Verification of Finite-State-Machine Refinements Using a Symbolic Methodology

Stefan Hendricx and Luc Claesen

Imec vzw/Katholieke Universiteit Leuven
kapeldreef 75, B-3001 Heverlee, Belgium
Stefan.Hendricx@imec.be

Abstract. The top-down design of VLSI-systems typically features a step-wise refinement of intermediate solutions. Even though these refinements do usually not preserve time-scales, current formal verification approaches are largely based on the assumption that both specification and implementation utilize the same scales of time. In this paper, a symbolic methodology is presented to verify the step-wise refinement of finite state machines, allowing for possible differences in timing-granularity.

1 Introduction

The top-down design of electronic VLSI-systems can, in general, be characterized in terms of well-defined (and mostly well-automated) refinement steps. Resource allocation, partitioning, control- and datapath synthesis and logic optimization are probably the best-known refinement steps encountered. With the aid of these refinements, high-level behavioural specifications of VLSI-systems — e.g. microprocessors — are progressively 'compiled' into physical implementations.

Despite the well-understood nature of these refinements, the history of electronic hardware design clearly illustrates that both the designer and the design-process may still be subject to a wide variety of shortcomings. Designers may suffer the occasional lapse of attention, design specifications may be ambiguous, incomplete or misinterpreted and there may even be bugs in the design-software itself. As a consequence, the possibility of refinement errors must always be kept in mind. Yet even with well-considered and far-reaching design, simulation and testing procedures, incidents such as the Pentium affair demonstrate that errors can still go undetected until the final release of the product [2].

The ever-growing complexity of modern VLSI-designs is partly to be blamed for limiting the effectiveness of traditional simulation-based verification and testing. Complex designs can no longer be verified exhaustively within realistic design-times and with non-exhaustiveness, there always exists the possibility that certain design-flaws remain undiscovered. Because of its crucial — and self-evident — role in producing qualitative designs, the ability to guarantee hardware correctness must therefore be procured through other alternatives. The most promising alternative is offered by formal verification [1]. Among other advantages, formal approaches allow exhaustive verification within a more-or-less

L. Pierre and T. Kropf (Eds.): CHARME'99, LNCS 1703, pp. 326–330, 1999.

acceptable time-frame. Not surprisingly, formal verification is currently one the focal points in electronic hardware-design and several applications — commercial as well as academic — have already been reported on.

Motivation The majority of formal techniques proposed so far, however, usually start from the assumption that both specification and implementation utilize the same scales of time. Whereas such an assumption is indeed valid for applications such as equivalence checking and finite-state-machine comparison, it is clearly not always applicable. In general, refinement does not preserve time-scales and timing-granularity tends to increase during the design-process. At the algorithmic level, for instance, executing a microprocessor-instruction may be specified in terms of a single (algorithmic) clock-cycle. At the physical level, however, executing the same instruction may span multiple physical clock-cycli.

In an effort to overcome this shortcoming, we recently proposed a symbolic approach dedicated to verifying the step-wise refinement of finite state machines, allowing for possible differences in timing [3].

Outline This research-paper presents a brief overview of our symbolic methodology. The next section discusses the underlying principles of our approach. In the poster accompanying this paper, the practical aspects of our approach are illustrated at the hand of a simple *paper-and-pen* example. The poster also demonstrates the step-wise refinement and verification of a simple microprocessor.

2 Symbolic Verification Methodology

The intuitive notion underlying our verification approach is very simple; when a finite state machine M_r is supposed to be a refinement of a state machine M, it is expected that a close relationship exists between a (non-empty) subset of M_r's states and the state-set of M. Based on this idea, the concept of a mapping-function is introduced.

2.1 User-Defined Mapping-Functions

Let S and S_r denote the high-level and low-level state-spaces respectively. In addition, let $S_{_M} \subseteq S_r$ be the (non-empty) subset of all low-level states that can be related to the high-level states. Specifically, the *mapping* $\Psi : (S_{_M} \subseteq S_r) \rightarrow S$ then denotes a function that maps — or relates — states in $S_{_M}$ to states in S.

In case of a refinement-relationship, Ψ exhibits the following properties:

- Ψ maps each state in $S_{_M}$ to exactly *one* state in S.
- Ψ may map multiple states in $S_{_M}$ to the same state in S.
- Ψ preserves the high-level outputs. That is, if $\mathbf{s} = \Psi(\mathbf{s}_r)$, the outputs relevant to the high-level state $\mathbf{s}$ should have the same values in both $\mathbf{s}$ and $\mathbf{s}_r$.

In general, it may be possible to determine more than one viable candidate for Ψ between any given pair of finite state machines. In this paper, however, we assume that the mapping of interest is supplied by the user.

2.2 State-Set-Bounded Trajectories

Together with the mapping-information provided by the user, the concept of *State-Set-Bounded* trajectories plays a key-role in our methodology.

Definition 1. *A* trajectory $\Delta_{\mathbf{s}_0 \to \mathbf{s}_n}$ *of a finite state machine M is a sequence of state-transitions* $\mathbf{s}_0 \xrightarrow{\tau_0} \mathbf{s}_1 \xrightarrow{\tau_1} ... \xrightarrow{\tau_{n-1}} \mathbf{s}_n$ $(\mathrm{n} \geq 1)$ *of M.*

Definition 2. *A trajectory* $\Delta_{\mathbf{s}_0 \to \mathbf{s}_n}$ *of M is* bounded *by states* $\mathbf{s}_0$ *and* $\mathbf{s}_n$*, if* $\mathbf{s}_0$ *and* $\mathbf{s}_n$ *appear* only *at the beginning or ending of the trajectory.*

Definition 3. *Assume* S *is a subset of states of M. A* state-set-bounded *trajectory or SSB-trajectory* $\Delta^{S}_{\mathbf{s}_0 \to \mathbf{s}_n}$ *of M is a trajectory* $\Delta_{\mathbf{s}_0 \to \mathbf{s}_n}$ bounded *by states* $\mathbf{s}_0, \mathbf{s}_n \in S$ and *containing no other states of* S.

In short, to verify that a finite state machine M_r is a refinement of a state machine M, we will examine if each low-level *SSB*-trajectory associated with the user-defined set $S_{_M}$ implies the existence of a high-level transition in M.

2.3 Finite-State-Machine Refinements

Let M and M_r be finite state machines with state-spaces S and $S_{_r}$ respectively. In addition, let Ψ be a (user-defined) function, which maps states in $S_{_M} \subseteq S_{_r}$ to states in S, such that for all low-level states $\mathbf{s}_{r1}$, $\mathbf{s}_{r2} \in S_{_M}$, there exist high-level states $\mathbf{s}_1$, $\mathbf{s}_2 \in$ S, for which $\Psi(\mathbf{s}_{r1}) = \mathbf{s}_1$ and $\Psi(\mathbf{s}_{r2}) = \mathbf{s}_2$.

M_r is called a refinement of M, if for each SSB-trajectory $\Delta^{S_M}_{\mathbf{s}_{r1} \to \mathbf{s}_{r2}}$ between states $\mathbf{s}_{r1}$ and $\mathbf{s}_{r2}$, a high-level transition T exists between $\mathbf{s}_1$ and $\mathbf{s}_2$, such that

$$Eval(\Delta^{S_M}_{\mathbf{s}_{r1} \to \mathbf{s}_{r2}}) \Rightarrow T_{\mathbf{s}_1 \to \mathbf{s}_2} \tag{1}$$

The function *Eval*( ) denotes the symbolic evaluation of the trajectories involved, with respect to the *de facto* stability conditions implied by the high-level state machine — i.e. during a transition, input signals are assumed to be stable.

2.4 Symbolic Evaluation of SSB-trajectories

Symbolic manipulation techniques and fixed-point calculations enable us to derive Boolean expressions for the symbolic evaluation of the SSB-trajectories associated with the mapping Ψ. As explained in [3], we simply need to consider a fixed-point for the generalized (n-steps) transition-relation $\delta^n(\mathbf{x}, \mathbf{y})$.

Informally, $\delta^n(\mathbf{x}, \mathbf{y})$ denotes the Boolean condition to reach an arbitrary state $\mathbf{y}$ in a sequence of n or *less* state-transitions, starting from state $\mathbf{x}$ (assuming stability of inputs during this sequence). For $n = 1$, $\delta(\mathbf{x}, \mathbf{y})$ simply corresponds to the conventional transition-relation of a finite state machine. For $n > 1$, we have the following recursive definition:

$$\delta^n(\mathbf{x},\mathbf{y}) = \delta^{n-1}(\mathbf{x},\mathbf{y}) \vee \left(\sum_{\forall \mathbf{q}} \delta^{n-1}(\mathbf{x},\mathbf{q}) \wedge \delta^*(\mathbf{q},\mathbf{y}) \right) \tag{2}$$

Above, $\delta^*(\mathbf{x},\mathbf{y})$ denotes the (1-step) transition-relation of a modified state machine M^*. [1] For a detailed discussion on M^*, the reader is again refered to [3].

For a finite state machine M_r and a user-defined set $S_{_M}$, we can show that there exists a finite value n_f, such that for all $m \geq n_f$,

$$\delta^{n_f}(\mathbf{x},\mathbf{y}) \equiv \delta^{m}(\mathbf{x},\mathbf{y}) \tag{3}$$

Using this fixed-point, a practical expression [2] can be derived for the symbolic evaluation of *all SSB*-trajectories associated with S_M:

$$Eval(\Delta^{S_M}_{\mathbf{x}\rightarrow\mathbf{y}}) \equiv \delta^{n_f}_{M_r}(\mathbf{x},\mathbf{y}) \wedge \chi_{S_M}(\mathbf{x}) \wedge \chi_{S_M}(\mathbf{y}) \tag{4}$$

Finally, expression 4 enables us to re-formulate the verification problem at hand (see section 2.3). To verify that M_r is a refinement of M — under the user-defined mapping $\Psi : (S_M \subseteq S_r) \rightarrow S$ — we need to prove that:

$$\delta^{n_f}_{M_r}(\mathbf{x},\mathbf{y}) \wedge \chi_{S_M}(\mathbf{x}) \wedge \chi_{S_M}(\mathbf{y}) \Rightarrow \delta_M(\Psi(\mathbf{x}),\Psi(\mathbf{y})) \tag{5}$$

So far, we have successfully applied expression 5 to verify — among others — the step-wise refinement of a simple microprocessor. That application example is illustrated in the poster accompanying this paper.

3 Conclusion

In this paper, a symbolic methodology to verify the step-wise refinement of finite state machines was presented. Key-elements to our approach are user-defined mapping-functions, that keep track of time-scale differences between the state-machines under consideration.

Acknowledgment

The research presented in this paper was supported by a scholarship from the Flemish Institute for the promotion of Scientific-Technological Research in Industry (IWT).

[1] Each transition in M starting in a state $\mathbf{s} \in S_M$ is replaced in M^* by a default transition to $\mathbf{s}$ itself. All other transitions (and states) of M are preserved in M^*.

[2] In expressions 4 and 5, $\chi_{S_M}(\,)$ represents the characteristic function of the set S_M.

References

1. P. Camurati and P. Prinetto. Formal Verification of Hardware Correctness: Introduktion and Sutvey of Current Research. *IEEE Computer*, 21(7):8-19, July 1988. 326
2. T. Coe, Mathisen T., C. Moler, and V. Pratt. Computational Aspects of the Pentium Affair. *IEEE Computational Science & Engineering*, pages 18-31, spring 1995. 326
3. S. Hendricx and L. Claesen. Symbolic Multi-Level Verification of Refinement. In *Nineth Great Lakes Symposium on VLSI*, Ann Arbor, MI 4-6 March 1999. IEEE Computer Society Press. 327, 328, 329