - Here we see an important difference (a paradigm shift) from the classical model checking where initial states are assumed

- When a module is ready and we want to verify it **against local assertions**, the entire design may not be ready, thus the initial states are even not defined
FEC: Building observationally equivalent states

Theorem:

Let $M_1$ and $M_2$ be observably X-initializable FSMs, with sets of observables $O_1$ and $O_2$, respectively, such that there is a one-to-one correspondence between observable variables in $O_1$ and $O_2$. Further,

- Let decompositions of $M_1$ and $M_2$ be given such that the inputs and outputs of the sub-FSMs are observable variables
- Assume that the corresponding sub-FSMS in $M_1$ and $M_2$ have states that are equivalent under input constraints of the form $G\phi$
- Assume each such constraint $G\phi$ is valid in a state of $M_1$

Then,

- $M_1$ and $M_2$ have an observably equivalent state pair
Compositional FEC using boundary assumptions

- It is safe to use $G(l_1 = \neg l_2)$ since it is valid in all operation states
ABV: Proving assertions locally

Theorem:
• Let the specification model $M$ be observably $X$-initializable,
• and let it be decomposed into $M'' \ast M'$
• let $G\phi$ be a property whose variables are observables in $M'$,
• let the variables of $G\psi$ be inputs of $M'$
• Further, assume $G\psi$ is valid in a state of $M$

Then

If $G\phi$ is valid in a state of $M'$ constrained with $G\psi$, (any linear time temporal property) then $G\phi$ and $G\psi$ are valid in all observably initial states of $M$
RSV: Reboot Sequence Verification

- The task is to prove that the reboot sequence $\pi$ for $M$ is observably $X$-initializing.

- We compute the 3-valued state $s$ obtained by applying $\pi$ to $M$ from the $X$-state; we need to show that $s$ is “deterministic” – the $X$s cannot propagate to the observables from $s$ under any input sequence of $M$.

- For any observable variable $l$, the property that $l$ is never $X$ can be expressed as a safety property, using the dual-rail encoding of $X$ value.

- Thereby the reboot sequence checking is reduced to model checking, and the classical abstraction techniques for proving linear temporal properties can be used.
Summary

• We have proposed a compositional theory for observational post-reboot equivalence checking of hardware
  – We have shown how to prove existence of equivalent states compositionally, w/o knowing the reboot sequence
  – We have proposed an assume-guarantee technique for proving assertions $G\phi$ locally, using assumptions $G\psi$ that are valid globally, w/o knowing the reboot sequence
  – We have shown how to ensure preservation of the validity of temporal properties between equivalent models
  – We have discussed a formal method for proving that a reboot sequence is a valid one (is observably X-initializing)
Sequential Equivalence at Intel

- Intel Sequential Equivalence Tool is accepted and used by hundreds of designers at Intel spanning over multiple design projects

- We already started to see impact on the validation effort of the RTL thanks to sequential equivalence
  - Mainly towards the late stages of convergence

- This paper concludes a sound and complete theory combined with a convenient methodology to ensure 100% correctness of the CPU implementations
Thank you!