Verification of a Production Cell using an
Automatic Verification Environment for VHDL

Ronald Herrmann*†
Siemens AG, Corporate R & D
Otto-Hahn-Ring 6
81730 Munich, Germany

Thomas Reielts ‡
OFFIS, FB 3
Westerstraße 10-12
26121 Oldenburg, Germany

Abstract
This paper presents from the users point of view the
automatic verification of nontrivial liveness properties
for a reactive system implemented using VHDL. The
aim is to make clear the simplicity, power and prac-
tical relevance of tools developed within the ESPRIT
project FORMAT. For the specialist this paper provides
a run through Assumption Commitment Style Ver-
ification and an overview of relevant publications.

1 Introduction
This paper describes the verification of a non trivial
case study called 'Production Cell' using an automatic
verification environment. The 'Production Cell' inte-
grates several different physical devices into one manu-
facturing method for modeling metal blanks. Due to
the fact that the machines are controlled by integrated
circuits (controller), a system level VHDL description
of these controllers should be formally verified. This
case study represents a safety critical reactive system.

Automatic verification tools provide early and fast
recognition of logical design errors which leads to short-
er design times. A lot of progress has been in the
development of automatic verification and validation
environments and a lot of complex CAD environments
with verification utilities have been developed (cf. [1]).
The tools we are using have been developed within the
ESPRIT III project FORMAT, can handle full VHDL
and combine graphical and textual specification lan-
guages so that it is very easy to use them.

This paper shows that nowadays it is possible to
verify a real life example by means of full automatic
verification tools that are the use of Symbolic Timing
Diagrams (STDs), Symbolic Model Checking (SMC)
and Tautology Checking (TC). STDs are used to spe-
cify the requirements which should be verified and the
description of the implementation is given in VHDL.
The paper [16] shows a different approach in that way
that they use STDs as specification language but they
do not use VHDL as implementation language. This
leads to the fact that they do not use SMC to establish
basic proofs.

Figure 1: Global view of the production cell

The paper is structured as follows: In the following
section we will introduce the 'Production Cell' in more
detail. The third section will give a survey of the veri-
fication methodology. Before we will set out concrete
examples we sketch the generation of the symbolic mod-
els used by the model checker in section 4. In the fifth
section we describe the verification of some nontrivial
properties before the last section summarizes the re-
results of this approach.

2 The Production Cell
To show the power of this verification approach we
are using a very vivid example referred to as 'Production
Cell' (PC). This example is first used as a com-
parative case study in the German Korsko project to
try out several different specification and verification
methods [12]. The PC represents a typical industrial
production cell used in a metal processing plant. It is
composed of two conveyor belts called feed belt and
deposit belt, a rotary elevating table, a two-armed robot, a press and a traveling crane as represented in figure 1. Metal plates are inserted into the system by putting them onto the feed belt. The feed belt moves the plates onto the rotary elevating table which has to be in the correct bottom position. Then the table changes its position to a position which is correct for the handing-over to the robot. The robot takes the plate with his first arm and moves it into the press. After the plate was forged, the robot moves the plate from the press to the deposit belt. The deposit belt transports the plate to the end of the belt and the crane takes it over to move it out of the cell.

Figure 2: Interface of the deposit belt controller

To control the movement of the cell components, a set of nontrivial controller ensure correct behavior. Because it is not possible to demonstrate the full verification process within this paper we concentrate on one simple component for detailed explanation: the deposit belt (DB). The full verification of the PC has been described in [14].

This conveyor belt should move plates, which are put on it by the robot, to the crane so that they can be taken over. Figure 2, 3 and 4 illustrate the DB controller: The input DB.in indicates that a sensor at the end of the deposit belt has detected a plate. The input R_A2M.On is connected with the robot controller and is set to high whenever the magnet on robot arm 2 is on. DB_loadToDBin is a special input port which is used for verification issues only (cf. section 3). The output port DB_Move controls the movement of the DB motor. DB_load_loaded transports the value of the internal signal DB_load, so that the environment has the information whether the DB is loaded or not. The features of the deposit belt controller can be described as follows:

- Disallow movement without a plate lying on it.
- Whenever a plate reaches the end of the DB the motor should stop so that the crane can take the plate and the plate will not fall from the DB.
- There is no special sensor indicating that a plate is lying on the belt so that the controller has to model this information: The controller recognizes that

the robot has put a plate on the DB (R_A2M.On) has been switched from '1' to '0'). An invariant (which has to be verified for the robot) guarantees that whenever the magnet is switched on, the corresponding arm is charged with a metal plate.

- The controller has to guarantee that the robot does not put an additional plate on the belt while there is a plate lying at the beginning of the belt. Because the controller has no information whether a metal plate has moved e.g. one meter or to the end, the total capacity which can be handled is two plates.

3 Verification Methodology

To cope with the verification task we are using one path of the verification methodology developed within the ESPRIT project FORMAT. The FORMAT system in general supports a specification language called VHDL/S providing four specification styles. VHDL/S covers operational and declarative as well as textual and visible specification mechanisms. The verification approach uses partitioning techniques so that parts of the designs can be verified by symbolic model checking techniques [2, 11]. The verification of properties for sub-components where the behavior of the environment has to be specified is summarized as Compositional Verification [3]. We are using a very powerful assumption - commitment style model checker developed by Siemens (ZFET SE 1) [10] which requires as input a symbolic finite state model representing the behavior of the VHDL implementation (cf. section 4), a set of assumptions and a commitment. The model checker solves the task whether the model guarantees the behavior specified in the commitment under the condition that the environment of the model behaves as specified in the set of assumptions.

\[
\text{model } + \text{ assumptions } \models \text{ commitment}
\]
architecture DB_control_body of DB_control is
begin
loaded : process
begin
wait on R_A2M_On until ( R_A2M_On = '0' );
DB_loaded <= '1' after 1 ns;
wait on DB_in until ( DB_in = '1' );
DB Loaded <= '0' after 1 ns;
end process loaded;
move: process
begin
wait on DB_in, DB_loaded
until ( DB_in = '0' and DB_loaded = '1' );
DB_Move <= forward after 1 ns;
wait on DB_in until ( DB_in = '1' );
DB_Move <= off after 1 ns;
end process move;
end DB_control_body;

Figure 4: VHDL architecture body of the DB controller

Tautology checking is now used to derive more complex properties for the composed system as follows: Let $A = \{ a_1, \ldots, a_n \}$ be a set of assumptions and let $C = \{ c_1, \ldots, c_m \}$ be a set of verified commitments each of it valid under a subset $A_i \subseteq A$ of assumptions or composed by tautology checking using the described method. Now the commitment $c_{\text{complex}}$ is guaranteed to be valid under the set of assumptions $A_{\text{new}} \subseteq A$ and commitments $C_{\text{new}} \subseteq C$ if and only if

$$\bigwedge_{a \in A_{\text{new}}} a \land \bigwedge_{c \in C_{\text{new}}} c \implies c_{\text{complex}}$$

is a tautology. The tautology checker is part of the model checker i.e. the tableau generation is used to perform this task.

Assumptions and commitments are specified using the graphical specification language referred to as Symbolic Timing Diagrams (STDs) as introduced in [15]. STDs are translated into First Order Temporal Logic and the temporal logic formulae are directly used as input to either the SMC or the TC. STDs have a declarative semantics in a sense that whenever an underlying model shows (referring to an interface) a behavior so that the timing diagram matches, then the timing diagram specifies exactly the expected behavior. A timing diagram matches at that moment the values of the specified input and output ports assumed from the model are identical to the values specified in the starting point of the timing diagram. The example STD $C_1$ shown in figure 5 matches or is active at that moment the input ports $R_A2M_On$ and $DB_{in}$ are assumed to be set to '1' resp. '0' and the output port $DB_{load\_loaded}$ has the value '0'. An event in an STD is a change of values for a certain port looked at. The arcs in this example STD define a (total) order on possible events on the ports. In this case each arc indicates that the target event may not happen at the same time as the source event and that the target event has eventually to occur. The precise semantics is described in [15, 3].

It may help in some verification sessions to associate events with auxiliary ports introduced in the design entity. These ports can be referred to within assumptions. Such an auxiliary port is called a probe. An example probe is $DB\_\text{inTo}\_\text{Mon}$ which is used to cope with the delay while setting the magnet of the crane. The value of the probe is set by the assumptions $A_{P_{\text{true}}}$ and $A_{P_{\text{false}}}$ (cf. section 5). The probe must not influence the behavior of the model. This has to be checked automatically.

Because of the fact that we model a non delta-delay VHDL design we have to guarantee that the environment of a module will infinitely often eventually perform a non delta-delay step every time we would like to proof a liveness property which depends on the progress of time. To do this we are using a special assumption $A_{EDP}$ which encodes that the environment finally and infinite often performs a step with a positive amount of time.

If a commitment is checked to be NOT TRUE, the model checker computes a counterexample which has either the form of a sequence of states coming up with the violation of the commitment or with a sequence of states terminating in a loop. The last will be the case at every time a violation of a liveness property has been detected. E.g. an event is expected but impossible to occur.

4 Computing the Symbolic Models

The first step to do symbolic model checking is the automatic translation of the VHDL architectures into symbolic finite state machines:
1. In the first step the VHDL source text will be compiled into a graph structure integrated into a LEDA design database\(^1\).

2. The VHDL graph structure is then compiled into an interpreted Petri net[4, 6] which has an associated state space. Special places are annotated with conditions over the state space, transitions are annotated with transformations.

3. The global state space is transformed into a binary state space. Within this translation step we are computing an initial binary variable ordering which is necessary for the computation of the BDDs. In the binary state space we distinguish between binary variables representing inputs and local values [9].

4. Next the Petri net is transformed into a control automaton (CA) representing the case graph of the net. The CA consists of states and transitions where each transition is annotated with a condition and a set of transformations. Some states of the CA are marked as WAIT states indicating that the model has reached a WAIT configuration\(^2\).

5. The annotations of the CA will be translated into a BDD representation\(^9\).

6. In a special reduction step we compose different transformations of the CA to more complex transformations. The effect is a dramatic reduction of states. The CA where each state represents a WAIT state is referred to as Macro Machine [13].

7. Finally the Macro Machine is transformed into a set of functions representing the next step function of the Input-deterministic model. Input-deterministic means that the model is deterministic for a given set of input values [13].

Table 6 summarizes the details of the model generation\(^3\).

5 The Verification of the Deposit Belt

We have to verify safety and liveness properties. A typical safety property is: Whenever a plate is lying on the feed belt and the motor mode is forward, the motor is switched to off as soon as the plate reaches the end of the feed belt. This fact is indicated by the event that the input sensor DB\(_{\text{in}}\) switches from '0' to '1'. This can very easily be expressed by one STD which can be verified without any additional assumptions.

The verification of liveness properties requires additional assumptions which have to be selected very carefully by the user. We will now demonstrate this by a 'global' liveness property for the DB. 'Global' says that this liveness property requires liveness from its adjacent components: At every time the robot puts a metal plate on the DB this metal plate will finally be picked up by the crane (cf. STD in figure 7). To verify \(C_{\text{global}}\) we have to check two liveness properties of the DB and one liveness property of the crane:

<table>
<thead>
<tr>
<th>Controller</th>
<th>EF</th>
<th>FB</th>
<th>T</th>
<th>R</th>
<th>P</th>
<th>DB</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>VHDL:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines of Code</td>
<td>25</td>
<td>36</td>
<td>48</td>
<td>128</td>
<td>23</td>
<td>23</td>
<td>56</td>
</tr>
<tr>
<td># processes</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Petri-net:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Places</td>
<td>46</td>
<td>64</td>
<td>121</td>
<td>251</td>
<td>73</td>
<td>50</td>
<td>157</td>
</tr>
<tr>
<td>Trans.</td>
<td>34</td>
<td>47</td>
<td>89</td>
<td>190</td>
<td>55</td>
<td>37</td>
<td>117</td>
</tr>
<tr>
<td>Model:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Input Bits</td>
<td>7</td>
<td>15</td>
<td>28</td>
<td>34</td>
<td>19</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td># State Bits</td>
<td>29</td>
<td>47</td>
<td>93</td>
<td>139</td>
<td>54</td>
<td>40</td>
<td>89</td>
</tr>
<tr>
<td>States</td>
<td>5</td>
<td>11</td>
<td>44</td>
<td>1602</td>
<td>9</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>Trans.</td>
<td>12</td>
<td>43</td>
<td>438</td>
<td>18227</td>
<td>17</td>
<td>17</td>
<td>30</td>
</tr>
<tr>
<td>Generation [sec]</td>
<td>20</td>
<td>39</td>
<td>668</td>
<td>31717</td>
<td>31</td>
<td>29</td>
<td>167</td>
</tr>
<tr>
<td>Reachability [sec]</td>
<td>3</td>
<td>17</td>
<td>3643</td>
<td>-</td>
<td>17</td>
<td>8</td>
<td>460</td>
</tr>
</tbody>
</table>

Figure 6: Symbolic models (EF external feed, FB feed belt, T table, R robot, P press, DB deposit belt, C crane)

\[\text{Bit } R_{\text{A2M} \text{On}} \quad = '1' \quad = '0'\]
\[\text{Bit } DB_{\text{load_loaded}} \quad = '1' \quad = '0'\]
\[\text{Bit } DB_{\text{in}} \quad = '0' \quad = '0'\]
\[\text{Boolean } DB_{\text{load_toDBIn}} \quad = '0' \quad = '1'\]
\[\text{Boolean } DB_{\text{in_toDBIn}} \quad = '0' \quad = '0'\]
\[\text{Bit } C_{\text{MN}} \quad = '0' \quad = '1'\]

Figure 7: The commitment \(C_{\text{global}}\) should be verified as final property.

\[\text{Bit } DB_{\text{load_loaded}} \quad = '0' \quad = '1'\]
\[\text{Bit } DB_{\text{in}} \quad = '0' \quad = '1'\]

Figure 8: Commitment: \(C_2\)

1. The STD \(C_1\) (cf. figure 5) has been described in section 3 and models the fact that a metal plate is hanging on the magnet of robot arm 2 \((R_{\text{A2M} \text{On}} = '1')\), there is no plate lying on the DB \((DB_{\text{load_loaded}} = '0')\) and there is no plate at the end of the feed belt \((DB_{\text{in}} = '0')\). When the magnet switches to off and the metal plates falls on the DB the signal \(DB_{\text{load_loaded}}\) should be set to '1'. This implies that the motor starts moving.

\(^1\)The LEDA VHDL System is a product of LEDA S.A. 35 Avenue du Granier 38240 Meylan France

\(^2\)This model can be computed without building the Petri net[7, 8].

\(^3\)All runs presented in this paper have been performed on a SPARC Station 10.
forward and under the assumption that $DB_{in}$ indicates that the plate has reached the end of the belt, the output_port $DB_{load\_loaded}$ should be set to '1'.

2. Figure 8 presents the STD $C_3$ modeling the fact that the output port $DB_{load\_loaded}$ is set to '1' before $DB_{in}$ is set to '1'.

3. The STD $C_3$ shown in figure 9 models the liveness property of the crane that whenever a metal plate reaches the end of the DB ($DB_{in} = '1'$) the crane will finally switch on his magnet $C_{MON} = '1'$: A special arc from the upper left corner of the timing diagram to the event $C_{MON} = '0'$ models that the crane has to switch off the magnet again.

For the verification of the DB commitments we need the following assumptions:

1. $A_1$ (cf. figure 10): If the magnet of the robot arm 2 has a metal plate, the robot puts the metal plate on the DB after $DB_{load\_loaded}$ has indicated that there is no longer an old metal plate lying at the beginning of the DB.

2. $A_2$ (cf. figure 11): The sensor $DB_{in}$ switches eventually to '1' if the MotorMode changes from off to forward.

3. $A_3$ (cf. figure 12): It is impossible for any metal plate to reach the sensor $DB_{in}$ before the motor mode has changed from off to forward.

4. $A_4$ (cf. figure 10, but with $DB_{in}$ instead of $R_{A2M\_On}$) The input port $DB_{in}$ has to be stable until $DB_{load\_loaded}$ has changed to '0'. In addition it describes the liveness property that $DB_{in}$ will eventually be set to '0'. This implies the fact that the crane will eventually take over the metal plate from the end of the DB.

Now we check the commitment $C_1$ on the model of the DB referred to as $M_{DB}$: The task

$$M_{DB} + (A_{EDP} \land A_1 \land A_2 \land A_4 \land A_5 \land A_6) \models C_1$$

is evaluated within 41 sec. (8 sec. for tableau generation, 12 sec. to compute fairness and 21 sec. for model checking), but the result is NOT TRUE. The counterexample is visualized in figure 13. The fault is that the robot puts two metal plates on the feed belt and the controller of the feed belt reacts to slow. The reason is the delay time 1 ns used in the specification and the unrealistic situation that the robot is such fast that the next plate is available in an time shorter than the specified delay. To avoid this we add the additional assumption $A_3$

$$M_{DB} + (A_{EDP} \land A_1 \land A_2 \land A_3 \land A_4 \land A_5 \land A_6) \models C_1$$

and the model checker checks this to TRUE. The preparation of a model with incorporated assumptions needs 41 sec. (4 second for the reachability on the original model, 11 sec. for the tableau generation, 11 sec. to compute reachability on the combined model and 15 sec. to compute fairness) and model checking takes 26 sec. $C_2$ can be verified with the same set of assumptions. After 32 sec. of model checking the model checker answers TRUE. The model checker reuses the tableau and the fairness computation because the set of used assumptions did not changed. The preparation of a combined model for the crane needs 2 h 45 minutes which includes two reachability computations, the tableau generation and the computation of fairness. We can verify the $C_3$ within 1 h 15 min to TRUE. This increase of time is caused by the fact that
we need 32 assumptions for the crane which are used to model the physical behavior of the cranes gripper.

To verify the global commitment $C_{\text{global}}$ we need the following assumptions of the crane:

1. $A_1$: Probe $DBinToMon$ is set to '1' at the same time the magnet of the crane switches on. The STD matches every time the magnet is '0' and the value of the probe is false.
2. $A_2$: $DBinToMon$ may not be set to 'true' before the crane has moved the already loaded metal plate out of the PC. This STD matches whenever the magnet of the crane is '1' and the value of the probe is false.
3. $A_3$: The probe is set to 'false' when the crane has taken the metal plate. The STD matches with probe equal to 'true' and $DBin = '1'$. $C_{\text{DBinit}}$ and $C_{\text{Cinit}}$ are commitments describing the initialization of the DB and the crane. Finally we can proof within 92.15 sec. that

$$C_1 \land C_3 \land C_2 \land C_{\text{DBinit}} \land C_{\text{Cinit}} \land A_1 \land A_3 \land A_{P_{\text{true}}} \land A_{P_{\text{false}}} \land A_7 \land A_8 \land A_9 \land A_{10} \implies C_{\text{global}}$$

is a tautology.

6 Conclusion and Outlook

In this work we have shown that the verification of safety and of liveness properties for reactive systems specified in VHDL can easily be performed in reasonable time. The verification of the 'Production Cell' controller is full automatic except that the user has to specify the requirements and to perform the modularization. To this end the VHDL module concept is suited to perform Assumption Commitment Style Verification to cope with complexity. The use of STDs helps the user to specify the properties which otherwise have to be expressed directly in temporal logic formulae. Other approaches (cf. [1]) have different underlying semantics and handle only delta delay designs. Further work will be the improvement of debug facilities, the performance of the model generation and the model checking techniques itself. An approach to the use of automatic state and data abstraction is presented in [5].

7 Acknowledgments

We would like to thank everyone participating in the success of the FORMAT verification environment.

References