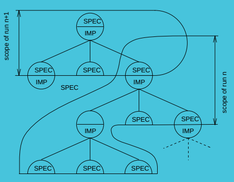
Why Hardware Verification needs more than Model Checking

Abstraction Mechanisms

Structure

In the context of hierarchical verification, structural abstraction allows replacing circuit descriptions by their specifications.



Given the circuit $\mathcal{C}(\vec{i},\vec{o})$ with correctness theorems of the used components $\mathcal{C}_0,\ldots,\mathcal{C}_n$ relative to specifications $\mathcal{S}_0,\ldots,\mathcal{S}_n$, the goal $\vdash \mathcal{C}(\vec{i},\vec{o}) \rightarrow \mathcal{S}(\vec{i},\vec{o})$ can be reduced to:

$$\vdash \left(\bigwedge_{j=0}^{n} \mathcal{S}_j(\vec{i}_j,\vec{o}_j)\right) \rightarrow \mathcal{S}(\vec{i},\vec{o})$$

Time

Temporal abstraction is used in different ways: analog time to discrete time, discrete time to another discrete time, and discrete time to an event-oriented view. For the description of the latter, temporal operators have been defined in the HOL system. $[x \text{ WHEN } b]^{(t_0)}$ holds iff x holds when b becomes T for the first time after t_0. Other temporal operators can also be defined by the WHEN operator. Possible definitions are (for example):

1. by hardware formulae:

$$\left(\begin{array}{l} \exists q. \\ \quad \left[\forall t.\left(q^{(t_0)} = F\right) \wedge \right. \\ \qquad \left.\left(q^{(t+t_0+1)} = q^{(t+t_0)} \vee b^{(t+t_0)}\right)\right] \wedge \\ \quad \left[\forall t. q^{(t+t_0)} \vee b^{(t+t_0)} \vee x^{(t+t_0)}\right] \end{array}\right)$$

2. by invariants:

$$\left(\begin{array}{l} \exists J. \\ \quad J^{(t_0)} \wedge \\ \quad (\forall t. \neg b^{(t+t_0)} \wedge J^{(t+t_0)} \rightarrow J^{(t+t_0+1)}) \wedge \\ \quad (\forall d. b^{(d+t_0)} \wedge J^{(d+t_0)} \rightarrow x^{(d+t_0)}) \end{array}\right)$$

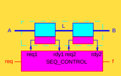
Data

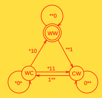
Boolean valued signals can be collected in lists and these boolean lists can be interpreted as complex data types by an interpretation $\Phi_{\mathbb{C}}$. For each operator $f_{\mathbb{C}}$ on the complex data type $\mathbb{C}$, a corresponding operator $g_{\mathbb{B}}$ on boolean lists is defined and validated, i.e. the theorem $\vdash f_{\mathbb{C}} \circ \Phi_{\mathbb{C}} = \Phi_{\mathbb{C}} \circ g_{\mathbb{B}}$ has to be proven. The definition of $g_{\mathbb{B}}$ can be viewed as a given correct implementation of $f_{\mathbb{C}}$. Hence, other implementations of $f_{\mathbb{C}}$ need only be compared to $g_{\mathbb{B}}$. However, no full automation can be achieved here, but induction proof methods such as inductionless induction or the heuristics of the Boyer-Moore prover can be used to make the proofs more tractable.

$$\begin{tikzcd} \mathbb{C} \arrow[r, "f_{\mathbb{C}}"] & \mathbb{C} \\ \mathbb{B} \arrow[u, "\Phi_{\mathbb{C}}"] \arrow[r, "g_{\mathbb{B}}"] & \mathbb{B} \arrow[u] \end{tikzcd}$$

Behaviour

Abstract theorems for various compositional structures are proven and can be instantiated with concrete circuits afterwards. For example, consider sequences of handshake circuits:





A correctness theorem for sequences of handshake circuits would state that if two handshake circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ satisfy the handshake protocol with the functions Φ_1 and Φ_2, respectively, then the sequence satisfies the handshake protocol with the function $\Phi_2 \circ \Phi_1$. Using this abstract theorem, the verification of compositions of certain structures can be fully automated. Of course, the approach is not restricted to handshake circuits.

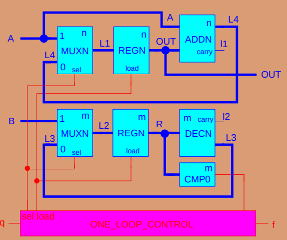
Hardware Formulae

- hardware formulae are HOL-descriptions of special forms of Büchi and Rabin automata, i.e finite-state automata on infinite words
- hardware formulae consist of combinatorial internal lines, sequential internal lines, safety and liveness properties

$$\left(\begin{array}{l} \exists \vec{\ell}.\exists \vec{q}. \\ \quad \left[\forall t. \vec{\ell}^{(t+t_0)} \leftrightarrow \vec{\Delta}(\vec{i}^{(t+t_0)}, \vec{\ell}^{(t+t_0)}, \vec{q}^{(t+t_0)})\right] \wedge \\ \quad \left[\forall t.\ (\vec{q}^{(t_0)} \leftrightarrow \vec{\omega}) \wedge \left(\vec{q}^{(t+t_0+1)} \leftrightarrow \vec{\Omega}(\vec{i}^{(t+t_0)}, \vec{\ell}^{(t+t_0)}, \vec{q}^{(t+t_0)})\right)\right] \wedge \\ \quad \bigvee_{j=0}^{f} \left[\forall t. \Phi_j(\vec{i}^{(t+t_0)}, \vec{\ell}^{(t+t_0)}, \vec{q}^{(t+t_0)})\right] \wedge \left[\exists t. \Psi_j(\vec{i}^{(t+t_0)}, \vec{\ell}^{(t+t_0)}, \vec{q}^{(t+t_0)})\right] \end{array}\right)$$

- hardware formulae are closed under $\neg$, $\wedge$ and $\vee$
- hardware formulae are closed under WHEN and NEXT Operators
- hardware formulae can be used for all abstraction mechanisms

Example: Summation Circuit (SIEMENS,Germany)



If the circuit is ready and a new request occurs at time t_0, then the two inputs A and B are read into the registers OUT and R, respectively. For the next $\mathrm{VAL}(B^{(t_0)})$ cycles the inputs A are added to the contents of OUT. In the meantime, the register R is decremented. If $\mathrm{VAL}(R^{(\delta+t_0)}) = 0$ holds, the process terminates and the register OUT contains the sum of the inputs A given in the interval $t_0,\ldots,\delta+t_0$. The formal specification in HOL is therefore as follows:

$$\forall t_0. f^{(t_0)} \rightarrow \left[\left[\left(\begin{array}{l} \lambda t. \left(\mathrm{VAL}(OUT^{(t)}) = \sum_{j=t_0}^{t} \mathrm{VAL}(A^{(t)})\right) \wedge \\ \qquad (t - t_0 = \mathrm{VAL}(B^{(t_0)})) \end{array}\right) \text{ WHEN } f\right] \text{ WHEN } req\right]^{(t_0)}$$

For the correctness proof, the following invariant has been used: $q^{(t)} \wedge \left(\mathrm{VAL}(OUT^{(t)}) = \sum_{j=t_0}^{t} \mathrm{VAL}(A^{(t)})\right) \wedge \left((t-t_0) + \mathrm{VAL}(R^{(t)}) = \mathrm{VAL}(B^{(t_0)})\right)$, where $q^{(t)}$ is a state of the controller which denotes that the circuit is in the computation state and not ready for new inputs. Note that the above specification of the circuit cannot be given in temporal propositional logic, since various data types are required. Boolean lists as inputs are read in and interpreted as natural numbers on the one hand, and as points of time on the other hand. Even for concrete bitwidths, there is no satisfying specification in temporal propositional logic!

University of Karlsruhe,
Institute of Computer Design & Fault Tolerance,
Project Automated System Design

Computer Research Center
Automation of Circuit Design