

(V)HDL-based Verification of Heterogeneous Synchronous/Asynchronous Systems

Hans Eveking

Johann Wolfgang Goethe-Universität Frankfurt
D-60054 Frankfurt am Main, Germany

Abstract

The modelling and verification of two types of heterogeneous systems is discussed: (i) systems consisting of synchronous subsystems with different clocks, and (ii) systems which are composed of synchronous and asynchronous subsystems. Concurrent behaviour is represented in VHDL by means of additional input signals, i.e., “concurrency is additional inputs plus fairness”. State space exploration techniques for mixed synchronous/asynchronous systems are presented.

1 Introduction

A number of powerful verification tools for state-equivalence proofs and model-checking were developed in the past. The tools are based on the principle of symbolic state space traversal [4, 3] where sets of states are efficiently represented by means of ROBDD's [2].

The verification tools typically accept system-descriptions in some HDL like SMV [10] or VHDL [1, 5]. In the case of VHDL, the domain of application is generally restricted to deterministic synchronous sequential circuits clocked by a single clock. Various description styles for this type of circuits were defined since the semantics of VHDL does not conform to such behaviours. In [5], the description style of VFORMAL is given which includes also the possibility of the resynchronization of asynchronous parts.

After a brief summary of transition systems in section 2, the extension of HDL-based symbolic state space traversal techniques to the verification of two types of heterogeneous systems will be discussed

(i) systems consisting of synchronous subsystems with different clocks (section 3), and (ii) systems composed of synchronous and asynchronous subsystems (section 4).

2 Transition systems

Some basic notations of transition systems are briefly reviewed in this section. A detailed introduction is given in [10, 8].

We model systems as binary encoded transitions systems. Each system has a set of n state variables (storage elements) $X = \{x_1, \dots, x_n\}$ and a set of m input variables $I = \{i_1, \dots, i_m\}$. In a deterministic system, a *transition function* (next-state function) $f_i(x_1, \dots, x_n, i_1, \dots, i_m)$ is associated with each state variable x_i . We will write $f_i(X, I)$ for simplicity. The set F is the set of the f_i : $F = \{f_1, \dots, f_n\}$.

To formulate the *transition relation*, an additional set of n auxiliary state variables $X' = \{x'_1, \dots, x'_n\}$ is introduced modeling the “new” states. The transition relation “relates” old, new and input variables. While the transition relation T_A of a system A is in general a relation $T_A(X', X, I)$, the transition relation of a deterministic system where the value of each x'_i is uniquely determined by a transition function f_i is given by

$$T_A = (x'_1 \equiv f_1(X, I)) \cdot \dots \cdot (x'_n \equiv f_n(X, I)) \quad (1)$$

or $T_A = (X' \equiv F(X, I))$ in more compact notation. The transition relation $T_{A=}$ represents that the state of the system A is not modified:

$$T_{A=} = (x'_1 \equiv x_1) \cdot \dots \cdot (x'_n \equiv x_n),$$

or simpler: $T_{A=} = (X' \equiv X)$.

The *symbolic forward traversal operation* calculates the characteristic function $\chi_n(X')$ of all successor states reachable from a set of states given by the characteristic function $\chi_o(X)$:

$$\chi_n(X') = \exists X, I : \chi_o(X) \cdot T_A(X', X, I).$$

In order to repeat this step, the “new” variables X' have to be replaced by the variables X which can be done by $\chi_n(X) = \exists X' : \chi(X') \cdot (X' \equiv X)$. In a symbolic *backward* traversal step the roles of the state variables X and X' are interchanged. Symbolic traversal steps are typically iterated until a fixed-point is reached.

3 Heterogeneous synchronous systems

Fig. 1 shows a system of two interacting synchronous systems which are clocked by two distinct physical clocks. To model the concurrent opera-

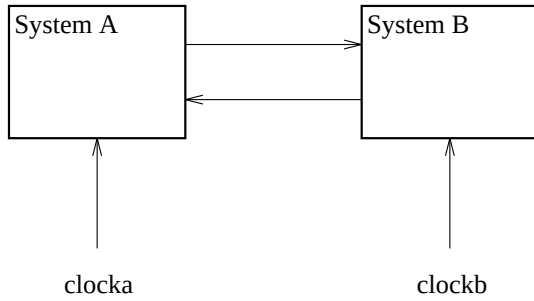


Figure 1: A heterogeneous synchronous system

tion of such a system typically two modes of concurrency are used [10]:

- the *interleaving* mode represents all interleaved executions; simultaneous executions are not modeled.
- the *simultaneous* mode where - in addition to all interleaved executions - simultaneous state changes are possible.

Fig. 2 exemplifies the difference of the two modes. The most commonly used mode is the interleaving

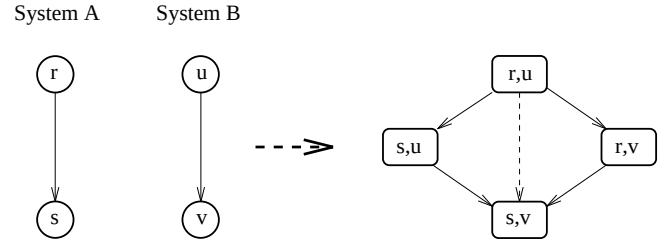


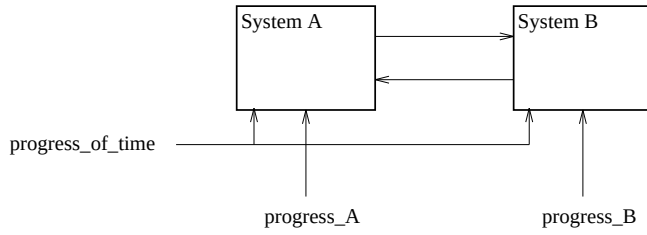
Figure 2: Interleaving and simultaneous mode (dashed line in addition) of concurrency

mode because in a model where, e.g., two processes share a common memory only a single variable (= memory address) changes at each point of time; thus, the state space of both processes shared by the common memory can not change simultaneously. This is of course not the case in our situation where simultaneous state-changes may occur. Hence, the simultaneous mode is our choice.

The formalization of the two modes of concurrency in terms of transition relations given in [10], for instance, is non-deterministic, and can not be brought into form (1) based on transition-functions. In the context of HDL's where simulation-based deterministic solutions prevail, we propose a different formalization of the simultaneous mode of concurrent operation which can be given in form (1). For each subsystem, an additional input signal called *progress-signal* is introduced. Since inputs are not under control of the system the values of the inputs are not determinate, and are thus used to model non-determinism. Fig. 3 shows the VHDL description principle of a heterogeneous synchronous systems. The input signal **progress_of_time** models the progress of discrete time (in a similar way as the clock-signals of various description styles are used to model synchronous systems).

While the introduction of the progress-signals as additional ports may be viewed as artificial since these signals are not “real” ports, our approach offers a number of advantages:

- in the models of concurrency given above it is possible that one system moves infinitely often while some other idles. This unrealistic



```

entity het_syn is
  port (progress_of_time,
        progress_A, progress_B: in bit;
        i1, ..., im: in bit);
end het_syn;

architecture behavior of het_syn is
  signal x1, ..., xn: bit;
  begin
    process
    begin wait on progress_of_time
      until progress_of_time = '1';
      if progress_A then
        ... -- description of system A
      endif;
      if progress_B then
        ... -- description of system B
      endif;
    end process;
  end behavior;

```

Figure 3: VHDL description principle of heterogeneous synchronous system

behavior is viewed to be “unfair” to the other system, and is excluded by means of *fairness-conditions* that ensure the progress of all sub-systems.

In our approach, fairness-conditions can be made explicit using the progress variables as reference signals. This is particularly useful to specify qualitative relationships between the clock-rates of the subsystems, for instance, that the clock-rate of one system is faster than the clock-rate of some other system (note that, in contrast, the models of concurrent behaviour comprise all possible sequences of execution, e.g., even behaviours where the clock-rates change in time).

- the global behavior remains deterministic; thus, verification techniques based on transition functions can be used rather than

transition-relation techniques.

- the resulting deterministic behaviour corresponds to the behavior of simulators. This offers the unique possibility to integrate the power of the highly-elaborated commercial simulation tools with our verification techniques. In the PREVAIL system [1], the formal verification tools provide counter examples in the case of unsuccessful verification (Fig. 4); the counter examples are transformed into input patterns of a commercial simulator. Well-directed by the counter examples the simulator can be used for the detailed analysis of erroneous behavior.

Note that in the models of concurrency used above a number of synchronization problems are ignored, and may thus lead to erroneous results if neglected (a detailed discussion is given in [11]). In particular, signals which are inputs of a synchronous system and arrive unsynchronized have to be sampled, e.g., by a *sample-flipflop* to synchronize with the system-clock. Even then a single input signal may not be synchronized properly due to metastable behavior of the sample-flipflop. In addition, several sampled input-signals may not reflect the actual changes if sampled during transitions. For instance, if two inputs change from 00 to 11 all possible combinations of values may be sampled during the transition. Hence, for the transmission of, e.g., data-vectors a handshaking protocol has to be observed.

4 Heterogeneous synchronous/asynchronous systems

We briefly rephrase some notions of asynchronous circuits in terms of transition relations. A more detailed introduction to asynchronous designs is given, for example, in [9, 7]. Formal verification of asynchronous circuits goes back to [6].

Our model of an asynchronous circuit is given in terms of a transition relation T involving a set of inputs I and two sets of “old” and “new” *feedback variables* Y , i.e., $T = Y' \equiv F_Y(I, Y)$. In the example of Fig. 5, we have

$$y' \equiv \bar{x}_1 \cdot x_2 + \bar{x}_1 \cdot y.$$

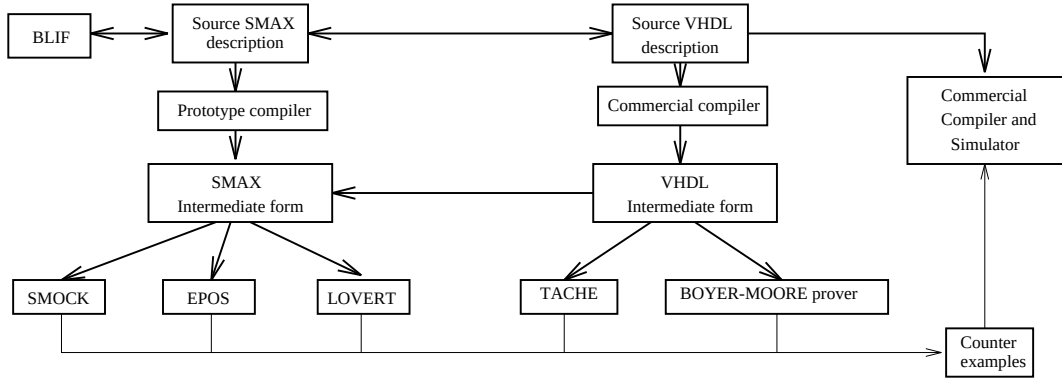


Figure 4: Interaction of formal verification and simulation in the PREVAIL system

An asynchronous circuit reacts on input-changes by a series of transition until a stable state is reached. The *stable states* of an asynchronous circuit are defined as all states where the “old” and “new” values of Y coincide, and are characterized

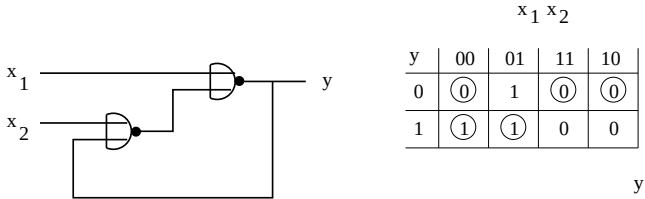


Figure 5: Example asynchronous circuit

by the characteristic function $\chi_{stab} = (Y \equiv F(Y, I))$. In the example of Fig. 5, we get

$$\chi_{stab} = (y \equiv f_y(x_1, x_2, y)) = \bar{x}_1 \cdot y + x_1 \cdot \bar{y} + \bar{x}_2 \cdot \bar{y}.$$

The stable states are encircled in Fig. 5.

Triggering by input-changes and stabilization of an asynchronous circuit can be modeled in VHDL by means of a process which is sensitive to all input and feedback signals, and where concurrent signal assignments are used to modify the feedback signals according to the following schema:

```

process
begin wait on i1, ..., im, y1, ..., yk;
  y1 <= ...;
  ...;
  yk <= ...;
end process;

```

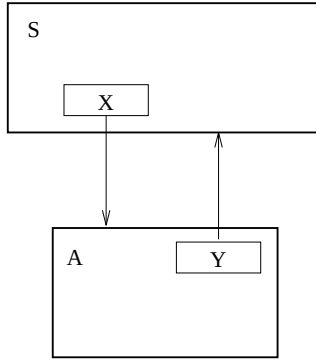
In Fig. 6, a heterogeneous system consisting of a synchronous system S and an asynchronous system A is shown. The outputs of the synchronous system work as inputs of the asynchronous system and vice versa.

The interaction between heterogeneous subsystems is based on the assumption that the time needed to reach a stable state is smaller than the clock-time following one of the basic assumption of the “fundamental mode” which requires that inputs do not change unless the system has finished its reaction on the last input change. Hence, the interaction of systems S and A of Fig. 6 can be discussed in terms of a master/slave behavior where the changes in A are “late” changes of the whole system.

Fig. 7 shows the VHDL description principle of a heterogeneous synchronous/asynchronous system. The asynchronous part is described as a process which is iterated until the values of all signals y_i are stable since all signals y_i are included in the **wait**-statement of this process.

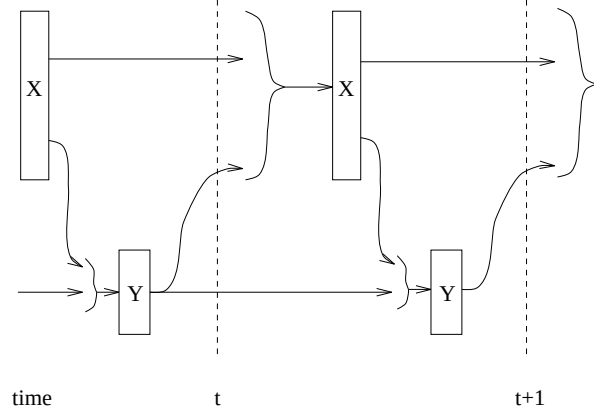
In the following, the circuit of Fig. 5 will be used as an example of the asynchronous subsystem of Fig. 6. It is assumed that the synchronous part consists of two flipflops x_1 and x_2 directly connected to the inputs of the synchronous subsystem.

Synchronous subsystem



Asynchronous subsystem

a)



b)

Figure 6: (a) Structural and (b) temporal relationships of a heterogeneous synchronous/asynchronous system

A *symbolic forward traversal step* of the heterogeneous system of Fig. 6 consists of the following parts:

1. the state changes of the synchronous subsystem are calculated by

$$\chi_{ns}(X', Y') = \exists X, Y, I : \chi_o(X, Y) \cdot (X' \equiv F_X(X, Y, I)) \cdot (Y' \equiv Y)$$

In the example of Fig. 5 assuming $\chi_o = y \cdot \bar{x}_1 \cdot \bar{x}_2$, i.e., $y = 1$, $x_1 = 0$ and $x_2 = 0$ and assuming that the synchronous system moves from state $x_1, x_2 = 00$ to either 10 or 01 we obtain $\chi_{ns} = y' \cdot (x'_1 \cdot \bar{x}'_2 + \bar{x}'_1 \cdot x'_2)$.

The “new” state variables in $\chi_{ns}(X', Y')$ are replaced by X, Y leading to $\chi_{ns}(X, Y)$. In the example we get $\chi_{ns} = y \cdot (x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2)$.

2. the transitions of the asynchronous subsystem are computed by

$$\chi_{na}(X', Y') = \exists X, Y, I : \chi_{ns}(X, Y) \cdot (X' \equiv X) \cdot (Y' \equiv F_Y(X, Y, I)).$$

In the example of Fig. 5 and χ_{ns} as above, we have

$$\exists x_1, x_2, y : y \cdot (x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2) \cdot$$

$$\begin{aligned} & (X' \equiv X) \cdot (y' \equiv f_y(x_1, x_2, y)) \\ &= x'_1 \cdot \bar{x}'_2 \cdot (y' \equiv f_y(1, 0, 1)) + \\ & \quad \bar{x}'_1 \cdot x'_2 \cdot (y' \equiv f_y(1, 0, 1)) \\ &= \bar{y}' \cdot x'_1 \cdot \bar{x}'_2 + y' \cdot \bar{x}'_1 \cdot x'_2 \end{aligned}$$

3. this is iterated until a fixed-point is reached. At each step, the reached stable states are collected using χ_{stab} . Stable states are not further considered for fixed-point iteration. Finally, stable states only have to be reached; otherwise, an oscillation is detected. In the example, we get finally $\bar{y} \cdot x_1 \cdot \bar{x}_2 + y \cdot \bar{x}_1 \cdot x_2$ as the characteristic function of the possible stable successor states of state $y \cdot \bar{x}_1 \cdot \bar{x}_2$.

In the *symbolic backward traversal procedure* for, e.g., the *EX*-operator of CTL [10] which calculates the set of all predecessor states, we proceed as follows:

1. starting with some characteristic function $\chi_n(X', Y')$ first all possible predecessor states of the asynchronous subsystem are determined while the state-variables of the synchronous system are fixed:

$$\chi_{na}(X, Y) = \exists X', Y', I : \chi_n(X', Y') \cdot (X' \equiv X) \cdot (Y' \equiv F_Y(X, Y, I))$$

```

entity syn_async is
  port (progress_of_time: in bit;
        i1, ..., im: in bit);
end syn_async;

architecture behavior of syn_async is
  signal x1, ..., xn: bit;
  signal y1, ..., yk: bit;
  begin
    process -- synchronous part
    begin wait on progress_of_time
          until progress_of_time = '1';
          ...
    end process;

    process -- asynchronous part
    begin wait on x1, ..., xn, y1, ..., yk;
          y1<= ...;
          ...
          yk<= ...;
    end process;
  end behavior;

```

Figure 7: VHDL description principle of heterogeneous synchronous/asynchronous system

This is iterated until a fixed-point is reached. Assume in the example $\chi_n = \bar{y} \cdot \bar{x}_1 \cdot \bar{x}_2 + y \cdot \bar{x}_1 \cdot x_2$ (states $y, x_1, x_2 = 000$ and 101). Then the possible predecessor states are 000 , 001 and 101 .

2. afterwards, the states of the asynchronous system are fixed and the possible predecessors of the synchronous subsystem are determined in a single step by

$$\chi_{ns}(X, Y) = \exists X', Y', I : \chi_{na}(X', Y') \cdot (X' \equiv F_X(X, Y, I)) \cdot (Y' \equiv Y).$$

3. from this set, the unstable states are removed. For instance, the state 001 is not a possible predecessor of state 000 .

Conclusions

The representation of concurrency in VHDL by means of “additional input signals plus fairness” has the significant advantage that the deterministic frame-of-reference typical for simulation-based

HDL’s can be maintained. This offers the opportunity for the combination of formal verification and analytical simulation tools and techniques for heterogeneous systems.

The representation and verification of mixed synchronous/asynchronous systems is feasible, too, since the stabilization of an asynchronous system, the stabilization of the representing VHDL-process as well as the stabilization of the associated fixed-point calculation coincide.

References

- [1] D. Borriore, L. Pierre, and A. Salem. Formal verification of VHDL descriptions in the Prevail environment. *IEEE Design and test of Computers*, 9(2):42–56, 1992.
- [2] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE C-35*, pages 677–691, 1986.
- [3] J.R. Burch, E.M. Clarke, K.L. McMillan, and D.L. Dill. Sequential circuit verification using symbolic model checking. In *Proc. 27th DAC*, pages 46–51, 1990.
- [4] O. Coudert, C. Berthet, and J.C. Madre. Verification of synchronous sequential machines based on symbolic execution. In *Proc. Workshop on Automatic Verification Methods for Finite State Systems, Grenoble*, pages 365–373, 1989.
- [5] A. Debreil and Phil. Oddo. Synchronous designs in VHDL. In *Proc. EuroDAC/EuroVHDL*, pages 486–491, 1993.
- [6] D. Dill and E.M. Clarke. Automatic verification of asynchronous circuits using temporal logic. In *Proc. IEEE*, pages 276–282, 1986.
- [7] L. Lavagno and A. Sangiovanni-Vincentelli. *Algorithms for synthesis and testing of asynchronous circuits*. Kluwer, 1993.
- [8] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, 1991.
- [9] E. J. McCluskey. *Logic Design Principles*. Prentice-Hall, 1986.
- [10] K. L. McMillan. *Symbolic Model Checking*. Kluwer, 1993.
- [11] T. H. Meng. *Synchronization Design for Digital Systems*. Kluwer, 1991.