Concurrent Design Methodology and Configuration Management of the SIEMENS EWSD - CCS7E Processor System Simulation

Thomas W. Albrecht, *Member, IEEE*
Siemens AG Austria
Engineering Center for Electronics, EZE
A-1030 Vienna
e-mail: Thomas.Albrecht@p0.hai.siemens.co.at

Abstract - This paper outlines our successful Concurrent Design Methodology and Configuration Management adopted for a Processor system simulation at both VHDL and GATE level. The complexity of the system simulated was 4 PENTIUMs and 28 ASICs with a total gate count of 2.4 MGates. It has been proven that simulation of such a complex system can be done on VHDL level and the efficiency of finding and correcting errors early in the design cycle was demonstrated.

I. Introduction

The challenge was to simulate a pair of fully redundant boards connected back to back in real configuration with identical operating characteristics. The system itself is the heart of the Enhanced CCS7E Unit of the EWSD-System developed by Siemens AG, Public Communications Network Group.

The system simulated consists of

- 4 Pentium microprocessor
- 28 ASICs (a total gate count of 2.4 MGates)
- 320 MByte DRAM
- standard components

Due to the complexity this system required very careful engineering. Simulation started on VHDL level as early as possible in the design approach and moved towards GATE level after the functionality of the system had been verified on VHDL level.

To minimize design cycle time, a concurrent development approach was adopted and the process segmented, so that a number of geographically dispersed groups using EDA tools by different vendors were responsible for developing the ASICs and the board, with the results combined into a single product. Such a large, complex project requires very elaborate mechanisms for linking ideas, configuration data, error reporting systems, feedback reports to ASIC designers, automated data exchange - all being facts that constitute configuration management.

II. Design Flow

It was decided that simulation would have to be done from the earliest steps of the process and continue throughout the cycle, which proved to be a key in minimizing problems later. This approach meant that changes made by one group could affect what the others did, so communication was of utmost importance, as extensive and repetitive simulation was.

The intention of this system simulation was to test ASIC interoperability that is to test board functionalities running across several ASICs and other board components. In order to achieve that goal the functionality of the system was splitted into several testcases, more than 50 in number. Each testcase was supposed to test a specific board functionality. Analysis of encountered problems was done by expert engineers who focused on only a few specific testcases and handled them across the ASICs.

Testcase development and ASIC development were aligned to provide those ASIC functionalities first which had to be tested firstly by board testbenches.

Simulation was started with early available versions of VHDL ASIC codes and moved towards GATE simulation after ASIC synthesis. GATE simulation was started after VHDL simulation had checked compliance to specification. For GATE simulation the same board testbenches were used as at VHDL simulation.

Fig.1 shows that during VHDL simulation many design cycles were necessary to achieve specified functionality. These design cycles were very short and overlapping. Each new ASIC delivery to board simulation was used immediately.

III. Tool Chain

Many tools of different vendors were used due to the fact that development was geographically dispersed and every group was used to its own tool set. Therefore it was decided to ensure tool interoperability where needed (Fig. 2).

As configuration management played an utmost important role VHDL configuration statements and Mentor Viewpoints were used to control simulation configurations respectively.

It is important to note that board testbenches were used at all levels of abstractions - from VHDL level down to GATE level. To check simulation results waveforms produced by the respective simulator and bus-cycle trace files were used. Especially these bus cycle trace files were very useful to check results of regression tests to support ASIC sign-off.

A. Board Development and Simulation

The tool chain for board development started with schematic capturing of the board netlist using Cadence's GED. This schematic representation was then converted automatically to Synopsys' SGE, Mentor's Design Architect and to our in-house layout tool. *Changes and bug fixing of the board netlist were only allowed in the GED board schematic as a common source.* The

32nd ACM/IEEE Design Automation Conference ®

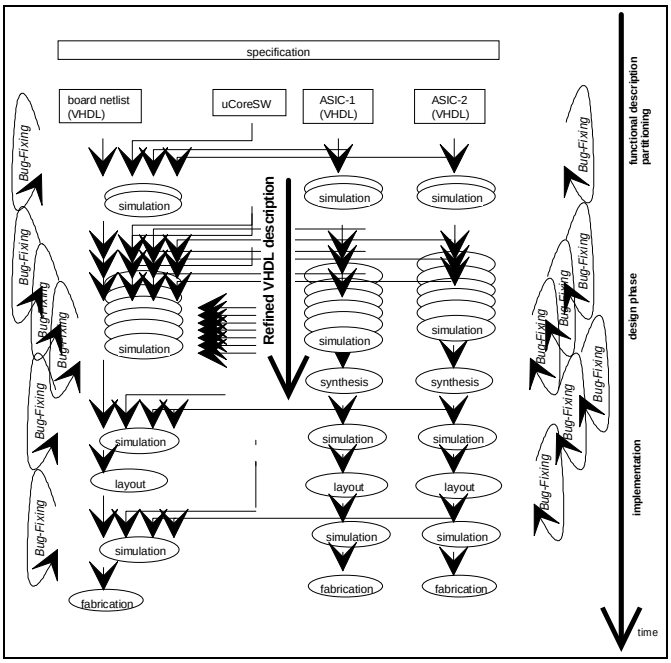

Fig. 1. The first step in the design was to prepare the specification. Each of the ASIC groups simulated its own design and afterwards board-simulation was done. The results were used to identify and correct errors, with the next version being used in additional simulations through final re-simulation for fabrication.

automated conversion was done by tools purchased from a third party vendor.

Simulation on VHDL level was done by the Synopsys VSS 3.0b simulator. The goal was to check functional compliance to specification. Simulation on GATE level was intended to check the timing behavior whereby all the worstcases (minimum and maximum) postlayout timing information of both all the ASICs and the board were included into the simulation database.

Simulation results were checked firstly by verification of waveforms generated by the simulator tools and secondly by verification of bus-cycle trace files for address and data buses (Fig. 3) generated by the VHDL testbench using VHDL textio constructs.

For each board component a VHDL model was available. For the PENTIUM either the bus-functional-model (BFM) or the LM1200-hardware model (HWM) was used. In the early stages of simulation, when memory accesses did not work properly, the PENTIUM BFM was used. After memory cycles had worked correctly we switched to the PENTIUM HWM to get closer to reality because the HWM functions equally in terms of generating more bus cycles in number, e.g. code pre-fetching cycles. For GATE simulation only the HWM was used.

The behavior of complete analog circuitries such as clock generation and clock synchronization was modeled in VHDL inside the Board Testbench were needed to verify board/system functionalities.

In case of locating an ASIC to be the source for board malfunctioning all pins of the respective ASIC were traced in

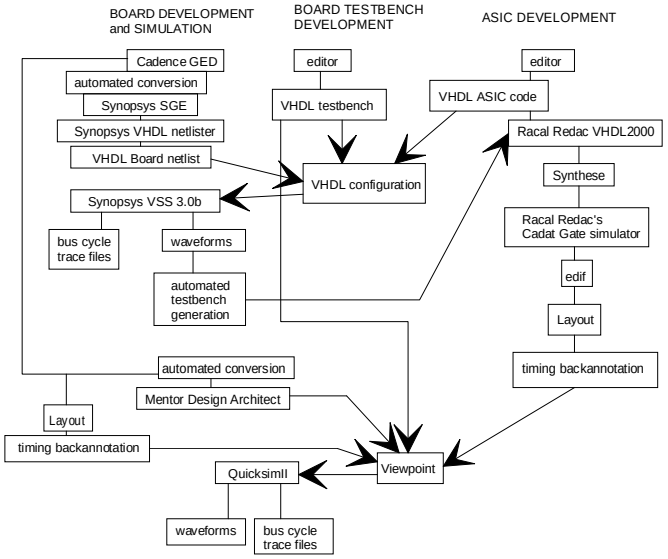

Fig. 2. This figure gives a brief overview about the tool chain and interfaces between tools by different vendors.

simulation. Out of the waveform file a VHDL ASIC testbench was generated automatically by one of our in-house tools and provided to the ASIC designers to investigate. This VHDL ASIC testbench contained all stimulus applied to the ASIC during the whole simulation and all monitored ASIC responses.

Following tool set were used:

VHDL-simulation-

- Synopsys VSS 3.0b
- PENTIUM Bus Functional Model by LMC and LM1200-HW-Model (since memory accesses had worked correctly)
- VHDL models for glue-logic components by Siemens Nixdorf AG

GATE-simulation-

- Mentor Quicksim II, V8.2
- System 1076, V8.2
- PENTIUM LM1200-HW-Model
- MENTOR models for glue-logic components

B. ASIC Development

To produce the ASIC VHDL code a standard texteditor was used. ASIC internal blocks were connected together by using a Racal Redac schematic editor and VHDL netlister. For each ASIC there was an ASIC specific testbench for the ASIC to be simulated before released to board simulation.

It was experienced that Racal Redac might have interpreted IEEE1076 differently than Synopsys did, e.g. concatenations (&) are not allowed in “case selector“ expressions for Synopsys. These and other differences were considered in ASIC VHDL code development or were automatically “corrected“ by an in-house conversion program.

C. Board Testbench Development

Board Testbench VHDL code, PENTIUM bus-functional-model code and assembler code were produced using a standard texteditor.

Since Mentor’s VHDL simulator System1076 does not fulfill the

```
========== S-BUS INTERFACE ==========
          Column1: Time
          Column2: Pipeline/Non-Pipeline
          Column3: Single/Burst
          Column4: Data/Code
          Column5: Read/Write
          Column6: Address [hex]
          Column7: BE_L [bin]
          Column8: Data [hex]
          Column9: Pipe-depth
   5805 NS: PI S C RD 0000FFF0 00000000 90909000_003000EA 1
   5945 NS: PI S C RD 0000FFF8 00000000 00000000_00000000 1
   6065 NS: PI S C RD 00000000 00000000 11111111_11111111 1
   6225 NS: PI S C RD 00000008 00000000 00CF9200_0000FFFF 1
   6525 NS: NP S C RD 00000010 00000000 00C09E00_0000FFFF 0
   6785 NS: PI S C RD 00003000 00000000 0F66100C_E0200F66 1
   6925 NS: PI S C RD 00003008 00000000 00001000_B866E022 1
   7065 NS: PI S C RD 00003010 00000000 0001B866_D8220F66 1
   7205 NS: PI S C RD 00003018 00000000 23EAC022_0F668000 1
  .....
  10365 NS: PI B C RD 00003018 00000000 23EAC022_0F668000 1
  10385 NS0001B866_D8220F66 1
  10445 NS00001000_B866E022 1
  10465 NS0F66100C_E0200F66 1
  10525 NS: NP S D RD 00001000 11110000 0040009B_00002003 0
  10785 NS: NP S D RD 00001000 11110000 0040009B_00002003 0
  10925 NS: NP S D WR 00001000 11110000 00000020_00002023 0
  .....
  58345 NS: NP S D RD 00010000 11110000 00000000_00000021 0
  58765 NS: NP S D RD 00010000 11110000 00000000_00000021 0
  59105 NS: NP S D RD 00010000 11110000 00000000_00000000 0
  59245 NS: PI S D WR 00010000 11110000 0C438005_0C438005 1
  59305 NS: PI S D WR F0300100 11110000 0C438005_0C438005 1
  59785 NS: NP S D RD 00010000 11110000 00000000_0C438005 0
  60285 NS: NP S D RD 00010000 11110000 00000000_0C438005 0
  60685 NS: NP S D RD 00010000 11110000 00000000_0C438005 0
  .....
  63265 NS: PI B C RD 000031A0 00000000 003E8300_00000207 1
  63285 NS0C438285_06C7FB75 1
  63345 NSF4B90C43_828507C7 1
  63365 NS71D8BEFE_E2000001 1
  63425 NS: NP S D RD 00010000 11110000 00000000_00010008 0
  .....
 434225 NS: NP S D RD F0290018 11111110 FFFFFFFF_00000005 0
 434525 NS: PI B C RD 00003260 00000000 00000000_00000000 1
 434545 NS00000000_00000000 1
 434605 NS00000000_00000000 1
 434625 NS00000000_00000000 1
 434665 NS: NP S D WR 00010000 11110000 0000000C_0000000C 0
 434745 NS: PI S D WR F0300100 11110000 0000000C_0000000C 1
 435185 NS: NP S D RD 00010000 11110000 00000000_0000000C 0
 435525 NS: NP S D RD 00010000 11110000 00000000_0000000C 0
 435905 NS: NP S D RD 00010000 11110000 00000000_0000000C 0
 436305 NS: NP S D RD 00010000 11110000 00000000_0000000C 0
 436705 NS: NP S D RD 00010000 11110000 00000000_0000000C 0
 437045 NS: NP S D RD 00010000 11110000 00000000_0000000C 0
 437505 NS: NP S D RD 00010000 11110000 00000000_00000000 0
 437825 NS: NP S C WR 00000000 11111011 004F0000_004F0000 0
```

Fig. 3. Example of a bus-cycle trace file generated automatically by VHDL textio constructs within board testbenches during simulation.

whole IEEE1076 standard, not supported constructs, e.g. VHDL configuration declarations, were considered in the board testbench VHDL code. Nevertheless VHDL configuration declarations were used extensively to control design hierarchy for VHDL level simulation to gain from its powerfulness and benefits. They were "replaced" by Mentor Viewpoints for GATE simulation to control design hierarchy and to connect timing backannotation information.

To program the PENTIUM either BFM code was loaded into the BFM controlled by VHDL configuration constructs or assembler code was assembled and built and loaded into FEPROM and DRAM area on the board considering the address management scheme designed for memory access cycles.

Longest simulation real times defined were about 850us for a single board configuration and 550us for a system/double board configuration.

Table I shows some performance figures measured on a SunSPARC10 workstation with 512MByte and 1.5GByte swap space equipped.

IV. VHDL-Configuration Management

This chapter addresses primarily how VHDL has been used for configuration management.

All activities shown in Fig. 1 were done in parallel - ASIC development, board development, board-testbench development, board simulation and bug fixing. Therefore many versions of each of the developed objects were produced. This resulted in following major requirements for configuration management:

- new versions must be included into simulation immediately with less effort
- including a new ASIC version in one board-testbench must not effect other testbenches
- it must be clear which version of each object is in use for current simulation runs
- it must be clear in which versions bugs were found and where these bugs have been corrected
- minimization of compilation effort
- a very effective incident reporting system
- re-simulation of ASICs as stimulated on the board at ASIC design site to investigate in ASIC bugs with no travel effort required between ASIC design site and board simulation site

A. Design Hierarchy

To control design hierarchy (Fig. 4) the VHDL configuration declarations were used.

Each delivered ASIC-VHDL code in a specific version was compiled into a unique VHDL library (design unit), e.g.

delivered container code	VHDL library
ASIC1_CZ_204	ASIC1_CZ_204
ASIC1_CZ_225	ASIC1_CZ_225

The naming scheme of delivered ASIC-VHDL codes was as follows:

<ASICname>_<VHDL leaf cell library><slice of functionality>_<version>

ASICname..... 6 types of ASICs are on this board
VHDL leaf cell library. different ASIC cell libraries were used
slice of functionality..... testbench development was aligned to
..... ASIC development
version..... unique version number

TABLE I
SOME PERFORMANCE FIGURES FOR BOTH VHDL AND GATE SIMULATION FOR 3 REPRESENTATIVE BOARD TESTBENCHES

VHDL	no. of boards	real-time [us]	simulation time [h]	real-time	:	simulation time
TC-1	1	850	19,5	1	:	82,6E+6
TC-2	2	250	13,5	1	:	194,4E+6
TC-3	2	500	23	1	:	165,6E+6
GATE	no of boards	real-time [us]	simulation time [h]	real-time	:	simulation time
TC-1	1	850	48	1	:	203,3E+6
TC-2	2	250	48	1	:	691,2E+6
TC-3	2	500	73	1	:	525,6E+6

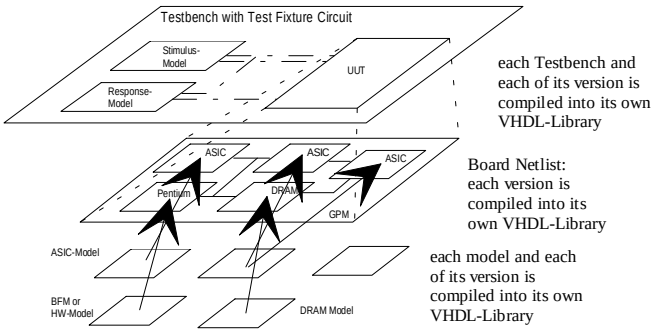

Fig. 4. A hierarchical representation of the board netlist including the design under test UUT

Only complete VHDL-ASIC codes were accepted by board simulation.

Not only ASIC versions were compiled into specific libraries, but also each version of the board netlist, each version of each testbench and the library containing all other board component VHDL models.

In order to both minimize compilation effort due to new versions used for simulation and keep up all 50 testbenches independently VHDL library clauses were used very carefully and only where really needed.

Subsequent examples show parts of the board netlist (Fig. 5), board testbench (Fig. 6) and VHDL-configuration file (Fig. 7) which is part of the testbench.

B. VHDL Board Netlist

- total line count: 5958
- compiled into VHDL-library, e.g. “GPS_288“

The board schematic itself was captured using the Cadence GED and then transformed to a Synopsys SGE input file. To generate the VHDL netlist (Fig. 5) the SGE VHDL writer was used.

As there are no board component or ASIC specific library clauses within the board netlist file this netlist can be kept up independently from ASIC VHDL code versions as long as it matches to the component declaration as specified in IEEE1076.

The VHDL board netlist entity header's port list contains all board component interconnection signals with mode "inout" and default signal expression =‘Z’ which is the weakest state defined in IEEE1164. Each board testbench can therefore use a different subset of board signals to be connected to for stimulation and monitoring. This was important because each board testbench addressed different functionality and needed therefore specific board signals to be stimulated or monitored.

C. Board Testbench

- compiled into VHDL library “WORK“

Every board testbench (Fig. 6) has its own interconnection to the unit-under-test, that are 2 boards connected back-to-back building up the system.

The entity declaration contains only a generic list in the formal generic clause to determine some values within the testbench, e.g. timing parameters.

The architecture declarative part contains signal declarations for board interconnection signals, component declarations but no component configurations.

Testbenches can use a different subset of board component interconnection signals, exactly those signals that need to be stimulated and monitored by the respective board testbench to check board functionality.

The architecture statement part contains the structural board interconnection and stimulus and response processes.

D. VHDL Configuration File

- for board testbench shown in Fig. 6
- compiled into VHDL library “WORK“

Out of the VHDL configuration file (Fig. 7) it could be seen easily which version of objects was used for simulation identified by an unique board testbench configuration identifier.

Having a new ASIC version to be used for board simulation following steps have to be done:

- the respective ASIC VHDL code has to be compiled into its unique VHDL library once
- board testbench VHDL configuration files have to be updated and compiled. Re-compilation of one testbench did not affect other board testbenches.

V. Incident Reporting System

After a malfunction has been nailed down to a specific component an incident report has been distributed to all board engineers and to the ASIC team leaders using the Internet e-mail system. The incident reports were numbered consecutively. This ensured a consequent update and follow-up of all incident reports. Each error report/incident report contains the information shown in Fig. 8.

Due to the fact that most of the functionality of this system/board is implemented within ASICs they were naturally the main source for generating incident reports.

The total number of IRs, IRs alive and finalised, was reported in a graphical representation (Fig. 9) on a weekly basis.

The curve number of IRs over time (Fig. 9) shows that during simulation phase many IRs were generated and alive. After fixing bugs and re-simulation the number of IRs alive became smaller and were related to not ASIC specific problems and the curve goes into saturation. This was one of the indicators to allow ASIC sign-off.

VI. Types of Errors Found

A total of more than 320 incident reports were raised during simulation phase.

More than 150 serious problems were found in ASICs within the system. Because they were found early through the board level simulation these errors did not appear at GATE simulation.

Most of errors encountered were bugs in the behavior of ASIC interoperability which were in large part related to misunderstandings and different interpretations of the specification.

Also many ASIC internal errors occurred due to internal counter or stack overflows.

About 35 problems were found in library models of board

```
-- VHDL Model Created from SGE Schematic gps.sch
-- May 11 13:40:50 1994

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

There are no library clauses for board components.

```
entity GPS is
      Port ( A_ADDRTCMS : InOut std_logic := 'Z';
               RSATM1I : InOut std_logic := 'Z';
               RSATM0I : InOut std_logic := 'Z';
               RSATM1O : InOut std_logic := 'Z';
               RSATM0O : InOut std_logic := 'Z';
      -- all component interconnection signals are declared
      -- within this port list with mode „inout“ and default
      -- signal expression := ‘Z’
                 .......
             );
end GPS ;

architecture SCHEMATIC of GPS is
   component GPELINK_1
      Port (  ATMADS : In    std_logic;
              .....
               TOUT : Out   std_logic );
   end component;
   component ATM30_2_1
      Port (     ACT : In    std_logic;
              .....
                 WR : InOut std_logic );
   end component;
   component GPXLINK_1
      Port ( AABTOUT : In    std_logic;
              .....
              XXADS : Out   std_logic );
   end component;
```

The architecture declarative part contains all component declarations of all board components but no component configurations.

```
-- component declarations for all board components

begin

  P2_8 : GPELINK_1
    Port Map ( ATMADS=>B_ATMADS_L, ATMBRDY=>B_ATMBRDY_L,
               .......
             RRUNBIST=>B_BSTE, TDO=>B_ELTDO, TOUT=>open );
  P18_8 : ATM30_2_1
    Port Map ( ACT=>B_ATMACT, AR=>B_ATMAR, BM0=>B_BM0, BM1=>B_BM1,
               ......
            TXD0=>A_ATMTXDO, TXD1=>A_ATMTXD1, WR=>B_OWR );
  P1_6 : GPXLINK_1
    Port Map ( AABTOUT=>B_IBTO_L, AADMANRM=>B_IDMA_L,
               ......
     XSINIT=>open, XSMUXOUT=>B_XSMUXO, XXADS=>B_XXADS_L )
```

This is a pure structural netlist.

```
-- port maps of all board components
end SCHEMATIC;
```

Fig. 5. Some important details of the VHDL board netlist

components others than ASICs and ASIC libraries.

More than 40 problems were encountered in tools or tool interfaces.

VII. Conclusion

Simulation of complex system level products can be done on a VHDL basis. The efficiency of finding and correcting errors through simulation early in the design cycle, rather than let them propagate to the end, was demonstrated.

It turned out that it is a better possibility to stress ASICs in a system/board environment where a processor is available to produce thousands of bus cycles which is a hard task to be achieved in an ASIC-testbench only by applying VHDL stimulus.

Consistency of the simulation databases was ensured by applying configuration management through the whole design approach consequently.

Such large, complex projects require very elaborate mechanisms for linking ideas, configuration data, error reporting systems, feedback reports to ASIC designers, automated data exchange - all the facts that constitute configuration management in a concurrent design environment.

```
------------------------------------------------------------------------
-- FILE: tb_soatm_01.vhd
-- AUTHOR: Thomas Albrecht
-- DEPARTMENT: SAG Oesterreich, EZE45
-- DESCRIPTION:
-- @(#) tb_soatm_01.vhd Version(1.15) 94/06/01
--
-- Copyright(c) SIEMENS AG 1994, ALL RIGHTS RESERVED
------------------------------------------------------------------------
--
library IEEE;
         use IEEE.std_logic_1164.all;
USE WORK.tracer.all;
use std.textio.all;
```

There are no library clauses for board components.

```
entity TB_SOATM_01_E is

 Generic  (
          --timing parameters--
          );

  --  Output files for Bus Tracer
  CONSTANT sbus_a0_trace_file : string := "sbus_a0_trace";

end TB_SOATM_01_E;

Architecture TB_SOATM_01_A of TB_SOATM_01_E is

  --  UUT0
  signal RSATM1I_0  : std_logic;
  signal RSATM0I_0  : std_logic;
-- all signal declarations for board_0

  --  UUT1
  signal RSATM1I_1  : std_logic;
  signal RSATM0I_1  : std_logic;
-- all signal declarations for board_1
```

The architecture declarative part contains signal declarations for signals used to interconnect both boards to a system but no component configurations.

```
  Component GPM
     Port (
     ------------------------------------------------------------------
     --  RSATM Interface
     ------------------------------------------------------------------
       RSATM1I      : InOut std_logic;
       RSATM0I      : InOut std_logic;
-- other signals to be stimulated or monitored
     );
  end component;
```

Only a subset of board component interconnection signals is used.

```
  Component icable_delay
     -- generic ();
     Port (P1 : INOUT std_logic;
           P2 : INOUT std_logic);
  end component;
```

Component declaration for a board interconnection model.

```
begin                          --test fixture circuit:
                               --wires board-netlist to testcase
  UUT0 : GPM
     Port Map (
       RSATM1I => RSATM1I_0,
       RSATM0I => RSATM0I_0,
       .....
     );
  UUT1 : GPM
     Port Map (
       RSATM1I => RSATM1I_1,
       RSATM0I => RSATM0I_1,
       .....
     );

--********Test Bench   -  User Defined Section*********
 ------------------------------------------------------------------
 --  interconnection cables between both boards
 ------------------------------------------------------------------
 XLK_X0D31: icable_delay  port map (P1 => X0D_0(31), P2 => X0D_1(31));
     .....

 TB_SOATM : block
 begin

-- stimuli processes like clock and reset generation, board settings
 .....
 ------------------------------------------------------------------
 --  Tracer procedure calls

 SBUS_A0 : PROCESS (A_CLK50PNT_0)
 -- declarations
 BEGIN  --  PROCESS SBUS_A0

 SBUS_TRACE (A_CLK50PNT => A_CLK50PNT_0,
             .....
             T_A_BE_2 => T_A_BE_2
 );

 END PROCESS SBUS_A0;

end block;
--********End Test Bench   -  User Defined Section********
end TB_SOATM_01_A;
```

The architecture statement part contains the structural board interconnection and stimulus and response models as concurrent statements.

Procedure call for bus-cycle-trace file as described in Chapter “Board Development and Simulation“.

Fig. 6. Some important details of the board testbench

```
-- ************************************************************
--  Copyright (C) 1994 - Siemens AG, Vienna EZE45.
-- ************************************************************
--  File:           tb_soatm_c_03.vhd
--  Title:          Configuration for TC_SOATM
--  Author:         Thomas Albrecht
--  Created:        Thu May 26 14:41:33 1994      <albrecht@kuldi>
--
--  @(#) tb_soatm_c_03.vhd Version(1.10) 94/08/22
--

library IEEE;
  use IEEE.STD_LOGIC_1164.ALL;

library GPS_288;     -- Board Netlist
library DAT_CZ_140; -- Library ASICs
library ADR_CZ_1240;
library XLK_CZ_122;
library ELK_CZ_229;
library P1_ATM30_V21_07;
library LRT_CZ_110;
library P1_EPLD_06;
library GPM_LIB_233; -- Library Elements
library PASSIV_DEV_08;
library MEMORY_DEV_08;
library TTL_DEV_08;
library UTIL_08;
  use UTIL_08.UTIL_P.ALL;
library VHDL_SHELLS_04;
library SYNOPSYS;
  USE SYNOPSYS.attributes.ALL;

configuration CFG_TB_SOATM_03 of TB_SOATM_01_E is
for TB_SOATM_01_A

for all : icable_delay
  use entity WORK.ICABLE_DELAY_E(ICABLE_DELAY_A)
  generic map (t_delay  => 8 NS);
end for;

for UUT0, UUT1 : GPM
 use entity GPS_288.GPS(SCHEMATIC);
 for SCHEMATIC
  for P46_8: R15N1B_PU_1
      use CONFIGURATION GPM_LIB_233.CFG_R15_PU_1;
  end for;
  for P90_8, P67_8, P39_8, P14_8,
      P1_8, P2_7, P1_7, P8_7: ABT16245_1
      use ENTITY TTL_DEV_08.SN74ABT16245(SN74ABT16245_A);
  end for;
  for P91_8, P93_8, P101_8, P98_8,
      P78_8, P26_8, P51_8, P50_7: R15PU_SIZE_4
      use CONFIGURATION GPM_LIB_233.CFG_R_PU_1;
  end for;

  for P2_8, P4_7: GPELINK_1
      use CONFIGURATION ELK_CZ_229.CFG_GPELINK_1;
  end for;

  for P18_8, P17_7: ATM30_2_1
      use configuration P1_ATM30_V21_07.CFG_ATM30_1;
  end for;
  for P1_6, P1_5: GPXLINK_1
      use CONFIGURATION XLK_CZ_122.CFG_GPXLINK_1;
  end for;
for P2_2, P2_1: PENTIUM_1
      use entity VHDL_SHELLS_04.PENTIUM_LM1200(STRUCTURAL)

          Generic Map (
             NewStartAddress => "00000000000000001111111111110000",
             P5_StartAddress => "11111111111111111111111111110000",
             Run_Bist  => FALSE
             );

             for STRUCTURAL
               for PENTIUM : P5LM1200
                   USE configuration GPM_LIB_233.CFG_PENTIUM_1
                     Generic Map (
                       timing => disabled,
                       delay  => typical
                     );
                end for;
              end for;
  end for;
.........
end for;                                  --  UUT

  for TB_SOATM
  end for;
 end for;                                   --  TB
end CFG_TB_SOATM_03;
```

Library clauses to control the versions (design units) to be used in this simulation.

Configuration declaration.

An unique board testbench configuration identifier allows to hold different configurations in parallel with reference to version of board testbench entity and architecture identifier.

Generic map aspects to control behavior of instanced components.

Configuration specification for a certain design unit.

Each ASIC has a top level configuration identifier.

Unique VHDL library where this ASIC version has been compiled into.

Generic map aspect to control behavior of instanced components; in this case to aviod the PENTIUM to run into BIST and to start program execution at a different DRAM address as it would be by default.

Fig. 7. Some important details of the VHDL configuration file

```
** INCIDENT REPORT **
** GP Board-Level Simulation **

IR_Number:          IR_BoardSim_<consecutive number>
Status:             LIVE | FINALISED

Owner of this IR:   responsible person
related IRs:

originator:         name of person who raised the IR
date:               , 1994

Name of Testcase:   e.g. TB_SOATM_01
Name of Testbench-Configuration:  CFG_TB_SOATM_03  (as in Fig. 7)

Problem:
 - brief description
 - in case of an ASIC bug where to retrieve the VHDL ASIC testbench
generated out of board simulation to investigate in ASIC bugs at ASIC design
site from a common data pool.

Solution:   - proposed or requested solution

** END of REPORT **
```

Fig. 8. Template for Incident Reports

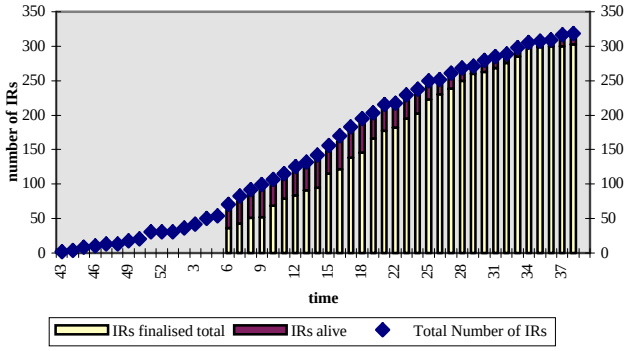

Fig. 9. Number of IRs over time

ACKNOWLEDGMENTS

I express my thanks to Mr. Johann Haslinger for the approval of publication, Mr. Horst Kukse and his team, Mr. Michael Bruenger and his team, all of Siemens Public Communications Network Group in Munich/Germany and Dr. Tobolka of Siemens Engineering Center of Electronics in Vienna/Austria for many helpful discussions and on comprehensive comments on early drafts of this work.

REFERENCES

[1] Peter Kensett. “Concurrent Engineering, Crisis or Opportunity“. *Presentation at EuroDAC ‘94*, September 1994

[2] Kenneth A. Radtke. “The AT&T 5ESS™ Hardware Design Environment: A Large System’s Hardware Design Process“. *In Proceedings 31st ACM/IEEE Design Automation Conference*, pages 527-531, June 1994

[3] IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-1987